Journal of Theoretical and Applied Information Technology
31st August 2014. Vol. 66 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SURVEY ON ADAPTIVE JOB SCHEDULERS IN MAPREDUCE

MAEDEH MOZAKKA, FARAMARZ SAFI ESFAHANI, MOHAMMAD H. NADIMI
Faculty of computer engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran
E-mail: maedeh.mozaka@gmail.com , fsafi@iaun.ac.ir, nadimi@iaun.ac.ir

ABSTRACT

In recent years the adaptive job schedulers became an attraction point to many researchers, but despite many efforts that have been made, there are still many challenges in this area. The aim of this paper is to provide a better understanding of adaptive job schedulers in MapReduce and identify important research directions in this area. In this paper, adaptive job schedulers in MapReduce are classified in four categories. The techniques used in them are described briefly, we present advantages and disadvantages of different adaptive schedulers and then their features and the application of each category of the schedulers are expressed. Finally, we will discuss some current challenges and future works.

Keywords: *Adaptive Job Scheduler, Cloud Computing, Hadoop, Job Scheduling, MapReduce.*

1. INTRODUCTION

One of research challenges in cloud computing is their software frameworks. Cloud computing provides a proper platform for hosting large-scale data-intensive applications. Typically, these applications leverage MapReduce frameworks such as Hadoop for scalable and fault-tolerant data processing [1]. MapReduce is a programming model as well as a framework that supports the model. The main idea of the MapReduce model is to hide details of parallel execution and allow users to focus only on data processing strategies [2]. Users specify the computation in terms of a *map* and a *reduce* function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks [3]. Hadoop is an open-source implementation for MapReduce. Job scheduling in multi-user environments is an open issue that has not been well addressed yet [2]. In MapReduce, the job submitted by user is divided into several tasks. There are two types of task in MapReduce: map task and reduce task. Each node is a physical machine with computational and storage capabilities. Hadoop uses the number of slots concept for each node in order to control the maximum number of tasks that can be executed concurrently on a node. Each slot of the node at any time is only capable of executing one task. In MapReduce, there are two types of slot: map slot, and reduce slot. Scheduling decisions are taken by a master node, called the JobTracker, and the worker nodes that called TaskTracker execute the tasks [4]. In general, job schedulers in MapReduce are classified in two categories: static schedulers (such as FIFO) and dynamic schedulers. Dynamic schedulers in MapReduce are classified in two categories: dynamic schedulers that operate based on non-environmental factors and dynamic schedulers that operate based on environmental factors which are called adaptive scheduler. In section 2, we briefly discuss the adaptive job schedulers and techniques used for them. In Section 3, advantages, disadvantages, features and the application of adaptive job schedulers are discussed, and Section 4 includes future works and conclusions.

2. THE ADAPTIVE JOB SCHEDULERS IN MAPREDUCE

2.1 Data Locality Ameliorator Schedulers

When input data is nearer to the computation node, it has a lower data transfer cost. Locality is a very crucial issue affecting performance in a shared cluster environment, due to limited network bisection bandwidth [5]. Leitao Guo (2009) et al. [6] offered a data distribution aware task scheduling strategy for MapReduce system. Their strategy has two main phases: In the initialization phase, statistics number of copies of data processed by each map task. At the same time, statistics the number of localizable tasks for each worker; in the scheduling phase, according to the information above, calculating the scheduling priorities for each

