Design and Implementation of Novel Job Scheduler for MapReduce System in Hadoop Platform

Mohammed Muzammil Mehdi¹ Prakash S²
¹M. Tech Student ²Associate Professor
¹,²Department of ISE
¹,²RNS Institute of Technology, Bangalore

Abstract— Apache Hadoop is an open source group of tools which is widely adopted in industries and as well as in academia due to its processing of large data in parallel. MapReduce is the most important component of Hadoop which is a programmable framework for pulling a data in parallel across the number of clusters. When lots of users submit their job at the same time they compete for the same resources as a result the job response time an important factor system performance seriously disgraced. So there is a need of efficient scheduler for such MapReduce clusters. To overcome this problem an efficient scheduler is designed and implemented which make use of the knowledge of workload patterns to improve the response time.

Key words: MapReduce, Hadoop Distributed File System (HDFS), Job Tracker, Task Tracker, Name Node, Data Node.

I. INTRODUCTION

Apache Hadoop and its open source implementation component MapReduce Programmable framework has been widely adopted in industries. Hadoop is evolved as an enterprise valuable solution for the two ingredients such as processing of large data which is greater than 10 Terabytes and another one is complex calculations. Hadoop takes all different types of data such as structured data, semi-structured data and unstructured data stores them analyze them and make better decisions.

The important component MapReduce which is called as Heart of Hadoop is a programmable framework in which huge amount of data processing jobs are processed and stored in parallel across many clusters of computer. MapReduce consists of two main tasks one is a map task and another one is reduce task. Map task divides the data in to small chunks in terms of key and value pair. Whereas reduce task fetch that data as its input, aggregates all the data and produces the final result.

MapReduce framework consists of two main components such as job tracker and the task tracker. The job tracker breaks the lengthier tasks or jobs in to a smaller chunks and it sends all small chunks of computations to the task tracker. The work of the task tracker is to process those small chunks of tasks of the particular node. Each MapReduce framework consists of one master node which manages all the distributed slave nodes. The master node communicates with the slave nodes through heartbeat messages. The scheduling of jobs is performed by the centralized Jobtracker present in the master node. The scheduler transfer the jobs to that particular slave nodes which consists of free resources, and it also responds to the master node through heartbeat messages.

Typically when flocks of users submits the jobs which consists of large amount of data and these data is of different sizes along with the different types of workload patterns that needs to be processed quickly. Under such scenario when users submits their jobs at the same time they compete for the same resources which cause, the important factor of system performance called average job response time seriously degraded. It is found that MapReduce framework often lagging behind in heavy-tailed (or long-tailed), in which MapReduce workloads consists of less but very huge jobs and many small jobs.

In such a MapReduce environment, there is a need for efficient scheduler to boost the system performance by improving the average job response time. However it is noticed that the existing schedulers adopted by the Hadoop do not work properly in heavy-tailed and diverse workloads. Hadoop by default uses a scheduler called FIFO (First in First Out), one of the biggest drawback of FIFO scheduler is that the smaller job has to wait for a longer time when it is submitted after a larger jobs. To overcome this in (7), Fair scheduler is proposed to boost the average job response time. In this Fair scheduler the resources are allocated fairly without considering workload patterns of different users.

In this paper we propose a novel Hadoop scheduler called job scheduler, whose objective is to improve average job response time of the Hadoop systems by making use of workload patterns that tunes is scheduling decisions among multiple users and for each single user as well. Here a self-scheduling policy is designed to schedule the jobs. Experimental results are shown by taking the Air Traffic data of the year 2005, which confirms the robustness and effectiveness of our solution.

Rest of the paper is structured as follows: section 2 provides an complete literature survey conducted. Section 3 provides the overview of the proposed scheduler. Section 4 gives the experimental results. Section 5 concludes the paper.

II. LITERATURE SURVEY

Apache Hadoop is becoming an important ideal and regular part of conversation in the coming months and years. Due to its open source enterprise valuable solution it is universally accepted in the industry area and as well as in academia. Hadoop is playing an significant role in statistical analysis and business intelligence in terms of simplicity and scalability.

Literature survey gives an overview of existing schedulers adopted by the Hadoop. In paper (1) author has proposed invented a new abstraction which is useful for the simple computations for those who want to process the data but hides the messy details of data distribution, concurrent data, fault tolerance and load balancing in a system. A huge amount of data which is more than terabytes of data which is running on thousand cluster of computers. By using this abstraction the user can run the thousands of MapReduce jobs on each day across the world.
proposed a scheduler called delay scheduler which increases the locality of data to improve the performance of fair scheduler. In this scheduler the task scheduler are delayed for a particular amount of time if that particular data is not local. The implementation of task level of this scheduler is merged with the proposed job-size scheduling scheme. To solve the conflicts occurs among the locality and fairness a simple algorithm called delay scheduler is proposed in this paper. When users submits the job, the tasks that are associated with this particular job are delayed when these task are not local. So the tasks which are not local have to wait for a longer time for getting executed. However this scheduler improved the performance of the fair scheduler but the local task has to wait and all the task level implementations is done based on the delayed tasks which are not local. The conflicts which are exists in the previously designed schedulers are overcome by this scheduler that are raised in between the locality and fairness.

