Improved Flexible Task Scheduling for Heterogeneous Cluster of Hadoop

Mr. Prashant P. Mali, Mrs.Sunita S. Dhotre

Abstract— Native task scheduling algorithm of Hadoop does not meet the performance requirements of heterogeneous Hadoop clusters. There are three native job schedulers of Hadoop i.e. First in First out (FIFO), Fair scheduler and Capacity schedulers. FIFO has a known drawback of no concept of prioritization of the jobs and no consideration of job size while scheduling the jobs. In Fair Scheduling the allocated resource may go unused if the user has submitted lesser number of jobs. The Capacity scheduler is more beneficial for larger jobs because jobs are prioritized based on their size. To overcome these drawbacks in this paper a flexible task scheduling is proposed which works on runtime workload adjustment strategy and size of job which is much like the adaptive scheduler but with a difference that instead of user defined business goals it relies on the node availability and run time task allocation.

Index Terms— Hadoop, Task Scheduling, Heterogeneous Cluster, Flexible Scheduling, dynamic workload.

1 Introduction

Cloud computing as we all know is a very highly scalable technology with broad availability. Many well-known organizations have developed public cloud computing platforms [3] and working towards enhancing cloud platform which is more dynamic, elastic and efficient. For example, Amazon Cloud [5], Google implement Google App Engine [2].

Hadoop [1] is a framework which provides distributed processing of large data sets across cluster of computers. It can be applied for structured and unstructured data search, data analysis and data mining. It is based on share nothing architecture i.e. nodes does not talk to each other. The file system of Hadoop is such that instead of data moving to processes, the processes move towards data thus the importance of effective job scheduling and task scheduling is primary objective. Job and task are different concepts in Hadoop. When a user submits a transaction, Hadoop will create a job and put it in the queue of jobs waiting for the job scheduler to dispatch it. Then, the job will be divided into a series of tasks, which can be executed in parallel [8], [9]. Scheduler plays a very important role to achieve performance levels in Hadoop system. Task scheduling technology is one way to control running task and allocate computer resource which is beneficial to improve performance of Hadoop platform.

There are five task scheduling strategies widely applicable to Hadoop.

1. LATE (Longest Approximate Time to End): LATE scheduler [6] always speculatively executes the task. And this scheduler execute those task which are farthest from into future. Basically, it works on following two assumptions: a) consistent Task progress on each node b) consistent computation of tasks at all nodes.
2. Dynamic Priority Scheduling Method: This methods supports distribution of capacity dynamically. And it is based on priorities of user concurrently capacity.
3. Delay Scheduling Method: Delay scheduling [7] method is an easiest way to achieve fairness and locality in Scheduling.
4. Deadline Constraint Scheduling Method: This Scheduling method [10] concentrate on the issue of deadlines but the main goal is on increasing system utilization.
5. Data Locality aware task Scheduling Method: This scheduling [11] start with getting request from requesting data node. Later, it schedules the task to node whose input data which is already present on that node. If tasks were unable to get then it will again select task which is nearer to data node. Selected node send the request. Waiting time of assigning task to the node which consist of input data.

As discussed above the original and the improved task scheduling strategies does not meet the performance requirements like stability, scalability, efficiency and load balancing. In this paper we have concentrated on the above mentioned issues in the heterogeneous computing environment. This paper contributes by presenting a flexible task scheduling approach based on run time workload allocations.

Before that, we have to provide a specification Hadoop factors and their related settings which is easiest way to implement job scheduling like fair and capacity scheduling [12] then, performance issues for these scheduler are measured by using simulation. Simulation of two job scheduler gives us response time of executing task either on homogeneous or heterogeneous cluster of Hadoop. It gives idea that which job scheduler is better to specific task.here are five task scheduling strategies widely applicable to Hadoop.

