

IMPROVED FAIR SCHEDULING ALGORITHM FOR TASKTRACKER IN HADOOP MAP-REDUCE

Mr. Santhosh S¹, Mr. Hemanth Kumar G²

¹PG Scholar, ²Asst. Professor, Dept. Of Computer Science & Engg, NMAMIT, (India)

ABSTRACT

Hadoop is an open source software framework that dramatically simplifies writing distributed data intensive applications. It provides a distributed file system, which is modeled after the Google File System, and a map/reduce implementation that manages distributed computation. Job scheduling is an important process in Hadoop Map Reduce. Hadoop comes with three types of schedulers namely FIFO, Fair and Capacity Scheduler. The schedulers are now a pluggable component in the Hadoop Map Reduce framework. When jobs have a dependency on an external service like database or web service may leads to the failure of tasks due to overloading. In this scenario, Hadoop needs to re-run the tasks in another slots. To address this issue, Task Tracker aware scheduling has introduced. This scheduler enables users to configure a maximum load per Task Tracker in the Job Configuration itself. The algorithm will not allow a task to run and fail if the load of the Task Tracker reaches its threshold for the job. Also this scheduler allows the users to select the Task Tracker's per Job in the Job configuration.

Keywords: *Big Data, Hadoop Map-Reduce, HDFS, Scheduler, Task Tracker*

I. INTRODUCTION

Hadoop Map-Reduce [1] [10] is a software framework and programming model for writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. Under such models of distributed computing, many users can share the same cluster for different purpose. Situations like these can lead to scenarios where different kinds of workloads need to run on the same data center. For example, these clusters could be used for mining data from logs which mostly depends on CPU capability. At the same time, they also could be used for processing web text which mainly depends on I/O bandwidth.

1.1 Motivation

Hadoop leverages the power of distributed computing with the help of Google Map Reduce Framework and Hadoop Distributed File System (HDFS) [2]. Map Reduce is the framework for processing large volume of datasets as key value pairs. Map Reduce divides each job in to two types of functions, map and reduce. The input is initially processed in distributed map tasks and aggregate the result with the help of reduce tasks [3] [9]. Hadoop uses HDFS as its primary data storage area [2]. HDFS is designed for storing very large files with streaming access in commodity hardware. The data is stored as blocks of independent unit with default size of 64 MB. The blocks are stored with a replication factor for handling the hardware failures. The default replication factor is 3.

1.2 Objectives

Hadoop uses large number of commodity hardware's for creating the cluster to solve Big Data problems. The Hadoop cluster offers a great potential of resources to multiple users in a shared manner. A good scheduler is required for sharing resources fairly between users. The default scheduler in Hadoop Map Reduce version 1 is FIFO scheduler. FIFO Scheduler schedules the jobs in the order in which it is submitted. There are two more schedulers comes with Hadoop Map Reduce named Fair Scheduler and Capacity Scheduler. These schedulers failed to serve their responsibility in some use cases. This paper exposes some of these use cases and a modified scheduler for handling the scenarios.

II. LITERATURE SURVEY

2.1 Hadoop Map-Reduce and Job Scheduling

The main advantage of Hadoop Map-Reduce is the data is distributed efficiently in the HDFS and the program is running over the local data wherever possible [62]. So the data movement between different nodes in the cluster is reduced and it gains the performance. So for efficient processing of Big Data using Hadoop, the data should be present in the HDFS. But in almost 90% of the real world use cases, the data is generated by some legacy systems or applications and moved in to HDFS for perform the analytics.

Hadoop is dealing with large volumes of data and most of the Hadoop jobs are running for a long time. So it's a huge loss if the jobs failed after running for a long time. To avoid this, the jobs need to be scheduled properly and user needs to get some control for scheduling the jobs.

The default scheduling algorithm in Hadoop operates off a first-in, first-out (FIFO) basis. Beginning in v0.19.1, the community began to turn its attention to improving Hadoop's scheduling algorithm, leading to the implementation of a plug-in scheduler framework to facilitate the development of more effective and possibly environment-specific schedulers. Since then, two of the major production Hadoop clusters – Facebook and Yahoo – developed schedulers targeted at addressing their specific cluster needs, which were subsequently released to the Hadoop community.

