An Adaptive Scheduling Technique for Efficient Task Assignments in Hadoop

[1] Mr. Jitesh J. Patil, [2] Prof. Rahul Jiwane
[1][2]Department of Computer Engineering Pillai HOC College of Engineering
[1] jiteshpatil14me@gmail.com, [2] rahul_jiwane@yahoo.com

Abstract: **The Map Reduce framework and its open source implementation in Hadoop is existing as a standard for Bigdata related processing in industry and academies. When a bunch of jobs are simultaneously submitted together to a Map Reduce cluster, bunch of jobs will compete for available resources by this the overall system performance may go down, this is because in Map Reduce cluster different kinds of workload is shared among multiple users. Existing scheduling algorithms which are supported by Hadoop always cannot guarantee good average response time with different workloads. Therefore it is a challenging ability to design an effective scheduler which can work with shared Map Reduce cluster. To solve this problem we proposed a new hadoop scheduler which works on the different workload patterns and reduces overall job response time by using the knowledge of workload patterns.. The scheduler will reduce the average job response time that are compared with existing Fair and FIFO Scheduler.**

***Keywords*—Map Reduce, Hadoop, Scheduling, Timsort**

I. INTRODUCTION

Map Reduce has emerged as an important paradigm in many large-scale data processing applications in modern data centers. The Map Reduce runtime consists of a single master process and a large number of slave processes. When a Map Reduce job is submitted to the runtime, it is split into a large number of Map and Reduce tasks, which are executed by the slave nodes. The runtime is responsible for dispatching tasks to slave nodes and ensuring their completion.

A Map Reduce 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. Typically the compute nodes and the storage nodes are the same, that is, the Map Reduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster. The Map Reduce framework consists of a single master Job Tracker and one slave Task Tracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client.

However, Hadoop is still a new framework that needs to be improved in some aspects. Task Scheduling technology, one of the key technologies of Hadoop platform, mainly controls the order of task running and the allocation of computing resources, which is directly related with overall performance of the Hadoop platform and system resource utilization.

Default scheduling algorithm that Hadoop platform provides is FIFO. The advantages of FIFO include simple idea and easy to be executed, light workload of job server,.etc. The disadvantages of FIFO lie in ignoring the different needs by different operations. For example, if a job analyzing massive data occupies computing resources for a long time, then subsequent interactive operations may not be processed timely. Therefore, this situation may lead to long response time and affect the user's experience.

Fair scheduler is proposed to improve the job response time by assigning all jobs with a equal share of resources. But there arises problem with the Fair Scheduler i.e., Fair scheduler makes scheduling decision without considering different types workload pattern by users. To address the above issues, we are building up a novel Hadoop scheduler for tasks assignment in Hadoop , which aims to enhance the average response time of Hadoop systems. The scheduler will make use of the history information available of the current user to make the scheduling decisions. To improve the performance of job completion we are using Timsort algorithm during sort and shuffle phase(by default quick sort is used). In this paper further will see: Section II discusses about existing core scheduling techniques. Section III discusses about proposed work, implementation details, introductory definitions and documentations.

II. LITERATURE SURVEY

A. FIFO scheduler

FIFO is the default Hadoop scheduler. The principle goal of FIFO scheduler to schedule jobs based on their priorities in first-come first-out of first serve order. FIFO stands for first in first out which in it Job Tracker pulls oldest job first from job queue and it doesn't consider about priority or size of the job. FIFO scheduler have numerous constraints for example: poor response times for short jobs compared to large jobs, Low performance when run different sorts of jobs and it gives good result just for single kind of job. To address these issues scheduling algorithms such as Fair and Capacity was introduced.

B. Fair Scheduler

Fair scheduling is a technique of assigning resources to jobs such that all jobs get, on average, an equivalent offer of resources over time. In the event that there is a single job running, the job uses the entire cluster. When other jobs are submitted, free task slots are alloted to the new jobs, so that every job gets generally the same measure of CPU time. It gives short jobs a chance to finish inside a sensible time while not starving long jobs. The objective of Fair scheduling algorithm is to do an equivalent conveyance of compute resources among the users/jobs in the system. The scheduler really composes jobs by resource pool, and shares resources reasonably between these pools. As a matter of course, there is a different pool for each user. The Fair Scheduler can restrict the quantity of simultaneous running jobs per user and per pool. Likewise, it can restrict the quantity of simultaneous running tasks per pool. The traditional algorithms have high data transfer and the execution time of jobs. Tao et al. presented an enhanced FAIR scheduling algorithm, which considers job characteristics and data locality, which diminishes both data transfer and the execution time of jobs. Consequently, Fair scheduling can cover some constraint of FIFO for example, it can function well in both small and large clusters and less mind boggling. Fair scheduling algorithm does not consider the job weight of each node, which is an important limitation of it.

C. Capacity scheduler

The configuration of capacity scheduling algorithm is fundamentally the same as fair scheduling. In any case, use of queues instead of pool. Each queue is allocated to an organization and resources are isolated among these queues. Scilicet, Capacity scheduling algorithm places jobs into various queues in accordance with the conditions, and designates certain system limit for each queue. On the off chance that a queue has heavy load, it looks for unallocated resources, then makes redundant resources allocated equally to each job . For augmenting resource utilization, it permits re-allocation of resources of free queue to queues using their full limit. When jobs arrive in that queue, running tasks are finished and resources are offered back to original queue. It additionally permits priority based scheduling of jobs in an organization queue . The capacity scheduler permits users or organization to simulate a separate Map Reduce cluster with FIFO scheduling for each user or organization . By and large,, capacity scheduling algorithm addresses the FIFO’s disadvantage such as the low utilization rate of resources. The most complex among three schedulers is a vital issue in capacity algorithm. The user needs to know system data and make queue set and queue select group for the job. In a large system, it will be one of enormous bottleneck of enhancing the overall performance of the system.