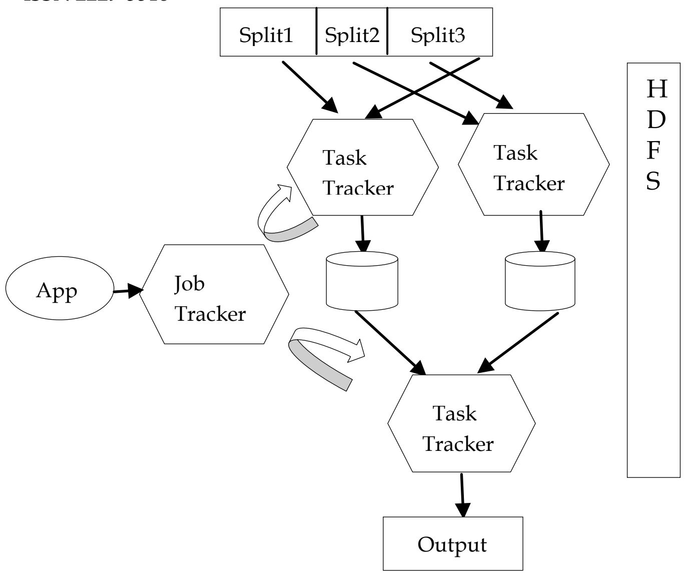


Figure 1. Hadoop Workflow

2 Related Work

Some of task scheduling strategies that have been proposed in recent years:

In 2009, M. Yong et al. [6] introduced task-tracker resource aware scheduler (TRAS). In this algorithm, each task-tracker collects its own resource information and then reports the same to the job-tracker for the next resource scheduling.

Reference [9] proposed a speculative task execution strategy (STES) in 2005. Even If the cluster has already finished most of the tasks, few of the tasks may be left due to insufficient hardware performance and become trailing task. In order to reduce the influence of trailing tasks on the whole job execution time, job-tracker will restart copies of trailing tasks running in parallel; once any one of the task is executed, the whole job is completed.

In 2013, Z. Tang et al. [16] showed Map reduce task scheduling algorithm for deadline constraint (MTSD), which allows user to specify a jobs deadline and makes it finished before the deadline.

A. *System Architecture of Hadoop:*

Hadoop is made up of two core parts: Hadoop Distributed File System (HDFS) and MapReduce [13]. HDFS provides redundant storage of massive amount of data. It provides reliability through replication. It stores files as block. HDFS operates on two types of nodes: Namenode (Master) which stores all metadata and Datanode that stores Files contents in blocks. MapReduce is programming platform for distributing a task across multiple nodes. It is composed of two types of nodes: Jobtracker and Tasktracker. Jobtracker coordinates all jobs using default scheduling strategy of Hadoop. Tasktrackers processes tasks and send reports back to Jobtracker.

B. *Task obtaining and scheduling process of Hadoop:*

To process multiple task at the same time, Hadoop by default use following task scheduling strategy.
1) Tasktrackers count running tasks.
2) It determines whether running task is less than fix task capacity or not.
3) Using flag, It determines whether node is available to obtain new task or not.

Remote Procedure Call (RPC) method is used by default by tasktracker that periodically sends heartbeat to jobtracker. When jobtracker get tasks, it will splits up those tasks and send them to tasktracker to execute.

C. *Problem Analysis*

In heterogeneous environment every nodes load changes dynamically which means different node performs differently which may lead to below discussed performance issues:

1) When the node is high performance computing node and the existing number of running tasks is equal to the allocated fixed slots, the task tracker is not expected to acquire new tasks. But if the node is still underutilized then which is a hunger situation and can run more tasks will lead to waste of resource
2) When the node is a low performance computing node and the number of tasks running is less than the allocated fixed slots then the node can acquire new task but if the node is heavily loaded and cannot host new task which is a saturation situation will lead to overload

Thus an ideal task scheduling and tracking mechanism which dynamically adjusts to the load at run time and the node capability to run tasks and acquire tasks accordingly will improve the clusters performance.

3 Flexible Task Scheduling Algorithm

Whole research is divided into two parts. In first part, we have worked on simulation of fair scheduler and capacity scheduler on single cluster and try to find out which scheduler is better to work on which type of task.

In second part, Our main focus to introduce a novel approach of task scheduling for heterogeneous Hadoop, where in a running cluster the task tracker on its own will make the adjustments to achieve the most optimal state based on runtime load adjustment for the resource available
The high level steps can be defined as below:
1. Task Tracker periodically weighs its load, which is termed as heartbeat interval based on predefined load parameters.
2. Dynamic allocation of max allocated slots for the tasks. This is a deviation from the Hadoop classical approach of one heartbeat and full allocation.

