International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 7, Issue 7, July 2017)

Phase Aware Job Scheduling for MapReduce in Hadoop

Snehal S. Kapade[1], Prof. J. V. Shinde[2], Prof. N. K. Zalte[3]

[1]PG Student, [2,3]Professor, Affiliated to Kalyani charitable trust Late G.N. Sapkal College of Engineering, Savitribai Phule, Pune University, Nasik, Maharashtra, India

***Abstract*— MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. MapReduce is a model for the data-intensive computation. However, despite recent efforts towards designing efficient scheduler to perform MapReduce, the available solutions focus on scheduling at the task-level offers suboptimal performance in executing jobs. This is because tasks can have highly varying resource requirements during their lifetime, which makes it difficult for schedulers to effectively utilize the available resources to reduce job execution time. PRISM-a fine grained phase and resource aware scheduler was introduced which mainly focuses on designing task level scheduler. The phase based resource aware scheduler offers high resource utilization and provides improvement in job running time. In my proposed system, I have used the virtual memory to overcome the disadvantage of PRISM and using the pausing phase. In this the resource will able to use virtual resource. Instead of going into paused phase it is possible to use resource virtually till the resource is available. This will surely help the job running significantly reducing the delay.**

***Keywords*—MapReduce, Scheduling, resource allocation, execution time.**

I. Introduction

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. Large-scale MapReduce clusters that routinely process petabytes of unstructured and semi-structured data represent a new entity in the changing landscape of clouds. Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically, both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

A key challenge is to increase the utilization of these MapReduce clusters. Our goal is to automate the design of a job schedule that minimizes the completion time of such a set of MapReduce jobs. The main objective is to smooth resources requirements by shifting slack jobs beyond periods of peak requirements. Although MapReduce has been widely adopted in a number of data centers, more improvements are still needed to meet the huge demands of big data computing. In the current MapReduce framework, each job consists of three dependent phases: map, shuffle, and reduce. The map and reduce phases typically deal with a large amount of data computations, while the shuffle phase handles the data transfer among different MapReduce workers. In terms of the resource demand, the map and reduce phases are CPU-intensive, while the shuffle phase is I/O-intensive. PRISM, a fine-grained resource-aware

MapReduce scheduler divides the tasks into phases, where each of the phases has a constant resource usage profile, and performs scheduling at the phase level. The importance of the Phase-level scheduling is demonstrated by showing the resource usage variability within the lifetime of a task using a wide range of MapReduce jobs. The phase-level scheduling algorithm improves execution time and resource utilization without introducing any stragglers with the help of Virtual Space. The main objective of the phase-level scheduling along with the Virtual Space is to reduce the delay that occurs during the "pause" stage of the node manager and also to increase the job running time and the resource utilization when compared with the task-level scheduling. Phase based scheduling algorithm. In order to overcome the issue of delay during the job execution due to the pause: stage, the virtual space is allocated to the node manager. Thus, the average job running time will be improved when compared with the phase based scheduling algorithm without virtual space.

II. Related Work

It shows existing system and the work done in Mapreduce. The original MapReduce work is to schedule the task in different levels.

A. Existing Work

In a MapReduce technique, it is a collection of jobs and can be scheduled concurrently on multiple machines, resulting in reduction in job running time. Many companies such as Google, Facebook, and Yahoo, they refer MapReduce to process large volume of data. But they refer at task level to perform these data. At the task level, performance and effectiveness become critical to day-to-day life. Initially the task level performs two phases one is mapper phase and other is reducer phase. In mapper phase, it takes data blocks hadoop distributed file system and it maps, merge the data and stored in the multiple files. Then the second, reducer phase will fetch data from mapper output and shuffle, sort the data in a serialized manner.

B. Previous Papers

In "Starfish: A self-tuning system for big data analytics" [3] can provide accurate resource information that can be used by the scheduler so that it can take effective scheduling decisions and reduce the job execution time. By using such job profiler integrated within the scheduler, the performance of the scheduler is effective than existing task level scheduler. It collects the past executed jobs profile information at a fine granularity for job estimation and automatic optimization. However, collecting detailed job profile information with a large set of metrics generates an extra overhead, especially for CPU-intensive applications. As a result, Starfish overestimate the execution time of a Hadoop job

In "Resource-aware adaptive scheduling for MapReduce clusters," [5] the slot-based resource allocation scheme, the physical resources on each machine are captured by the number of identical slots that can be assigned to tasks. The problem with that slot-based resource allocation is that the run-time resource consumption varies from task to task and from job to job.

