© 2016 IJSRSET | Volume 2 | Issue 4 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

A Time Sharing Scheduler with Multiple Priority Based Queues for Improving Scheduling in Hadoop Cluster - Cloud Environment

Bakul Panchal[1], Priya Chak[2], Jayesh Mevada[2]

[1]Information Technology, L. D. College of Engineering, Ahmedabad, Gujarat, India
[2]Computer Engineering, Merchant Engineering College, Basna, Gujarat, India

ABSTRACT

Today, In the Era of Big data, it is in need of high levels of scalability and efficiently processing is main issue. So there is lot of challenges to handling data like how to store, retrieve and to process data efficiently. Hadoop is a distributed software platform for processing big data on a large cluster, which implements basic mechanism of Google's MapReduce. The MapReduce job-scheduling algorithm is one of the core technologies of Hadoop. The default job scheduler of Hadoop is FIFO, which will start the job in the order as it is submitted, and this causes the job to be started later when it is submitted later. This paper uses the Time Sharing with increased time slot algorithm to solve this problem. With this scheduler, the job which is submitted late, will get quick response and started without long delay**.**

Keywords: Cloud Computing, FIFO, FAIR, Hadoop , MapReduce , Scheduling, SLS, Time Sharing.

I. INTRODUCTION

Big data has become very popular in cloud computing area. Hadoop is an open source distributed software platform, which is good for processing big data. And hadoop is an implementation of MapReduce programming model, which can run applications in the cluster that consists of large number of commodity nodes. It also provides a reliable and scalable distributed file system [9].

In Hadoop, there is a master node and some slave nodes. In the master node, there is a process named JobTracker and in each slave node, there is a process named TaskTracker. The JobTracker is responsible for splitting one job into some tasks and then assign these tasks to TaskTracker. The TaskTracker is the node in which the task is actually running [2].

Job scheduling algorithm is one of the core technologies of Hadoop. The job scheduling algorithm controls the order in which each task will run and also controls the allocation of resources. In Hadoop, the default job scheduler is FIFO. Usually the size of jobs are not the same, some jobs are very short. And for these short jobs, they will be delayed for a long time until jobs ahead of them in the queue are completed, and this will result in a very long response time for these jobs [1].

So we are going to propose a method which can solve this problem. In this method the time is divided into slots and each job will get a time slot fairly. In addition, if job is not completed in given time slot then it will be moved to the next queue which has more time slot than earlier one.

This paper is organized as follows. In section II, we review the current job schedulers in Hadoop. Then we describe the procedure of job scheduling in Hadoop and the problems we want to resolve in section III. In section IV we present the Scheduler design, Pseudo code and implementation of our scheduler. Then Experimental results are showed in section V. Finally, we conclude our algorithm in section VI.

II. METHODS AND MATERIAL

1. Related Work

There are many researches has been done on Hadoop Scheduling. FIFO is a default scheduler of Hadoop.

The main objective of FIFO scheduler is to schedule jobs based on First In First Out. It does not consider priority or size of the job. FIFO scheduler allows jobs to utilize the entire cluster capacity so it has many limitations such as poor response times for short jobs compared to large jobs, Low performance when run multiple types of jobs etc. [1]

Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average an equal share of resources over time. It lets short jobs complete within a reasonable time while not starving long jobs [2]. The objective of Fair scheduling algorithm is to do an equal distribution of compute resources among the users/jobs in the system. Fair scheduling can covers some limitation of FIFO such as it can work well in both small and large clusters and it is less complex. Fair scheduling algorithm gives better response time but as number of jobs in queues increases all over throughput decreases, Because it does not consider the job weight of each node, which is an important disadvantage of it. The Capacity Scheduler is designed to allow sharing of large cluster while giving each organization a minimum capacity guarantee. The central idea is that the available resources in the Hadoop Map-Reduce cluster are partitioned among multiple organizations that collectively fund the cluster based on computing needs and an organization can access any excess capacity no being used by others. This provides elasticity for the organizations in a cost-effective manner [3]. Workload Characteristic and Resource Aware Scheduling Considers the job characteristics and node status. WCRA is less complex and It can provide fast response time for small jobs. However WCRA has a Disadvantage that it does not consider the weight of large jobs [4].

2. Proposed Methodology

Fair Scheduler gives a fair share of the resources available. In this scheduling each job will get one slot for every round, if the number of jobs in the cluster is N and the running time of each task is t, then the average turnover time is N*t. so when the number of jobs get larger, the turnover time will also become larger, and the same as the waiting time for each job. Fair scheduler increases average response time of jobs but not throughput of entire cluster. The number of the jobs in the cluster can be very large, at this situation; the completion time of each job can be very long, especially for the jobs in the tail. And the throughput of the cluster will be brought down as the number of jobs grows. So we use multiple ready queues to hold all the jobs in the cluster. When the multiple queues with different priorities are applied, the jobs will be transferred from queue with high priority to queue with low priority, and the slots that each job get will also get larger and larger. So for a job with x tasks, the times of round will be less than x, and it will increase the throughput of Hadoop.