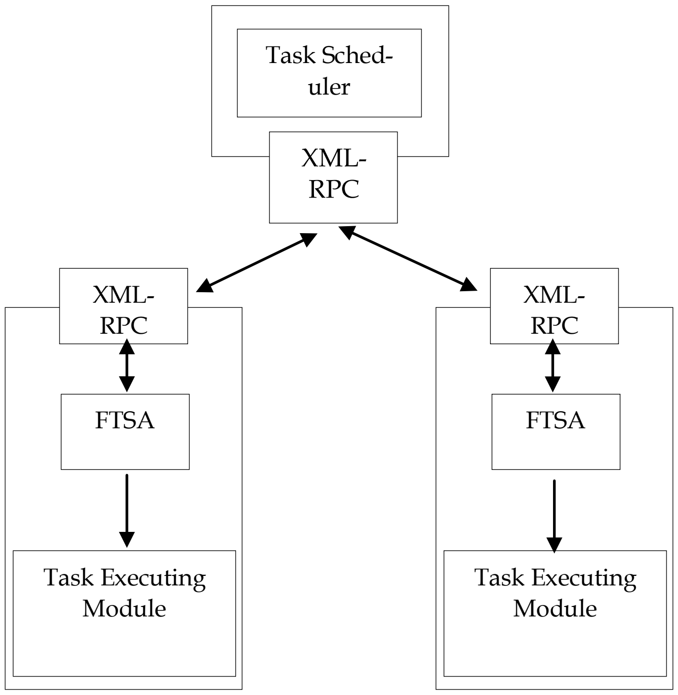


Figure 2. Environment of FTSA

4 Algorithm Description

The load parameters considered here are CPU Usage (CUsage), Memory Usage (MUsage) and Average Length Of Task Queue(AvgTQ) these information can be calculated based on the user-mode time (*user*), low-priority user-mode time(*prior*), system-mode time (*sys*), idle task-mode time (*idle*), hard disk I/O preparing time (*ioprep*), hardware interrupting time(*ihrq*), and software interrupting time (*isrq*) from the File System of OS[15].

$$CUsage = (user+prior+sys)/ total \times 100 \qquad (I)$$

$$total = (user+prior+sys+idle+ioprep+ihrq+isrq) \qquad (II)$$

AvgTQ is a parameters from file /proc/loadavg which reflect the average task queue length of a single core. The numerator is the execution time of non-system idle process and denominator is the total execution time of CPU.

$$MemUsage= (MemTotal-MemFree-Buffers-Cache)/MemTotal \qquad (III)$$

The following is the pseudo code of our algorithm

Input: Current tasktracker load information
Max, Min: the maximum and minimum number of CPU cores of the heterogeneous cluster nodes
Initial value of MaxTasksCapacity
X: heartbeat intervals
Cthre, Lthre: threshold parameter

Output: AskForNewTask, MaxTasksCapacity

```
1 for (i = 0; i<3; i + +) {
/ *In a heartbeat interval, collect three sets of load
Information, deposit them in a corresponding load
array*/
2 CpuQueue[i] = ResourceMonitor.getCpuInfo ();
/ *Get CPU utilization by calling getCpuInfo () */
3 AvgTQ [i] =
ResourceMonitor.getLoadAverage ();
4 MemQueue[i] = ResourceMonitor.getMemInfo ();
/ *Get MEM utilization by calling getMemInfo () */
5 Thread.sleep (WaitTime);
/ *Wait forWaitTime, in order to reach information
Acquisition cycle */
6}
7 for (j = 0; j <3; j + +) {
8 countCpu+ = CpuQueue[j];
9 countLoad+ = AvgTQ [j];
10 countMem+ = MemQueue[j];
11}
12 CpuUsage = countCpu/3;
/ *CPU utilization in this heartbeat interval */
13 AvgTQ = countLoad/3;
/ *The average length of single core task queue in this
Heart beat interval */
14 MemUsage = countMem/3;
/ *MEM utilization in this heartbeat interval */
15 if (CpuUsage<Cthre&&AvgTQ<
Lthre&&MemUsage<Mthre) {
/ *Load judgment, get the value of StatusCount*/
16 StatusCount = StatusCount + 1;
17}
18 HeartBeatCount = HeartBeatCount + 1;
19 AskForNewTask = RunningTasks<
MaxTasksCapacity;
/ *Set the flag of obtaining new task
(AskForNewTask)*/
20 if (HeartBeatCount == X) {
/ *After reaching X heartbeat intervals, make corresponding
decision */
21 if (MaxTasksCapacity<=Max) {
22 MaxTasksCapacity = MaxTasksCapacity + 1;
23 StatusCount = 0;
24 HeartBeatCount = 0;
25}
26 if (MaxTasksCapacity>=Min) {
27 MaxTasksCapacity = MaxTasksCapacity− 1;
28 StatusCount=0;
29 HeartBeatCount = 0;
30}
31}
```

