Scientific Journal of Impact Factor(SJIF): 3.134

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 1,Issue 1, December -2013

HFSP: Bringing Size-Based Scheduling To Hadoop

Minal Sable[1], Tanmayee Sathe[2] , Vibhawari Jawale[3], Aashish Ramdasi[4]

[1]*Department of Computer Engineering, SAOE(Kondhwa), minalsable5@gmail.com*
[2]*Department of Computer Engineering, SAOE(Kondhwa), sathe_tanmayee@yahoo.in*
[3]*Department of Computer Engineering, SAOE(Kondhwa), vibhawari.jawale05@gmail.com*
[3]*Department of Computer Engineering, SAOE(Kondhwa), apramdasi.sae@sinhagad.edu*

Abstract — *Size-based scheduling is becoming older day by day, it has been recognized as an powerfull approach to assure fairness and near optimal system response time.We introduce HFSP, a scheduler acquainting this technique to Hadoop which real, multi-server, complex and widely used system.*

Initial job information is needed in sized based scheduling, which is not available in hadoop. HFSP develops such information by evaluating it on-line during job execution.

Our experiments, which are based on realistic workloads generated via standard benchmarking suite, recognizes a significant decrease in a system response time by using Hadoop Fair Scheduler, and reflects that HFSP is largely tolerant to job size estimation.

Keywords- *MapReduce, Performance, Data Analysis ,Scheduling*

I. INTRODUCTION

The arrival of large scale data analytics, encouraged by parallel processing framework such as MapReduce, has created a need to control the resources to compute clusters that function in a shared , multi-occupant environment. In the same company, many users share the same cluster to avoid redundancy and may present extent cost saving. Data-intensive scalable computing framework such as MapReduce was initially designed for very large batch processing jobs. Now-a-days MapReduce is used by many companies for production , repetitive and even experimental data analysis jobs. This diversity is approved by recent studies that examine a variety of production-level workloads.

From previous work, an important fact arises that there is a strict need for short response time. Data research , introductory study and algorithm tuning on small datasets often involve interactivity, in the that there is a human involved in loop pursuing answers with a trial and error process. In addition, Oozie a workflow scheduler contribute to workload diversity by generating a number of small “orchestration” jobs. Even if larger production jobs are in execution, interactive and “orchestration” jobs should not wait for long before being served. The task of cluster administrator involves the manual setup of a number of “pools” to allocate resources to different job categories and fine tuning of parameters that look after the resources allocation. This process is tedious, prone to error and it cannot adapt easily to changes in the workload compositon.

II. EXISTING SYSTEM

A. Scheduling:-

First Come First Serve(FCFS) and Processor Sharing(PS) are two most simple and universal scheduling disciplines used in many systems. For example, the FIFO and Fair Scheduler motivated by these two approaches. In First Come First Serve(FCFS), job scheduling is done in the order of their submission. In Processor Sharing(PS) division of resources is done evenly so that each active job keeps progressing. The above disciplines have served shortcoming in loaded system: in FCFS, large running jobs can delay very significantly small ones that are waiting to be executed. In PS, each additional job delays the completion of all the other jobs.

Size-based scheduling adapts the idea of giving priority to small jobs so that they will not delay the larger ones. The Shortest Remaining Processing Time(SRPT) policy, which gives priority to jobs that need the least amount of work to complete. SRPT is the one that minimizes the mean sojourn time or response time, that is the time that passes between a job submission and its completion. Policies like SRPT may drw in starvation ie. If smaller jobs are continuously submitted, larger ones may never get scheduled.

B. Comparison between PS and SRPT:-

The figure below compares PS and SRPT scheduling discipline with an example. Consider three jobs-j1,j2 and j3,where j2 and j3 small jobs which are submitted while large job j1 is running. In PS the three jobs j1,j2 and j3 run slow and parallel. While in SRPT, j1 is preempted and j2 & j3 are completed earlier. It is worth noting that, the completion time of j1 does not suffer for preemption, somewhat counter to instinct, this is often case for SRPT.

