# SCHEDULING JOB QUEUE IN HADOOP

R. Jamroth Banu[1] , E. Jamuna[2] , S. Janani[3]

[1] *Student, Department of Computer science, Panimalar Engineering College, Tamil Nadu, India*
[2] *Student, Department of Computer science, Panimalar Engineering College, Tamil Nadu, India*
[3] *Student, Department of Computer science, Panimalar Engineering College, Tamil Nadu, India*

**ABSTRACT**

Size-*based scheduling has been recognized as an effective approach to guarantee fairness and near optimal system response times. HFSP is a scheduler used in this technique to a real, multi-server, complex and widely used system such as Hadoop. It is a widely used bigdata processing engine with a master slave setup. Our solution implements a size-based and pre-emptive scheduling discipline. Scheduling decisions use the concept of virtual time and cluster resource. The jobs are focused according to their priorities that are computed through aging. This ensures that neither small nor large jobs suffer from starvation. The Size based scheduling in HFSP, gives the priority to small jobs that they will not be slowed down by large ones. The Shortest Remaining Processing Time (SRPT) concept gives priority to the jobs that requires minimum amount of work to complete. It 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.*

**Keywords**: -*Map Reduce, SRPT, Performance, Scheduling.*

---

## 1. INTRODUCTION

The advent of large-scale data analytics, stimulated by parallel frameworks like Hadoop. Spark and Naiad has created the requirement to control the resources of compute clusters that operates in a shared Environment. The users within the same company, share the same cluster and hence this avoids redundancy in physical deployments and in data storage. It represent enormous cost savings. Initially it was built for very large batch processing jobs, data-intensive scalable computing frameworks such as Map Reduce. Nowadays map reduce are used by many companies for production, recurrent and experimental data analysis jobs.

The most important fact that emerges from previous works is that there exists a need for short system response time. The operations, such as data exploration, preliminary analyses, and algorithm tuning are involved in interactivity. In addition to this, the workflow schedulers such as Oozie contribute to workload heterogeneity by Generating a number of small "orchestration" jobs. There are many batch jobs running at same time that works on big datasets. These jobs are important part of the workloads, since they transform data into value. Due to the heterogeneity of the workload, it is significant to find the right trade-off in assigning the resources to interactive and batch jobs.

In our work, we address the problem of job scheduling that allocates the resources to a number of concurrent jobs. It focus on Hadoop the most widely adopted open-source implementation of Map Reduce.
At current there are two different strategies used to schedule jobs in a cluster. The first strategy is to split the cluster resources equally among all the running jobs. An example of this strategy is the Hadoop Fair Scheduler. This strategy preserves fairness among jobs, when the system is overloaded. It increases the response time of the jobs. The second strategy is to serve one job at a time and it avoids resource splitting. An example of this strategy is First-In-First-Out (FIFO), in which the job that arrived first is served first. The disadvantage with this strategy is being blind to job size, the scheduling choices lead inevitably to poor performance: small jobs may find large jobs in the queue, thus they may incur in response times that are disproportionate to their size. As a result, the interactivity is difficult to obtain. Both strategies have drawbacks that prevent them
From being used directly in production without any precautions. Commonly, a manual configuration of both the scheduler and the system parameters is required to overcome such drawbacks. It involves the manual setup of a number of pools that 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 our paper, we present the design of a new scheduling protocol that caters both fair and efficient utilization of cluster resources. While it strives to achieve short response times. This approach satisfies the interactivity requirements of small jobs and the performance requirements of large jobs. It can thus coexist in a cluster without requiring manual setups and complex tuning. Our technique automatically adapts to resources and workload dynamics.

Our solution implements a size-based, pre-emptive scheduling discipline. The scheduler allocates cluster resources such that job size information. It is inferred while the job makes progress toward its completion.
Scheduling decisions use the concept of virtual time, in which jobs make progress according to an aging function. The cluster resources are focused on jobs according to their priority. This ensures that neither small nor large jobs suffers from starvation. The result of our work materializes as a full-fledged scheduler implementation that integrates seamlessly in Hadoop. We call our scheduler HFSP, to acknowledge an influential work in the size based scheduling literature.