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

task and each works that requesting task, and scheduling the task to works based on this priority. This strategy regarding the distribution of data, schedules map tasks on the nodes that most likely contain relevant data, and reduces network overhead and improves the performance of system. Hadoop schedules reduce tasks at requesting nodes without considering data locality leading to performance degradation. Mohammad Hammoud (2011) et al. [7] proposed LARTS, a practical strategy which uses network locations and partition sizes to improve data locality. JobTracker schedules reducers only in TaskTrackers which are selected by reducers. If some of TaskTrackers receive more reducers than others, scheduling skew will occur. Because the TaskTrackers that are available but not selected by any reducer, remain unused and leading to a reduction in exploiting parallelism and system efficiency. LARTS attempts to schedule reducers as close as possible to their *maximum* amount of input data and conservatively switches to a *relaxation* strategy seeking a balance between scheduling delay, scheduling skew, system utilization, and parallelism. In general, the performance of map tasks is affected by waiting time and transmission time. The waiting time of a task is the shortest time that the task has to wait before it is scheduled to one of the nodes storing the input data. The transmission time is the time needed to copy the input data of the task to the node requesting tasks. Xiaohong Zhang (2011) et al. [8] proposed an effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments. The objective of their method is to make a tradeoff between waiting time and transmission time at runtime when schedule a task to a node and obtain optimal task execution time. In brief, after receiving a request from a requesting node, the method preferentially schedules the task whose input data is stored on the requesting node. If there exist no such tasks, the method first selects the task whose input data is nearest to the requesting node, and then 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 storing the input data. Otherwise, it schedules the task to the requesting node. Despite previously proposed optimizations related to management of straggler tasks, MapReduce implementations were still weak in heterogeneous clusters. Faraz Ahmad (2012) et al. [9] proposed some optimization methods for improving MapReduce performance on heterogeneous clusters as Tarazu. Tarazu consists of (1) Communication-Aware Load Balancing of map computation (CALB) across the nodes, (2) Communication-Aware Scheduling of map computation (CAS) to avoid bursty network traffic and (3) Predictive Load Balancing of reduce computation (PLB) across the nodes. Shadi Ibrahim (2012) et al. [10] suggested a replica aware scheduling method as Maestro to reduce map tasks which process remote data and cause excessive traffic network. Maestro schedules the map tasks considering chunk locality and node availability. The scheduling of Maestro is in two waves: first wave scheduler and run time scheduler. The first wave scheduler is responsible for filling the empty slots of each data node based on the number of hosted map tasks and on the replication scheme for their input data. Runtime scheduling takes into account the probability of scheduling a map task on a given machine depending on the replicas of the task's input data. These two waves lead to a higher locality in the execution of map tasks and to a more balanced intermediate data distribution for the shuffling phase.

Xiaohong Zhang (2012) et al. [11] proposed a scheduling method in order to improve data locality. After receiving a request from a requesting node, their method preferentially schedules the task whose input data is stored on the requesting node. If no such tasks exist, their method will select the task whose input data is nearest to the requesting node, and then make a decision on whether to reserve the task for the node storing the input data or schedule the task to the requesting node by transferring the input data to the requesting node on the fly. This method increases the performance of the system.

Mohammad Hammoud (2012) et al. [12] proposed COGRS. COGRS is a locality-aware skew-aware reduce task scheduler for saving MapReduce network traffic. This scheduler attempts to schedule every reduce task at its center-of-gravity node determined by the network locations of that task's feeding nodes and the skew in the sizes of that task's partitions. By scheduling reducers at their center-of-gravity nodes, they argue for reduced network traffic which can possibly allow more MapReduce jobs to co-exist on the same system. CoGRS controllably avoids scheduling skew, a situation where some nodes receive more reduce tasks than others, and promotes pseudo-asynchronous map and reduce

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

phases. If there is a tendency for a job to be assigned the same slot repeatedly, sticky slot occurs. This problem aroused because following a strict queuing order forces a job with no local data to be scheduled [4].

Yanrong Zhao (2012) et al. [13] have designed and implemented a new job scheduling algorithm (TDWS) for Tencent Distributed Data Warehouse (TDW) which is based on Hadoop MapReduce. This scheduler avoids from occurring sticky slots and starvation of small jobs. Firstly, TDWS organizes jobs into several separate groups and shares resources between these groups to meet the needs of different applications, each group for one type of job. Secondly, TDWS takes memory heterogeneity into account and implements a memory-awareness mechanism to gather memory information to handle the problem. TDWS uses Delay scheduling strategy [14] in order to increase data locality.

Arun Kumar K. (2012) et al. [15] offered CASH, which is a Context Aware Scheduler for Hadoop. This scheduler increases the performance in heterogeneous Hadoop clusters. CASH Classifies the jobs as CPU or I/O bound. This scheduler classifies the nodes as Computational or I/O good. Then it maps the tasks of a job with different demands to the nodes which can fulfill the demands. Thus, by implementing CASH the performance of the heterogeneous cluster and the aggregate execution times of the jobs can be improved.