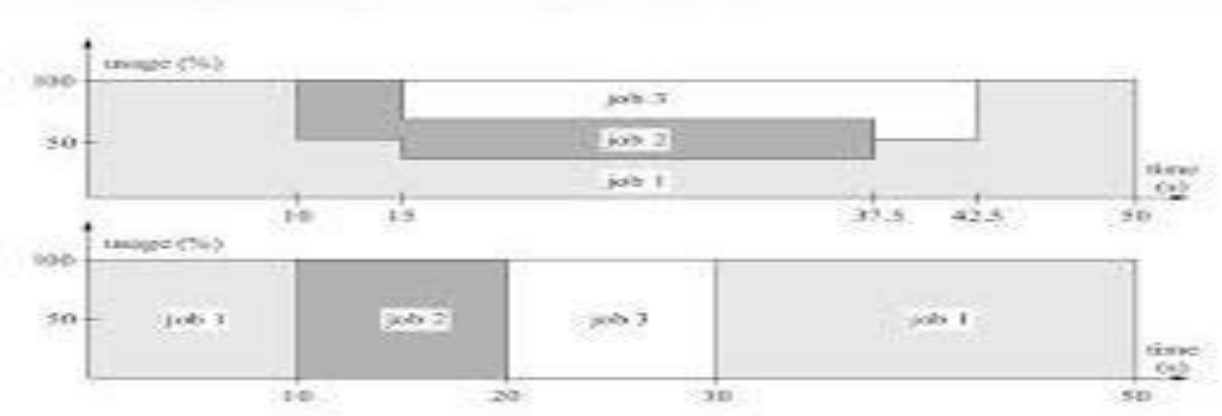

Fig.2 Comparison between PS and SRPT

III. PROPOSED SYSTEM

In Proposed System we presented an 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.

The contribution of our work can be summarized as follows:

• We design and implement the system architecture of HFSP , including a component to estimate job sizes and a dynamic resource allocation mechanism that strives at efficient cluster utilization.

• Our scheduling discipline is based on the concepts of virtual time and job aging. These techniques are conceived to operate in a multi-server system, with tolerance to failures, scale-out upgrades, and multi-phase jobs – a peculiarity of MapReduce.

• We reason about the implications of job sizes not being available a-priori from an abstract. Our results indicate that size-based scheduling is a realistic option for Hadoop clusters, because HFSP sustains even rough approximations of job sizes.

A.Algorithm 1 HFSP resource scheduling for a job phase.

```
function ASSIGNPHASETASKS(resources)
for all resource s 2 resources do
if 9 (Job in training stage) and Tcurr < T then
job   select job to train with smallest initial
virtual size
ASSIGN(s, job)
Tcurr  = Tcurr + 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 RELEASERESOURCE(task)
if task is a training task then
Tcurr  = Tcurr – 1
```


end if
end function

B.Scheduling Algorithm:

1)The procedure AssignPhaseTasks is responsible for assigning tasks for a certain phase. First, it checks if there are jobs in training stage for that phase.

2) If there are any, and the number of current resources used for training tasks Tcurr is smaller or equal than T, the scheduler assigns the resource to the first training task of the smallest job. Otherwise, the scheduler assigns the resource to the job with the smallest virtual time.

3)When a task finishes its work, the procedure releaseResource is called. If the task is a training task, then the number Tcurr of training slots in use is decreased by one.

IV. CONCLUSION

Realizing that MapReduce has advanced to the point where shared clusters are used for larger workloads which include non-insignificant fraction of interactive data processing task has motivated our work.As a result we have seen the raise of deployment based practices in which long response times due to fair sharing of resources among competing jobs compensated by over-dimensioned hadoop cluster.

To overcome previous work limitations we study the benefits of new scheduling discipline the targets at the same time short response times and fairness among jobs. Thus, we have proposed size based approach to scheduling jobs in hadoop, we call it as HFSP.

Our work brought up several challenges: evaluating job size on-line without wasting resources, avoiding job starvation both on small and large jobs, and guaranteeing short sojourn time despite estimation errors were the most noteworthy. We solved these problems in the context of a multi-server system using virtual time and aging, that is built to be tolerant to failures, scale-out upgrades, and supports the composite job structure of MapReduce.

REFERENCES

1. A. Verma, L. Cherkasova, and R. H. Campbell, “Aria: automatic resource inference and allocation for MapReduce environments,” in *Proc. Of ICAC*, 2011.

2.“Two sides of a coin: Optimizing the schedule of MapReduce jobs to minimize their makespan and improve cluster performance,” in *Proc. of IEEE MASCOTS*, 2012.

3. M. Zaharia *et al.*, “Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling,” in *Proc. of ACM EuroSys*, 2010.

4. Y. Chen, A. Ganapathi, R.Griffith, and R. Katz, “The case for evaluating MapReduce performance using workload suites,” in *Proc. of IEEE MASCOTS*, 2011.

5. H. Chang *et al.*, “Scheduling in MapReduce-like systems for fast completion time,” in *Proc. of IEEE INFOCOM*, 2011.

6. “Performance analysis of coupling scheduler for MapReduce/ Hadoop,” in *Proc. of IEEE INFOCOM*, 2012.

