An Adaptive Scheduling Technique for Improving the Efficiency of Hadoop

Ms Punitha R
Computer Science Engineering
M.S Engineering College, Bangalore,
Karnataka, India.

Mr Malatesh S H
Computer Science Engineering
M.S Engineering College, Bangalore,
Karnataka, India.

***Abstract*− Hadoop is a framework of tools, it closely resembles the map reduce framework. MapReduce which is used for processing and generating large datasets, it is a programming model. MapReduce cluster which partitioned jobs into small tasks, when the flocks of jobs are submitted to a MapReduce it shares all the resources and checks the system performance in terms of jobs response times. Thus the challenge is the ability of efficient scheduling.**

We have many conventional scheduling algorithms supported by Hadoop these cannot always guarantee good average response time under different workloads in address of this we present an adoptive scheduling technique size-based optimization scheduler for MapReduce, multi-job workload that aims at improving efficiencies across the machines while observing completion time goals, under variety of workload compare to FIFO and Fair scheduling techniques.

Keywords: Hadoop, MapReduce, workload characteristics, scheduling, diverse workload.

I. INTRODUCTION

Hadoop is an open source implementation of map reduce, map reduce is now standard in industry and academia for processing large-scale datasets, such as yahoo, Facebook, Twitter, etc. to support batch processing for large jobs submitted from different users, (i.e. MapReduce workload). The input of MapReduce can be given as key value pairs and output as map functions [3]. There are 2 types of tasks in map reduce, one is map tasks another one is reduce tasks. Map reduce implemented as a parallel programming model can be classified on its simplicity, flexibility and ability to hide low-level issues such as scheduling, failures, data locality network bandwidth, machine and resource availability from the user. Despite of many studies in optimizing MapReduce/Hadoop, there are several key challenges.

The base Apache Hadoop frameworks composed of the following modules:

(1). Hadoop Common−contains libraries and utilities needed by other Hadoop modules.

(2). Hadoop Distributed File System (HDFS) −a distributed file system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.

(3). Hadoop YARN−a resource management platform responsible for managing compute resources in cluster and using them for scheduling of user's applications.

(4). Hadoop MapReduce−a programming model for large scale data processing.

All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are common thus should be automatically handled in software by the framework. For effective scheduling work, every Hadoop –compatible file system should provide location awareness. The data size in Hadoop can't be measured; data may be of variable size therefore, data cannot be predicted at early stage once the data is fetched as an input the data size

willbe predicted, else known.

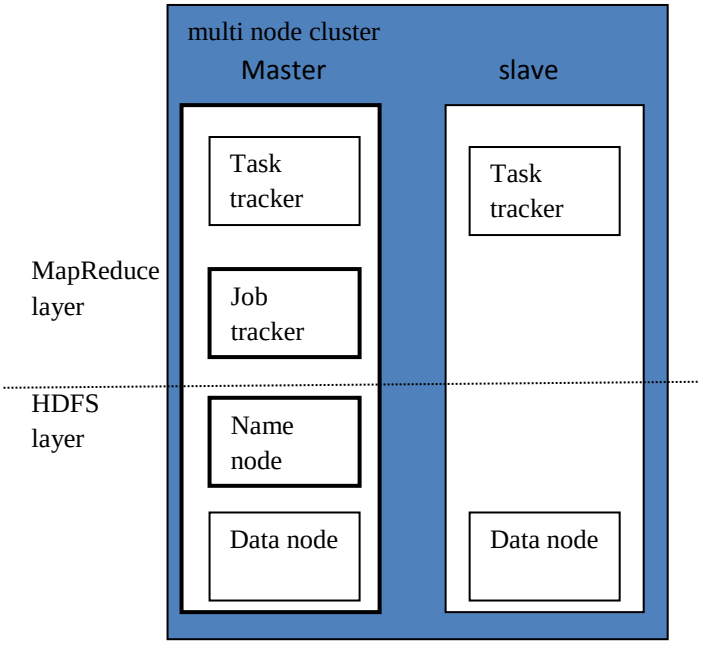

Fig.1:Multi node Hadoop cluster.

A small Hadoop cluster includes a single master and multiple worker nodes. The master node consists of a JobTracker, TaskTracker, NameNode and DataNode. A slave or worker node acts as both a DataNode and TaskTracker, though it is possible to have data-only worker nodes and compute only worker nodes. These are normally used only in non-standard applications.