2.2. Adaptive Schedulers Based on Speculative Execution

These schedulers identify slow tasks and then launch several backup tasks for them on the other nodes. When a slow task or a backup copy of that terminates, all the other similar tasks (slow task or the other backup tasks) are killed. If a backup task ends earlier than slow task, then the overall performance improves.

Quan Chen (2010) et al. [16] proposed SAMR scheduling algorithm. *SAMR* is inspired by LATE [17] scheduling algorithm. This scheduling algorithm holds historical information on each node. The TaskTracker, reads the historical information and sets the parameters using these informations. Quan Chen et al. proposed several equations for finding slow tasks and slow TaskTrackers. The scheduler can launch backup tasks for slow tasks according to these equations. SAMR launches backup tasks for map tasks on the rapid nodes or on the slow reduce nodes, and launches the backup tasks for reduce tasks on the rapid nodes or on the slow map nodes. When a task or several tasks execute on TaskTracker, execution informations feedbacks to TaskTracker, and historical informations update using them. SAMR computes progress score of tasks more accurate than LATE, thus this scheduler launches backup tasks for really slow tasks that prolong job execution time.

Although the LATE scheduler is proposed to resolve some problems that occur in heterogeneous environments, but this scheduler is designed to minimize the response time of first job in the job queue, and will prolong the response time of the other jobs in the job queue. LATE scheduler uses the past information to estimate the time to finish of tasks and is not suitable for environments with dynamic loading.

Hsin-Han You (2011) et al. [18] offered load-aware scheduler for MapReduce framework in heterogeneous cloud environments, and abbreviated it as LA scheduler. This scheduler improves the overall performance of Hadoop clusters.

Quan Chen (2011) et al. [19] suggested HAT scheduler for heterogeneous environments. This scheduler calculates the progress of tasks more accurate than previous methods and adapts with different environments automatically. HAT uses historical informations which are stored on each node to set the parameters and identifies slow tasks dynamically. Quan Chen et al. proposed several equations to calculate the weight of new phases, the progress score of map and reduce tasks, to detect slow tasks and slow nodes. Based on the accurate-calculated progress of tasks, HAT estimates the remaining time of tasks accurately and further launches backup tasks for the tasks that have the longest remaining time. HAT estimates progress of a task accurately since it tunes the weight of each phase of a map task and a reduce task automatically according to the historical values of the weights. HAT, further classifies slow nodes into map slow nodes and reduce slow nodes. In this way, HAT can launch backup tasks for reduce straggler tasks on map slow nodes and vice versa.

Self-Adaptive MapReduce scheduling algorithm (SAMR) uses historical information to adjust stage weights of map and reduce tasks when estimating task execution times. However, SAMR does not consider the fact that for different types of jobs their map and reduce stage weights may be different. Even for the same type of jobs, different

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

datasets may lead to different weights.

Xiaoyn Sun (2012) [20] proposed the ESAMR algorithm to overcome these problems. ESAMR classifies the historical information stored on every node into k clusters using a machine learning technique. If a running job has completed some map tasks on a node, ESAMR records the job's temporary map phase weight (i.e., M1) on the node according to the job's map tasks completed on the node. The temporary M1 weight is used to find the cluster whose M1 weight is the closest. ESAMR then uses the cluster's stage weights to estimate the job's map tasks' TimeToEnd on the node and identify slow tasks that need to be re-executed. If a running job has not completed any map task on a node, the average of all k clusters' stage weights are used for the job. In the reduce stage, ESAMR carries out a similar procedure. After a job has finished, ESAMR calculates the job's stage weights on every node and saves these new weighs as a part of the historical information. Finally, ESAMR applies k-means, a machine learning algorithm, to re-classify the historical information stored on every worker node into k clusters and saves the updated average stage weights for each of the k clusters. By utilizing more accurate stage weights to estimate the TimeToEnd of running tasks, ESAMR can identify slow tasks more accurately than SAMR, LATE, and Hadoop default scheduling algorithms.