2.2 Scheduling in Hadoop

2.2.1 Default FIFO Scheduler

The default Hadoop scheduler operates using a FIFO queue. After a job is partitioned into individual tasks, they are loaded into the queue and assigned to free slots as they become available on Task Tracker nodes. Although there is support for assignment of priorities to jobs, this is not turned on by default.

2.2.2 Fair Scheduler

The Fair Scheduler [4] was developed at Facebook to manage access to their Hadoop cluster, which runs several large jobs computing user metrics, etc. on several TBs of data daily. Users may assign jobs to pools, with each pool allocated a guaranteed minimum number of Map and Reduce slots. Free slots in idle pools may be allocated to other pools, while excess capacity within a pool is shared among jobs. In addition, administrators may enforce priority settings on certain pools. Tasks are therefore scheduled in an interleaved manner, based on their priority within their pool, and the cluster capacity and usage of their pool.

As jobs have their tasks allocated to Task Tracker slots for computation, the scheduler tracks the deficit between the amount of time actually used and the ideal fair allocation for that job. As slots become available for scheduling, the next task from the job with the highest time deficit is assigned to the next free slot.

Over time, this has the effect of ensuring that jobs receive roughly equal amounts of resources. Shorter jobs are allocated sufficient resources to finish quickly. At the same time, longer jobs are guaranteed to not be starved of resources.

2.2.3 Capacity Scheduler

Yahoo's Capacity Scheduler [5] [7] addresses a usage scenario where the number of users is large, and there is a need to ensure a fair allocation of computation resources amongst users. The Capacity Scheduler allocates jobs based on the submitting user to queues with configurable numbers of Map and Reduce slots.

Queues that contain jobs are given their configured capacity, while free capacity in a queue is shared among other queues. Within a queue, scheduling operates on a modified priority queue basis with specific user limits, with priorities adjusted based on the time a job was submitted, and the priority setting allocated to that user and class of job.

When a Task Tracker slot becomes free, the queue with the lowest load is chosen, from which the oldest remaining job is chosen. A task is then scheduled from that job. Overall, this has the effect of enforcing cluster capacity sharing among users, rather than among jobs, as was the case in the Fair Scheduler.

2.3 Problem Statement

All scheduling methods implemented for Hadoop to date share a common weakness:

1. Failure to monitor the capacity and load levels of individual resources on Task Tracker nodes, and
2. Failure to fully use those resource metrics as a guide to making intelligent task scheduling decisions.

Instead, each Task Tracker node is currently configured with a maximum number of available computation slots. Although this can be configured on a per-node basis to reflect the actual processing power and disk channel speed, etc available on cluster machines, there is no online modification of this slot capacity available. That is, there is no way to reduce congestion on a machine by advertising a reduced capacity.

2.4 Proposed System

The following are the desired functionalities for the proposed scheduler

- Control the number of concurrent tasks for a particular job in a Task Tracker
- This value should be configurable via the Job configuration itself. So user can change this value dynamically for different jobs.
- Fence some Task trackers from Job execution.
- Execute different jobs in user selected Task Trackers.
- It should support all the functionalities currently provided by the Fair Scheduler.

III. SYSTEM DESIGN

The proposed scheduler schedules the jobs according to the current status of the Task Tracker's. So the scheduler is named accordingly. The proposed system shown in Fig.1 is divided into two components, the core scheduler module which will handle the actual scheduling part and a preprocessing module. When a Heartbeat is received from a Task Tracker, the Task Tracker information and List of scheduled Jobs should hand over to the preprocessor. The preprocessing module first compare the hostname of the Task Tracker against the list of Task Trackers specified for the Job. If this check succeeds then it will compute the number of tasks currently running for the Job in the Task Tracker.

