International Journal of Natural and Engineering Sciences 9 (2): 58-61, 2015
ISSN: 1307-1149, E-ISSN: 2146-0086, www.nobel.gen.tr

(PAMRS H-PSFP) Performance Analysis of Map Reduce Scheduler's with H-PFSP

Ahmed MATEEN[1*] Rabia ZAFAR[1]
[1]Department of Computer Sciences, University of Agriculture Faisalabad, Pakistan

[*] Corresponding author:
Email: ahmedmatin@hotmail.com

Received: May 21, 2015
Accepted: 03 July, 2015

Abstract

Map Reduce offers a data parallel programming, functional model and computing framework that is being used for processing of big data. Data processing are developed using open-source Hadoop implementation of map reduce. Scheduler is main module of Map Reduce which runs on master server by performing centralized control of Map Reduce cluster. We analyse FIFO Scheduling enhancing it with Hybrid Parallel pessimistic Fair Schedule Protocol (H-PFSP) by using a statistical technique called EWMA (exponential weighted moving average) and conclude that H-PFSP scheduler can integrate it in fair scheduler and hence every job can be scheduled by H-PFSP in every pool. During comparison of H-PFSP and Fair scheduler, function to analyze that Fair scheduler has poor performance than H-PFSP in most cases The most prominent feature of H-PFSP is that it can close each job before Fair scheduler does. Contrasting Fair scheduler, with moderately precise job progress estimation H-PFSP can progress the per-job performance of Map Reduce systems.

Keywords: Fifo, map reduce, h-pfsp, hadoop

INTRODUCTION

Hadoop was designed for large batch jobs [1]. Hadoop enables the distributed processing of large data sets across cluster of commodity servers and offer an open source software scheme [2]. The basis of its design is to expand from an only server to hundreds of machines with an overload degree of fault tolerance and flexibility of these clusters comes from the capacity to identify and handle any damage at an application layer.

It delivers reliable shared storage through HDFS (Hadoop distributed file system) and data analysis using map reduce [3].

In this study we have exposed disadvantages of existing schedulers that were used in Hadoop and their mean flow calculation. We have reviewed that existing schedulers of map reduce behave poorly and mortify the through put and response time [4].

Because map reduce nature is to handle big data so scheduling task is considered one of the main issue that comes in map reduce framework [5].

RELATED WORK

Map Reduce

Map Reduce is a programming model used in hadoop frame work for processing of big data which rests on functional programming. It expanded to process huge volume of data on its clusters at Google. Map reduce is used for unstructured data in batch oriented processing by using parallel algorithm. The model is appropriate for web indexing, data mining, simulation and few scientific calculations which also called inherently parallel application. Map reduce has the capacity to handle the algorithm for big data processing.

Map Reduce Scheduling

Map reduce scheduling is unlike traditional scheduling because of data locality requirement or placing map task. Map reduces feature fault tolerance by restarting failed tasks; while parallelization and distribution are automated. Hadoop provides status and monitoring tools.

In Heterogeneous environment Map Reduce job execution is volatile. The code of Map Reduce is to split job in to independent task using multiple resources for parallelize job execution. In current scenario, Processor scheduler's implementation shapes into additional form of fair scheduling whose running resembles Round Robin. Map Reduce uses a block-level runtime scheduling with a speculative execution. [6]Map reduce has the capacity to handle the algorithm for big data processing.

ANALYSIS

Problem in Existing schedulers of Hadoop

Hadoop's primary scheduler executes jobs in FIFO order, many other schedulers also used in Hadoop. There tabular representation is as under

Table 1: comparison of scheduler

Serial Number	Map reduce Scheduler		
	Name	**Developed by**	**Working**
1	FIFO SCHEDULER	Hadoop default scheduler	Inappropriate for Processing of small jobs Non-Pre-emptive
2	FAIR SCHEDULER	Face book	It eradicates the short coming of FIFO scheduler. It justify equal share of all the resources all the time. Pre-emptive
4	CAPACITY SCHEDULAR	Yahoo	while maximizing the throughput and utilization of the cluster while running map reduce application. Non-Pre-emptive
5	LATE(longest approximate time to end	Face book	Pre-emptive It used for map reduce in heterogeneous environment
6	WAITED SCHEDULER	Google	Pre-emptive

The disadvantage of FIFO scheduling is poor response times for short jobs in the presence of large jobs. It is also not suitable for multiple job scheduling.

Easy scheduling policies like First in first out and Processor sharing are not priority-based policies. FIFO scheduler has also Convoy effect i.e. shortest job behind the longest job. FIFO scheduler like mean response time grant low performance inverse to those policies like priority based which optimize mean response time just like shortest process response time (SPRT) does, and provides service differentiation [7]. Furthermore Priority based scheduling can also leads to starvation. However, acceptance of SRPT (shortest remaining process time) has been delayed by troublesome fears about unfairness: Shortest remaining time is advantageous because short processes are handled very quickly [8] .This is the policy which always gives priority to small job and seems unfair for larger job.