So we decided to modify the existing Fair scheduler to support the desired functionalities. The Hadoop framework consists of a JobTracker in the master node and a set of TaskTrackers in the slave nodes. And the JobTracker is responsible for scheduling the jobs and assigning the tasks of each job to TaskTracker. TaskTracker sends the heartbeat signal to JobTracker continuously and will get the tasks that assigned to itself in the message of response from JobTracker.

3. Implementation Strategy

✓ *Scheduler Design*

Fig 1 describes a Time Sharing Scheduler with multiple queues based on priority. The tasks that are not completed are shifted to next queue having low priority. The jobs that are completed are removed from queue.

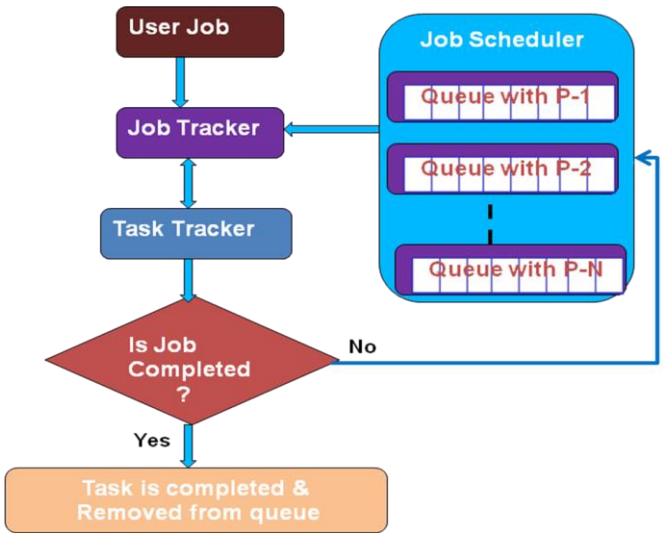

Figure 1. Procedure of Time sharing with multiple priority queues scheduler

✓ ***Algorithm:***

[1] There are N queues in the cluster and each queue is offered a priority.
[2] For each queue, there is a quota allotted which will be consumed in each round and the quota is larger as the priority is less.
[3] At the beginning, the Job Tracker will choose The queue with highest priority, and for this queue, the specified slots will be consumed by jobs in the queue in Time Sharing manner. After that, the jobs, which are not completed, will be Transferred to the next queue with lower priority, so that in next round, these jobs can get more slots. If the queue with high priority become empty, then the queue with low priority can get the slots.

✓ ***Implemented Pseudo Code***

```
void scheduler()
  {
  freeslots=count the information of
heartbeat;
  Jobqueue jobqueue =
GetCurrentjobqueue();
  while(freeslots > 0)
  {
If (current job of head job is not
completed)
    Then
move the head job of this queue to next
queue with lower priority;
    else
    remove it from this queue
    endif
    if (jobqueue.isempty)
        then
        move to next job queue
  endif
      update head job information
       }
    }
```

III. RESULTS AND DISCUSSION

1. Experimental Results

We have used Scheduling load Simulator provided by Apache. Several optimizations are also made to improve scheduler performance for different scenarios and workload. Each scheduler algorithm has its own set of features, and drives scheduling decisions by many factors, such as fairness, capacity guarantee, resource availability, etc. Unfortunately, currently it is non-trivial to evaluate a scheduler algorithm. Evaluating in a real cluster is always time and cost consuming, and it is very hard to find a large-enough cluster. Hence, a simulator, which can predict how well a scheduler algorithm for some specific workload, would be quite useful. It is very important to evaluate a scheduler algorithm very well before we deploy in a production cluster. The Scheduler Load Simulator (SLS) is such a tool, which can simulate application loads in a single machine [20].

We have used 3 Computers having 4 GB RAM and 360 GB hard Disk. We installed Single node setup of Hadoop and given different size of data as input to check the output time by using simple program of WordCount.java. We got better performance for FAIR with compared to FIFO. Then we compared output of FAIR scheduling with our proposed scheduling output. Outputs are shown in below table. We have given input File of Different Size and observed their Output in Seconds. Comparison chart is also shown below. Detailed analysis shows that at initial stage when input file size is small, FAIR scheduler and MyScheduler gives same efficiency but as the input file size increases MyScheduler gives better efficiency than FAIR scheduler gives.

TABLE I
OUTPUT OF DIFFERENT SCHEDULERS

Input Size	Time in Second		
	FIFO	FAIR	MyScheduler
64 MB	1.3	0.9	0.9
128 MB	2.3	1.9	1.9
192 MB	3.1	2.6	2.3
256 MB	3.9	3.3	3

