Hadoop Scheduler with Deadline Constraint

Geetha J[1], N UdayBhaskar [2], P ChennaReddy[3] ,Neha Sniha [4]

[1,4] Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, Bangalore, Karnataka 560054.
[2]Department of Computer Science, Government College (UG & PG) Anantapur, A.P., India.
[3]Department of Computer Science and Engineering, JNTU College of Engineering, Pulivendula, kadapa dist AP.

Abstract

A popular programming model for running data intensive applications on the cloud is map reduce. In the Hadoop usually, jobs are scheduled in FIFO order by default. There are many map reduce applications which require strict deadline. In Hadoop framework, scheduler with deadline constraints has not been implemented. Existing schedulers do not guarantee that the job will be completed by a specific deadline. Some schedulers address the issue of deadlines but focus more on improving system utilization. We have proposed an algorithm which facilitates the user to specify a jobs deadline and evaluates whether the job can be finished before the deadline. Scheduler with deadlines for Hadoop, which ensures that only jobs, whose deadlines can be met are scheduled for execution. If the job submitted does not satisfy the specified deadline, physical or virtual nodes can be added dynamically to complete the job within deadline[8].

Keywords

Hadoop, Mapreduce, Hadoop Schedulers, HDFS, Namenode, Datanode, Jobtracker, Tasktracker, Deadline

1. INTRODUCTION

Apache Hadoop is an open source implementation of the Google's MapReduce [1] parallel processing framework. The details of parallel processing, including distribution of data to processing nodes, restart failed subtasks, and consolidation of results after computation is hidden by Hadoop framework. This framework allows developers to write parallel processing programs which focus on computation. Hadoop includes: 1) Hadoop Distributed File System (HDFS) [2], a distributed file system that store large amount of data with high throughput access to data on clusters. 2) Hadoop MapReduce: a software framework for processing of distributed data on clusters.

These problems are well-dealt by Hadoop; a reliable, scalable, distributed computing platform developed by Apache [5].It is an open source implementation of Google's MapReduce framework that allows for the distributed processing of huge data sets transversely clusters of computers using simple programming models. It is designed to level up from single datacenter to thousands of machines, each offering local computation and storage. At the application layer the library is

DOI : 10.5121/ijccsa.2014.4501

designed to detect and handle failures, Instead of relying on hardware to deliver high-availability, so a highly-available service on top of a cluster of computers can be delivered each of which may be prone to failures.

Hadoop has an underlying storage system called HDFS-Hadoop Distributed file system. To process the data in HDFS, Hadoop provides a MapReduce engine that runs on top of HDFS. This engine has master-slave architecture. The master node is called the JobTracker and the slaves are called TaskTrackers. MapReduce jobs are automatically parallelized across a large set of TaskTrackers. The JobTracker splits the job into several maps and reduce tasks. Hadoop divides the input to the job into fixed size pieces called input splits. The outputs of the map tasks are stored in the local disk. This intermediate output serves as input to the reduce tasks. The consolidated output of the reduce task is the output of the MapReduce job and is stored in HDFS.

As Hadoop jobs have to share the cluster resources, a scheduling strategy is used to determine when a job can execute its tasks. As the pluggable scheduler was implemented, several scheduler algorithms have been developed for it like FIFO Scheduler, Fair Scheduler [3], and Capacity Scheduler [4].

2. FOUNDATION

A. Problem Definition

Can a given task that translates to a MapReduce job J and has to process data of size N be completed within a deadline D, when run in a MapReduce cluster having M nodes with map task slots, reduce task slots and possibly k jobs executing at the time[8].

B. Deadline Estimation Model

We develop an initial estimation model based a set of assumptions

The cluster consists of heterenegous nodes, so that the processing of unit cost for each map or reduce node is equal; Key distribution of the input data is uniform, so that each reduce node gets equal amount of reduce data to process; Reduce tasks starts after all map tasks have completed; The input data is already available in HDFS. To derive the expressions for the minimum number of map tasks and reduce tasks, we extend the model used in for Equal Load Partitioning technique[8]. To estimate the duration of the job we consider map completion time, reduce completion time and data transfer during reduce copy phase. Hadoop supports pluggable schedulers and we have implemented Constraint Scheduler using the minimum task scheduling criteria. It is developed as a contrib module using Hadoop version 1.1.2 source code. Hadoop config file needs to be modified to use the scheduler. We have also implemented a web based interface that allows the user to specify the deadline for a given job.

C. Working

The job submission process implemented by Job Submitter does the following: User submits the job. (Step 1) .Scheduler will compute the minimum number of map and reduce slots required for the job completion (step 2). If schedulability test fails user will be notified to enter another deadline value. If passed, job will be scheduled .(Step 4).