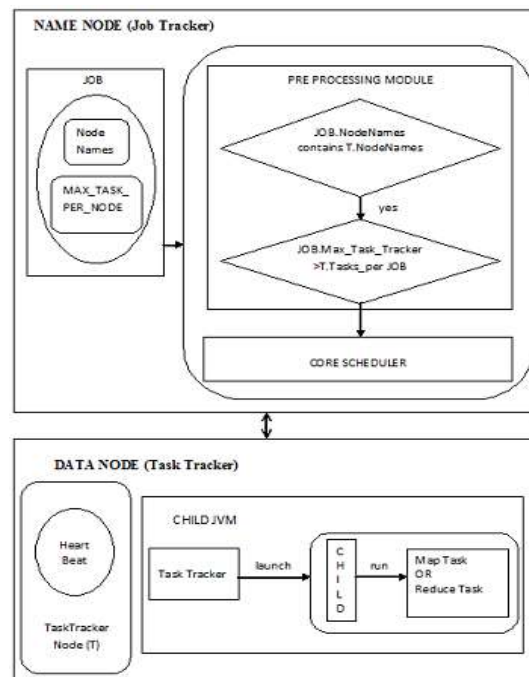


Figure 1: System Architecture of Task Tracker Aware Scheduler

If the number of currently running tasks is less than the number specified in the Job Configuration, then the Job object and Task Tracker information is hand over to the Core Scheduler module. The Core scheduler is a modified Fair Scheduler [6] [7] with a priority enhanced algorithm.

3.1 The Task Tracker Aware Scheduler Algorithm

The Task Tracker Aware Scheduler algorithm is given below.

- Each Job 'j' is initialized with $j.wait=0$, $j.priority=Normal$
- Read and set value of $j.max_task_task$ Tracker and $j.hosts_array$ from job configuration
- Heartbeat is received from node 'n'
- For each job 'j' sort by hierarchical scheduling policy with priority
- For j in jobs do
 - If $j.hosts_array$ contains $n.hostname$ and $j.Max_task_task$ tracker $>$ running tasks for job 'j' on node 'n' then
 - if 'j' has node local task 'T' on 'n' then
 - Return task 'T' to 'n'
 - Else if
 - $j.Priority==Priority.HIGH$
 - then
 - Set $j.wait = 0$;
 - Set $j.Priority =Priority. NORMAL$;
 - Return task 'T' to node 'n'
 - Else
 - $j.wait = j.wait + W1$
 - End if
 - Else if $j.wait>Max_Wait_time$

- Set J.Priority = HIGH
 - Else
 - J.wait = j.wait + W1
 - End IF
- End For

where, W1 - The time difference between Heartbeats

IV. IMPLEMENTATION

Selection of tasks within a job is mostly done by the Job In Progress class, and the Job Scheduler class. JobIn Progress exposes two methods, obtain New Map Task and obtain New Reduce Task, to launch a task of either type. Both methods may either return a Task object or null if the job does not wish to launch a task. The Task Tracker Aware Scheduler overrides assign Tasks method with an algorithm called Task Tracker Aware Scheduling to control the Task Tracker selection. The implementation introduced two configuration properties,

- `mapred.tascheduler.tasks.max` – Maximum number of tasks that can be run on a Task Tracker for a Single Job. Value is in Integer. Value ≤ 0 should be treated as maximum slots available
- `mapred.tascheduler.hosts` – Hostnames of the Task Trackers in which jobs needs to be run. Values in Strings separated with comma. String “ALL” should be treated as select all available Task Trackers.

User needs to configure the above two properties in the Job Configuration object. A java class Job Schedulable For TAS is implemented which extends the Schedulable class present in the original Fair Scheduler. The configuration properties are read from the Job Object and applied the preprocessing logic at this point.

