IJIRMET ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 3, Issue 4, April 2018

Bringing Size Based Scheduling To Hadoop

[1] R.Sagunthala Devi, [2] Dr.Ar.Arunachalam,
[1] B.E., M.Tech Bharath University
[2] Professor & Head, Bharath University.

Abstract: ***Size-based scheduling with aging has been recognized as an effective approach to guarantee fairness and near optimal system response times.We present HFSP, a scheduler introducing this technique to a real, multi-server, complex and widely used system such as Hadoop.Size based scheduling on Hadoop is an effective approach to avoid starvation and minimize response time using hybrid scheduler.Size-based scheduling requires a priori job size information, which is not available in Hadoop.HFSP builds such knowledge by estimating it on-line during job execution.Size based scheduling in HFSP adopts the idea of giving priority to small jobs that they will not be slowed down by large ones.HFSP is a size based and preemptive scheduler for Hadoop. HFSP is largely fault tolerant and tolerant to job size estimation errors. The Scheduling decisions use the concept of virtual time and cluster resources are focused on jobs according to their priority, computed through aging. This protocol never faces Starvation Problem for small and large jobs.To minimize the response time the least remaining processing time will be calculated for each job.***

I. INTRODUCTION

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a Parallel and distributed computing environment. It makes Use of the commodity hardware Hadoop is Highly Scalable and Fault Tolerant. Hadoop runs in cluster and eliminates the use of a Super computer. Hadoop is the widely used big data processing engine with a simple master slave setup.Big Data in most companies are processed by Hadoop by submitting the jobs to Master. The Master distributes the job to its cluster and process map and reduces tasks sequencially.But nowadays the growing data need and the competition between Service Providers leads to the increased submission of jobs to the Master. This Concurrent job submission on Hadoop forces us to do Scheduling on Hadoop Cluster so that the response time will be acceptable for each job.Size-based scheduling with aging has been recognized as an effective approach to guarantee fairness and near optimal system response times. We present HFSP, a scheduler introducing this technique to a real, multi-server, complex and widely used system such as Hadoop.Size based scheduling on Hadoop is an effective approach to avoid starvation and minimize response time using hybrid scheduler.

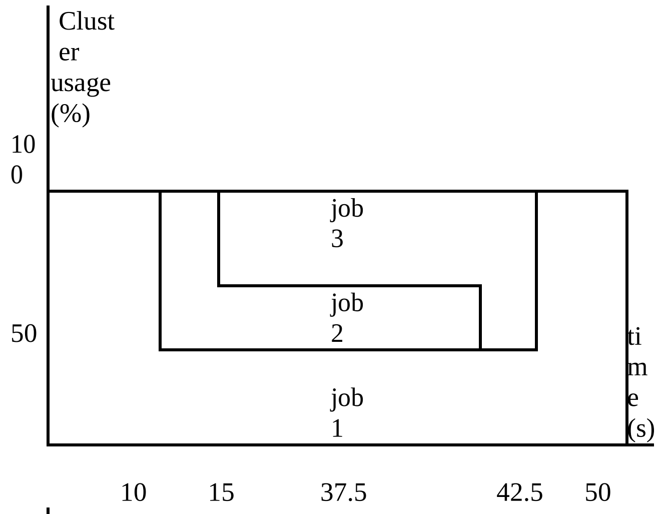
II. LITREATURE SURVEY

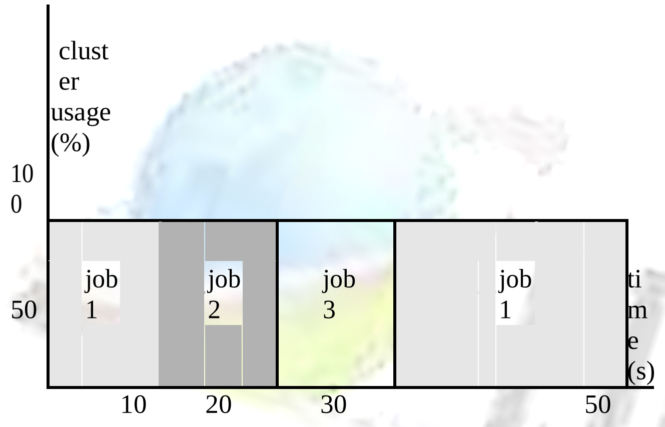
Traditional Scheduling

First Come First Serve (FCFS) and Processor Sharing (PS) are the 2 simplest scheduling methods used in many of systems. FIFO and Fair are two schedulers for Hadoop, the first inspired by FCFS method and the second by PS method.

In FCFS, jobs which are scheduled in submission order and thing in PS resources are divided equally among all, so that the job which is active keeps progressing. In loaded systems, these types of mentioned process have severe shortcomings that are one case is in FCFS, large running jobs can delay significantly small ones.

In PS, job delays are unpredictable and this additional delay of jobs affects the job completion of all the others. In order to improve the performance of the system in terms of delay, it is important to mind the size of jobs. Job scheduling based on size adopts the idea of giving priority to small jobs.





