World Journal of Engineering Research and Technology
WJERT
www.wjert.org
SJIF Impact Factor: 3.419

AN IMPROVED JOB SCHEDULING METHOD BY USING RELEASED RESOURCES FOR MAPREDUCE

Indumathi. S[1], Pavithra. T. N[2], Shruti. M. Masali[3], Thriveni. M. D[4*], Malatesh. S. H[5]

[1234]B. E, Student of M.S.E.C, Bengaluru, India.

[5]HOD (MTech (CSE), PhD, MCSI, MISTE), Department of Computer Science and Engineering, M.S.E.C, Bengaluru, India.

Article Received on 14/03/2016 **Article Revised on 04/04/2016** **Article Accepted on 24/04/2016**

***Corresponding Author**
Thriveni. M. D.
B. E, Student of M.S.E.C, Bengaluru, India.

ABSTRACT

Big data is defined as large amount of data which requires new technologies and architectures to make possible to extract value from it by capturing and analysis process. Big data has emerged because we are living in a society which makes increasing use of data intensive technologies. Due to such large size of data it becomes very difficult to perform effective analysis using the existing traditional techniques. Big data concept means a datasets which continue to grow so much that it becomes difficult to manage it using existing database management concepts and tools. 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 has become one standard for big data processing. However job scheduling in this model is always a challenge for the research fraternity and several job scheduling algorithms have a already have been proposed by researchers addressing various issues. All such algorithms mostly emphasize on meeting of job deadline without taking cognizance of additional released resources being available through intermediate release of resources during processing. In this paper, we propose a new job scheduling algorithm which satisfies job deadline for all scheduled jobs along with utilization of the unused resources so that more number of jobs can be scheduled and the overall system efficiency will be increased.

KEYWORDS: MapReduce; Big Data; Job Scheduling; Improved Fair Scheduling.

INTRODUCTION

Hadoop is an open-source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Distributed processing of large sets across clusters of computers using simple programming modules. A hadoop frame-work application works in an environment that provides distributed storage and computation across clusters of computers. In recent times, data generation rate has been increased to a great extent. Huge volume of data is being generated by various web applications as well as different scientific and technical experiments.

Hadoop framework includes following 4 modules.

- **Hadoop common**

These are java libraries and utilities required by other hadoop modules. These libraries provide file system and OS level abstractions and contain the necessary java files and scripts required to start hadoop.

- **Hadoop YARN**

This is a framework for job scheduling and cluster resource management.

- **Hadoop Distributed file System(HDFS)**

A distributed file system that provides high throughput access to application data.

- **Hadoop MapReduce**

This is a YARN based system for parallel processing of large data sets.

MapReduce is a distributed programming model for parallel processing on Terabytes of data, which has become one standard for processing Big Data. It is scalable and parallel processing programming model. There are two functions – a Map functions and a Reduce functions.

- **The map task:** This is the first task which takes input data and converts it into set of data where individual elements are broken down into tuples (key/value pairs).
- **The reduce task**: This task takes the output from a map task as input and combines those data tuples into a smaller set of tuples. The reduce task is always performed after the map task.

There is a technique which decides the execution order of the tasks in the slots, called job Scheduling Algorithms. Different Job scheduling algorithms have already used for the parallel processing framework which takes care of the Data Locality, resource utilization, MapReduce Interdependence and meeting job deadline. when jobs are smaller data locality is considered and cannot ignore the data travelling and network cost during the calculation of Map Task Completion Time(MTCT). The data locality is considered as a very little part when job is with a relatively higher execution time. Hadoop's default scheduler is FIFO. For MapReduce jobs FIFO schedules jobs in FIFO order with different priority levels. These job scheduling algorithms lacks from meeting the job deadline and resource utilization. The other scheduler FAIR scheduler focuses on guaranteeing FAIR share for all the users and providing better resource utilization. Here in the FAIR scheduler it lacks in meeting the deadline of a job and also the data locality, MapReduce interdependence are not addressed here. In this paper we are focusing to improve the FAIR scheduler by achieving the job deadline by utilizing the released resources and improving the overall efficiency the system.

HADOOP MAPREDUCE SCHEDULING AND ISSUES

Hadoop allows the user to submit the job and also to configure the job, control the job execution. Every job consists of some independent tasks, and all those tasks need to have a system slots to run. In Hadoop all scheduling decisions are made on a task for both map and reduce phases. The Hadoop scheduling model is a Master/Slave/Worker cluster structure. The master node i.e. Job Tracker coordinates the worker machines i.e, .TaskTracker. Job Tracker is a process which manages the jobs and Task Tracker is which manages the tasks. Map Reduce tasks are independent in each machine.