5 Evaluation and Experimental Results

Evaluation section is segregated in three sections namely, Environment Setup, Execution Parameters and Experimental Results.

Environment Setup

We used the VMware virtual machine (VM) platform to set up an experimental pseudo heterogeneous cluster system within a personal computers. Each VM is configured as a Hadoop node, and the experimental heterogeneous platform contains 2 nodes with different hardware configurations. There is one master node with CORE i3 CPUs and 1-GB memory, which are marked as hadoop217 which is set as Job Tracker, and another data nodes with a core i3 CPU and 1-GB memory, which are marked as hadoop218. One node with a CORE i3 CPU and 512-MB memory is set as the task tracker. The spaces of hard disks are all 20 GB. The operating systems installed in all VMs are Red Hat 5 Enterprise, and the bandwidth of the network is 100 MB. Table I shows the detailed configuration parameters of nodes.

Table I
Configuration Parameter of Nodes

Node No.	CPU	Memory
1	Intel Core i3 CPU	1GB
2	Intel Core i3 CPU	1GB
3	Intel Core i3 CPU	512MB

Execution Parameters

As execution parameters [16] we have used following set of configuration first is Hadoop configuration which we have defined as default set by Hadoop as shown in Table II.

Table II
Configuration Parameter of Hadoop

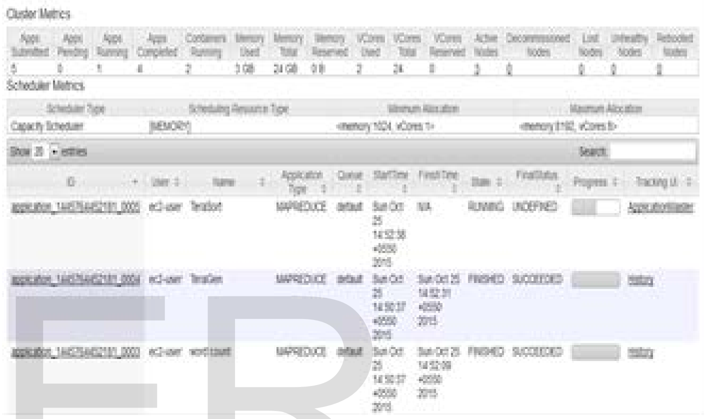
Parameter	Default Value
dfs.block.size	64MB
dfs.replication	3
dfs.Heartbeat.interval	3s

Experimental Results

Scheduling is a main factor for task execution in Hadoop environment. For the evaluation of our approach we have focused on computation intensive job. Our algorithm was tested for different job sizes and multiple runs for the algorithm was done the results displayed were for four runs and the time is computed for each run and finally averaged. The comparative analysis is provided with four other algorithms whose run results were calculated from job scheduling simulator named SLS (Scheduler Load Simulator). As can be analyzed from the below illustration our approach of flexible task scheduling considerably reduces the time for execution of Tasks. In order to guarantee the fairness of experiments, we choose different sizes (3072, 4096, and 5120 MB) of data to sort. The average numbers of tasks generated by each task tracker are 96, 128, and 160, respectively. To further be assured of the results each set of data is tested four times.

We compile the source code based on Hadoop-1.2.0 with Eclipse and create the jar (job) using maven. The all jar deployed on all nodes in both cluster and run word count program to count similar words from particular file, Teragen MR job and Terasort MR job using Hadoop cluster and we try to show all information regarding task like how many mapper and reducer are formed, what is a response time to execute task etc. in GUI using hadoop scheduler for e.g. FIFO, Fair, Capacity Scheduler.

As discussed in previous work of scheduling, we try to execute Capacity scheduler practically. And using this, we implement custom scheduler. Following are the benchmarking result of capacity scheduler for various job with job run Information.



