International Journal of Contemporary Research in Computer Science and Technology (IJCRCST) e-ISSN: 2395-5325
Volume3, Issue 7 (July '2017)

SCHEDULING WITH TASKTRACKER FOR MAPREDUCE IN HADOOP

Poornima K S,
Assistant Professor,
Department of Computer Science and Engineering,
Dayananda Sagar College of Engineering,
Bangalore, Karnataka,India.

Akshatha G,
Assistant Professor,
Department of Computer Science and Engineering,
Dayananda Sagar College of Engineering,
Bangalore, Karnataka,India.

Abstract: Hadoop is a framework for processing large amount of data in parallel. It uses the Hadoop Distributed File System (HDFS) and MapReduce framework for processing. Job scheduling is an important process in Hadoop MapReduce. There are three types of schedulers mainly used in Hadoop Framework. They are namely FIFO, Fair and Capacity Scheduler. In Hadoop Framework the schedulers are a pluggable component now a day. There are jobs which have dependency on an external service like database or web service. This condition will lead to the failure of tasks due to overloading. In this case, Hadoop needs to re-run the entire tasks in another slots. To address this issue, the TaskTracker aware scheduling has been introduced. The TaskTracker scheduler enables users to configure a maximum load per TaskTracker in the Job Configuration itself. If the load of the TaskTracker reaches its maximum limit then the algorithm will not allow the task to run and stops the task from failing. Also, the TaskTracker aware scheduler allows the users to select the TaskTracker per Job which is configured in the Job configuration.

Keywords: Hadoop, HDFS, BigData, MapReduce, JobTracker, TaskTracker, Scheduler

I .INTRODUCTION

Hadoop is the heart of a complex flow of data. Apache Hadoop is a software framework for processing Big Data such as data in the range of petabytes[1]. Doug Cutting is the developer of Hadoop Framework and he is also the creator of Apache Lucene, which is part of his Web Search Engine Apache Nutch. Hadoop MapReduce is a programming model and software framework for developing applications that quickly process huge amounts of data in parallel on large clusters of compute nodes. Hadoop leverages the power of distributed computing with the help of Google Map Reduce Framework and Hadoop Distributed File System (HDFS)[2]. Hadoop applications can function towards Business success today which heavily depends on two kinds. First one is Ability to store and analyse large data sets. Second one is an Ability to extract other organizations data.

Hadoop is the heart of a complex flow of data. Data is often collected from many distinguished systems. This data is then imported into the Hadoop Distributed File System (HDFS). While processing, some form of processing takes place using MapReduce or one of the several languages built on top of MapReduce (Hive, Pig, Cascading, and so on). Finally, the filtered, transformed, and aggregated results are exported to one or more external systems.

Hadoop Distributed File System (HDFS): Hadoop leverages the power of distributed computing with the help of Google Map Reduce Framework and Hadoop Distributed File System (HDFS). Data is often collected from many disparate systems. This data is then imported into the Hadoop Distributed File System (HDFS). The main advantage of Hadoop MapReduce is the data is distributed efficiently in the HDFS and the program is running over the local data wherever possible. So the data movement between different nodes in the cluster is reduced and it gains the performance. So for efficient processing of BigData using Hadoop, the data should be present in the HDFS.

MapReduce Framework: Hadoop MapReduce is the framework for processing large volume of datasets as key value pairs. MapReduce divides each job in to two types of functions, map and reduce. Both Map and Reduce functions take input as key value pairs and emits the result as another set of key value pairs. Each job is divided in to number of map tasks and Reduce tasks. The input is initially processed in distributed map tasks and aggregate the result with the help of reduce tasks. In conventional programming, the data is copied to the location where the actual job is running. Where as in Hadoop MapReduce framework, the job is copied to the location where the actual data exists. HDFS is been used as primary data storage area for achieving the data locality in Hadoop. 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.

Schedulers: 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[2]. The default scheduler in Hadoop MapReduce version 1 is FIFO scheduler. FIFO Scheduler schedules the jobs in the order in which it is submitted. FIFO, Fair Scheduler and Capacity Scheduler are the three more schedulers with Hadoop MapReduce. These schedulers failed to serve their responsibility in some use cases. Scenarios are handled by modified scheduler.

TaskTracker&JobTracker: It is a node in the cluster that is capable of performing tasks such as Map, Reduce and Shuffle operations - from a JobTracker. Every TaskTracker has a configuration which contains the set of slots. This information shows the maximum number of tasks it can accept. If it does not find then it looks for an empty slot on a

machine in the same rack. TaskTracker assigns JVM processes to separate the actual work. Because this can ensure that process failure does not bring down the task tracker. In case of process failures, the affected processes are monitored by the TaskTracker by capturing the output and exit codes. It notifies that JobTracker, when the process finishes, successfully or not.In order to ensure that the TaskTracker is alive and working fine, it sends out heartbeat messages to the JobTracker, usually every few minutes. This message also has the information such as the number of available slots so that the JobTracker can decide on further assignment of the tasks to that particular TaskTracker or to any other TaskTracker. Specific nodes in the cluster that are ideally the nodes that have data or in the same track are assigned MapReduce task within the Hadoop by the JobTracker.