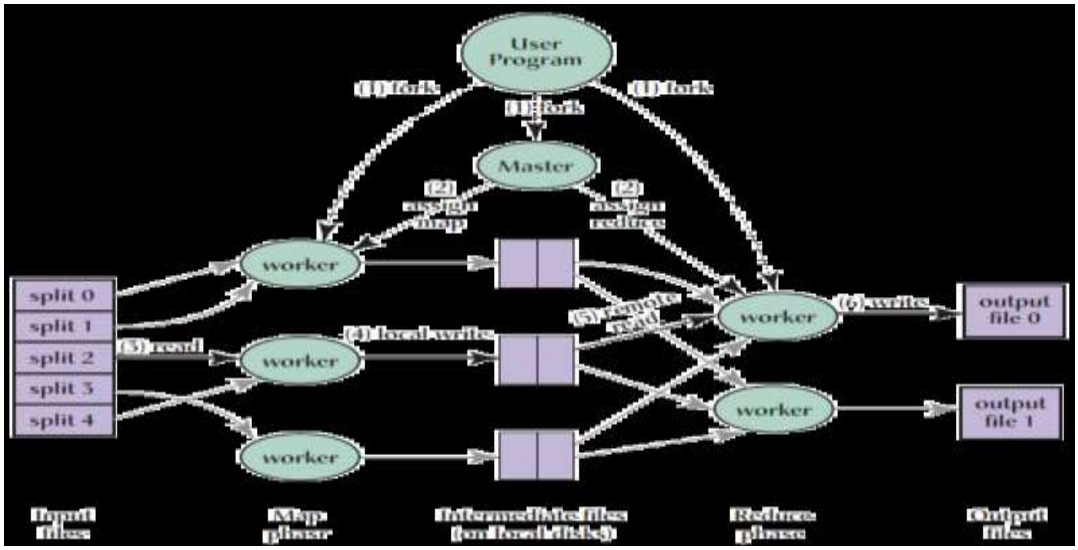

Fig.1. Overview of a MapReduce job

The Hadoop Map Reduce scheduling issues comes here. There are three important scheduling issues in MapReduce such as synchronization, locality and fairness. locality is a very crucial issue affecting performance in a shared cluster environment. The time when scheduler tries to assign Map tasks to slots available on machine in which the underlying storage layer holds the intended to be processed is the concept of data locality in MapReduce. Synchronization is the process of transferring the intermediate output of the map process to the reduce processes as input and this is also considered as a factor which affects the performance. Generally reduce process will start when the map process completes. . Due to this dependency, a single node can slow down the whole process, causing the other nodes to wait until it is finished .So synchronization of map and reduce phases, made up essential help to get overall performance of MapReduce Model. The demands of the workload can be elastic, so fair workload to each job sharing cluster should be considered. Synchronization overhead could affect the fairness. To solve these important issues many scheduling algorithms have been proposed in the past decades. in the next section we describe these algorithms.

SCHEDULING ALGORITHMS

The important issues that we described, and many more problems in scheduling of MapReduce, the Scheduling is one of the most critical aspects of MapReduce. There are many algorithms to address these issues with different techniques and approaches. Some of the algorithms get focus to improvement of data locality and some of them implements to provide Synchronization processing. Also, many of them have been designed to minimizing the total completion time. In this section we describe briefly some of these algorithms.

FIFO SCHEDULING ALGORITHM

The default scheduler of Hadoop is FIFO.FIFO scheduler chooses the homework to execute by considering the priority of homework and their arriving.. First come, and first go. FIFO is quite simple. In the platform of Hadoop there is only one homework queue. And the submitted homework will be queued by time .The new incoming task will be inserted at the tail of the task queue. The advantage of this scheduling is simple and easy to realize. But the disadvantage of this scheduling is obviously. It sees all the homework as the same without regarding of the time of homework. And this kind of scheduling is bad to small homework.

FAIR SCHEDULING STRATEGY

This method configures the task pool in the system, and a task pool can execute a task. These tasks are the result of a big homework divided. When one user submits much homework,

every task is assigned to some task pool to execute the tasks. Fair uses a two level hierarchy for assigning slots to the users. FAIR creates a pool per user. First it allocates task slots across pools, and at the second level, each pool allocates its slots among various jobs in the pool. Each pool is given a minimum share, which can be zero and it ensures that each pool will receive its minimum share, and the sum over the minimum shares of all pools does not exceed the Cluster Capacity.

When a pool is having some free slots from its minimum share, those slots are distributed amongst other pools. When a new job enters the system then two different timeouts are given to it. A newly started job must get its minimum share before the given timeout expires; otherwise the scheduler starts killing other pools' tasks and re allocates them to the job. If the job has not got its fair share by the second timeout, FAIR kills more tasks. While killing task it chooses the most recently launched task in a job which is having more slots than its fair share.

This scheduler mostly focuses on resource utilization but it can't ensure the job deadline. So, whether a scheduled job can be finished within time or not that question is not answered by this scheduling algorithm. FAIR scheduler is also not focussed towards MapReduce interdependence and data locality.

IMPROVED FAIR SCHEDULER

Improved fair scheduler is one of the job scheduling algorithm based on fair scheduler. Improved fair scheduler algorithm differentiates the job as CPU bound and I/O bound. Jobs with a smaller data transfer time than the Map Task Completion Time (MTCT) is called CPU bound job and jobs with a larger data transfer time than the Map Task Completion Time (MTCT) is called I/O bound job is task completion.