2.3 Performance Manager Schedulers

These schedulers are capable to manage the scheduling in line with user goals. Jorda Polo (2009) et al. [21] proposed a QOS-Oriented scheduler. The goal of this scheduler is to execute running jobs using as few resources as possible while trying to meet the user-provided deadline for each job. The deadline scheduler works by assigning available task slots to the job with the largest positive need of CPU, which roughly translates as the job that needs more resources to be completed on time. The need is defined as the difference between the estimated number of slots required to meet the deadline and the number of running tasks. Since the scheduler should have an idea of the job's needs as soon as possible, jobs with no completed and no running tasks always take precedence over other jobs. For jobs that their deadline is expired, if it isn't possible to meet their deadlines, the scheduler tries to at least complete them as soon as possible, prioritizing them over any other kind of job, which in turn also helps to avoid jobs starvation.

Jaideep Datta Dhok (2010) [22] proposed LSCHED for task scheduling in Hadoop. This scheduler is able to maintain user specified level of utilization on cluster nodes. LSCHED builds a list of candidate jobs. For each job in the queue of the scheduler, one candidate instance for Map part and one (or zero, if the job does not have a reduce part) for the Reduce part is added in the list. Then LSCHED classifies the candidate jobs into two classes, good and bad, using a pattern classifier. Tasks of good jobs do not overload resources at the TaskTracker during their execution. Jobs labeled bad are not considered for task assignment. If the classifier labels all the jobs as bad, no task is assigned to the TaskTracker. If after classification, there are multiple jobs belonging to the good class, then LSCHED chooses the task of a job that maximizes expected utility (E.U. (J)). Cluster administrator determines the task priority, and this priority can be used for execution policy. Once a task is assigned, LSCHED observes the effect of the task from information contained in subsequent heartbeat from the same TaskTracker. Based on this information, if the TaskTracker is overloaded, LSCHED concludes that last task assignment was incorrect. The pattern classifier is then updated (trained) to avoid such assignments in the future. The Adaptive scheduler [23] dynamically predicts the performance of concurrent MapReduce jobs and adjusts the resource allocation for the jobs. It allows applications to meet their performance objectives without overprovisioning of physical resources. But this scheduler is completely unaware of unique capabilities of the hardware.

Jorda Polo (2010) et al. [24] improved the Adaptive scheduler to meet user defined high level performance goals. They exploited from the capabilities of hybrid systems transparently and efficiently. The completed adaptive scheduler is hardware-aware and is able to co-schedule accelerable and non-accelerable jobs on a heterogeneous MapReduce cluster.

Ying Li (2011) et al. [25] proposed a power-aware scheduling algorithm for MapReduce jobs in heterogeneous cloud resources in order to energy saving. This scheduler considers users' SLAs (Service Level Agreements). Since MapReduce framework is generally for data-intensive cloud computing, they considered energy saving both in processing elements and in disk storages.

Jorda Polo (2011) et al. [26] proposed RAS in order to maximize the utilization of system

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

resources and to meet the users' job completion time goals. RAS extends the abstraction of 'task slot' to 'job slot', and leverages resource profiling information to obtain better utilization of resources and improves application performance. RAS seeks to meet soft-deadlines via a utility-based approach and adapts to changes in resource demand by dynamically allocating resources to jobs. This scheduler differentiates between map and reduce tasks when making resource-aware scheduling decisions.

2.4 Resource Contention Reducer Schedulers

This category of schedulers with consideration of the resource metrics of each TaskTracker (such as CPU utilization, number of page faults per unit of time, etc.), try to allocate tasks to the TaskTrackers so that avoid race conditions. This category of schedulers, increase the resource utilization of TaskTrackers.

Mark Yong (2009) et al. [27] proposed two resource-aware scheduling mechanisms to minimize resource contention on machines: Dynamic free slot advertisement mechanism and Free slot priorities/filtering mechanism. In Free slot priorities/filtering mechanism, cluster administrators retain the fixed maximum number of compute slots per node at configuration time. As TaskTracker slots become free, they are buffered for some small time period and advertised in a block. TaskTracker slots with higher resource availability are presented first for scheduling tasks on. Instead of scheduling a task onto the next available free slot, job response time would improve by scheduling it onto a resource-rich machine, even if such a node takes a longer time to become available.