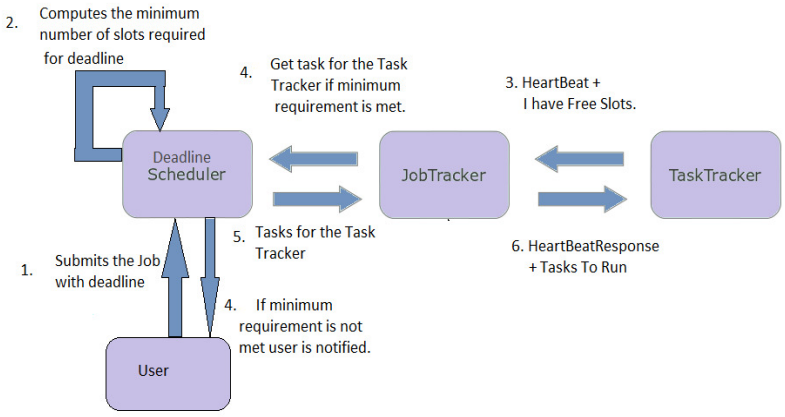

Fig. 1 Working of Deadline Scheduler

3. DESIGN AND IMPLEMENTATION

A. Design Goals

The design goals for Constraint Scheduler were:

1) To be able to give users immediate feedback on whether the job can be completed within the given deadline or not and proceed with execution if deadline can be met. Otherwise, users have the option to resubmit with modified deadline requirements.
2) Maximize the number of jobs that can be run in the cluster while satisfying the time requirements of all jobs.

B. Sequence Diagram

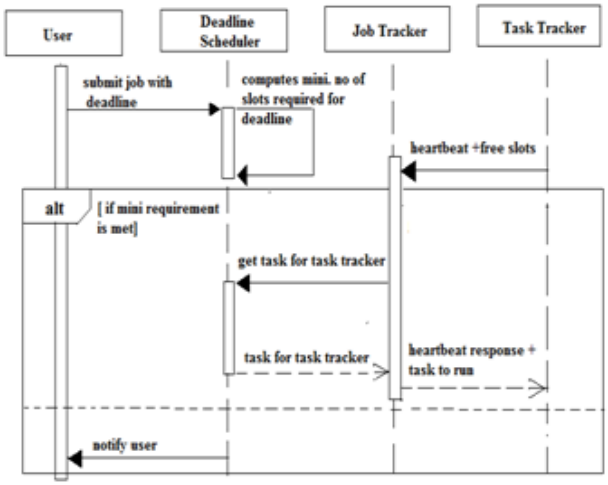

Fig. 2. Working of Deadline Scheduler

.

C. Deadline Estimation Model

Constraint Fair Scheduler

ScheduleJob(Cluster c)
freemapslots <-c.freeMapSlots
freereduceslots <-c.freeReduceSlots
REPEAT
j <-nextJob(Pool)
arrivaltime <-getStartTime(j)
mapcostperunit <-getMapCostPerUnit()
reducecostperunit <-getReduceCostPerUnit()
shufflecostperunit <-getShuffleCostPerUnit()
mapinputoutputfraction <-getMapInputOutputCostPerUnit() reducestarttime <-getReduceStartTime()
deadline <-getFloat()
inputsize <-0

REPEAT

inputsize<-inputsize+length(inputsplits) UNTIL(endOf(inputspilts))
Compute MaxReduceStartTime
IF MaxReduceStartTime < arrivaltime
THROW EXCEPTION Compute minMapSlots Compute minReduceSlots
IF minMapSlots > freemapslots
OR
minReduceSlots >freereduceslots
THROW ConstraintFailException
UNTIL (EndOf (jobs in Pool))

ConstraintFairSchedulerParamsEstimator (job word count)
Run the job randomgenerator Output <-"randomoutput" Repeat n times

Run the job wordcount taking random generator output as input for this job
Compute the parameters total map cost, total reduce cost, total io fraction, total time to start reduce, total time for job completion
Compute the average of each parameters.

A Set-up

Experiments were conducted in physical cluster. It consisted of 4 physical cluster with one jobtracker and 3 tasktrackers. The machine had 4 GB memory with 64 bit Intel i5 processor and Ubuntu server present in each node. We also had virtual node in the cluster which can be added to the existing cluster dynamically installed in oracle virtualbox on one of the nodes. Both the

environments had two map slots and two reduce slots on each machine.

B. Result

After the set up, we estimated the values of the parameters like map cost, reduce cost, shuffle cost, filter ratio, we submitted the job with deadline and took observation for different deadline values keeping input size constant, which shows the variation of required map and reduce slots as shown in Fig. 3. We also submitted the job with different input sizes and took observations. Our observation shows that for a particular data size with increasing deadlines, resource demand will decrease.