FAIR SCHEDULING

Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average, an equal share of resources over time. When there is a single job running, that job uses the entire cluster.

When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. Unlike the default Hadoop scheduler, which forms a queue of jobs, this lets short jobs finish in reasonable time while not starving long jobs.

It is also a reasonable way to share a cluster between a numbers of users. Finally, fair sharing can also work with job priorities - the priorities are used as weights to determine the fraction of total compute time that each job should get.

CAPACITY SCHEDULING

The Capacity Scheduler is designed to allow sharing a 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. There is an added benefit that an organization can access any excess capacity no being used by others. This provides elasticity for the organizations in a cost-effective manner.

HADOOP FAIR SOJOURN PROTOCOL

The Hadoop Fair Sojourn Protocol (HFSP) is a Job scheduling based on size with aging for Hadoop. Implementing HFSP raise number of challenges.

A few of them come from Map Reduce itself and the fact that a job is composed by tasks – while others come from the scheduler which is a size based in a context where the size of the jobs is not known a priori.

In this section we describe the challenges which have to face and the proposed solutions. In Map Reduce, jobs are scheduled according to the tasks which it handles, and they consist of two phases, called MAP and REDUCE. We evaluate the sizes of jobs by executing the subset of tasks for each job; however, REDUCE tasks can be launched only after the MAP phase is getting complete. Our scheduler thus divides the job logically in to two phases and treats them independent and individually so the scheduler assumes the job as consisting of two parts with two different sizes, one for the MAP and the other for the REDUCE phase.

When a resource is provided available for scheduling the MAP (resp. REDUCE) task, the scheduler sorts and evaluates jobs according to their specific virtual MAP (resp. REDUCE) sizes, and provides resources to the job with smallest size for that phase.

AGING MODULE

The aging module takes as input the estimated sizes to compute virtual sizes. The use of virtual sizes is a technique applied in many practical implementations of well-known schedulers it consists in keeping track of the amount of the remaining work for each job phase in a virtual “fair” system, and update it every time the scheduler is called. The output is that, although if the jobs not receive resources and thus its real size do not decrease, in the virtual system the job virtual size slowly decreases with time.

OBJECTIVES OF THE PROJECT

Size-based scheduling requires a priori job size information, which is not available in Hadoop: HFSP builds such knowledge by estimating it on-line during job execution.

Size-based scheduling with aging has been recognized as an effective approach to guarantee fairness and near optimal system response times. We present HFSP, a scheduler introducing this technique to a real, multi-server; complex and widely used system such as Hadoop

Hadoop is a free, Java-based programming framework that supports the processing of large data sets in a Parallel and distributed computing environment.

It makes Use of the commodity hardware Hadoop is Highly Scalable and Fault Tolerant. Hadoop runs in cluster and eliminates the use of a Super computer. Hadoop is the widely used big data processing engine with a simple master slave setup.

Big Data in most companies are processed by Hadoop by submitting the jobs to Master. The Master distributes the job to its cluster and process map and reduce tasks sequencially.But nowadays the growing data need and the and competition between Service Providers leads to the increased submission of jobs to the Master.

This Concurrent job submission on Hadoop forces us to do Scheduling on Hadoop Cluster so that the response time will be acceptable for each job.

III. EXISTING SYSTEM

There are mainly two different strategies used to schedule jobs in a cluster i.e. (PS & FCFS). The first strategy is to split the cluster resources equally among all the running jobs and this strategy in Hadoop is called Hadoop Fair Scheduler.

The second strategy is to serve one job at a time, thus avoiding the resource splitting. An example of this strategy is First-In-First-Out (FIFO), in which the job that arrived first is served first.

The problem with this strategy is that, being blind to job size, the scheduling choices lead inevitably to poor performance. Both strategies have drawbacks that prevent them from being used directly in production without precautions.

Commonly, a manual configuration of both the scheduler and the system parameters is required to overcome such drawbacks. This involves the manual setup of a number of pools to divide the resources to different job categories, and the fine-tuning of the parameters governing the resource allocation. This process is tedious, error prone, and cannot adapt easily to changes in the workload composition and cluster configuration.

In FCFS, jobs are scheduled in the order of their submission, while in PS resources are divided equally so that each active job keeps progressing. In loaded systems, these disciplines have severe shortcomings.

In FCFS, large running jobs can delay significantly small ones; in PS, each additional job delays the completion of all the others.

IV. PROPOSED SYSTEM

We present the design of a new scheduling protocol that caters both to a fair and efficient utilization of cluster resources, while striving to achieve short response times. Our solution implements a size-based, preemptive scheduling discipline.

The scheduler allocates cluster resources such that job size information is inferred while the job makes progress toward its completion. Scheduling decisions use the concept of virtual time and cluster resources are focused on jobs according to their priority, computed through aging.

This ensures that neither small nor large jobs suffer from starvation. The outcome of our work materializes as a full-fledged scheduler implementation that integrates seamlessly in Hadoop named HFSP.

The size based scheduling in HFSP adopts the idea of giving priority to small jobs that they will not be slowed down by large ones.

