International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 3 903 - 910

Improving Map Reduce Performance in Heterogeneous Distributed System using HDFS Environment-A Review

Shraddha Thakkar
M.E. Scholar, Dept. of CE
LDRP-ITR, Gandhinagar Gujarat
thakkarshradhdha@gmail.com

Prof. Sanjay Patel
Associate Professor, Dept. of CE
LDRP-ITR, Gandhinagar Gujarat
sanjaypatel54@gmail.com

Abstract:- Hadoop is a Java-based programming framework which supports for storing and processing big data in a distributed computing environment. It is using HDFS for data storing and using Map Reduce to processing that data. Map Reduce has become an important distributed processing model for large-scale data-intensive applications like data mining and web indexing. Map Reduce is widely used for short jobs requiring low response time. The current Hadoop implementation assumes that computing nodes in a cluster are homogeneous in nature. Unfortunately, both the homogeneity and data locality assumptions are not satisfied in virtualized data centers. Hadoop's scheduler can cause severe performance degradation in heterogeneous environments. We observe that, Longest Approximate Time to End (LATE), which is highly robust to heterogeneity. LATE can improve Hadoop response times by a factor of 2 in clusters.

Keywords: *Hadoop, HDFS, Map-reduce, Scheduling Algorithm, LATE.*

*****

1. Introduction

Cloud computing groups together numbers of commodity hardware servers and other resources to offer their combined capacity on an on-demand, pay-as-you-go basis. The users of a cloud have no idea where the servers are physically located and can start working with their applications. That is the primary advantage [4] of cloud computing which distinguishes it from grid or utility computing.

A main advantage of Map Reduce is that it automatically handles failures, hiding the complexity of fault-tolerance from the programmer. If a node crashes, Map Reduce reruns its tasks on a different machine. Same as, if a node is available but is performing poorly, a condition that we call a straggler, Map Reduce runs a speculative copy of its task (also called a "backup task") on another machine to finish the computation faster. Without this mechanism of speculative execution, a job will be as slow as the misbehaving task. Stragglers can arise for many reasons, including faulty hardware and misconfiguration. Google [8] has noted that speculative execution can improve job response times by 44%.

Here the problem is that how speculative execution can be done to maximize performance. Scheduler starts speculative tasks based on a simple heuristic comparing each task's progress to the average progress. Although it works well in homogeneous environments where stragglers are obvious, which can lead to severe performance degradation when its heterogeneous.

Hadoop's homogeneity assumptions lead to incorrect and often more speculative execution in heterogeneous environments, and can even degrade performance observations. In some cases, as many as 80% of tasks were speculatively executed.one might expect speculative execution to be a simple matter of replicating tasks that are sufficiently slow. In reality, it is a complex issue for several reasons.
(1) Speculative tasks are not free – they compete for certain resources, such as the network, with other running tasks.
(2) Choosing the node to run a speculative task on is as important as choosing the task. (3) In a heterogeneous environment, it may be difficult to distinguish between nodes that are slightly slower than the mean and stragglers. Finally, stragglers should be identified as early as possible to reduce response times.

LATE is based on three principles:

Prioritizing tasks to speculate, selecting fast nodes to run on, and capping speculative tasks to prevent thrashing. We show that LATE can improve the response time of Map Reduce jobs by a factor of 2 in large clusters.[3]

This paper is organized as follows.
Section 1 Introduction
Section 2 Describes Hadoop's introduction and the architectures
Section 3 Scheduling in hadoop
Section 4 Assumptions to be held in hadoop's scheduler and **S**hows how these assumptions break in heterogeneous environments.
Section 5 Discussion
Finally, we conclude in **Section 6.**

2. HADOOP

Hadoop is Popular open source Implementation. Hadoop has been used by many companies (AOL, Amazon,Facebook,Yahoo and New York Times) in

production for large scale data analysis in cloud computing. [10]

Hadoop hides the details of parallel processing, including data distribution to processing nodes, restarting failed subtasks, and consolidation of results after computation. Hadoop framework allows developers to write parallel processing programs which focus on their computation problem, in place of parallelization issues. Hadoop [9] includes 1) **Hadoop Distributed File System (HDFS):** a distributed file system that store large amount of data with high throughput access to data on clusters and 2) **Hadoop Map Reduce:** a software framework for distributed processing of data on cluster. Further classification is as per shown in figure 1