With respect to paper (3) author designed a platform for going beneath the benchmarks for the performance evaluation of MapReduce. In this paper a vocabulary for determining MapReduce workloads is analyzed and examined by comparing two bearing MapReduce tasks. This scenario focused on realistic workloads for the performance evaluations. It enhances new cluster operators to predict the workload related resource bottlenecks and workload related options of MapReduce task schedulers. In this framework each user can realistically predict a workload patterns which are considered as a critical tasks for performing the more evaluation of jobs under a different scenarios by using the two important components of MapReduce tasks that are analyzed by using the vocabulary of determining MapReduce workloads. In paper (4) proposed the fair scheduler which provides resources equally among each users and performance isolation is provided among the users. In fair scheduler the job scheduling decisions are made without taking in to account workload pattern of various different users. In this scheduler private Hadoop cluster are given to each user by using which the user can start and experiments their jobs within the seconds and interactive queries can be made by using appropriate share cluster efficiency. In this scheduler the resources are shared in an equal manner without taking in to account the workload patterns of various different users. Some people apply for accessing the data which may be equal size and some other users can request for downloading if different sizes, sharing the resources equally results in wastage of resources. The key solution for this problem is to design an correct scheduling policy which allocates the resources based on the different workloads of users. In this scheduler when users apply their jobs at the same time the resources of equal distribution compete and results in low response time which is one of the drawback of this scheduler.

With respect to paper (5) the authors estimated the execution time of the task of running jobs by taking in to consideration of execution time of previously completed task of that particular job and all the slot numbers are calculated and the task execution of various jobs is estimated. In this scenario the complete information of each slot and the execution time of various jobs is tracked and information is updated based on the how many slots are available and how many jobs are finished their execution. The numbers of slots that are free are allocated to the next coming jobs based on the information that is collected. This information will helps in assigning a slots based on the different workloads of various users.

III. SYSTEM ARCHITECTURE

The novel scheduler called as job size scheduler invented in this project aims to improve the job response time of a hadoop system by utilizing the job size patterns to launch its scheduling schemes among number of users and as well as for each single user. The proposed novel job-size scheduler improves the job response time under the various workloads in the comparison with the already adopted schedulers by MapReduce framework such as Apache Hadoop as a standard scheduling discipline.

![Architecture Overview of Job-Scheduler](image)

**Fig. 1: Architecture Overview of Job-Scheduler.**

In the above figure, the users submits the job to the master node which consists of job tracker to breaks the larger job in to smaller pieces by using workload information collector. In this proposed scheduler scheduling is carried out at two different levels such as

A. **Level 1 Scheduling**

At tier one all the nodes which are having free resources are noticed. It calculates the slots share by using the information collected by job tracker based on how many slots are free.

B. **Level 2 Scheduling**

At level 2, using the information about number of slots available the scheduling policy is tuned among each and every single user to choose its appropriate scheduling scheme.

IV. EXPERIMENTAL RESULTS

Experimental results are shown by taking the Air Traffic data of the year 2005, which confirms the robustness and effectiveness of our solution.
Fig. 2: Performance Evaluation of Air Traffic Data.

In the above figure 2, a performance evaluation on air traffic data is conducted to show the performance of novel job-size scheduler compared to the current Hadoop scheduler. In this the performance analysis of air traffic is displayed and the data consumed reports are represented in the form of bars in which x-axis is assigned to data size and the time of execution has been assigned to y-axis.

Here the proposed scheduler is compared with the existing Fair scheduler which shows that, this novel scheduler improves the average job response time as compared to the previous schedulers used by the Hadoop.

V. CONCLUSION

In this paper we have proposed a novel job scheduler to improve the efficiency of Hadoop system that processes multiple MapReduce jobs. A new scheduling policy called self-tuning scheduling policy is designed for scheduling the Hadoop jobs which is carried out at two levels such as, the resources which are shared across the number of users are set based on the predicted job size of each user and the scheduling of job is again tuned to adopt that particular job size distribution. The experimental results are conducted by taking the Air Traffic data of the year 2005, to confirm the robustness and effectiveness of our solution.

REFERENCES