Estimated parameter values for Wordcount

1) Deadline = 45 secs
2) Mapcost = 0.00018423391
3) Reduce cost = 0.000019585172
4)Shuffle cost = 0.00000186264
5)IO Fraction = 1.3711414

The fig 3 shows observation for input data size of 500 MB taking deadline values on x-axis and minimum number of map reduce slots required on y-axis. In the setup we had 6 free map and 6 free reduce slots available. So, it is observed from figure that when deadline value of 20 seconds or more is given, minimum requirement of map reduce slots is met and job is scheduled.

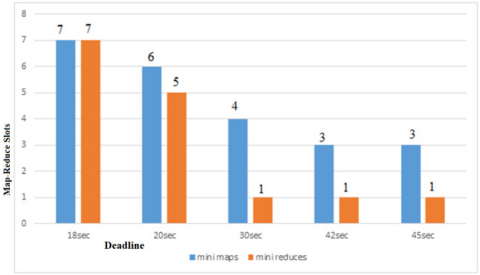

Fig. 3 Graph for 500 MB input data size

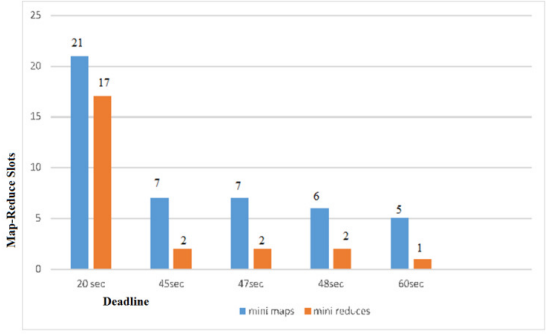

Fig. 4 Graph for 1.2GB input data size

The fig 4 shows observation for input data size of 1.2 GB taking deadline values on x-axis and minimum number of map reduce slots required on y-axis. In the setup we had 6 free map and 6 free reduce slots available. So, it is observed from figure that when deadline value of 48 seconds or more is given, minimum requirement of map reduce slots is met and job is scheduled.

The fig 5 shows observation for input data size of 5 GB taking deadline values on x-axis and minimum number of map reduce slots required on y-axis. In the setup we had 6 free map and 6 free reduce slots available. So, it is observed from figure that when deadline value of 190 seconds or more is given, minimum requirement of map reduce slots is met and job is scheduled.

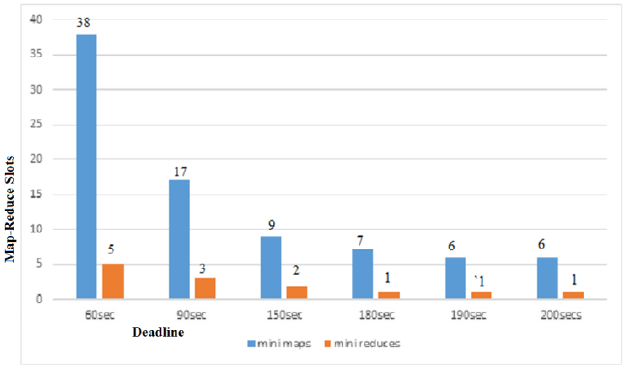

Fig. 5 Graph for 5GB input data size

4. CONCLUSION

We extended the approach of real time cluster scheduling to derive minimum map and reduce task count criteria for performing task scheduling with deadline constraints in Hadoop. We computed the amount of resource required to complete a job for a particular deadline. For this, we proposed the idea of estimation of the values of parameters: filter ratio, cost of processing a unit data in map task, cost of processing a unit data in reduce task, communication cost of transferring unit data.

Our observation shows that for a particular data size with increasing deadlines, resource demand will decrease. Also, if the data size increases and deadline is kept constant, resource demand will increase. If resource demand increases, we can meet the demand by adding physical or virtual node to the existing cluster dynamically or provide a feasible deadline. We have implemented the same.

REFERENCES

[1] Ricky Ho, How MapReduce works Dzone publishing, retrieved from http://architects.dzone.com/articles/how-hadoop- mapreduce-works
[2] Robert C,Hirang K and Sanjay R, Architecture of open source applica- tions HDFS.
[3] Apache Hadoop publishing, Hadoop MapReduce NextGeneration-Fair Scheduler, December 2013, Version 2.0.4-alpha.
[4] Arun M,Understanding Capacity Scheduler,July 2012,unpublished.
[5] Shafer J,Rixner S, and Cox, Balancing portability and performance March 2010, Performance Analysis of Systems and Software (ISPASS), 2010 IEEE International Symposium on publishing.

[6] Deadline: www4.ncsu.edu/~kkc/papers/rev2.pdf.
[7] Multi Node Cluster: http://www.michael-noll.com/tutorials/ running-Hadoop- on-ubuntu- linux-multi- node-cluster/.
[8] Scheduling Hadoop Jobs to Meet Deadline, Kamal Kc, Kemafor Anyanwu, IEEE CloudCom 2010.