The Shortest Remaining Processing Time (SRPT) policy, which prioritizes jobs that need the least amount of work to complete, is the one that minimizes the mean response time (or sojourn time), that is the time that passes between a job submission and its completion.

We Extend HFSP to pause jobs with Higher SRPT and allow other waiting jobs in Queue based on FCFS.

LIST OF MODULES

- **Big Data and Environment**
- **Running a batch job through FCFS**
- **Size based scheduling on concurrent jobs**
- **Extending HFSP for job mistreatment**

MODULE DESCRIPTION:

BIG DATA AND ENVIRONMENT

Huge Collection of data is retrieved from open source datasets that are publicly available from major Application Providers like Amazon. Big Data Schemas were analyzed and a Working Rule of the Schema is determined. The CSV (Comma separated values) and TSV (Tab Separated Values) files are Stored in HDFS (Highly Distributed File System) and were read through Master and manipulated using Java API that itself developed by us which is developer friendly, light weighted and easily modifiable.

RUNNING A BATCH JOB THROUGH FCFS

A batch job is a backend job running in hadoop clusters and also called as long running jobs as it is scheduled to process bulk data so that the application would makes use of the results produced for updation.sample jobs are submitted to hadoop master and hadoop master will run the jobs based on a well known technique called First come first serve manner (FCFS).Parallel execution of job is done by hadoop cluster and the results are shown through a well known Framework called Map Reduce. The Mapper task is done first in slave nodes and reduce task will be done in Master to throw the output.

SIZE BASED SCHEDULING ON CONCURRENT JOBS:

Here n number of jobs are submitted to the Hadoop Master and Master will schedule the jobs based on FCFS and PS in a hybrid way.

The Capacity of cluster will be analyzed so as to share resources between concurrent jobs arriving to Master.

A threshold will be maintained to balance load in slaves and Resource scheduling will not be done further if limit is reached.

The Arriving jobs will put in queue until resource gets free in cluster.

EXTENDING HFSP FOR JOB MISTREATMENT i.e.STARVATION

As jobs may find long waiting time in queue, we extend our hybrid Approach which clubs FCFS and PS to put running jobs on hold for some time, if the particular job has high Shortest Remaining Processing Time (SRPT).

Depending upon aging of the waiting jobs and SRPT the long running jobs may be put on hold and the waiting jobs which have high priority will be executed for a while and constant evaluated for SRPT for new jobs to arrive for execution. Our Proposed methodology shows high throughput in job completion.

ALGORITHM USED

```
Function ASSIGNPHASETASKS(resurces)
   for all resource s € resources do
          if ∃ (Job in training stage) and $T_{curr}$ < T then
          job ← select job to train with smallest initial
                virtual size
      ASSIGN(s, job)
      $T_{curr}$ ← $T_{curr + 1}$

          else
              job ← select job with smallest virtual time


ASSIGN(s, job)
end if
end for
end function
  function ASSIGN(resource, job)
      task ← select task with lower ID from job
      assign task to resource
  end function
  function RELEASERESOURSE(task)
      if task is a training task then
    $T_{curr}$ ← $T_{curr - 1}$
     End if
  end function
```

V. CONCLUSION

Resource allocation plays an increasingly important role in current Hadoop clusters, as modern data analytics and workloads are becoming more complex and heterogeneous.

Our work was motivated by the increasing demand for system responsiveness, driven by both interactive data analysis tasks and long-running batch processing jobs, as well as for a fair and efficient allocation of system resources.

In this paper we presented a novel approach to the resource allocation problem, based on the idea of size-based scheduling. Our effort materialized in a full-fledged scheduler that we called HFSP, the Hadoop Fair Sojourn Protocol, which implements a size-based discipline that satisfies simultaneously system responsiveness and fairness requirements.

Our work raised many challenges: evaluating job sizes online without wasting resources, avoiding job starvation for both small and large jobs, and guaranteeing short response times despite estimation errors were the most noteworthy.

HFSP uses a simple and practical design: size estimation trades accuracy for speed, and starvation is largely alleviated, by introducing the mechanisms of virtual time and aging.

FUTURE ENHANCEMENT

Currently we are extending HFSP such that it can use recent job preemption primitives a necessary condition to allow even faster response time

We will consolidate our code base and contribute it to the Hadoop commodity, casting HFSP to work for modern frame work such as YARN and Meros.

REFERENCES

[1] Apache, "Hadoop: Open source implementation of MapReduce," http: //hadoop.apache.org/.
[2] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in Proc. of USENIX OSDI, 2004.
[3] Apache, "Spark," http://spark.apache.org/. [4] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, 2012, pp. 2–2.
[4] Microsoft, "The naiad system," https://github.com/ MicrosoftResearchSVC/naiad.
[5] D. G. Murray, F. McSherry, R. Isaacs, M. Isard, P. Barham, and M. Abadi, "Naiad: A timely dataflow system," in Proceedings of the 24th ACM Symposium on Operating Systems Principles, 2013, pp. 439– 455.