ORIENTAL JOURNAL OF COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
Published By: **Oriental Scientific Publishing Co., India.**
www.computerscijournal.org

ISSN: 0974-6471
March 2017,
Vol. 10, No. (1):
Pgs.194-200

Improved Fair Scheduling Algorithm for Hadoop Clustering

SNEHA and SHONEY SEBASTIAN

Department of Computer Science, Christ University, Bengaluru,India.

http://dx.doi.org/10.13005/ojcst/10.01.26

(Received: March 11, 2017; Accepted: March 18, 2017)

ABSTRACT

Traditional way of storing such a huge amount of data is not convenient because processing those data in the later stages is very tedious job. So nowadays, Hadoop is used to store and process large amount of data. When we look at the statistics of data generated in the recent years it is very high in the last 2 years. Hadoop is a good framework to store and process data efficiently. It works like parallel processing and there is no failure or data loss as such due to fault tolerance. Job scheduling is an important process in Hadoop Map Reduce. Hadoop comes with three types of schedulers namely FIFO (First in first out), Fair and Capacity Scheduler. The schedulers are now a pluggable component in the Hadoop Map Reduce framework. This paper talks about the native job scheduling algorithms in Hadoop. Fair scheduling algorithm is analysed with its algorithm considering its response time, throughput and performance. Advantages and drawbacks of fair scheduling algorithm is discussed. Improvised fair scheduling algorithm is proposed with new strategy. Analysis is made with respect to response time, throughput and performance is calculated in naive fair scheduling and improvised fair scheduling. Improvised fair Scheduling algorithms is used in the cases where there is jobs with high and less processing time.

Keywords: HDFS, Map reduce, Scheduling, Fair scheduling, Job tracker, Task tracker.

INTRODUCTION

Hadoop is a distributed framework for large amount of data. It is a solution for big data which deals with complexities of high volume, velocity and wide variety of data. Hadoop has a set of open source projects to provide common services. Hadoop is fundamentally infrastructure and software for storing and processing large amount of data which is also an open source project under Apache[1]. Storage in Hadoop is in distributed manner. In Hadoop all large files are fragmented and stored in different nodes around cluster for processing.

Importance of Hadoop

Hadoop transforms commodity hardware into a service that stores petabytes of data reliably and allows huge distributed computations[2]. Key attributes of Hadoop isreliable and redundant in which is useful in case of failures. There is no data loss, extremely powerful, batch processing centric, easy to program distributed applications, runs on commodity hardware. Data is held on multiple Data nodes, automatically replicated when machine fails or task fails that is automatically redone. Hadoop is scalable like you have one application running in one machine and automatically that can be scaled

up to 100 0r 1000 machines, scales linearly[3]. It provides simple APIs for both compute and data access and it is very powerful as you can process huge amount of data (in petabytes). Processing in parallel allows for the timely processing of massive amount of data.

Framework of Hadoop

Hadoop distributed file system is a cluster on which the data is stored. It has master slave kind of architecture where 'Name node' is the master which contains information about metadata and manages all the data nodes where the actual data gets stored in terms of blocks[4,5]. There is one Name node in the cluster and there can be thousands of 'Data node' in the cluster. Name node is not a high availability machine i.e. it has high memory and redundant power supplies also if power supply goes down performance is affected. Secondary Name node takes the backup of Name node which will be useful at times of failure. If the Name node fails, it can be set automatically with the log of secondary Name node which has a configuration file of Name node.

Map Reduce is a framework - a programming(java) model of Hadoop. To run any of the batch processing jobs which is distributed across the cluster the appropriate program should be written in map reduce. The map reduce paradigm takes care of scheduling jobs, monitoring jobs, allocating resources, mining jobs and managing failures[6]. Job tracker is the core service or the thread running all the time which is used for scheduling the jobs on the data nodes and to monitor these tasks. The jobs are submitted to a queue in the Job tracker which in turn schedules the tasks to various Data nodes. Job tracker also monitors the submitted job[7]. The map reduce program will run on the data nodes and the data is fetched from the data node based on map and reduce program through the task tracker. Both components are necessary to run the jobs on data node and program can be executed parallelly on thousands of machines.