II .LITERATURE SURVEY

The present scheduler calculates the actual capacity at each node by monitoring the resource load levels per node. Monitoring the resources such as CPU utilization and number of page faults is done by each TaskTracker in the TaskTracker Resource Monitoring framework. It had the basic three resources. To improve the load balancing these three resources are tracked at all the time. The capacity of the machine's virtual memory management state was considered. This in turn results in monitoring page faults and virtual memory. These factors are useful for tracking the free memory.

Delay Scheduling: There is conflict between fariness and data locality during cluster sharing. To solve the problem, an algorithm called delay scheduling[3] was proposed. This algorithm ensures that when the job that should be scheduled next according to fairness cannot launch a local task, it waits for a small amount of time, letting other jobs launch tasks instead. This is the tradeoffs between utilization and fairness. In cluster computing infrastructure, there exists some conflict i.e. it provides great fairness but poor utilization when there are separate clusters, and in a single FIFO cluster, it provides great utilization but no fairness. Delay scheduling can be applied under various types of scheduling techiniques than fair sharing due to its simplicity.

Improved Scheduling in Cloud Environment: Cloud Computing is emerging technology today. For processing huge data on Clouds, Hadoop MapReduce is used as a powerful Computation Model[4]. The FIFO scheduler is the default scheduler available in all Hadoop implementations. The FIFO order is used to schedule the jobs and also priority is also considered accordingly. Different schedulers which can be used with Hadoop implementation were studied and some instructions on how to enhance the scheduling technique in Hadoop in Cloud Environments were given[4].
The following are the desired functionalities for the system

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

III. PROPOSED SYSTEM

Figure 3 gives the System Architecture of the TaskTracker Aware Scheduler system. The scheduler schedules the jobs according to the current status of the TaskTracker. So the scheduler is named accordingly. The architecture is divided into two parts. They are pre-processing module and core scheduler module. The core scheduler module will handle the actual scheduling part. Some jobs access information from the external services and databases. Because of this the unavailability of the data occurs whenever there is an issue with the external service and database. This in turn causes the failure of task due to overloading or unavailability.

In the below Figure 3, consider a TaskTracker node T, which send out heart beat messages. We have a Job, which has the node names if the TaskTrackers specified for the job and maximum tasks per node in its Job Configuration. When a Heartbeat is received from a TaskTracker, the TaskTracker information and List of scheduled Jobs should hand over to the pre-processor. The pre-processing module first compare the hostname of the TaskTracker against the list of TaskTrackers specified for the Job. If this check succeeds then it will compute the number of tasks currently running for the Job in the TaskTracker.

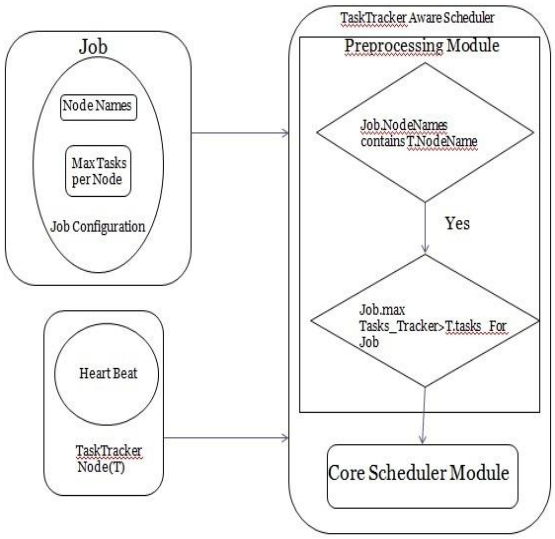


Figure 1: System Architecture of TaskTracker Aware Scheduler System

If the total number of currently running tasks is less than the number specified in the Job Configuration, then the Job object and TaskTracker information is handed over to the Core Scheduler module. The Core scheduler is a modified Fair Scheduler with a priority enhanced algorithm.

TaskTracker Aware Scheduler Algorithm

The Task Tracker Aware Scheduler is introduced to overcome the limitations of the other scheduling algorithms. When the jobs have dependencies on the external database or source, there is a chance of overloading and failure. In this case, the entire task needs to be re run in another slot. To address this issue, the Task Tracker Aware Scheduler is introduced. Hence, we use Task Tracker Scheduler algorithm for the scheduler[1]. This enables the user to configure a maximum load per TaskTracker in the job configuration

itself and it will not allow a task to fail if the TaskTracker has reached its threshold. The algorithm is explained below.
Each Jobn'j' is initialized with j.wait=0,
j.priority = Normal
Read and set value of j.max_task_taskTracker 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_tasktracker>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
Hierarchical Scheduling- It is sorting the jobs with maximum number of task remaining to execute comes first in the job queue.