The contribution of our work can be summarized as follows:
(i) We design and implement the system architecture of HFSP, to estimate job sizes and a dynamic resource allocation mechanism that strives at efficient cluster utilization.
HFSP is available as an open-source project.
(ii)Our scheduling discipline is based on the concepts of virtual time and job aging. These two techniques are conceived to operate in a multi-server system, with tolerance to failures, scale-out upgrades, and multi-phase jobs.
(iii) Our results indicate that size-based scheduling is a realistic option for Hadoop clusters, because HFSP sustains even rough approximations of job sizes.
(iv)We perform an experimental campaign, where we compare the HFSP scheduler to a prominent scheduler used in production-level. For the experiments, we use Pig Mix, a standard benchmarking suite that performs real data analysis jobs. Our results show that HFSP represents a sensible choice for a variety of workloads, catering to fairness, interactivity and efficiency requirements.

## 2. RELATED WORK:

In general Map reduce and Hadoop have received considerable attention recently, both from the industry and academia. We focus on job scheduling, and we consider here the literature pertaining to this domain. Theoretical Approaches: Several theoretical works tackles the scheduling in multi-server systems, a recent example is the work by Moseley and Fox. These works are elegant and significant contributions to the domain. It provides performance

bounds and optimality results based on simplifying assumptions on the execution system. Some works provides approximability results that are applied to simplified models of Map Reduce. In contrast to this, we focus on the design and implementation of a scheduling mechanism taking into account all the details and Intricacies of a real system.

Fairness and QoS: most of the works take a system-based approach to scheduling on Map Reduce. For example, the Fair scheduler and its improvement with a delay scheduler is an important example to which we compare our results. Several other works focus on resource allocation and strive to achieve fairness across jobs, but it do not aim at optimizing response times.Sandholm and Lai studied the resource assignment problem through the lenses of a bidding system to achieve a dynamic priority system and then implemented quality of service for jobs. Kc and Anyanwu addressed the problem for scheduling jobs to meet the user-provided deadlines, but assumed job runtime to be an input to the scheduler. Flex is a proprietary Hadoop size-based scheduler. In Flex, fairness is defined as avoiding the job starvation and guaranteed by allocating a part of the cluster according to Hadoop's Fair scheduler; size-based scheduling. It is then performed only on the remaining set of nodes. In contrast to this, by using aging, our approach can guarantee more fairness while allocating all cluster resources to the highest priority job, and thus completing it as soon as possible. Job Size Estimation: Various recent approaches proposed the techniques to calculate the query sizes in recurring jobs. Agarwal et al report that recurring jobs are 40% of all those running in Bing's production servers. Our estimation module, works on-line with any job submitted to a Hadoop cluster, but it has been designed so that the estimator module can be easily plugged with the other mechanisms, benefitting from more advanced solutions. Complementary approaches: Task size skew is a problem for Map Reduce applications, since large tasks delay the completion of a whole job; skew makes job size estimation more and more difficult. The approach of Skew Tune greatly mitigates the issue of skew in task processing times with a plug-in module that integrates in Hadoop,
Which can be used in conjunction with HFSP. Tian et al. proposed a mechanism where IO-bound and CPU-bound jobs run concurrently, benefitting from the absence of conflicts on resources between them. We remark that in this case it is possible to benefit from size-based scheduling, as it can be applied separately on the IO- and CPU-bound queues. Tan et al. proposed a strategy to adaptively start the REDUCE phase in order to avoid starving jobs; also this technique is orthogonal to the rest of scheduling choices and can be integrated in our approach. Hadoop offered a Capacity Scheduler which is designed to be operated in multitenant clusters where different organizations submit jobs to the same clusters in separate queues, obtaining a guaranteed amount of resources. We remark this idea is complementary to our proposal, since jobs in each queue could be
Scheduled according to a size-based policy such as HFSP, and reap according benefits. Framework Schedulers: Recent works have said the idea of sharing cluster resources at the framework level, for example to enable Map Reduce and Spark applications to run concurrently. Monolithic schedulers such as YARN and Omega used a single component to allocate resources to each framework, while two-level schedulers have
a single manager that negotiates resources with independent framework-specific schedulers. We believe that such framework schedulers imposes no conceptual barriers for the size-based scheduling, but the implementation would require a very careful engineering. In particular, the size-based scheduling should only
be limited to batch applications rather than streaming or interactive ones that require continuous progress.