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
5	0	1	4	2	3 GB	24 GB	0 B	2	24	0	3	0	0	0	0

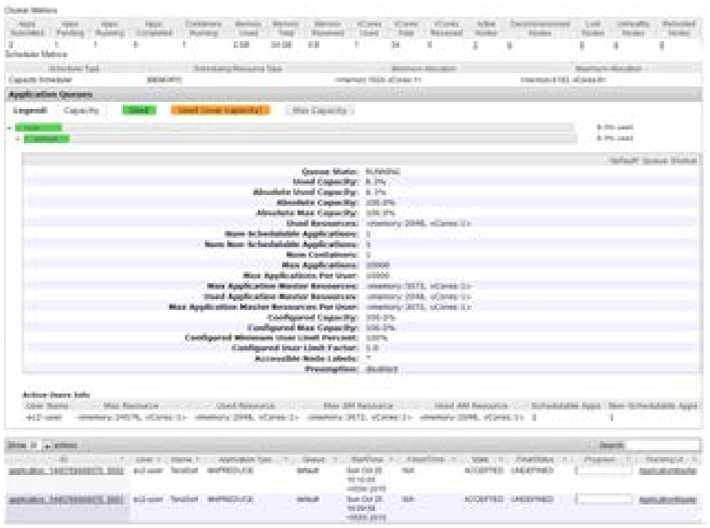
Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1445764452181_0005	ec2-user	TeraSort	MAPREDUCE	default	Sun Oct 25 14:52:38 +0550 2015	N/A	RUNNING	UNDEFINED		ApplicationMaster
application_1445764452181_0004	ec2-user	TeraGen	MAPREDUCE	default	Sun Oct 25 14:50:37 +0550 2015	Sun Oct 25 14:52:31 +0550 2015	FINISHED	SUCCEEDED		History
application_1445764452181_0003	ec2-user	wordcount	MAPREDUCE	default	Sun Oct 25 14:50:37 +0550 2015	Sun Oct 25 14:52:09 +0550 2015	FINISHED	SUCCEEDED		History

Figure 3 Sample Job Run Information



Cluster Metrics

Scheduler Metrics

Application Queues

Legend: Capacity Used Used (over capacity) Max Capacity

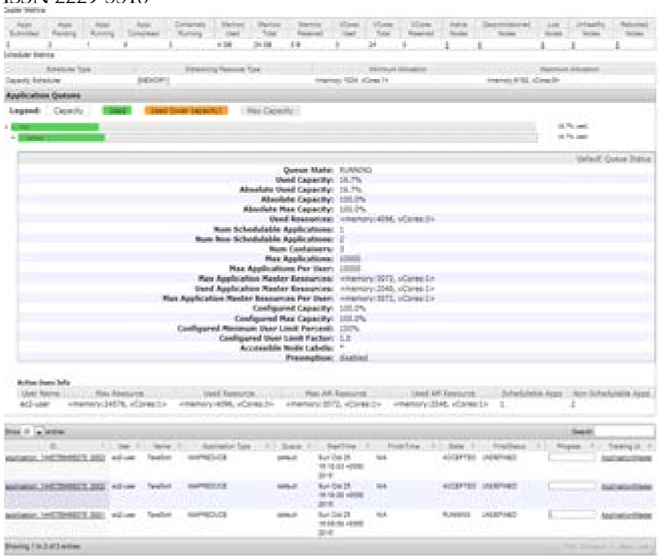
'default' Queue Status

[illegible]

Active Users Info

[illegible]

Figure 4 Sample Capacity Scheduler Job I



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
3	2	1	0	3	4 GB	24 GB	0 B	3	24	0	3	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:8>

Application Queues

Legend: Capacity 🟩 Used 🟧 Used (over capacity) Max Capacity

root 16.7% used
default 16.7% used

'default' Queue Status

Queue State:	RUNNING
Used Capacity:	16.7%
Absolute Used Capacity:	16.7%
Absolute Capacity:	100.0%
Absolute Max Capacity:	100.0%
Used Resources:	<memory:4096, vCores:3>
Num Schedulable Applications:	1
Num Non-Schedulable Applications:	2
Num Containers:	3
Max Applications:	10000
Max Applications Per User:	10000
Max Application Master Resources:	<memory:3072, vCores:1>
Used Application Master Resources:	<memory:2048, vCores:1>
Max Application Master Resources Per User:	<memory:3072, vCores:1>
Configured Capacity:	100.0%
Configured Max Capacity:	100.0%
Configured Minimum User Limit Percent:	100%
Configured User Limit Factor:	1.0
Accessible Node Labels:	*
Preemption:	disabled