In "Mapreduce: Simplified data processing on large clusters," [6] new abstraction that allows us to express the simple computations where given which tries to perform but hides the messy details of parallelization, fault tolerance, data distribution and load balancing in a library was introduced. Still the operation is applicable at task level only that can be further divided into phases and the further parallelization in phases can cause the overhead problem. Than the PIRSM was introduced which the importance of phase-level. In a phase-level, phase-level scheduling algorithm which improves execution parallelism and performance of task. Phase and Resource Information Aware Scheduler for MapReduce at the phase-level refers at the phase-level to improve resource utilization and performance.

It overcomes the problem of varying resources at the task level and divided them into phase level. Finally, even though the flexibility of phase-based scheduling should allow the scheduler to improve both resource utilization and job performance over existing MapReduce schedulers, realizing such a potential is still a challenging problem. This is because pausing the task execution at run-time may delay the completion of the current and subsequent tasks, which may increase the job completion time Thus, the scheduler must avoid introducing stragglers when switching bet Author phases. In the following sections, Author will describe how PRISM overcomes this challenge.

III. Proposed System

This is our complete experimental setup; Job progress monitor is the web GUI where the statistics of all the nodes are displays. It contains one container at one slave and all resources are virtually allocated in that container. This container will hold all the resources for each and every phase. When map phase starts it will hold resources for map phase than for shuffle, reduce etc. It will then discuss with master and allow more if requires for the task to execute. When you start something then map phase starts first follows by reduce and all phases. Task tracker actually performs all the phases that are slaves. We can have multiple task tracker. In our case we have 3 slave nodes so 3 task trackers. Job request is sent by the client and here master is acting as the client.

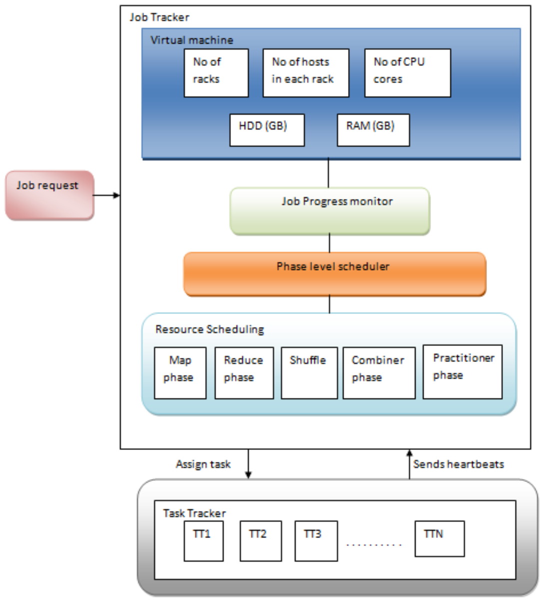


Fig. 1. Proposed Architecture

The existing system is the PRISM that uses the phase level scheduling. The problem with the phase-level scheduling algorithm is, when the task has completed the execution of phases, and if no sufficient resources are available then the subsequent phases may not be scheduled immediately. The phase execution will be ”paused” when no sufficient resources are available. The” paused” execution will avoid contention of resource by delaying task completion [8]. But the execution of the job will be paused and the request for the additional space from the phase-base scheduler will be done by the node manager and thus the ”pause” stage will always result in long job execution time. In order to overcome this problem of long job execution time, the virtual space will be allocated to node manager. This additional space of virtual mechanism apart from the node manager will be utilized during the job execution of the paused stage of the job execution. When the Node Manager retrieves the original space from phase base scheduler, the node manager will shift the loaded content in the virtual space to the original space in node manager. The figure shows the use of virtual memory in pause duration of execution state that helps avoiding execution of long job.

IV. System Analysis

This section describes in detail the design and working of Phase-base scheduling algorithm with virtual space.

