Improving Job Scheduling in Hadoop

Hawai Abdullah, Sohail Ansari, Hamzah Ansari, S.A.Gindi

1,2,3,4 Student, 4 Assistant Professor
1,2,3,4 Department of Information Technology
1,2,3,4 MHSSCOE, Mumbai, India

Abstract—Hadoop is an open-source software framework. Hadoop is used for storage and large-scale processing of data-sets on clusters of commodity hardware. MapReduce is a one type of programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster. Scheduling is required in hadoop for Time Optimization and CPU Utilization. By default Hadoop uses FIFO, and optional 5 scheduling priorities to schedule jobs from a work queue. Various scheduling algorithms such as FIFO, Fair and Capacity are good as some time means if load is low then Fair scheduler are good one but if load increased at some level then capacity scheduler are good choice. All this scheduler having some problem. If jobs are scheduled in FIFO order, it easily causes the starvation of small jobs in the event of resources being utilized by large jobs, while Fair Scheduler is inefficient when handling large jobs and it leads to sticky slots problem. For resolving such issue some customized algorithms have been developed by enhancing existing algorithms. Now our work is to enhance the scheduling algorithm for task allocation by minimizing the process time and more resource utilization. We will improve TDWS (Tencent Distributed Data Warehouse Scheduling) algorithm by including parameters such as CPU Utilization, Free Memory (RAM), Load average of the resources.

Key words: Hadoop, HDFS, MapReduce, FIFO Scheduling, Fair Scheduling, Capacity Scheduling, TDWS Scheduling

I. INTRODUCTION

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware[1]. The volume and size of the data is increasing day by day with the exponential growth of the enterprises. People upload videos, take pictures on their cell phones, update their Facebook status, leave comments around the web, click on ads etc. By doing these all activities machines are generating and keeping more and more data. These types of growth of data are first challenges to businesses such as Google, Yahoo, Amazon, and Microsoft. They required going through Terabytes and Petabytes of data. So need of processing and study of large volume of data is increasing and so Hadoop File System are Developed. Hadoop is one type of Programming framework which is used to support the processing of large volume of data in a distributed environment. Hadoop was developed by Google’s MapReduce. MapReduce is a programming model and an associated implementation for processing and generating large data sets with a parallel, distributed algorithm on a cluster [3]. A popular data processing engine for big data is Hadoop MapReduce [18]. Job Scheduling for the Task Allocation to the Map reduce is equally important. Hadoop takes care for storing the data and replication as well but main concern is for retrieving the important and appropriate data with less lost cost in time and CPU Usage is the key part. Scheduling is required in hadoop for Time Optimization and CPU Utilization. By default Hadoop uses FIFO, and optional 5 scheduling priorities to Schedule jobs from a work queue [5].

II. TDWS (Tencent Distributed Data Warehouse) ALGORITHM

If jobs are planned in FIFO order, it simply causes the starvation of small jobs in the event of resources being utilized by big jobs, while Fair Scheduler is ineffective when handling large jobs and it leads to sticky slots problem. Here, we propose a new job scheduling algorithm TDWS. The scheduling algorithm takes account characters of various dissimilar applications to meet their different requirements. In addition, it is also highly robust to heterogeneity and easy to accomplish optimal data locality [5].

TDWS are usually used for raised a data locality. TDWS ordered a job into group just as same as fair scheduler and also maintains advantage of FIFO. Comparing to Fair, tasks of large jobs could be executed with much higher parallelism, and avoids the sticky slots problem. Comparing to FIFO, short jobs will not starve and long jobs can still finish in reasonable time, while critical jobs that come up could be done as soon as possible [5]. TDWS has been adopted in Tencent data warehouse, and plays an important role in daily Tencent operations.

A. Following is the existing TDWS Algorithm

If n has a free slot then

Sort Jobs in TDWS Style

For j in jobs do

If n has enough memory resource for job j

If j has unlaunched task t with data then

Launch t on n

Set j.skipCount=0

Break
Else If j has unlaunched task t then
    If j.skipCount >= D then Launch t on n
    Break
Else
    Set j.skipCount = j.skipCount + 1
End if
End if
End for

In Existing TDWS Algorithm, only memory associated with the resource is taken under consideration. Following are statistic with respect to other scheduling algorithms for turnaround time for large jobs.

B. TDW Scheduling

TDW organizes jobs into two groups:

1. Nrtgroup
2. Rtggroup

It assigns a share to each group, it also have special group known keygroup, which is only used for urgent jobs.