Active Users Info

User Name	Max Resource	Used Resource	Max AM Resource	Used AM Resource	Schedulable Apps	Non-Schedulable Apps
ec2-user	<memory:24576, vCores:1>	<memory:4096, vCores:3>	<memory:3072, vCores:1>	<memory:2048, vCores:1>	1	2

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_[illegible]	ec2-user	TeraSort	MAPREDUCE	default	Sun Oct 25 [illegible] 2015	N/A	ACCEPTED	UNDEFINED		ApplicationMaster
application_[illegible]	ec2-user	TeraSort	MAPREDUCE	default	Sun Oct 25 [illegible] 2015	N/A	ACCEPTED	UNDEFINED		ApplicationMaster
application_[illegible]	ec2-user	TeraSort	MAPREDUCE	default	Sun Oct 25 [illegible] 2015	N/A	RUNNING	UNDEFINED		ApplicationMaster

Showing 1 to 3 of 3 entries

Figure 5 Sample Capacity Scheduler Job II

5 Conclusion

As discussed and demonstrated the approach of Flexible Task scheduling considerably reduces the execution time of the tasks and also enhances the chances of task completion without failure. The approach is applicable to both heterogeneous as well as homogeneous Hadoop clusters. Flexibility also induced an additional feature of better load balancing technique thus avoiding overloading of nodes. Though our approach assures task completion but failures of tasks or jobs due to other system parameters cannot be avoided, thus the scope of better fault tolerance is still open and left for future research.

References

[1] Apache. (2012, Aug.). Hadoop, the Apache Software Foundation, ForrestHill, MD, USA. [Online]. Available: http://hadoop.apache.org/

[2] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *IEEE Micro*, vol. 23, no. 2, pp. 22–28, Apr. 2003.

[3] M. Armbrust*et al.*, "A view of cloud computing," *Commun. ACM*, vol 53, no. 4, pp. 50–58, Apr. 2010.

[4] Y. H. Wu, "Research of scheduling policies in cloud computing", M. S. thesis, College Comput, Shanghai Jiao Tong Univ., Shanghai, China, 2011.

[5] Amazon. (2011, May). Amazon Elastic Compute Cloud (Amazon EC2), Amazon Web Services, Inc., Seattle, WA, USA. [Online]. Available: http://aws.amazon.com/ec2/

[6] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, "Improving MapReduce performance in heterogeneous environment", 8th USENIX Symposium based on Operating Systems Design and Implementations.

[7] M. Zaharia, Dhruba Borthakur, J. S. Sarma, K.Elmeleegy, "Delay Scheduling: simple technique to achieving locality and fairness in cluster scheduling"

[8] S. Ghemawat and J. Dean and "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[9] T. White, *Hadoop: The Definitive Guide*. Sebastopol, CA, USA:O'Reilly Media, 2012, pp. 167–188.

[10] Z. Tang, J. Q. Zhou, K. L. Li, and R. X. Li, "MapReduce task scheduling algorithm for deadline constraint," *Cluster Comput.*, vol. 16, no. 4, pp. 651–662, Dec. 2013.

[11] X. Bu, J. Rao and C. Xu, "Locality-Aware Task Scheduling for MapReduce Applications in Virtual Clusters"

[12] Matei Zaharia "Job scheduling with fair and capacity schedulin Hadoop summit 2009

[13] Apache. (2013, Apr.). MapReduce tutorial, The Apache Software Foundation, Forrest Hill, MD, USA. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[15] R.Walker, Examining Load Average. Linux Journal, Dec. 2006. [Online].Available: http://www.linuxjournal.com/article/9001

[14] Apache. (2013, Apr.). MapReduce tutorial, The Apache Software Foundation, Forrest Hill, MD, USA. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[16] Apache, "Hadoop," The Apache Software Foundation, Forrest Hill, MD,USA,Sep.2012.[Online].Available: http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/ClusterSetup.html