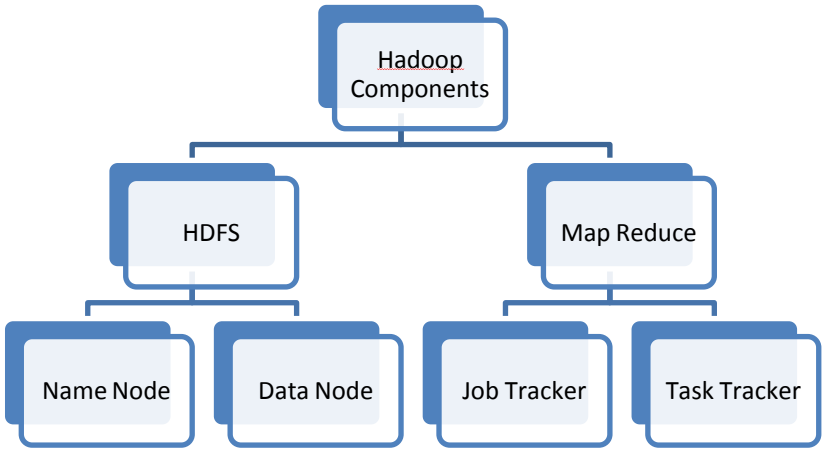

Fig.1.Hadoop Components

2.1 Hadoop Distributed File System

HDFS is the file system component of Hadoop. While the interface to HDFS is patterned after the UNIX file system, Faithfulness to standards was sacrificed in favor of improved performance for the applications at hand

A. Name Node

The Name Node [10] manages the namespace tree and the mapping of file blocks to Data Nodes (the actual location of file data). An HDFS client wanting to read a file first asks the Name Node for the addresses of data blocks which will comprising the file and then reads block contents from the Data Node closest to the client. When writing data, the client requests the Name Node to nominate a suite of three Data Nodes to host the block replicas. The client then writes data to The Data Nodes in a pipeline fashion as per shown in fig 2.

The existing record of the image stored in the local host's home files system which is called a Checkpoint. The Name Node also saves the Modification log of the image known as the journal in the local host's home file system. For improving durability, duplicate copies of the checkpoint and journal can be made at other servers.

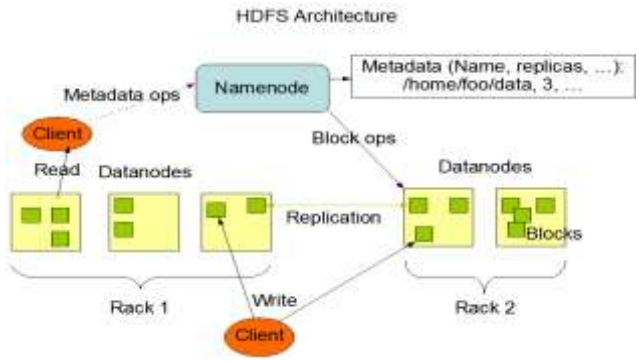

Fig 2.HDFS Architecture

B. Data Nodes

For starting any process Data Node contacts to the Name Node and performs a handshake. The reason of the handshake is to verify the namespace ID and the software version of the Data Node. If either does not match, the Name Node the Data Node automatically shuts down as per shown block. After the handshake the Data Node registers with the Name Node.

2.2 Map-Reduce Architecture

Map Reduce framework (Refer Figure 3) splits the job into various blocks of chunks which the Map the tasks process in parallel. The outputs from the map tasks are sorted by the framework and given to Reduce tasks as input. Both the input and output of the tasks are stored in a file system. The

 ISSN: 2321-8169
903 - 910

framework takes care of scheduling, monitoring the tasks and re executing the failed tasks.