Map reduce programming paradigm

The file which need to be stored and processed into the database will be splitted into different chunks of size 64MB[8]. This data operates according to the key value pair. Only the required data is selected from this and remaining is filtered out. Only the required data is selected in the mapper phase based on the programming model. This mapper function will run on all the fragmented blocks. The reducer will combine the output of mapper which is used for further processing. This is very efficient way to process the cluster, where the same code can be applied to all the required and necessary clusters.

Job Scheduling

High performance computing systems are comprising of multiple resources. These resources include servers, storage, memory, software, GPUs, networks etc., When user submits the job they specify the resources they need. For example, consider (nodes) m5 and m4 are big memory nodes, some jobs may require many processors, some jobs may require few processors but consumes lot of time[9]. Some jobs may require special resources such as InfiniBand or large amount of system memory. As user submits job, the job schedules the resources where job is ready to run. At some point the cluster will fill up, at this point the job scheduler will hold the job in wait queue until the running jobs completes, thus increasing the availability of resources. The job scheduler allows queue jobs to run. Another task of a job scheduler is to assign a priority to each job waiting in a queue. The highest priority job sits in top of the queue waiting for computer resources to be available. Every couple of minutes the job queue is re-evaluated and adjustments are made to the queue, based on waiting jobs on the respective priority levels. Then the job scheduler rearranges the jobs in queue based on priority to designate priority of individual jobs and the cycle repeats again.

Job scheduling in Hadoop

Since Hadoop is a very large clustered environment many tasks will be parallelly executing, scheduling should take care of allocating resources on the priority basis and on what basis the priority should be given. Job Scheduling is allocating resources in a best possible way among multiple tasks[9]. Scheduling happens in task tracker in case of Map Reduce 1 and resource manager in case of Map Reduce 2, Few such scheduling algorithms are discussed in the following sections.

Literature Review

Hadoop has general scheduling algorithms and there are few efficient algorithms but that is suitable only for some scenarios and it is not generic like resource aware scheduling, LATE fair scheduling, delay scheduling, deadline constraint scheduling and so on.

Job Scheduling Algorithms in Hadoop

FCFS Scheduling

H. Patel and R. Sonaliya[11] mentions that first in first out is the default and very traditional way of job scheduling. The jobs are stored in a queue, and the scheduling priority is given to the jobs whichever occurred first in the queue and the new job will be assigned to the end of the queue[2]. Priority can be assigned to each job in a queue which avoids the starvation of the jobs in a queue. Advantage of FCFS is simple and easy to realize. The disadvantage of FCFS is that smaller jobs must wait for longer time if there is any job which takes longer time to complete, which reduces resource utilization.

Capacitive Scheduling

M. Zaharia *et.al*[12] describes that capacitive scheduling is initially implemented by yahoo for the scenario where the number of users is large and to ensure that fair allocation of resources is allocated. This takes a little different stands to multi user scheduling. In this case queues are divided on the basis of users or groups of users which is termed as organizations and there are multiple queues specific for the organization, each queue is given portion of resources in a cluster[13]. Consider if a new job enters into organization A, so it would be picked up as there is no job running. Moreover this would take up as many resources are available and cluster is effectively utilized. When job in organization B appears, tasks of the first job will be killed to free up slots for the new job. The main features of capacitive scheduling is capacity guarantees, elasticity, multitenancy, hierarchical queues and security.

Deadline Constraint Scheduler

Geetha *et al*[14] describes that Deadline Constraint Scheduler addresses the issue of deadlines but focuses more on increasing system utilization. By default, the jobs are scheduled based on the priorities assigned to the job by the user. But in deadline constraint scheduler user submits the job with deadline. Before allotting the job to the task tracker, job tracker computes the minimum number of slots required by the heartbeat message sent by task tracker to the job tracker. If the minimum number of slots meet the requirement, then the jobs are allocated for processing else user is notified saying slots are not available anduser can modify the deadline of job and can be resubmitted. The job can be scheduled for execution. Here the jobs are not killed and freed the slots, instead deadline is modified which doesn't affect the currently executing jobs. The throughput of the job is not taken into consideration and if the minimum number of slots is not available then the user should wait for a longer time.