A. Algorithm Description

Following are the steps for phase base scheduling algorithm

a. Scheduler replies to the heart beat message with scheduling request every time when a task has to be scheduled,
b. The task execution is launched by the node manager. When there is no space available to execute the job then the execution of the job will be paused.
c. The node manager will than load the content in paused state due to the insufficient resources into the virtual space.
d. When the actual memory from phase-base scheduler is allocated, the node manager will deactivate the virtual space and load the content into original space.
e. By finishing executing a particular phase, the task asks permission to start the next phase from the node manager
f. Then the node manager forwards the request for the permission to scheduler through heart beat message.
g. When the phase-level resource requirement and the current progress information is known, the scheduler decides whether to start the new task or paused task to begin next phase
h. This information is given to the node manager and when the execution of all tasks is completed, the task status is received by node manager and forwards it to node manager.

In the scheduling algorithm node manager is responsible for reporting the resource availability on the node and it must select the next phase to should be scheduled on the node. There are J number of jobs, specifically $j \in J$ and consists of two tasks called Map M and Reduce R. Resource $r(t) \in \{M,R\}$. The utility function is

$$U(i, n) = U\,fairness(i, n) + \alpha \,.\, Uper\,f(i, n)$$

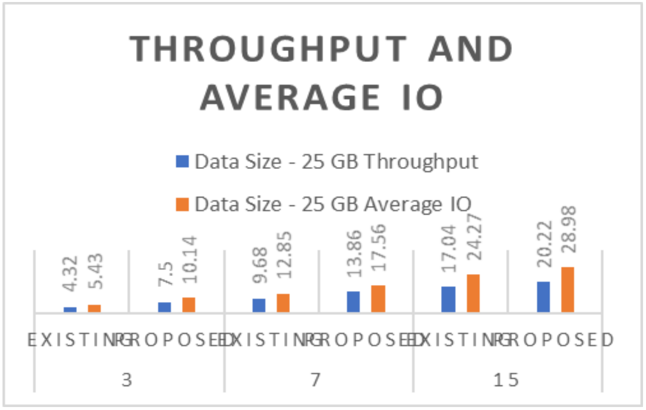
The fairness of each phase is calculated as:

$$U\,fairness(i,n) = U\,bfairness(i,n) - Uafairness(i,n)$$

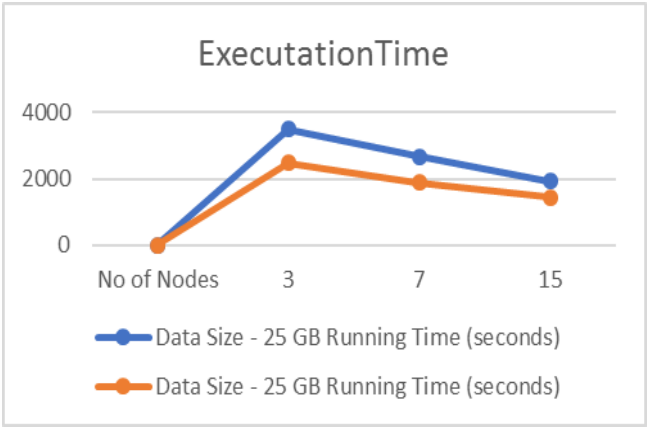
Ufairness and Uperf represent the utilities for improving fairness and job performance, respectively, and a is an adjustable weight factor. If we set a close to zero, then the algorithm would greedily schedule phases according to the improvement in fairness.

V. Experiments

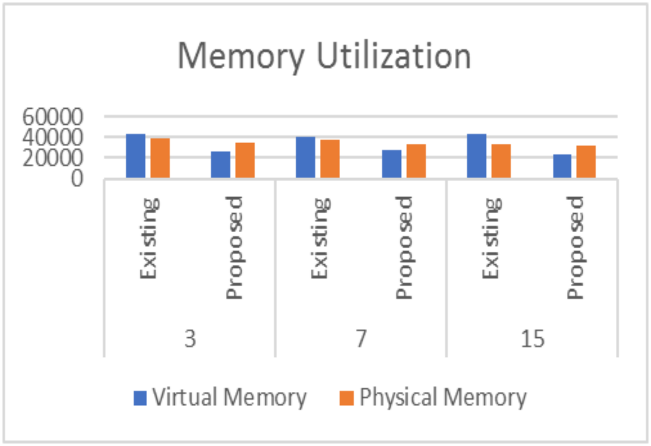
The system has been implemented in Hadoop architecture in a cluster of 15 compute nodes. To evaluate the benefit by using virtual memory, is necessary to compare proposed system to the PRISM. In the experiments, I have compared both system with 3,7 and 15 nodes Hadoop with different size of text input. We started our experiment with 3 nodes Hadoop in pseudo distributed mode with 5GB text input. The CPU time and Memory utilization on 3 nodes cluster the performance of the scheduler started degrading with the increase in size of the text file. But in the distributed cluster the result of job completion time is performance better for those 3 nodes in pseudo distributed nodes. To provide accurate evaluate again we tested the performance on fully distributed mode on 7 and 15 nodes Hadoop.