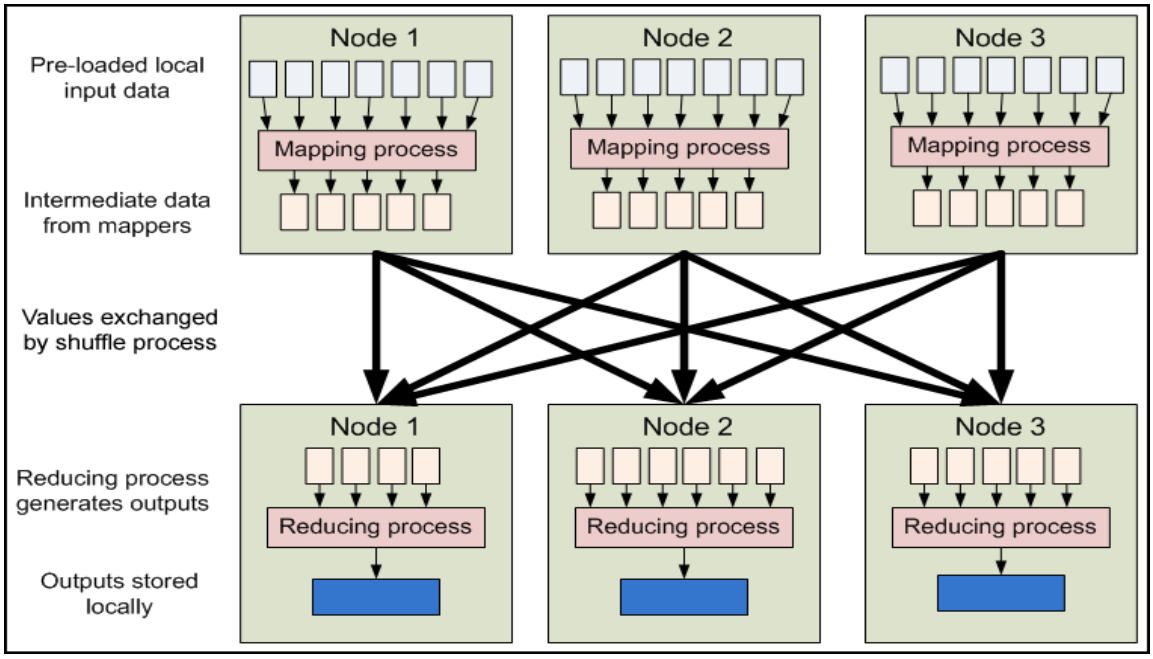

Fig.3 Map-Reduce Architecture

3. Scheduling in Hadoop

During normal operation Data Nodes send heartbeats to the Name Node. The default heartbeat interval is three seconds. If the Name Node does not receive a heartbeat from a Data Node in ten minutes the Name Node considers the Data Node to be out of service. The Name Node schedules creation of new replicas of those blocks on other Data Nodes. These all process done via Job tracker and Task tracker by scheduler which has been discussed as follows and depicts in fig.4

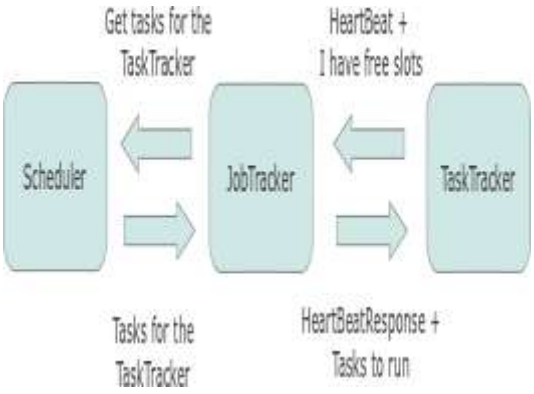

Fig 4.Scheduling in Hadoop

A. Job Tracker

Each cluster has only one Job Tracker which is actually a daemon service for submitting and Tracking Map Reduce jobs in Hadoop. So it is a single point of failure for Map Reduce service and hence if it goes down all running jobs is halted. The slaves are configured to the node location of the Job Tracker and perform tasks as directed by the Job Tracker.

B. Task Tracker

Each slave node has only one Task Tracker (Refer Figure 4) which keeps track of task instances and notifies the Job Tracker about the status.

3.1 Scheduling Algorithms in Hadoop

As per above discussion in Hadoop schedules are run on the basis of scheduling algorithms. It has three existing algorithm like FIFO, FAIR, CAPACITY which is also known as native or core scheduling algorithm. Many researchers have also improved scheduling policies as per the requirement. They are depicted as follow Fig.5