MapReduce is a good fit for problems that can be solved by easily dividing a job into a number of smaller pieces, which can thus be solved independently; these methods is called as divide and conquer. MapReduce algorithm can go long way in improving the performance of particular cluster [5].

Hadoop is a framework of software tools and open source implementation of MapReduce provided by the Apache Software Foundation. The Hadoop architecture follows the master/slave method. The roles are performed by the 'JobTracker' and 'TaskTracker' processes, the work units are represented by 'tasks'. The tasks are controlled by the TaskTracker the resources of Hadoop are abstracted into typed slots [7]. In the TaskTracker the number of slots per each TaskTracker is determined the maximum number of concurrent tasks followed to run in worker machine. A slot is bound to a particular type of tasks. The default schedulers in Hadoop are FIFO (First-In First-Out) scheduling technique. More recently the fair scheduler assigns heavy-tailed and diverse workload jobs to 'pools' and then it guarantee a minimum number of slots to each pool.

II. RELATED WORK

A. Input and Output Types

The inputs provided for various MapReduce libraries which provide support for reading inputs in several different formats. For example, "text files", "sequential files", "string" mode of input which treats each line of input as key/value pair: the key is the offset in the file and the value is the contents of the line. Each input type implementation knows how to split itself into meaningful ranges for processing as a separate map tasks.

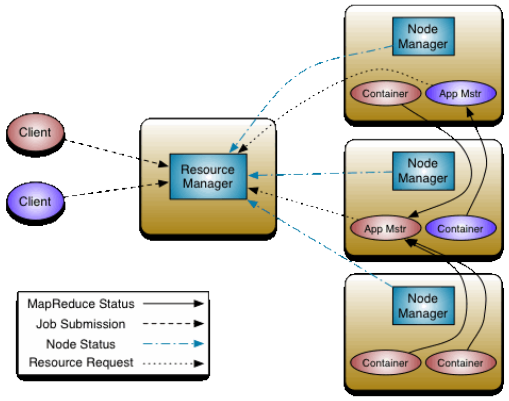

Fig. 2: The YARN architecture

YARN – Yet Another Resource Negotiator, which is the new implementation referred as MRv2. Implementation of Apache Hadoop. It addresses the issue of scalability, and performance issue based on Google original paper on MapReduce. It has Resource manager which keeps track of all the resources via CPU,memory etc.., Job submitted on Application master will have to negotiate with the MapReduce for resources to execute the task. Node manager acts as a controller. In essence, YARN performs well compared to MRv1.

B. Cluster Configuration

All of the programs are executed on a cluster that consisted of approximately 1800 machines. Each machine has 4GB of memory, two 160GB IDE disks, and a gigabit Ethernet link. The machines were arranged in a two-level tree-shaped switched network with approximately 100-200 Gbps of aggregate bandwidth, all the machines were in same hosting facility and therefore the round-trip time between any pair of machines were less than a millisecond

III.LITERATURE SURVEY

This section shows the existing system and its disadvantages and overview of the proposed system.

A. Existing System

Typically, multiple users compete for the available slots in a MapReduce cluster when these users concurrently submit jobs to the cluster. As a result, the average job response times is an important efficient performance concern in MapReduce systems, might be seriously degraded.We found that MapReduce workloads often exhibit the heavy-tailed (or long-tailed) workload characteristics [8]. Where a MapReduce clusterworkflow consists of few but extremely large jobs and many small jobs.Default scheduler, conventionally works on FIFO scheduling, which gives chances to first come first serve basis, which makes jobs with high priority to wait for long time if it happens to place at last in cluster. As job submission on cluster can happen concurrently.

B.Disadvantages of Existing System:

- This leads to poor average utilization.
- Serve diverse workloads since small jobs will experience extremely long waiting time when submitted after a large job.
- The Fair scheduler makes its scheduling decision without considering workload patterns of different users.

C.Proposed System

In this part of the paper, we represent a queuing model that is developed to emulate a Hadoop system. The main purpose of this model is to compare various Hadoop scheduling schemes, and give the proof of our new approach. This model does not include all the details of the complex Hadoop system, but provide a general guideline to clients which are useful for efficient performance evaluation.

We propose a novel Hadoop scheduler, called **size-based optimization scheduler,** which aims to improve the average job response time of Hadoop systems by leveraging the job size patterns to tune its scheduling schemes among the multi-job users and for each user as well. Specifically, we first develop a lightweight information collector that tracks statistic information of recently finished jobs from each user. A self-tuning scheduling policy is then designed to scheduler Hadoop jobs at two levels: the resource shares across multiple users are tuned based on the estimated job size of each user; and the job scheduling for each individual user is further adjusted to accommodate to that user's job size distribution.