IV. IMPLEMENTATION

All schedulers in Hadoop, including the TaskTracker Aware Scheduler, inherit from the TaskScheduler abstract class. The TaskTracker Manager is accessed through the TaskTracker abstract class i.e. it acts as an interface to the JobTracker and as a Configuration instance. The scheduler implements three abstract methods. Those methods are the lifecycle methods start and terminate, and assign Tasks method to launch tasks on a given TaskTracker. In Hadoop task assignment is an active process. Every TaskTracker node periodically sends out heartbeat messages to the JobTracker with their TaskTrackerStatus. This status contains a list of running tasks on the node, the number of available slots on the node, and other information.

In order to launch the tasks, the Job Tracker calls assign Tasks on the scheduler. After the launch, these are returned with the heartbeat message. The Job in progress class and the Job Scheduler class does the selection of tasks within a job. Job In Progress exposes two methods, obtain NewMap Task and obtain NewReduceTask. Either of the two methods can be launched and both methods may either return a Task object or if the job does not wish to launch a task return a null. The TaskTracker Aware Scheduler overrides assign Tasks method with an algorithm called TaskTracker Aware Scheduling to control the TaskTracker selection. The implementation introduced two configuration properties:

• mapred.tascheduler.tasks.max – Maximum number of tasks that can be run on a TaskTracker 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 TaskTrackers

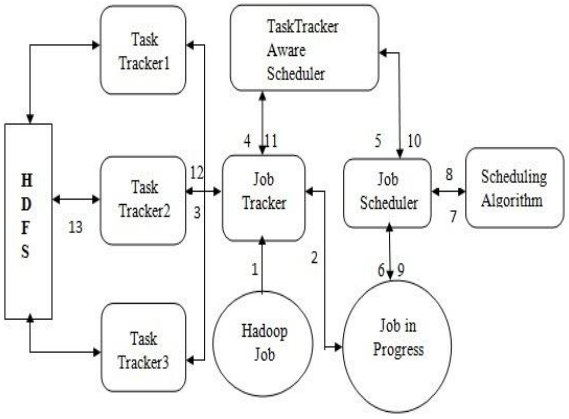


Figure 2: TaskTracker Aware Scheduler System

The Figure 2 shows the implementation and functionality of the TaskTracker Scheduler System. The step by step processing of the TaskTracker is numbered accordingly and the explanation is as follows.

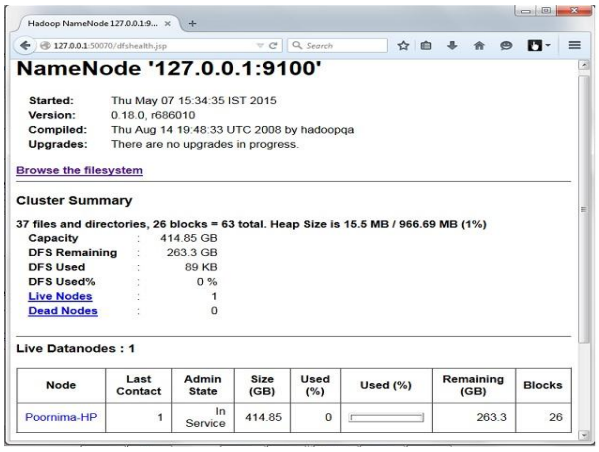
1. Submit Hadoop Job to the Job Tracker with JobConf
2. Job Tracker will create a JobInProgress Object
3. Heartbeat received from Task Tracker at Job Tracker
4. JobTracker Passes the TaskTracker information to the TaskTracker Aware Scheduler.
5. TaskTracker Aware Scheduler passes the TaskTracker availability and Jobs in queue to the Job Scheduler.
6. JobScheduler iterates through the queued jobs and picks the corresponding JobInProgress object.
7. Handover the JobInProgress object to the Scheduling algorithm and find out that if the task present in the object matches with the TaskTracker 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 JobInProgress Object from the Queue and continue from step 7.
9. If 8 is success, update the status in JobInProgress.
10. Handover the task object to Scheduler
11. Handover the task object to Job Tracker
12. Handover the information to the TaskTracker
13. Execute the task in TaskTracker and update the result in HDFS.