III. IMPLEMENTATION DETAILS

In this section the overview of the system is discuss with algorithm.

A. System Overview

The following figure 1 shows the architectural view of the proposed system.

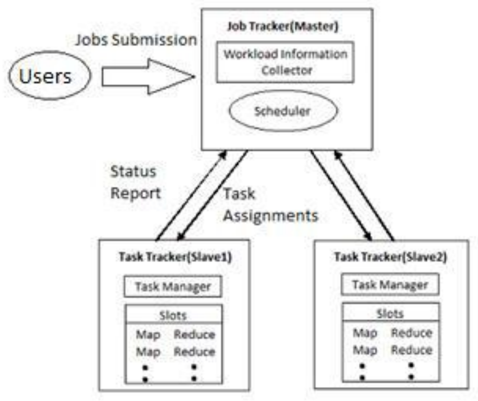

Figure 1: System Architecture *Job tracker* The essential function of the job tracker is dealing with the task trackers, tracking resource accessibility. The Job tracker is a node which controls the job execution process. Job tracker performs Map Reduce tasks to particular nodes in the cluster. Client submits jobs to the Job tracker. At the point when the work is finished, the Job tracker overhauls its status. Client applications can approach the Job tracker for information. *Workload Information collector* It contains the information about the jobs which are executed through log file such as task execution time etc. In our case it would be third party API which provides live metrics of job and their details. *Scheduler* It is responsible for scheduling jobs . *Task tracker* It takes after the orders of the job tracker and overhauling the job tracker with its status intermittently. Task tracker run tasks and send the reports to Job tracker, which keeps a complete record of each job. Every Task tracker is designed with a set of slots, it indicates the number of tasks that it can accept.

B. Algorithm Input :

- ➢ Set of Users U = {u1,u2,.......un} where n is the number of users.
- ➢ Jobs submitted by each user jui .
- ➢ Available slots S in each queue S = {s1,s2,......,sq} where q is the number of queues defined.

Output : Jobs scheduled according to their size. 1. All details of jobs currently available in queue will be extracted using workload information collector (in our case it would be third party API which provides live metrics of job and their details). 2. Jobs will be grouped by user and average size of each job will be calculated. 3. Based on load on each queue all jobs are first arranged in descending order by their size and it will be decided in which queue with how many slots should be allocated for job completion. 4. To improvise the performance of Map Reduce task we will employ Timsort algorithm during sort and shuffle phase to get best performance from parallel processing. 5. To break tie between or priority precedence FIFO will be used to decide which user gets slot first.

C. Experimental Setup

The system can be developed using Hadoop on Linux platform. The system requires 3 laptops with minimum 8gb RAM, Core 2 Duo processor and Ubuntu 14.04 Desktop LTS. One laptop operates as a master node(Job Tracker) and other two as slave 1 and slave 2(Task Trackers).

IV. RESULTS AND DISCUSSION

Table 1 shows the difference between existing and proposed system.

	Existing System	*Proposed System*
Average response Time	High	Comparatively low

In above table, theoretically it has been proved that, proposed system minimizes the average response time than existing system.

V. CONCLUSION

Task assignment in Hadoop is an fascinating issue in light of the fact that efficient task assignment can altogether decrease runtime, or enhance hardware utilization. The proposed scheduler will make use of historic information collected from the job history statistics for current user for making the scheduling decisions between users. The scheduler will reduce the average job response time that are compared with existing Fair and FIFO Scheduler. To improve the performance of job completion Timsort algorithm is used during sort and shuffle phase.

Acknowledgment The authors would like to thank the researchers as well as publishers for making their resources available and teachers for their guidance. We are thankful to the authorities of and concern members of “LsPS: A Job Size-Based Scheduler For Efficient Task Assignments In Hadoop” for their constant guidelines and support. We are also thankful to the reviewer for their valuable suggestions. We also thank the college authorities for providing the required infrastructure and support. Finally, we would like to extend a heartfelt gratitude to friends and family members.

REFERENCES

[1] Yi Yao, Jianzhe Tai, Bo Sheng, and Ningfang Mi, LsPS: A Job Size-Based Scheduler for Efficient Task Assignments in Hadoop, IEEE Transactions on Cloud Computing, no. 1, pp. 1, PrePrints PrePrints, doi:10.1109/TCC.2014.2338291.

[2]] J. Dean, S. Ghemawat, and G. Inc., Map Reduce: Simplified data processing on large clusters, in Proc. 6th Conf. Symp. Operating Syst. Des. Implementation, 2004, p.10.

[3] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, Job scheduling for multi-user Map Reduce clusters, University of California, Berkeley, CA, USA, Tech. Rep. UCB/ EECS-2009-55 , Apr. 2009.

[4] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, The case for evaluating Map Reduce performance using workload suites, in Proc. IEEE 19th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst., 2011, pp. 390399.

[5] https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

[6] https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html