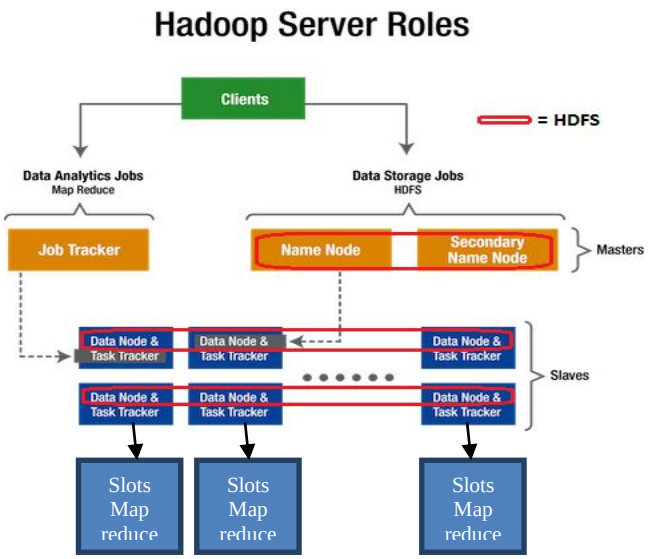

Fig. 3: Architecture of the model.

The user or client submits the jobs to the JobTracker in which the data analysis of jobs are done and user supplies jobs to the HDFS, which consists of JobTracker and TaskTacker in which the data is stored in HDFS. HDFS consists of Name Node and Data Node. Name Node and Secondary Name Node acts as master node and Data Node and TaskTracker acts as slave node. The TaskTracker divides jobs into slots by MapReduce functions, Map tasks will prefer to use map slots and reduce tasks prefer to use reduce slots [2].

D.Modules in Phase

1. Traffic Analytics
2. Custom M R
3. Size Base Scheduler
4. Performance Analysis

1.Traffic Analytics: Here we are using Data set 2005.Here Monthly air traffic analysis, and efficient response time can be calculated.

2. Custom MR: Air Traffic Run report based on data size modulated report. In our model, we consider to change the distributions of map/reduce task numbers for investigating various job size patterns, while fixing the uniform distribution to draw the execution times of map/reduce tasks.

3. Size Base Scheduler: The Number of Jobs which assigned to Hadoop. This Scheduler (size-based scheduler)assign the task based on the size with the Help of Job Tracker.

4. Performance Analysis: Comparison Analysis between Fair and LsPs based on data size and time. It gives more Efficient Performance than the Fair.

E. Advantages of Proposed System

This system can powerfully handle characteristic workloads in a corporate network.

- Size-based scheduler achieves non-negligible improvements of overall job response times no matter which user has smaller job sizes.
- Improves the efficiency of Hadoop systems that process heterogeneous MapReduce jobs.
- Improves the performance in terms of job response times under a variety of system workloads.
- There is a remarkable work done by the effectiveness & robustness under the non-stationary workloads.
- It can consistently improve the average job response times through dynamically adapting the scheduling to the workload changes.

IV. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the performance analysis of air traffic in flights. We first evaluate the individual impact of each optimization technique of job sized adaptive scheduling technique for improving the efficiency in Hadoop.

A. Experimental Setup

We run our experiments in a cluster consisting of 4 compute nodes, each with two Intel i5 processor, 40GB memory and 50GB hard disks, 8GB RAM. We configure one node as master and NameNode,and the other 3 nodes as slaves andDataNodes. The latest version of Hadoop 1.2.1 is chosen in our experiment.We generate our test bed workloads by choosing MapReduce Benchmark Suit and using some of the provided datasets such as Word Count, Grep, Sort, Pi Estimator, Inverted Index, Classification, Histogram Movies, Histogram Ratings, Sequence Count, and Self join, so on, where the input data sizes are chosen according to the capability of cluster. Ex. When we take real time system, in this variety of IT Company shares are the inputs and compare it with one company to the other that shares are the datasets used as input and outputs will be of variation between the shares from one Company to another.

B. Performance Evaluation

In this section, we first show the tasks execution processes for airtraffic.Then we evaluate and compare the performance improvement under different slot configuration. Third monthly airtraffic analysis through pie graph with the datasets information of2005 year and shows the variation between all months in a year. Fourth cancellation flight analysis in same dataset and variation between all months in a year.