Resource aware Scheduler

A. Patel[15] describes that Whenever the user submits job, job tracker checks whether the resources (cpu space, time, disk space etc.,) are available in the slots which is required to execute the jobs. If the resources are available in the task tracker, then the jobs are allotted else either the jobs are in waiting state until the resources are freed up or the jobs are fragmented depending on the required size and then jobs are allotted to each slot. Resource Aware Programming in Hadoop has prettify one of the Explore Challenges [5-6] in Cloud Computing. Programming in Hadoop is centralized, and initiated. Planning decisions are confiscated by a combatant node, called the Job Tracker, whereas the miss nodes, called Task Trackers are obligated for task execution.

Few of the existing job scheduling algorithms are discussed in detail in the above section. Fair scheduling is also one of the traditional and default algorithm in Hadoop.

Fair Scheduling

This is implemented by Facebook for the scenario of fair allocation of resources for the users. A. Patel *et.al*[16] describes that the jobs are assigned to a pool and not to queue before allocating, once the pool is full, fair amount of resources are allocated to the jobs irrespective of the job type (prioritized job, longer job, small job etc,) Conceptually it is like capacitive scheduler with minor differences. With capacitive schedulers, the queues are divided and termed as poolsand the jobs would be picked up

from the pools and would be given their portions of the resources[17]. Suppose if another job comes into the pool then this is processed like First in first out or FIFO with priority, in this case a small high priority job should wait for a long time this is improved in fair scheduler that the jobs which was waiting the queue is picked up and processed parallel to give a better user experience. A simple way to share a cluster fairly between jobs is to always assign free slots to the job that has the fewest running tasks. If slots become free quickly enough, the resulting allocation will satisfy max-min fairness.

Process allocation of fair scheduling

Advantage of the fair scheduling is that it supports the scheduling of divided homework. The different type of tasks will get the different assignment of resources. It will promote the quality of service, and adjust the amount of homework executed the same time and it will use resources deeply.

Disadvantage is that it ignores the node of the balance states, and it will result in imbalance.

Delay Scheduling

An improvisation to the Fair Scheduling is delay scheduling. While storing data in a Hadoop, we fragment the file and then it is stored and it can be stored in different clusters[18,19]]. So the objective of delay scheduling is to launch the jobs to a node where relevant data is available so that it takes less time and we get high throughput. In case if the node is not free, then the job is skipped for D times and the job is launched to the node where data is available. If the node is still busy even after skipping D times, then the job is launched on the nodes whichever is free.

Alogrithm: 1 Navir Fair Scheduling

```
when a heartbeat is received from node n:
        if n has free slots then
sort job in increasing order of number of runnuing
task for j in job do
        if j has unlaunched task t with data on n
        then launch t on n
        end if
    end for
end if
```

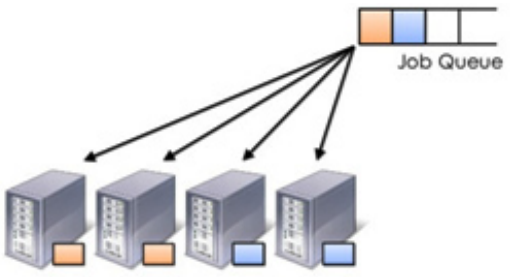

Fig.1:Process allocation of fair scheduling

Proposed work

This is like fair scheduling with little modifications. Whenever job is submitted, the jobs are sorted according to the processing time. Smaller jobs are assigned to the high performance systems which takes very less time to process the job.

Process allocation of Improvised Fair Scheduling

Improvised fair scheduling algorithm

The fair amount of resources is given to the jobs in a cluster. The jobs may take long duration to execute or it executes quickly which depends on the scenario. So all the jobs are given a fair share. Drawback of fair scheduling is that it ignores the node of the balance states, and it will result in imbalance.