Hong Mao (2011) et al. [28] proposed a load-driven task scheduler in order to decrease runtime cost of MapReduce jobs and to improve hardware resource utilization rate. This scheduler uses Ganglia [29] to monitor the status of Hadoop cluster. When the cluster is running MapReduce job, Ganglia will get all the performance metrics and load condition of the nodes. Tasks allocate to the TaskTrackers according to workload of slave nodes. To this end, the scheduler uses a Dynamic Slot Controller (DSC). DSC adjusts the number of map slots and number of reduce slots for TaskTrackers adaptively. By DSC, the workload of each slave node running TaskTracker would be between lower limit and upper limit.

Radheshyam Nanduri (2011) et al. [30] proposed a job-aware scheduling algorithm for MapReduce in order to reduce the jobs execution time. From the list of available pending tasks, the scheduler selects the one that is most compatible with the tasks already running on that node. This scheduler employs an event capturing mechanism on theTaskTrackers [31] which listens to events related to cpu, memory, disk and network to monitor resource usage characteristics of that particular task.

Yi Yao (2013) et al. [32] proposed LsPS, which leverages the knowledge of workload patterns to improve the system performance by dynamically tuning the resource shares among users and the scheduling algorithms for each user.

Zhe Wang (2013) et al. [33] proposed a scheduling strategy for heterogeneous clusters based on job type classification. This scheduling strategy includes two parts. 1) Divide the job dynamically into two types based on cluster historical operating data: CPU - intensive and I/O-intensive. 2) To remove the influence of noise data on the reliability of historical data, they offered a scheduling strategy-- CICS: CPU and I/O Characteristic estimation Strategy. This strategy is mainly based on classical FCFS and has been modified intensively on Fairness.

3. ADVANTAGES, DISADVANTAGES, FEATURES AND APPLICATIONS OF ADAPTIVE JOB SCHEDULERS

Advantages and disadvantages of adaptive job scheduling methods are expressed in Tables 1, 2, 3 and 4. Reduce tasks are divided into three sub-phases: shuffle, sort and reduce. Shuffle sub-phase, copies the map outputs from maps host machines, to the reducer machines.

If reduce tasks begin after completion of a certain percentage of map tasks (e.g. 5%), during the maps executions, the shuffling of partitions will be done and thus the turnaround time of MapReduce jobs will decrease, and early shuffle is done. Sweet spot of a program is the spot at which early shuffle is triggered and provides the best performance for the program. But in LARTS and COGRS, the sweet spot is determined statically, which is the disadvantage of these schedulers. The features of adaptive job schedulers are expressed in Table 5 and the application of adaptive job schedulers are expressed in Table 6.

Journal of Theoretical and Applied Information Technology
31st August 2014. Vol. 66 No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Data Locality Ameliorator Schedulers

Method	Advantages	Disadvantages
Leitao Guo et al.(2009)	• Reduce network overhead • Improve system efficiency	-
LARTS	• Improve scheduling delay, scheduling skew, system utilization, and parallelism • Reduce network traffic • Increase of performance	• Static sweet spot determination
Xiaohong Zhang et al.(2011)	• Reduce normalized runtime • Reduce job response time	-
Maestro	• Reduce network traffic • Reduce runtime • Suitable for homogeneous and heterogeneous environments	-
Xiaohong Zhang et al.(2012)	• Increase of performance	-
COGRS	• Reduce network traffic • Reduce job runtime	• Static sweet spot determination
TDWS	• Suitable for heterogeneous environments • Avoid job starvation • Avoid occurrence of sticky slots • Rapid execution of urgent jobs • Maintain FIFO scheduling benefits	-
CASH	• Reduce runtime • Increase overall throughput	-

Table 2: Adaptive schedulers based on speculative execution

Method	Advantages	Disadvantages
SAMR	• Reduce runtime • Save system resources • Scalability	• Ignore different weights for different job types and different dataset sizes • Ignore data locality for launching backup tasks
LA	• Reduce response time • Increase of cluster utilization	• Ignore data locality for launching backup tasks
HAT	• Increase of system performance • Scalability	• Ignore different weights for different job types and different dataset sizes • Ignore data locality for launching backup tasks
ESAMR	• Reduce runtime • Save system resources • Scalability	• Ignore data locality for launching backup tasks