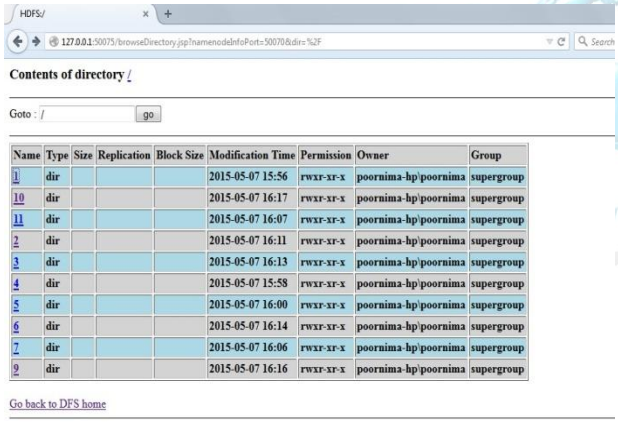
V. EXPERIMENTS AND RESULTS

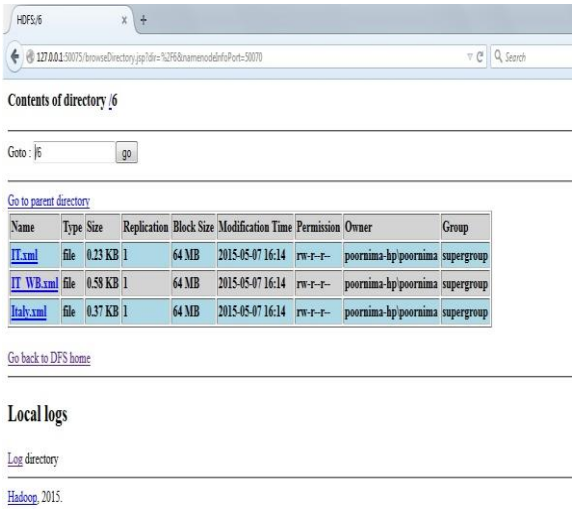
A Hadoop cluster is setup in the lab with Apache Hadoopversion 1.04. The available core in the machine was Intel i3.Each machine is configured with 2 map slots and 2 reduce slotsas shown in figure 3. The source code of Fair Scheduler is downloaded from Apache Hadoop website. TaskTracker Aware Scheduler is implemented by adding the pre-processor module and TaskTracker aware scheduler algorithm logic to the original Fairscheduler.

The below use cases and combinations are tested andverified the results:

1. Submit the job with a single host name, shown in figure 4.
2. Submit the job with hostnames="ALL'
3. Submit the job with max_TasksPerNode=1, showing figure 5
4. Submit the job with max_Tasks_PerNode=2







VI.CONCLUSION

The TaskTracker Scheduler system is implemented using TaskTracker aware scheduler algorithm and tested successfully. The main aim of the system is to configure maximum load per TaskTracker in the Job Configuration itself. The system prevents the task from running as well as failing if the load of the TaskTracker reaches the threshold for the job. It allows the user to configure maximum load per TaskTracker in the Job Configuration. User can configure the value via Job Configuration so that user can change the value dynamically for different jobs. It can control the number of concurrent tasks for a particular job in a TaskTracker. The TaskTracker scheduler allows the users to select the TaskTracker per Job in the configuration. It also can fence some TaskTasker from Job Execution.

It overcomes the limitations of the existing schedulers and gives more control to the users for job execution. Meanwhile it supports all the functionalities currently provided by the Fair Scheduler. The proposed system is a dynamic system. It uses the dynamic data obtained from the GeoName Services.

VII.REFERENCES

[1]. Jisha S Manjaly, Varghese S Chooralil, "TaskTracker Aware Scheduling for Hadoop MapReduce", 2013 IEEE, Third International Conference on Advances in Computing and Communications.

[2]. Mark Yong, Nitin Garegrat, Shiwali Mohan, Computer Science and Engineering, University of Michigan, Ann Arbor, worked on, "Towards a Resource Aware Scheduler in Hadoop", in December, 2009.

[3]. MateiZaharia, DrubaBorthakur, Joydeep Sen Sarma, KaledElmeleegy, Scott shenker, Ion Stoica worked on, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling", in ACM, 2010.

[4]. B. Thirumala Rao, Dr. L.S.S Reddy, Lakireddy Bali Reddy College of Engineering, worked on, "Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments", and presented in International Journal of Computer Applications (0975 – 8887)Volume 34– No.9, November 2011.

[5]. Tom White, "Hadoop: The Definitive guide", third edition, 2012.

[6]. Top five ways to get started with Big Data, IBM software, Through Leader White Paper, June 2014.

[7]. Sonja Zillner, HienerOberkampf, Claudia Bretschneider, AmrapaliZaveri, Sabrina Neururer worked on, "Towards a Technology Roadmap for Big Data Applications in the Healthcare Domain", in 2013.

[8]. Puneeth Singh Duggal and Sanchita Paul, Department of Computer Science &Engineering of Birla Institute of Technology, worked on, " Big Data Analysis: Challenges and Solutions", and presented it in International Conference on Could, Big Data and Trust 2013.