To avoid the imbalance in the cluster some criteria's is to be followed while processing the job. Instead of randomly allocating job to the nodes, we are allocating shortest job to the nodes which has high performance. Sort the jobs in ascending order based on their processing time and also sort the nodes based on its performance. Whenever the user submits job, all the jobs are sorted according

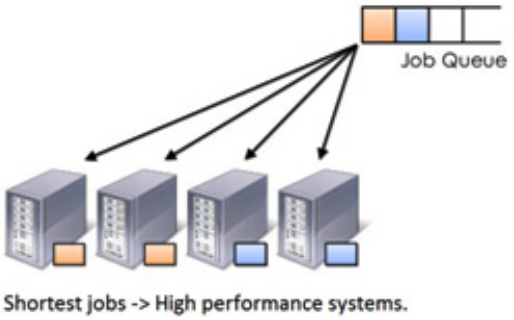

Fig. 2: Process allocation of Improvised Fair Scheduling

to the processing time. Nodes are categorized based on their performance. As shortest jobs give high throughput they are processed by high speed systems so that response time will be high, accordingly jobs are processed. Advantage of this approach is that it is very efficient when compared to the default fair scheduler and throughput is high. Disadvantage is that there are cases where we have to take care of longer jobs and overloading issues in a node.

Alogrithm: 2 Improve Fair Scheduling Algorithm

pre: Sort the notes accordingly with their o performance.
Sort the job accordingly to their processing time.

1 when a heartbeat is received from node n:
1.1 if n has free slots then
1.2.1 sort job in increasing order of number of running tasks.
1.2.2 for j in job do

2.1 if j has unlaunche task t with data on n
2.1.1 then launch t om n
else
2.2 if j has unlaunched task t then
2.2.1 then launche t in n
2.3 end if
3 end for
4 end if

Experiments and results

Tests are made with fair scheduling and with the new approach (improvised fair scheduling) considering the parameters like tasks divided, response time, performance and throughput.

Table1: Response time

No. of Tasks	Fair Scheduling (MicroSecs)	Improvised Fair Scheduling (MicroSecs)
5	3240	2820
10	6420	4850
20	9060	7090

Response Time

Response times is amount of time taken for job to complete its processing time.
Throughput is number of jobs completed its processing in a unit of time.

The response time is 2820s when we use new approach of fair scheduling algorithm, where as in fair scheduling algorithm the response time is 3240s in first case. So we get better response time compared to the naive fair scheduling.

System Performance

There is no difference between the throughput when their less number of tasks. As the tasks increases in number we get better performance than the naive fair scheduling. Here we have high response time in 2nd case, but fair scheduling has average response time. When we use this algorithm with increased number of tasks, performance is better than the fair scheduling.

Graphs Plotted

When we use Fair Scheduling

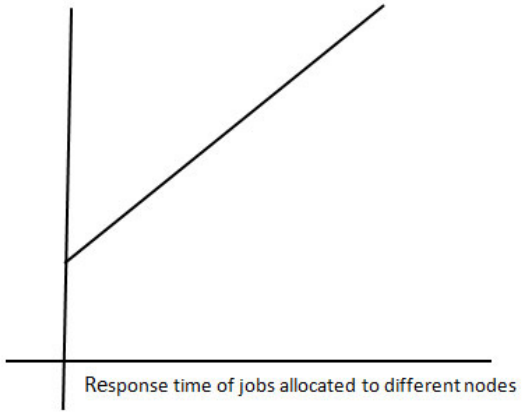

Fig. 3: Response time graph for Fair Scheduling

Table 2: System performance

No of Tasks	Fair Scheduling (MicroSecs)	Improvised Fair Scheduling (MicroSecs)
5	High	High
10	Average	High
20	Low	Average

When we use Improved Fair Scheduling

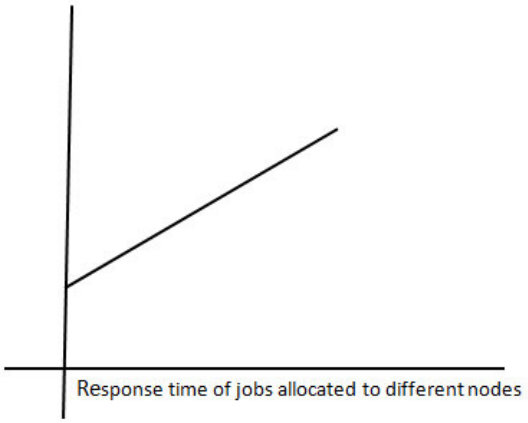

Fig.4: Response time graph for Improvised Fair Scheduling

CONCLUSION

Hadoop framework is scalable up to 40,000 nodes , the problem arises when the cluster increases and all slots should be used effectively. So algorithms should be selected in such a way that output should be effective and high throughput should be produced. Fair scheduling algorithm gives high throughput and less starvation but resources are wasted, with the use of improvised fair scheduling algorithm throughput is high and better efficiency with proper use of resources.