Table 3: Performance manager schedulers

Method	Advantages	Disadvantages
Jorda Polo et al. (2009)	• Ability of performance prediction and performance management • Save physical resources	• Inappropriate for applications with different performance objectives
LSCHED	• Ability of fast learning • Ability to achieve the user specified level of node utilization • Increase of performance	• Low utilization of resources in learning phase • Ineffective for tasks with unpredictable behavior • Decrease of performance in very large clusters
Jorda Polo et al. (2010)	• Ability to achieve user specified high level performance goals even in the presence of hybrid systems and accelerable jobs	-
Ying Li et al. (2011)	• Save energy consumption • Consider Users' SLA	-
RAS	• Meet users' completion time goal • Improve resource utilization • Increase of performance	• Lack of support for preemption of reduce tasks • Need for additional monitoring and forecasting capabilities to manage network bottlenecks

Table 4: Resource contention reducer schedulers

Method	Advantages	Disadvantages
Mark Yong et al. (2009)	• Reduce contention for CPU resources and I/O on the worker machines • Increase the performance of cluster	-
Hong Mao et al. (2011)	• Reduce runtime • Improve the utilization of CPU and other cluster resources	• Lack of support for multi job environments
Radheshyam Nanduri et al. (2011)	• Reduce runtime • Maximize the utilization of nodes resources • Ability to plug into FAIR and Capacity schedulers • Ability to implement in any distributed environment	-
LsPS	• Reduce runtime • Suitable for workloads which include jobs with different sizes	• Ignore the factor of priority for the share assignment
Zhe Wang et al. (2013)	• Reduce runtime • Improve resource utilization	• Having linear relative with historical data

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table5: Features of adaptive schedulers

Method \ Features	Runtime reduction	Network traffic reduction	Resource utilization increment	Suitable for heterogeneous environments	Scalability	Satisfaction of users' high level performance goals
Leitao Guo et al. (2009)		✓				
LARTS		✓	✓			
Xiaohong Zhang et al. (2011)	✓	✓		✓		
CAS	✓	✓		✓		
Maestro	✓	✓		✓		
Xiaohong Zhang et al. (2012)	✓	✓				
COGRS	✓	✓				
TDWS	✓	✓		✓		
CASH	✓	✓		✓		
SAMR	✓		✓	✓	✓	
LA			✓	✓		
HAT				✓	✓	
ESAMR	✓		✓	✓		
Jorda Polo et al. (2009)			✓			✓
LSCHED			✓	✓		✓
Jorda Polo et al. (2010)			✓	✓		✓
Ying Li et al. (2011)	✓					✓
RAS	✓		✓			✓
Mark Young et al.(2009)			✓			
Hong Mao et al. (2011)	✓		✓			
Radheshyam Nanduri et al. (2011)	✓		✓			
LsPS	✓					
ZheWang et al. (2013)	✓		✓	✓		

Table 6: Application of each category of schedulers

Scheduling type	Application
Data locality ameliorator schedulers	• Network traffic reduction • Performance enhancement
Adaptive schedulers based on speculative execution	• Performance enhancement in heterogeneous environments
Performance manager schedulers	• Satisfaction of Users' high level performance goals (performance management)
Resource contention reducer schedulers	• Resource utilization enhancement • System performance enhancement