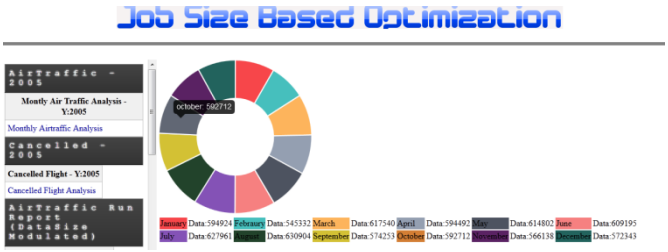

Fig.4: The monthly air traffic analysis of dataset 2005

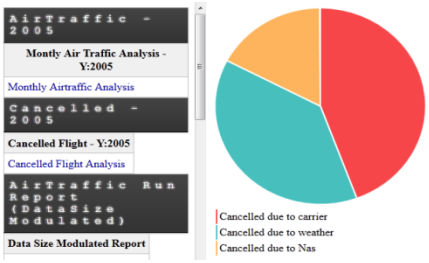

Fig.5: Cancellation of flights in dataset 2005.

The below fig shows the performance evaluation in air traffic with using the differentlogs, each logs contain same data but in variable data size. Fig 3 shows the data size over time.We evaluate and compare the performance variation between Fair and LsPS.

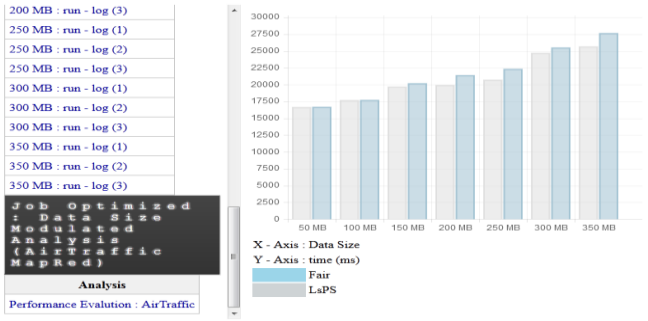

Fig. 6: Performance evaluation of air traffic.

V. CONCLUSION

We have proposed size-based optimization scheduler, an adaptive scheduling technique for improving the efficiency of Hadoop systems that process heterogeneous MapReduce jobs. MapReduce workloads of contemporary enterprise clients have revealed the diversity of job size. Our new online policy captures the present job size patterns of each user and leverages this knowledge to dynamically adjust the slot shares among all active users and to further on-the-fly tune the scheme for scheduling jobs within a

single user. Using real experiments in Amazon EC2. It reduces the overall job response times by a factor over FIFO and Fair, respectively.

Our real time experimental results from cloud cluster will show that our scheduler shows the traffic analysis, and reduces the average MapReduce job response time.

REFERENCES

[1] Chao Tian, Haojie Zhou, Yongqiang He, Li Zha, "A Dynamic MapReduce Scheduler for Heterogeneous Workload", APRIL 2009.
[2] Shanjiang Tang, Bu-Sung Lee, Bingsheng He "DynamicMR: A Dynamic Slot Allocation Optimization Framework for MapReduce Clusters".
[3] Benamin Moseley AnirbanDasgupta, Ravi Kumar, TamasSarlos, " On Scheduling in MapReduce and Flow-Shops" june [4-6]2011.
[4] Yanpei Chen, ArchanaGanapathi, Team Griffith, Randy Katz,"The Case for Evaluating Mapreduce Performance Using Workload Suites".
[5] Visalakshi P, Karthik T U, "MapReduce Scheduler Using Classifiers for Heterogeneous Workloads" IJCSNS International Journal of Computer Science and Network Security, VOL.11, No.4, April 2011.
[6] Jeffrey Dean and Sanjay Ghemawat, "MapReduce Simplified Data Processing on Large Clusters" OSDI 2004.
[7] Jorda Polo, Claris Castillo, David Carrera, Yalanda Becerra Iran Whalley, MalgorzataSteinder, Jordi Torres, and Eduard Ayguade,"Resource-aware Adaptive Scheduling for MapReduce Clusters".
[8] Yi Yao, Jianzhetai, Shang, NingfangMi, "LSPS: a Job Size-Based Scheduler for Efficient Task Management in Hadoop", IEEE 2013-2014.