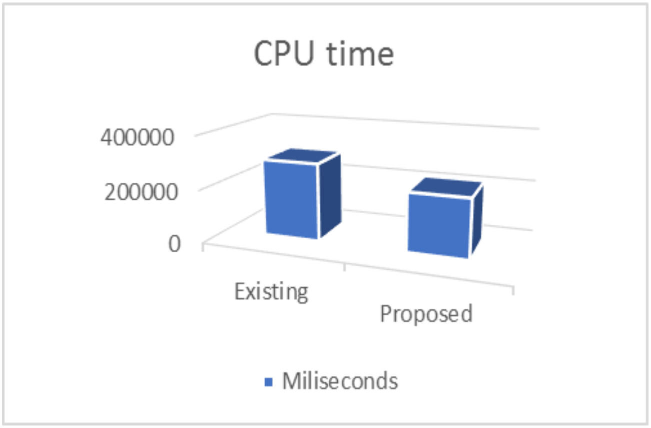
(a) Throughput and Average IO



(b) Execution time



(c) Memory utilization



(d) CPU time

VI. Conclusion

MapReduce is programming model for cluster to perform a data-intensive computing. In this work we mainly demonstrate that, the scheduler allocate the resources dynamically at the run time and the pause duration occurs due to insufficient resources can be overcome. The use of virtual space from all the resources results in better outcomes compare to PRISM. PRISM demonstrates that, how run-time resources can be used and also how it varies over the long-life time. Thus, our system improves job execution algorithm and performance of resources without introducing stragglers.

REFERENCES

[1] Qi Zhang, Mohamed Faten Zhani, Yuke Yang, Raouf Boutabai, and Bernard Wong, ”PRISM: Fine-Grained resource Aware Scheduling for MapReduce,” in IEEE Transactions on Cloud Computing, Vol. 3, no.

[2] Clustering on Big Data Using Hadoop MapReduce,International Conference on Computational Intelligence and Communication Networks,2015

[3] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, and S. Babu, Starfish: A self-tuning system for big data analytics, in Proc. Conf. Innovative Data Syst. Res., 2011, pp. 261272.

[4] Phase and resource information- aware scheduling for MapReduce jobs scheduling, International journal of latest trends in Science and technology.

[5] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder, J. Torres, and E. Ayguade, Resource-aware adaptive scheduling for MapReduce clusters, in Proc. ACM/IFIP/USENIX Int. Conf.Middleware, 2011, pp. 187207

[6] J. Dean and S. Ghemawat, Mapreduce: Simpli_ed data processing on large clusters, Commun. ACM, vol. 51, no. 1, pp. 107113, 2008.

[7] A. Verma, L. Cherkasova, and R. H. Campbell, Resource provisioning framework for mapreduce jobs with performance goals, in ACM/USENIX Middleware 2011, pp. 165186.

[8] Hadoop Distributed File System [Online]. Available: hadoop.apache.org/docs/hdfs/current/, 2015

[9] R. Boutaba, L. Cheng, and Q. Zhang, On cloud computational models and the heterogeneity challenge, J. Internet Serv. Appl.,vol. 3, no. 1, pp. 110, 2012.

[10] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce online," in Proc. USENIX Symp. Netw. Syst. Des. Implementation, 2010, p.21.

[11] Savitri D.H, Narayana H.M, "PRISM: Allocation of Resources in Phase-level using MapReduce in Hadoop," in International Journal of Research in Science and Engineering Vol. 1, Special Issue: 2.

[12] The Next Generation of Apache Hadoop MapReduce[Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html, 2015.