4.1 Data Flow Diagram of TaskTracker scheduler

The data flow diagram for Job execution using Task Tracker Aware Scheduler is shown in Fig.2

The steps are shown below:

1. Submit Hadoop Job to the Job Tracker with Job Conf
2. Job Tracker will create a Job In Progress Object
3. Heartbeat received from Task Tracker at Job Tracker
4. Job Tracker Pass the Task Tracker information to the Task Tracker Aware Scheduler
5. Task Tracker Aware Scheduler pass the Task Tracker availability and Jobs in queue to the Job Scheduler
6. Job Scheduler iterates through the queued jobs and picks the corresponding Job In Progress object.
7. Handover the Job In Progress object to the Scheduling algorithm and find out that if the task present in the object matches with the Task Tracker configuration.
8. Handover the task slot to Job Scheduler if it matches with algorithm, else return null. If the result is null, Job scheduler picks next Job In Progress Object from the Queue and continue from step 7.
9. If 8 is success, update the status in Job In Progress.
10. Handover the task object to Scheduler
11. Handover the task object to Job Tracker
12. Handover the information to the Task Tracker
13. Execute the task in Task Tracker and update the result in HDFS

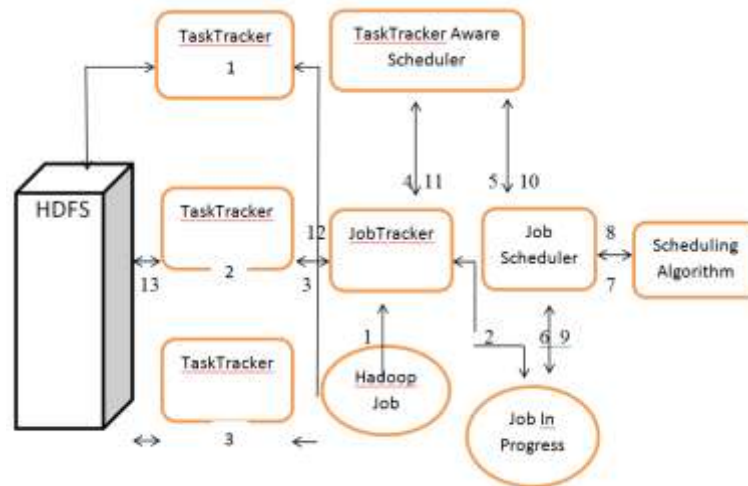


Figure 2: Data Flow Diagram of the TaskTracker

V. EXPERIMENTS AND RESULTS

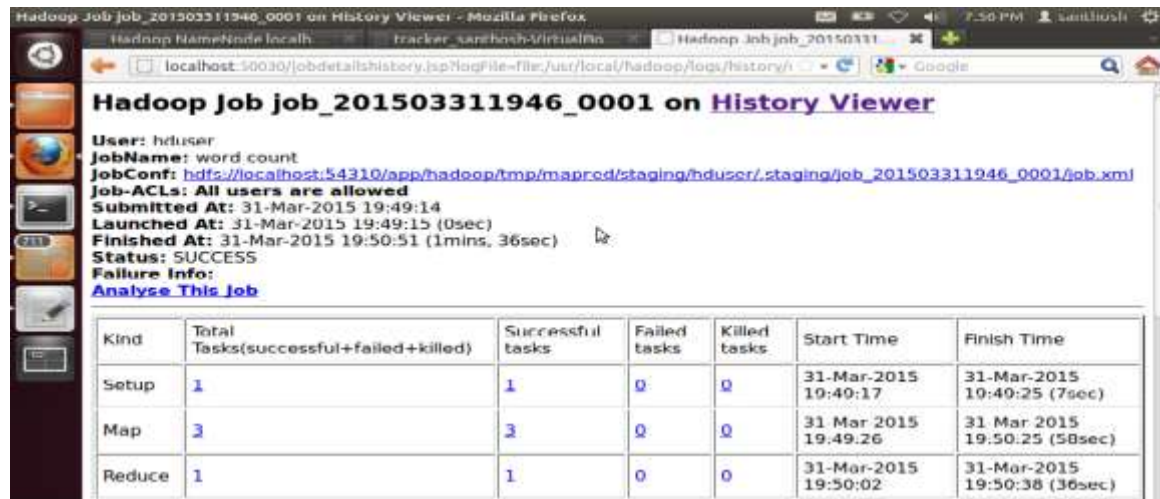
A Hadoop cluster is setup in the lab with Apache Hadoop version 1.0.3. The available core in the machine was AMD A4 processor. The machine is configured with 2 map slots and 2 reduce slots as shown in figure 3. The source code of Fair Scheduler is downloaded from Apache Hadoop website. TaskTracker Aware Scheduler is implemented by adding the preprocessor module and Priority handling logic to the original Fair scheduler. Hadoop is configured to Task Tracker Aware scheduler by adding the jar file to the lib folder and updated the property name `mapred.jobtracker.taskScheduler` to `org.apache.hadoop.mapred.Task Tracker Aware Scheduler` [8]. A word count sample program is written for testing.