Fair scheduler is used to assign resources to a job so that each hob get on average an equal share of resources over time which behaves opposite to default Hadoop scheduler which is FIFO scheduler[9]. Fair scheduler addresses the problem with a FIFO scheduler when short jobs have to wait for long running jobs to complete [10]. Yahoo! developed a multi-queue scheduler called

Capacity Scheduler [11]. It has same functionality as fair scheduler except we define a certain limit to slots for assigning of jobs [12]. In Delay scheduling if a node request a task and head of line job unable to start on a local task it is skipped and looked at a following jobs until we wait for free of other slots for processing of existing job [13].

Fair scheduling is like a processor sharing (PS) type of scheduler which in multiuser environment mortify the per job performance Fair scheduler works by means of polling. In all slot it nepotism fair allotment. it make sure that the work is done fairly. In real the Fair algorithm broken down the cluster in to slices and real resources will take only a portion of a cluster for each job. Hence, when it takes over the resources then the response time of every job will be too delayed. Fair scheduling approximate the status of current job by using the technique called historical job profiling. Hence, the server performance may vary due to back ground work load and failure. Historical execution information is complex for calculation of estimated flow time, Because of multiplicity of job and volatility of cluster environment. It should be updated on run time execution.

EXPERİMENTAL DESIGN

Scheduling Optimization

During analysis of H-PFSP, Map Reduce system contains a hybrid protective scheduler. It also favors multiuser environments on a cloud computing platform or inside a company. HPFS makes scheduling decision on the base of the parameter called estimated job execution time. An inaccurate estimation may make all scheduling unacceptable or may debase the whole performance. Thus, to attain reasonable precision, we use run time approach I.e. to confine the status of clusters and running job it updates the estimates at interval during the running process. Present job by self will be used for the historical execution information.

H-PFSP Design and Algorithm

The mechanism of H-PFSP scheduler consists of following:-

H-PFSP Design

We embed H-PFSP scheduler into implemented fair scheduler to job in every group can be scheduled using H-PFSP.H-PFSP has beaten the performance of fair scheduler in most set up.

H-PFSP comparator decides that a scheduler assign a task when it will get an idle slot. H-PFSP comparators forever choose the task initially in fair phase. If greater than single job is present, scheduler chooses a job with smallest number of task scheduled and with SRPT under H-PFSP.

H-PFSP and PS scheduling

H-PFSP scheduling used PS built in scheduling mechanism to renew and compute the slack system. Consequently an real job position in both H-PFSP and PS are needed to be updated. A major issue exist when we execute large data sets on a map reduce frameworks which follow FIFO scheduling which is unfair for short jobs because in presence of large jobs it takes hours and day to complete [14].

A mechanism of PS scheduling priority queue is established. Once we submit a job, in the queue we place virtual copy of this job. If same copies of job received we processed the job on first come basis. According to the lasting processing time it order the job copies under processing scheduler.

Slack scheme

During analysis of slack system in order structure for each job. Under PS and H-PFSP It consisted of job residual time as well as slack value of the job. To reduce to bare bones of the calculation of the slack system; Storage and recalculations at each interval is needed for supporting estimates. These estimates will contain the information of standard time of Map and Reduce tasks and recent development of Map and Reduce phases

Re-evaluation and slack scheme update

Updating has to be made under slack system every now and then. It will revise the slack system with the evaluation of the execution progress and lasting point in synchronous mode.

H-PFSP algorithm

In this study H-PFSP algorithm is main core of our proposed model which is implemented by H-PFSP comparator. It will search for an idle slot and instruct a scheduler to assign a task. It always follows the policy of fair phase first for selection of job. If greater than one job found than scheduler chooses with shortest remaining process time less than H-PFSP and one with the smallest number of job scheduled the purpose of this Fair Phase is to find out if a job is in its initial stages. Under two conditions we can identify that a task is in fair phase. If a map progress is less than 0.95 then the stage will be considered as map stage otherwise it considered as reduced stage.

Design and Evaluation

Fair scheduler can provide platform to fix the H-PFSP implementation. We can use H-PFSP and fair scheduler's jointly to distribute the resources between different pools. The code is an autonomous connecting unit. Thus the previous scheduling policies i.e. approximate execution policies and delay scheduling are still present.

An outcome for mean flow time

We analyze that with a smaller number of task there exist no difference between fair scheduler and H-PFSP and core responsibility of H-PFSP is to lessen the average flow time of job of a system similar to map reduce without dejection fairness.

The result shown that the large FAIRLIMIT can degrade the performance. Hence moderately higher value of FAIRLIMIT is suggested especially when the environment is volatile or each split tasks is too small.