 ISSN: 2321-8169
903 - 910

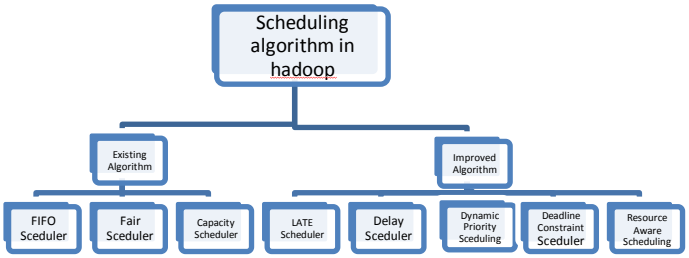

Fig 5.Classification of Hadoop Scheduling Algorithms

3.2 Comparison of different Scheduling Algorithms of Hadoop:

Scheduling Algorigihm	Taxonomy	Idea to Implementation	Advantages	Disadvantages
Fifo	adaptive	schedule jobs based on their priorities in first-come first- out.	1. cost of entire cluster scheduling process is less. 2. simple to implement and efficient.	1. designed only for single type of job. 2. Low performance when run multiple types of jobs. 3. poor response times for short jobs compared to large jobs.
Fair Scheduling	adaptive	do a equal distribution of compute resources among the users/jobs in the system.	1. less complex. 2. works well when both small and large clusters. 3. it can provide fast response times for small jobs mixed with larger jobs.	1. does not consider the job weight of each node.
Capicity	adaptive	Maximization the resource utilization and throughput in multi-tenant cluster environment.	1. ensure guaranteed access with the potential to reuse unused capacity and prioritize jobs within queues over large cluster .	1. The most complex among three schedulers.
Dynamic priority Scheduler	adaptive	designed for data intensive workloads and tries to maintain data locality during job execution	1. is a fast and flexible scheduler. 2. improves response time for multi-user Hadoop environments.	
LATE	adaptive	Fault Tolerance	1. robustness to node heterogeneity. 2. address the problem of how to robustly perform speculative execution to maximize performance.	1. only takes action on appropriate slow tasks. 2. However it does not compute the remaining time for tasks correctly, and cannot find real slow tasks in the end. 3. poor performance due to the static manner in computing

				the progress of the tasks.
Deadline Constraint Scheduler	adaptive	To improve Map Reduce in terms of saving the time of the execution and the system's resources.	1. decreases the execution time of map reduce job. 2. improve the overall Map Reduce performance in the heterogeneous environments.	1. do not consider the data locality management for launching backup tasks.
Delay Scheduling	adaptive	To address the conflict between locality and fairness.	1. Simplicity of Scheduling	1. No particular
Resource-aware Scheduler	adaptive	To optimizations for jobs using the same dataset	1. optimizations for jobs using the same dataset.	

3.3 Speculative Execution of Hadoop

In scheduling policy if any node has an empty task slot, Hadoop chooses a task for it as following.(1) Any failed tasks are given highest priority. This is done to detect when a task fails repeatedly due to a bug and stop the job. (2) Non-running tasks are considered. For maps, tasks with data local to the node are chosen first.

Finally, Hadoop [3] executes task speculatively. To choose speculative tasks, Hadoop considers progress score between 0 and 1. For a map, the progress score is the fraction of input data read. For a reduce task, the execution is divided into three phases, each of which accounts for 1/3 of the score:

Sr No.	Phase	Function
1	Copy	Task fetches Map outputs
2	Sort	Map outputs are sorted by key
3	Reduce	User-defined function is applied to the list of map outputs with each key.

Table 1.Phases of speculative execution

In each phase, the score is the instance of data processed.

For example [5], a task halfway through the copy phase has a progress score of $\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$, while a task halfway through the reduce phase scores $\frac{1}{3} + \frac{1}{3} + (\frac{1}{2} \cdot \frac{1}{3}) = \frac{5}{6}$

Hadoop considers at the average progress score of each phase to define a threshold for speculative execution: When a task's progress score is less than the average for its category minus 0.2, and the task has run for at least one minute, it is marked as a straggler. All tasks beyond the threshold are considered "equally slow," and ties between them are broken by data locality. The scheduler guarantees that at most one speculative copy of each task is running at a time. Hadoop works reasonably well in homogenous environments because tasks tend to start and finish in "waves" at roughly the same times and speculation only starts when the last wave is running. Hadoop uses a FIFO discipline where the earliest submitted job is asked for a task to run, then the second etc.

4. Assumptions to be held in Hadoop's Scheduler

Hadoop's [3] scheduler makes several implicit assumptions:

1. Nodes can perform work at roughly the same rate.
2. Tasks progress at a constant rate throughout time.
3. There is no cost to launching a speculative task on a node that would otherwise have an idle slot.
4. A task's progress score is representative of fraction of its total work that it has done. Specifically, in a reduce task, the copy, sort and reduce phases each take about 1/3 of the total time.
5. Tasks tend to finish in waves, so a task with a low progress score is likely a straggler.
6. Tasks in the same category (map or reduce) require roughly the same amount of work.

Assumptions 1 and 2 break down in a virtualized data center due to heterogeneity.

Assumptions 3, 4 and 5 can break down in a homogeneous and may cause Hadoop to perform poorly.