5.1 Steps and Snapshots of Tests and Results

1. Submit the job with a single host name, shown in figure 3.
2. Submit the job with `hostnames="ALL"`
3. Submit the job with `max_TasksPerNode=1`, shown in figure 4
4. Submit the job with `max_Tasks_PerNode=2`

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	0.00%	3	1	2	0	0	0 / 0
reduce	0.00%	1	1	0	0	0	0 / 0

Figure 3: Job with Single Hostname



Hadoop Job job_201503311946_0001 on History Viewer

User: hduser
 JobName: word count
 JobConf: hdfs://localhost:54310/opp/hadoop/tmp/mapred/staging/hduser/.staging/job_201503311946_0001/job.xml
 Job-ACLs: All users are allowed
 Submitted At: 31-Mar-2015 19:49:14
 Launched At: 31-Mar-2015 19:49:15 (0sec)
 Finished At: 31-Mar-2015 19:50:51 (1mins, 36sec)
 Status: SUCCESS
 Failure Info:
[Analyze This Job](#)

Kind	Total Tasks(successful+failed+killed)	Successful tasks	Failed tasks	Killed tasks	Start Time	Finish Time
Setup	1	1	0	0	31-Mar-2015 19:40:17	31-Mar-2015 10:40:25 (7sec)
Map	3	3	0	0	31 Mar 2015 19:49:26	31 Mar 2015 19:50:25 (58sec)
Reduce	1	1	0	0	31-Mar-2015 19:50:02	31-Mar-2015 19:50:38 (36sec)

Figure 4: Job with Max_TasksPerNode=1

VI. CONCLUSION

A Hadoop cluster is setup in the lab with 3 nodes. Different types of schedulers are tested and verified the results. Task Tracker Aware scheduling algorithm is implemented and tested successfully. The advantages over the existing algorithms are verified. The proposed solution overcomes the limitations of the existing schedulers described here and gives more control to the users for Job execution.

REFERENCES

Websites

- [1] Apache Hadoop, ” <http://hadoop.apache.org>”.
- [2] “http://hadoop.apache.org/docs/r1.0.4/hdfs_design.html”, Hadoop Distributed file system
- [3] “http://hadoop.apache.org/common/docs/current/mapred_tutorial.html”, Hadoop Map Reduce tutorial
- [4] Hadoop Fair Scheduler, “http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html”
- [5] “http://hadoop.apache.org/docs/stable/capacity_scheduler.html”, Hadoop Capacity Scheduler

Journals

- [6] B.Thirumala Rao,Dr.L S S Reddy “Survey on Improved Scheduling in Hadoop Map Reduce in Cloud Environments” in International Journal of Computer Applications (0975 – 8887)Volume 34– No.9, November 2011
- [7] Matei Zaharia, Dhruba Borthakur, and Joydeep Sen Sarma, Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling, Yahoo! Research, University of California, Berkeley ,2009
- [8] Jisha S Manjaly, Varghese S Chooralil “TaskTracker Aware Scheduling for Hadoop Map Reduce” in Third International Conference on Advances in Computing and Communications, 2013

Proceedings paper

- [9] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, and Randy Katz Ion Stoica, “Improving Map Reduce Performance in Heterogeneous Environments,” Proceedings of the 8th conference on Symposium on Operating Systems Design & Implementation

Books

- [10] Tom White, “Hadoop The Definitive guide”, Third Edition, 2012.