The idea of improvisation for the fair scheduling is discussed and analysed above. It is used in most of the scenarios where there is combination of small and large jobs availability. Thereis no starvation for the jobs and resources are fairly distributed to all the jobs. This can be enhanced in the future by adding few constraints and making it more generic.

REFERENCES

1. M. Ramla, "Significance of various Hadoop job schedulers – A retrospective", international *journal of engineering sciences & research technology*, No: 2277-9655, 2016.
2. H. Bhosale and D.Gadekar, "A Review Paper on Big Data and Hadoop", *International Journal of Scientific and Research Publications*, **4**(10), 2014.
3. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Communications *of the ACM*, **51**(1), Pp. 107-113, 2008.
4. A. Gates *et al*, "Building a High-Level Dataflow System on top of MapReduce: The Pig Experience," *Proceedings of the VLDB Endowment*, **2**, (2), Pp. 1414-1425, 2009.
5. A. Kadam and P. Deshmukh, "A Review on Distributed File System in Hadoop", *International Journal of Engineering Research & Technology,* **4***(2), 2015.*
6. H. Liao *et al*,"Multi-Dimensional Index on Hadoop Distributed File System**",** *Fifth IEEE International Conference on Networking, Architectureand Storage***,** 2010.
7. https://blog.cloudera.com/blog/2008/11/job-scheduling-in-hadoop/.
8. *M. Almeer***,"***Hadoop MapReduce for remote sensing image analysis", **2**, Pp. 4, 2012.*
9. Z. Zhao, "User-Based Collaborative-Filtering Algorithms on Hadoop", *Third International conference on knowledge discovery and Data Mining,* 2010.
10. S. Pakize, "A Comprehensive view of Hadoop MR Scheduling Algorithms", *International journal of computer networks and communications security***, 2**(9), 2014**.**
11. H. Patel and R. Sonaliya, "Improving Job Scheduling in Hadoop Mapreduce**",***International journal of innovative research in technology***,** **2**(1) , 2015.
12. M. Zaharia *et al*, "Improving MapReduce performance in heterogeneous environments.", *Operating systems design and implementation: Proceedings of 8th USENIX conference,* Pp. 29-42**,** 2008.
13. http://docplayer.net/14486352-Job-scheduling-with-the-fair-and-capacity-schedulers.html
14. Geetha *et al*, "Hadoop Scheduler with

Deadline Constraint", *International Journal on Cloud Computing: Services and Architecture*, **4**(5), 2014.

15. A. Rasooli and D. Down "A Hybrid Scheduling APproach for Scalable Heterogeneous Hadoop Systems" *SC Companion: High Performance Computing, Networking Storage and Analysis,* Pp. 1284-1291, 2012.
16. A. Patil *et al*, "Recent Job Scheduling Algorithms in Hadoop Cluster Environments: A Survey", *International Journal of Advanced Research in Computer and Communication Engineering*, **4**(2),2015.
17. A. Patil et al, "workload analysis security aspects and optimization of workload in Hadoop clusters*", International Journal of Computer Engineering Technology*, **6**(3), 2015.
18. G. Sasiniveda andN. Revathi," Performance Tuning and scheduling of Large data set analysis in Map Reduce Paradigm by Optimal Configuration using Hadoop", *International Journal of Computer Applications*, **70**(21), 2013 .
19. Y. XIA *et.al*, "Research on Job Scheduling Algorithm in Hadoop*", Journal of Computational Information Systems,*2011.
20. http://hadoop.apache.org.
21. Hadoop fair scheduler - http://hadoop.apache.org/common/docs/r0.20.1/fair_scheduler.html.
22. B. Andrews and Binu, "Survey on Job Schedulers in Hadoop Cluster", *Journal of Computer Engineering,* **15**(1), Pp. 46-50, 2013.
23. Kamal and K. Anyanwu, "Scheduling Hadoop Jobs to Meet Deadline", *Second International Conference on Cloud Computing technology and Science,* Pp. 388-392, 2010.