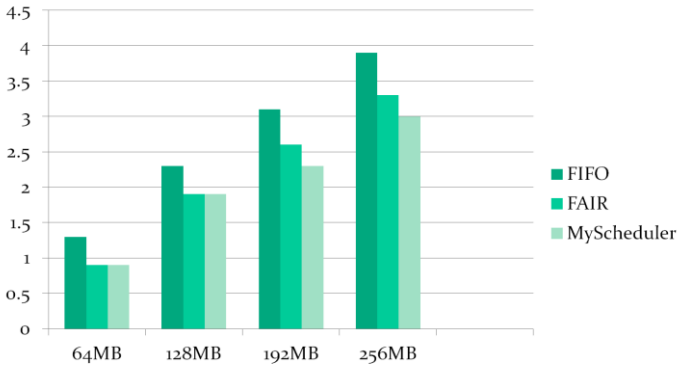

Figure 2. Comparison Chart of FIFO, FAIR and MyScheduler

IV. CONCLUSION

We can see that initially for small input file size FAIR and My Scheduler algorithm runs equally but as file size increases My Scheduler gives better performance than default scheduling algorithms of hadoop. Using the proposed Method of scheduling, we can maintain average response time as well as Increase average job completion time and thus efficiency of scheduling in Hadoop is increased more than its default scheduling algorithms like FIFO and FAIR.

V. REFERENCES

[1] Jiayin Wang, Yi Yao, Ying Mao, Bo Sheng, Ningfang Mi, "Fair and Efficient Slot Configuration and Scheduling for Hadoop Clusters" , 978-1-4799-5063-8/14 © 2014. IEEE DOI 10.1109/CLOUD.2014.

[2] Fair scheduler [online] https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

[3] Capacity scheduler [online] https://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html

[4] Divya M, Annappa B, "Workload Characteristics and Resource Aware Hadoop Scheduler " 978-1-4799-8349-0/15 ©2015 IEEE.

[5] Yi Yao, Jianzhe Tai, Bo Sheng, and Ningfang Mi,Member, IEEE, "LsPS: A Job Size-Based Scheduler for Efficient Task Assignments in Hadoop " , 2168-7161 © 2014 IEEE.

[6] R.Thanga selvi, R.Aruna, "Longest Approximate Time to end Scheduling Algorithm in HADOOP Environment" ISSN (ONLINE): 2454-9762,ISSN (PRINT): 2454-9762. 2016.

[7] Bin Ye, Xiaoshe Dong, Pengfei Zheng, Zhengdong Zhu*, Qiang Liu, Zhe Wang"A delay scheduling algorithm based on history time in heterogeneous environments "978-0-7695-5058- 9/13© 2013 IEEE.

[8] Dazhao Chang, Jio Rao, Changjun jiang and Xiaobo Zhou, "Resource and deadline-aware job scheduling in Dynamic Hadoop Clusters " , 1530-2075/15 @ 2015 IEEE, DOI 10.1109/IPDPS.2015.

[9] Garima Sharma, Dr. Anita Ganpati "Performance evaluation of fair and capacity scheduling in Hadoop YARN" , 978-1-4673-7910-6/15/$31.00 ©2015 IEEE

[10] Seyed Reza Pakize, " A Comprehensive View of Hadoop MapReduce Scheduling Algorithms ", ISSN 2308-9830.

[11] Aprigio Bezerra_†, Porfidio Hern´andez_, Antonio Espinosa_ and Juan Carlos Moure_Escola d'Enginyeria "Job Scheduling in Hadoop with Shared Input Policy and RAMDISK ",978-1- 4799-5548-0/14/$31.00 ©2014 IEEE

[12] Shen Li ∗, Shaohan Hu ∗, Shiguang Wang ∗, Lu Su †, Tarek Abdelzaher ∗, Indranil Gupta∗, Richard Pace, "WOHA: Deadline-Aware Map-Reduce Workflow Scheduling Framework over Hadoop Clusters" 1063-6927/14 $31.00 © 2014 IEEE,DOI 10.1109/ICDCS.2014.

[13] Rakesh Verma " Survey on MapReduce and Scheduling Algorithms in Hadoop" paper ID: SUB151194. International Journal of Science and Research (IJSR),ISSN (Online): 2319-7064.

[14] Mr.A.U.Patil1, Mr T.I Bagban2, Mr.A.P.Pande, "Recent Job Scheduling Algorithms in Hadoop Cluster Environments " , ISSN : 2278-1021, 2011.

[15] Peng Qin, Bin Dai, Benxiong Huang, and Guan Xu, " Bandwidth-Aware Scheduling With SDN in Hadoop:A New Trend for Big Data " , 1932-8184 © 2015 IEEE.

[16] Deveeshree Nayak, Venkata Swamy Martha, David Threm,Srini Ramaswamy,Summer Prince and Gunter Fahrnberger, "Adaptive Scheduling in the Cloud – SLA for Hadoop Job Scheduling " , 2015

[17] Jisha S Manjaly, Varghese S Chooralil, "TaskTracker Aware Scheduling for Hadoop MapReduce" in Third International Conference on Advances in Computing and Communications, 2013.

[18] Prajesh P Anchalia," Improved MapReduce k-Means Clustering Algorithm with Combiner", 978-1-4799-4923-6/14 © 2014 IEEE,DOI 10.1109/UKSim.2014.

[19] Akram Roshdi, Mahboubeh shamsi, " Review: Big data on Cloud computing" ISSN: 2319, july 2015.

[20] https://hadoop.apache.org/docs/r2.4.1/hadoop-sls/SchedulerLoadSimulator.html.