## 3. EXISTING METHODOLOGY

There are two important different strategies that are used to schedule jobs in a cluster i.e. (PS & FCFS). The work of 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 work of the second strategy is to serve one job at a time, and thus avoiding the resource splitting. An example of this strategy is First-In-First-Out (FIFO), in which the job that are arrived first is served first. The problem with this strategy is that, being very blind to job size, the scheduling choices lead inevitably to the poor performance. Both the strategies have drawbacks that prevent them from being used directly in production without any precautions.

Commonly, a manual configuration of both the scheduler and the system parameters is required to overcome the above mentioned drawbacks. This involves the manual setup of a number of pools that divides all the resources to the different job categories, and the fine-tuning of the parameters governs the resource allocation. This process is difficult, error prone, and cannot adapt easily to all the changes in the workload composition and cluster configuration.

In FCFS, all the jobs are scheduled in the order of their submission, while in PS resources are divided equally so that each active job always keeps progressing. In loaded systems, these disciplines have more shortcomings: in FCFS, large running jobs can delay significantly the small running jobs, in Process sharing each additional job delays the completion of all the other jobs.

## 4. PROBLEMS IN THE EXISTIMG SYSTEM

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

It is blind to job size, the scheduling choices lead to poor performance.

The response time becomes lower, when the number of process get increased.

## 5. PROPOSED WORK

In this proposed work we present the design and the idea of a new scheduling protocol that caters both to a fair and efficient utilization of the cluster resources, while striving to achieve the short response times. Our solution implements the size-based pre-emptive scheduling discipline. The scheduler allocates the cluster resources so the job size information is inferred while the job makes progress toward its completion. Scheduling decisions use the concept and idea of virtual time and cluster resources are focused on the jobs according to their priority, computed through aging. This ensures that neither small nor large jobs suffer from starvation problem. The outcome of our work materializes as a full-fledged scheduler implementation that integrates in Hadoop named HFSP.Size based scheduling in HFSP uses the idea of giving priority to the small jobs and that they will not be slowed down by the large jobs. The Shortest Remaining Processing Time (SRPT) policy, which prioritizes jobs that need the minimum amount of work to complete, is the one that minimizes the mean response time that is the time that passes between a job submission and its completion. We extend the HFSP to pause jobs with Higher SRPT and allow other waiting jobs in Queue based on FCFS.