To kill the task an improved fair scheduler considers the job type and data locality. Though improved FAIR scheduler tries to improve the resource utilization and considers Data Locality but it not completely meets job deadline and does not take care of Map Reduce interdependence.

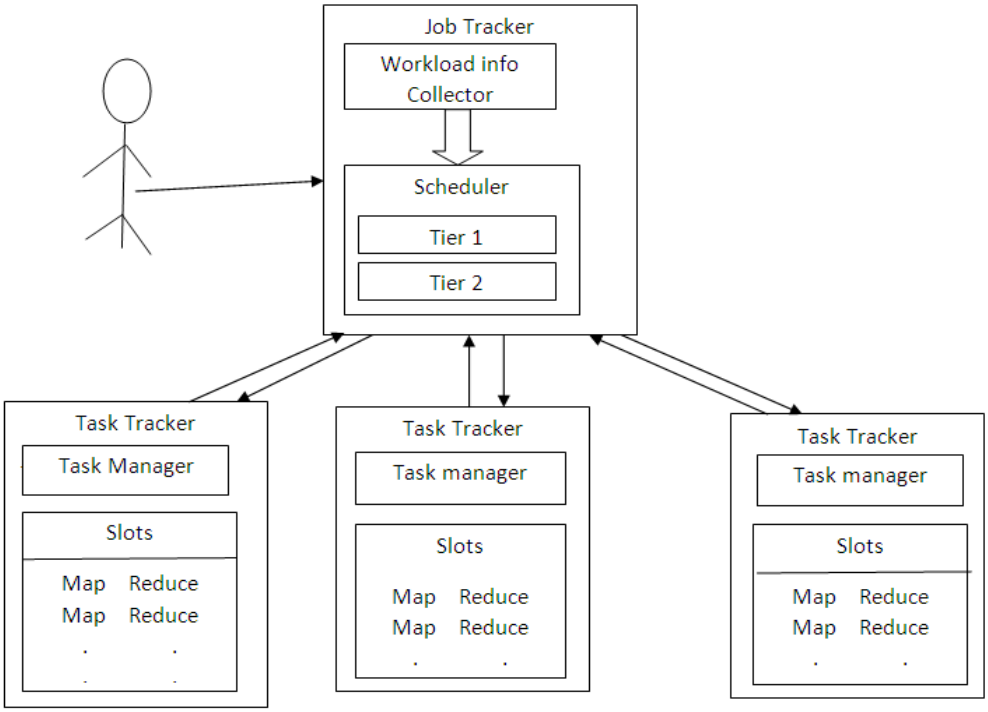

Fig 2: System Architecture

IMPLEMENTATION

We propose a new job scheduling algorithm, which takes care of the MapReduce and satisfies Job Deadline along with better Resource Utilization.

For the word count a MapReduce functions implemented and is tested in machines of various capacities.

Input:Jobs_Relaxed_Share,Jobs_Minimum_Share,Map_Task_Completion_Time,Jobs_Start_Time, Current time

Output: Jobs_Share

Step I: Kill the most recently launched task from Jobs_Relaxed_Share of the job except for a job that has got its full share.

Step II: If the job has not got its full Jobs_Minimum_Share then do the following.

Step III: Check the number of slots needed to fulfill the Jobs_Minimum_Share.

Step IV: Find a job whose Jobs_Relaxed_Share can fulfill the demand.

Step V: If we get one then we need to check the Estimated Completion Time from {Map_Task_Completion_Time – (Current time – Jobs_Start_Time)}.

Step VI: If Estimated Completion Time _ {CEILING (Total Slots/ (Present Share of the Job+ Jobs_Relaxed_Share) X Completion Time of a Single Map_Wave}.

Then wait for the slots to complete their execution.

Else Kill the Jobs_Relaxed_Share.

CONCLUSION

In this paper, we have proposed a new job scheduling algorithm, which improves the utilization of the unused resources it will allow more number of jobs to be scheduled and satisfies job deadline for a scheduled job. When the job is complete every job releases its resources and till new job appears these resources are unused. Instead of waiting for a new job to appear and utilize the leftover resources, we allocate the released resources to the existing jobs and eventually the jobs with additional resources are finished earlier. So the resources are utilized efficiently and overall system efficiency will be increased.

REFERENCES

1. Tom White,"Hadoop: The Definitive Guide", O'Reilly, Yahoo! Press, ISBN: 978-0-596-52197-4.
2. Rajkumar Buyya, James Broberg, Andrej Goscinski,"Cloud Computing: Principles and Paradigms", Wiley, ISBN: 978-0-470-88799-8.
3. Kamal Kc, Kemafor Anyanwu, "Scheduling Hadoop Jobs to Meet Deadlines," CloudCom In IEEE, Dec 2010.

Indumathi S

Pavithra T N

Shruti.M.Masali

Thriveni M D

Malatesh S H