4. CONCLUSION AND FUTURE WORKS

In this paper, adaptive job schedulers categorized into four groups and each one of them were briefly described. All schedulers in each group have the same goals and schedule tasks in similar ways. In this paper, the positive and negative aspects of each scheduler and the features of them were expressed. As mentioned in previous sections, data locality ameliorator schedulers, by reducing network traffic, schedulers based on speculative execution, by launching backup tasks for slow tasks, and resource contention reducer schedulers, through increasing utilization of system resources, increase the performance of system. And performance manager schedulers are capable to meet high level performance goals of users. In the following we mention the important research challenges in this area. In the proposed method by Leitao Guo (2009) et al., the optimization of data distribution-aware scheduling strategy, such as the management of used resources by jobs and the schedule algorithm based on the loads of all nodes, is a Promising direction for future works. There are four main future directions For LARTS: First, sweet spots can be located dynamically rather than statically. Second, LARTS can be applied to speculative tasks in addition to regular ones. Third, exploring LARTS's potential in shared (or heterogeneous) computation environment with a large-scale cluster. Testing and analyzing LARTS with various scientific applications is also an imperative future direction. Next step in proposed method by Xiaohong Zhang (2001) et al. is to focus on the technique which can transmit data sets in Hadoop environments more effectively. Improving the Maestro algorithm to work in dynamic settings in public clouds, and evaluation of Maestro implementation in shared environments with the possible integration with existing schedulers such as the Fair scheduler, are the works that has not been done yet. Dynamic determination of sweet spots in COGRS is a promising research direction. We can extend CASH in three directions. First, implementing a new 'data placement policy' where the input data of a computationally/disk intensive job is placed on a computationally/disk better node. Second, finding a finer classifier for jobs and nodes. Third, taking into account the network traffic before assigning a task to a node. One of future works for HAT is to address the data locality problem when launching backup tasks. Another possible future direction for HAT is to profile a small-scale of the MapReduce application first before the real execution. In ESAMR algorithm, lack of consideration of data locality when launching backup tasks, is one of the problems that are still unresolved. With consideration of energy saving techniques in communication devices, we

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

can improve the proposed method by Ying Li (2001) et al. RAS considers three resource capacities: CPU, memory and I/O. It can be extended easily to incorporate network infrastructure bandwidth and storage capacity of the TaskTrackers. Future work of proposed method by Radheshyam Nanduri (2011) et al. includes tuning of MapReduce framework with different configuration parameters for finding best runtime of the jobs through machine learning techniques. presenting scale up/down algorithms with resource-aware scheduling to switch on/off the virtual machines/nodes based on the resource usage of the cluster in order to save energy, is another promising research direction for proposed method by Radheshyam Nanduri et al. The ability to make Hadoop scheduler resource aware is one of emerging research problems that grabs the attention of many of researchers. We hope the present article can take a small pace in introduction of adaptive job schedulers, survey of positive and negative aspects of these schedulers and identification of current challenges in this area.

REFRENCES:

[1] Q. Zhang, L. Cheng et al., "Cloud computing: state-of-the-art and research challenges", *Journal of Internet Services and Applications*, 2010, Vol. 1, No. 1, pp. 7-18.

[2] K. H. Lee, Y. J. Lee et al., "Parallel data processing with MapReduce: a survey", *ACM SIGMOD Record*, 2011, Vol. 40, No. 4, pp. 11-20.

[3] J. Dean, S. Ghemawat, " MapReduce: simplified data processing on large clusters", *Communications of the ACM*, 2008, Vol. 51, No. 1, pp. 107-113.

[4] B. T. Rao, L. S. S. Reddy, "Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments", *International Journal of Computer Applications*, 2011, Vol. 34, No. 9, pp. 29.

[5] D. Yoo, K. M. Sim, "A comparative review of job scheduling for MapReduce", *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference*, 2011, pp. 353-358.

[6] L. Guo, H. Sun et al., "A Data Distribution Aware Task Scheduling Strategy for MapReduce System", *Cloud Computing*, 2009, pp. 694-699.

[7] M. Hammoud, M. F. Sakr, "Locality-Aware Reduce Task Scheduling for MapReduce", *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference*, 2011, pp. 570-576.

[8] X. Zhang, Y. Feng et al., "An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments", *Cloud and Service Computing (CSC)*, 2011 International Conference, 2011, pp. 235-242.

[9] F. Ahmad, S. T. Chakradhar et al., "Tarazu: optimizing MapReduce on heterogeneous clusters", *Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems, ACM*, 2012, pp. 61-74.

[10] S. Ibrahim, H. Jin et al., "Maestro: Replica-Aware Map Scheduling for MapReduce", *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium*, 2012, pp. 435-442.