In fact, Yahoo disables speculative execution on some jobs because it degrades performance, and monitors faulty

machines through other means. Facebook disables speculation for reduce tasks [17].

Assumption 6 is inherent in the Map Reduce paradigm. Tasks in Map Reduce should be small, otherwise a single large task will slow down the entire job.

When Assumptions break down:

4.1 Heterogeneity

The first two assumptions are about homogeneous. Hadoop assumes that any detectably slow node is faulty. In a non-virtualized data center, there may be multiple generations of hardware. In a virtualized data center where multiple virtual machines run on each physical host, such as Amazon EC2, co-location of VMs may cause heterogeneity [14].

For resolving degradation of Map reduce performance in heterogeneous environments and found solutions to improve its performance. Each approach improves one of the Map reduce features in a heterogeneous cluster. The algorithms that represent improved feature are divides into two categories as follows:

4.1.1. Data Locality Algorithms.

4.1.2. Fault Tolerance Algorithms.

4.1.1. Data Locality Algorithms

Data locality is a decision parameter for the Map reduces performance. Here, the algorithm that has been developed to improve data locality management in a heterogeneous Hadoop cluster has describes.

A. Data Placement in Heterogeneous Hadoop Clusters

Data placement strategy is efficient for a homogeneous environment having 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 nodefinished 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 Map Reduce performance in heterogeneous environments is to reduce the amount of data moved between slow and fast nodes in a heterogeneous clustering

J. Xie et al. [2] 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.

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 com-putting capacities. Second, the data redistribution algorithm which reorganized the file fragments to solve the data skew problem.

B. Initial Data Placement

The initial data placement [4] algorithm starts 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 handled 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.

C. Data Redistribution

Input file fragments distributed by the initial data placement algorithm [6] 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
(3) New data computing nodes are added into an existing cluster. To address this dynamic data load-balancing problem, we implemented a data redistribution algorithm to reorganize file fragments based on computing ratios.

D. Data Locality Aware Task Scheduling Method for Heterogeneous Environments

In this research, the work is built upon the method to improve data locality of Map reduce 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.

X. Zhang et al. **[7]** 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 goal 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.

4.1.2. Fault Tolerance Algorithms

A main advantage of Map reduce is that it automatically manages 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 specu-lative 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.[6]

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

4.2.2.1 LATE: Longest Approximate Time to End Algorithm

Here if the 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 observes 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 re-duce 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 Progress Score/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. [6]

It assumes that tasks make progress at a roughly constant rate. There are cases where
this heuristic can fail, which we describe later, but it is effective in typical Hadoop jobs. To really get the best chance of beating the original task with the speculative task, we should also only launch speculative tasks on fast nodes --not stragglers. We do this through a simple heuristic -don't launch speculative tasks on nodes that are below some threshold, Slow Node Threshold, of total work performed (sum of progress scores for all succeeded and in progress
tasks on the node). This heuristic leads to better performance than assigning a speculative task to the 1st available node. Another option would be to allow more than one speculative copy of each task, but this wastes resources needlessly.

Finally, to handle the fact that speculative tasks cost resources, we augment the algorithm with two heuristics:

- A cap on the number of speculative tasks that can be running at once, which we denote Speculative Cap.
- A Slow Task Threshold that a task's progress rate is compared with to

Determine whether it is \slow enough" to be speculated upon. This prevents needless speculation when only fast tasks are running.[3]

In summary, the LATE algorithm [2] works as follows:

If a node asks for a new task and there are fewer than Speculative Cap Speculative tasks running then following has been takes place:

- Ignore the request if the node's total progress is below Slow Node Threshold.
- Rank currently running tasks that are not currently being speculated by estimated time left.
- Launch a copy of the highest-ranked task with progress rate below
  Slow Task Threshold.[2]

As Hadoop's scheduler [5] , we also wait until a task has run for 1 minute before evaluating it for speculation. In practice, we have found that a good choice for the three parameters
to LATE are to set the Speculative Cap to 10% of available task slots and set the Slow Node Threshold and Slow Task Threshold to the 25th percentile of node progress and task progress rates respectively.

- **Advantages of LATE**

The LATE algorithm [5] has several advantages.
It is robust to node heterogeneity, because it will re-launch only the slowest tasks, and only a small number of tasks.

LATE prioritizes among the slow tasks based on how much they affect job response time.

LATE also caps the number of speculative tasks to limit contention for shared resources. Comparatively, Hadoop's native scheduler has a fixed threshold, beyond which all tasks that are slow enough and have an equal chance of being launched. This fixed threshold can cause excessively many tasks to be speculated upon.