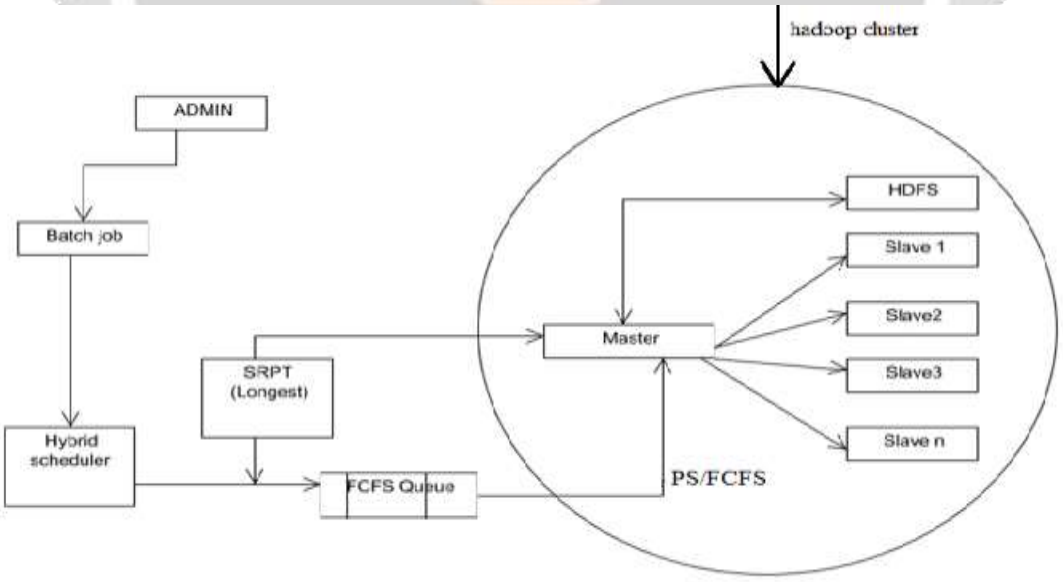


**Fig -1** Architecture diagram of proposed work

### 5.1. MODULES

#### 5.1.1. 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.

#### 5.1.2. 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.

#### 5.1.3. 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.

#### 5.1.4. EXTENDING HFSP FOR JOB MISTREATMENT ie.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.

## 6. RESULTS AND DISCUSSIONS

Our system 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.
It 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.The enhancement of our work is Hybrid Approach for Scheduling.

## 7. CONCLUSION

Resource allocation plays a very important role in current Hadoop clusters, as modern data analytics and workloads are becoming more and more complex and heterogeneous. Our work was motivated by the increasing demand for the system response, driven by both interactive data analysis tasks and the long-running batch processing jobs, as well as for a fair and efficient allocation of the system resources. In this paper we presented a novel approach to the resource allocation problem, based on the idea of the size-based scheduling. Our effort is to materialize in a full-fledged scheduler that we called HFSP, the Hadoop Fair Sojourn Protocol,

which implements a size-based discipline that satisfies simultaneously system response and the fairness requirements.
Our work raised many challenges: evaluating job sizes online without wasting the resources, avoiding the job starvation for both small jobs and large jobs, and guaranteeing short response time despite estimation error. HFSP uses a simple and practical design: size estimation trades accuracy for speed, and starvation is largely alleviated, by introducing the mechanisms of the virtual time and aging.

## 8. REFERENCES

[1]. HFSP: Bringing Size-Based Scheduling To Hadoop
Mario Pastorelli, Damiano Carra, Member, IEEE, Matteo Dell'Amico and
Pietro Michiardi, November 26, 2014

[2] Map Reduce: simplified data processing on large clusters Jeffrey Dean, Sanjay Ghemawat, Communications of the ACM, 2008

[3] Map Reduce optimization using regulated dynamic prioritization
Thomas sandholm, kelvin lai, 2009 ACM SIGMETRICS Performance
Evaluation Review.

[4] Scheduling Hadoop Jobs to Meet Deadlines
Kamal Kc, Kemafor Anyanwu, 2010 IEEE Second International Conference on
Cloud Computing Technology and Science.

[5] Fairness and scheduling in single server queues
Adam Wierman , 2011 Surveys in Operations Research and Management

[6] Dynamic Proportional Share Scheduling in Hadoop
Thomas Sandholm, Kevin Lai, 2010 15th International Workshop

[7] Performance analysis of Coupling Scheduler for Map Reduce/Hadoop
Jian Tan, Xiaoqiao Meng, Li Zhang, INFOCOM, 2012 Proceedings IEEE

[8] Effective Straggler Mitigation: Attack of the Clones
Ganesh Ananthanarayan, Ali Ghodsi, Scott Shenker, I Stoica - NSDI, 2013

[9] Revisiting Size-Based Scheduling with Estimated Job Sizes
Matteo Dell' Amico, Damiano Carra, Mario Pastorelli, Pietro Michiardi ,
Modelling, Analysis & Simulation of Computer and Telecommunication
Systems (MASCOTS), 2014 IEEE 22nd International Symposium

[10] Challenges for Map Reduce in Big Data
Katarina Grolinger, Michael Hayes, Wilson A. Higashino
Alexandra L' Heureux, David S. Allison, Miriam A.M. Capretz,
Services (SERVICES), 2014