Outcomes for mean slowdown

Following are results that came upfront for mean slow down I.e. total processing time and total waiting time, graphical representation is as follow:

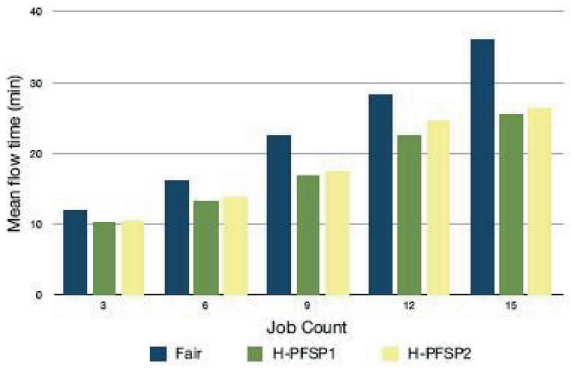


Figure 1. Total Processing time and total waiting time of Fair scheduling Vs H-PFSP Vs Job count

Table 2. Total Processing time and total waiting time of Fair scheduling Vs H-PFSP Vs Job count

Schedulers'	Job count	Per job flow time
Fair Scheduler	3,6,9,12	12,16,23,28
H-PFSP 1	3,6,9,12	10,13,16,22
H-PFSP-2	3,6,9,12	10,14,17,25

In H-PFSP, if we set two different FAIR LIMIT values i.e. H-PFSP1 sets FAIR LIMIT equal to 2 and H-PFSP2 sets a value greater than 5.The analysis of results showed that as much we mark a lower value it will increases the performance. Instinctively, the performance of H-PFSP2 with higher value it lies between H-PFSP1 and Fair scheduling policy.

Further detail is as follow

- H-PFSP has smaller mean slowdown.
- If we set the larger fair limit. It makes the scheduling algorithm inefficient because of multiple job run in fair phase
- Randomness to the execution time of each task is included.
- Normal distribution will be supposed
- The mean slowdown time first reduce due to larger FAIR LIMIT and system with longer FAIR phase can build more exact estimation.

The multi job scheduler due to starvation burdensome the FIFO scheduling i.e if a multi-tasking system always switches between the first two tasks while a third never gets time to run, then the third task is being famished of CPU time. Fair scheduler has very important feature of polling. Which assure fairness at each moment and assure fair allotment to every slot.. Hence when the resources will monopolize then the reaction time of every job will be greater than before. [15]

CONCLUSION

Therefore we analyse Map Reduce FIFO Scheduler by enhancing it with H-PFSP. H-PFSP performs every task effectively as compare to Fair scheduler. The map reduce system highlight the per job performance with moderately exact job progress estimation. During running process it updates the estimates at intervals to record the current status of running job and cluster to calculate expected flow time by historical execution information is difficult due to the variety of job and unpredictability of cluster environment. It should be updated on run time execution. H-PFSP connecting module Hadoop is capable of to be used in a scheduling pool. The outcomes show that HPFSP without forfeiting the fairness lessens the average flow instance of jobs respectively to the job size and number

REFERENCES

[1] https://people.csail.mit.edu/matei/talks/2009/msr_mapreduce_scheduling.pdf|(last access 10-12-2015)

[2] Apache hadoop. [Online]. Available: http://hadoop.apache.org(last acces 10-12-2015)

[3] http://ce.sysu.edu.cn/hope/UploadFiles/Education/2011/10/201110221516245419.pdf (last access 20-12-2015)

[4] http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-55.html(last access:02.11.2015)

[5] ALQudah, O. ; Jararweh, Y. ; Yaseen, Q.,2014, Multi-threading based Map Reduce tasks scheduling, Information and Communication Systems (ICICS), 2014 5th International Conference on ,pp 1-6

[6] Elghoneimy, E. , Bouhali, O. . Alnuweiri, H. 2012, Resource Allocation and scheduling in Cloud Computing. , IEEE workshop on computing, networking and communication,pp.309-314

[7] Lee KH, Hyunsik C, Bongki M.2011.Parallel data processing with map reduces,ACM Volume 40, no 4,pp. 11-20

[8] https://en.wikipedia.org/wiki/Shortest_remaining_time(last access 10-12-2015)

[9] https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html(last access 10-12-2015)

[10] Simon, T. ; Halem, M. ; Chapman, D.,2012, A Hybrid Scheduling Algorithm for Data Intensive Workloads in a Map Reduce Environment, Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on ,pp 161-167

[11] Capacity Scheduler http://hadoop.apache.org/common/docs/r0.19.2/capacity_scheduler.html(last access 11-12-2015)

[12] http://blog.cloudera.com/blog/2008/11/job-scheduling-in-hadoop/(last access 05-12-2015)

[13] http://arxiv.org/ftp/arxiv/papers/1207/1207.0780.pdf(last access 10-12-2015)

[14] Ghit, B. Epema, D.2015, Reducing Job Slowdown Variability for Data-Intensive Workloads, Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2015 IEEE 23rd International Symposium on ,pp 61-70

[15] Li Y, Chuang L, Fengyuan R.2013. Efficient Hybrid Parallel PFSP Protected Scheduling for Map Reduce System. 12 th IEEE International conference on Trust, Security and Privacy in Computing and Communications ,pp.1099-1106