LATE takes into account node heterogeneity when deciding where to run speculative tasks. Whereas, Hadoop's core scheduler assumes that any node which finishes a task and asks for a new one is likely to be a fast node, i.e. that slow nodes will never finish their original tasks and never be member for running speculative tasks.

At last but not least, by focusing on estimated time left instead of progress rate, LATE speculatively executes only those many tasks that will improve job response time, rather than any slow tasks.

5. Discussion

1. Decreases decisions Time: Instead of waiting to base decisions on measurements of mean and variance.
2. **Use estimated time left:** prioritize among tasks to speculate instead of considering Progress Rate.
3. Nodes are Heterogeneous: Ignore assigning speculative tasks to slow nodes.
4. **Resources are Limited:** Caps should be used to protect against overloading the system.

6. Conclusion

Efficiency to re-design Hadoop scheduler resource aware is most critical research problem This paper summarizes advantages and disadvantages as well as working comparison of various Scheduling policies of Hadoop Schedulers developed by different communities. Scheduler considers the resources like CPU, Memory, Job deadlines and IO etc. Schedulers discussed in this paper pointing out one or more problem(s) in scheduling in Hadoop. All the schedulers discussed above assume homogeneous Hadoop clusters. Future work will consider scheduling in Hadoop in Heterogeneous Clusters.

7. Future Work

We will upgrade a Scheduling algorithm, Longest Approximate Time to End (LATE) that is highly robust to heterogeneity.

References

[1] A Comprehensive View of Hadoop Map Reduce Scheduling Algorithms- Seyed Reza Pakize .Department of Computer, Islamic Azad University, Yazd Branch, Yazd, Iran--International Journal of Computer Networks and Communications Security-- VOL. 2, NO. 9, SEPTEMBER 2014, 308–317

[2] Improving Map Reduce Performance through Data Placement in Heterogeneous Hadoop Clusters- Jiong Xie, Shu Yin, Xiaojun Ruan, Zhiyang Ding, Yun Tian, James Majors, Adam Manzanares, and Xiao Qin -Department of Computer Science and Software Engineering Auburn University, Auburn, AL 36849-5347

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

[4] B.Thirumala Rao, N.V.Sridevi, V.Krishna Reddy, L.S.S.Reddy "Performance Issues of Heterogeneous Hadoop Clusters in Cloud Computing" Global Journal of Computer Science and Technology Volume 11 Issue 8 Version 1.0 May 2011

[5] Andy Konwinski--Improving MapReduce Performance in Heterogeneous Environments Technical Report No. UCB/EECS-2009-183

[6] An Empirical Analysis of Scheduling techniques for Real-time cloud based data processing-linh T.X. Phan Zhuoyao zhang, Qi Zheng Boon Thau Loo Unniversity of pennsylvania

[7] S. Khalil, S.A. Salem, S. Nassar and E.M. Saad, " Mapreduce Performance in Heterogeneous Environments: A Review ", International Journal of Scientific & Engineering Research, Vol. 4, NO. 4, April - 2013, pp. 410-416.

[8] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In Communications of the ACM, 51 (1): 107-113, 2008.

[9] "Hadoop home page." http://hadoop.apache.org/.

[10] Critical study of hadoop Implementation and performance issue-Conference: Research In IT: Exploring the Horizon" National Conference, At Patkar College

[11] Hadoop's FIFO Scheduler http://hadoop.apache.org/common/docs/r0.20.2/fifo_scheduler.html

[12] Hadoop's Fair Scheduler http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html

[13] Hadoop's Capasity Scheduler http://hadoop.apache.org/common/docs/r0.20.2/capasity_scheduler.html

[14] Dynamic Proportional share Scheduling in Hadoop Thomas sandholm and Kevin Springer Berlin Heidelberg Volume 6253, 2010, pp 110-131

[15] Hadoop Preemptive Deadline Constraint Scheduler-IEEE Explorer Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2014 International Conference on 13-15 OCT 2014

[16] Coupling task progress for Map Reduce resource-aware scheduling-IEEE Explorer-INFOCOM, 2013 Proceedings IEEE

[17] Personal communication with the Yahoo! Hadoop team and with Joydeep Sen Sarma from Facebook.

[18] "Hadoop home page." http://hadoop.apache.org/.

[19] White, T., 2012, " Hadoop: The Definitive Guide ", ed. Third, Tokyo: Yahoo press.