[11] X. Zhang, Y. Ding, "A distribution aware scheduling method in MapReduce", *Electrical & Electronics Engineering (EEESYM), 2012 IEEE Symposium*, 2012, pp. 128-131.

[12] M. Hammoud, M. S. Rehman et al., "Center-of-Gravity Reduce Task Scheduling to Lower MapReduce Network Traffic", *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference*, 2012, pp. 49-58.

[13] Y. Zhao, W. Wang et al., "TDWS: A Job Scheduling Algorithm Based on MapReduce", *Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference, IEEE*, 2012, pp. 313-319.

[14] M. Zaharia, D. Borthakur et al., "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling", *Proceedings of the 5th European conference on Computer systems, ACM*, 2010, pp. 265-278.

[15] K. A. Kumar, V. K. Konishetty et al., "CASH: context aware scheduler for Hadoop", *Proceedings of the International Conference on Advances in Computing, Communications and Informatics, ACM*, 2012, pp. 52-61.

[16] Q. Chen, D. Zhang et al., "Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment", *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference*, 2010, pp. 2736-2743.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[17] A. Konwinski, "Improving mapreduce performance in heterogeneous environments", *Technical Report No. UCB/EECS-2009-183, University of California, Berkeley*, 2009.

[18] H. H. You, C. C. Yang et al., "A load-aware scheduler for MapReduce framework in heterogeneous cloud environments", *Proceedings of the 2011 ACM Symposium on Applied Computing*, 2011, pp. 127-132.

[19] Q. Chen, M. Guo et al., "HAT: history-based auto-tuning MapReduce in heterogeneous environments", *The Journal of Supercomputing*, 2011, pp. 1-17.

[20] X. Sun, "An Enhanced Self-Adaptive MapReduce scheduling algorithm", *Master Thesis, University of Nebraska, Lincoln*, 2012.

[21] J. Polo, D. D. Nadal et al., "Adaptive task scheduling for multijob mapreduce environments", *Technical report UPC-DACRR-CAP-2009-28, Departament d'Arquitectura de computadors, universitat polit `ecnica de catalunya*, 2009.

[22] J. D. Dhok, "Learning Based Admission Control and Task Assignment in MapReduce", *Master Thesis, International Institute of Information Technology, Hyderabad, India*, 2010.

[23] J. Polo, D. Carrera et al., "Performance-driven task co-scheduling for mapreduce environments", *Network Operations and Management Symposium (NOMS), 2010 IEEE*, 2010, pp. 373-380.

[24] J. Polo, D. Carrera et al., "Performance management of accelerated mapreduce workloads in heterogeneous clusters", *39th International Conference on Parallel Processing (ICPP2010)*, 2010.

[25] Y. Li, H. Zhang et al., "A Power-Aware Scheduling of MapReduce Applications in the Cloud", *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference*, 2011, pp. 613-620.

[26] J. Polo, C. Castillo et al., "Resource-aware adaptive scheduling for mapreduce clusters", *Middleware 2011*, 2011, pp. 187-207.

[27] M. Yong, N. Garegrat et al., "Towards a Resource Aware Scheduler in Hadoop", *Proc. ICWS*, 2009, pp. 102-109.

[28] H. Mao, S. Hu et al., "A Load-Driven Task Scheduler with Adaptive DSC for MapReduce", *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference, IEEE*, 2011, pp. 28-33.

[29] Ganglia monitoring system, Available: http://ganglia.sourceforge.net/.

[30] R. Nanduri, N. Maheshwari et al., "Job Aware Scheduling Algorithm for MapReduce Framework", *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference, IEEE, 2011*, pp. 724-729.

[31] JobTracker Architecture, Available: http://hadoop.apache.org/ common /docs/current/mapred_tutorial.html.

[32] Y. Yao et al. "Scheduling heterogeneous MapReduce jobs for efficiency improvement in enterprise clusters." *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE*, 2013, pp. 872-875.

[33] Wang, Zhe, et al. "A New Schedule Strategy for Heterogenous Workload-aware in Hadoop." *ChinaGrid Annual Conference (ChinaGrid), 2013 8th. IEEE*, 2013, pp. 80-85.