1. Nrtgroup:
   - This group works for large jobs only.
   - Most of the slot resources of the cluster are occupied by this group.
   - 80% of slot resources are occupied by this group.
2. Rtggroup:
   - This group works for small jobs only.
   - The rest of cluster resources is engaged for this group to prevent the starvation of small jobs.
   - If we set 80% of total slot resources to nrtgroup, the left 20% of total slot resources will be allocated to rtggroup.

There is also a special group named keygroup which works for urgent jobs, keygroup has the highest privilege to get all the slot resources of the cluster. When new jobs are submitted to the keygroup, all tasks slots that are free will be assigned to this group.

III. RELATED WORK

Yongcai Tao, Qing Zhang, Lei Shi, Pinhua Chen et al. (2011): This research paper is based on FAIR scheduling, which proposes an improved FAIR scheduling algorithm, which takes the job character and data locality while tasks are killed for making a slot for new users. Performance evaluation results show that the improved FAIR decreases the data progress, speeds the execution of jobs, consequently improving a system performance. Drawback of this paper is I/O Bottleneck while killing the task and performance degradation while reallocation the task.

Yanrong Zhao, Weiping Wang, Dan Meng, Yong Chun Lv, Shubin Zhang, Jun Li (2012): In this paper, author proposed a new job scheduling algorithm TDWS. The scheduling algorithm takes account characters of different applications to meet their different requirements. In addition, it is also highly robust to heterogeneity and easy to accomplish best possible data locality. The experiments demonstrate the probability and a efficiency.

Aysan Rasooli, Douglas G. Down (2013): This paper analyzes the performance of commonly used Hadoop schedulers including FIFO and Fair sharing and compares them with the COSHH scheduler, which has been developed by the authors. So it is a one of the customized algorithm. It is used for overcoming a limitation of existing scheduler. So, a Hybrid solution is introduced for scalable and heterogeneous Hadoop systems.

Ji and Lang Tong, Ting He, Jian Tan, Kang-won Lee, and Li Zhang (2013): The proposed scheduler maintains data specific to the opportunistic environment. Using a Hadoop-based implementation, author compares scheduler with the current Hadoop Fair Scheduler. Author experiments verify that opportunistic scheduler can significantly reduce the variability in job completion times.

Jisha S Manjaly, Varghese S Chooralil (2013): A New TaskTracker aware scheduling has introduced. This scheduler enables users to configure a maximum load per TaskTracker in the Job Configuration itself. The algorithm will not allow a task to run and fail if the load of the TaskTracker reaches its threshold for the job. Also this scheduler allows the users to select the TaskTracker's per job in the Job configuration.

IV. PROPOSED WORK

To Enhance TDW Scheduling algorithms which we will consider in the instant Load and CPU Utilization of the resources available in cluster. We will enhance the TDW algorithm while taking following parameter including memory of resources available in the pool for the scheduling job while scheduling the task.

- CPU Utilization
- Load average of the resources

Following is proposed algorithm for enhancing the existing algorithm.

A. Hybrid Algorithm Based On System Information:

If n has a free slot then
    Sort Jobs in <Customized -Algorithm>
    For j in jobs do
        If n has enough memory resource and CURRENT CPU UTILIZATION/LOAD AVERAGE for job j
            If j has unlaunched task t with data then Launch t on n
                Set j.skipCount = 0
            Break
        Else
            If j has unlaunched task t then
                If j.skipCount >= D then Launch t on n
                Break
            Else
                Set j.skipCount = j.skipCount + 1
            End if
        End if
    End for
End if
As shown in above algorithm we sort job in customized type in place of TDWS style in existing TDWS algorithm. We take CURRENT CPU UTILIZATION/LOAD AVERAGE in place of memory in TDWS style algorithm. So job is executing to much speedy. This will improve the turnaround time for jobs.

B. Comparison of Existing and Customized Algorithms

1) Existing Algorithm
List<Resource> totalResource; // Total configured Data Nodes
Resource allocateResource; // Node to which task will be assigned
List<Resource> availResource = getAvailResource(totalResource); // List the active Data Nodes
List<Resource> highPriorityResource = filterByPriority(availResource); // Find high priority Data Node
allocateResource = allocateRandomResource(highPriorityResource); // Assign Randomly from high priority Data Nodes

2) Customized Algorithm
List<Resource> totalResource; // Total configured Data Nodes
Resource allocateResource; // Node to which task will be assigned
List<Resource> availResource = getAvailResource(totalResource); // List the active Data Nodes
List<Resource> highPriorityResource = filterByPriority(availResource); // Find high priority Data Node
List<Resource> lessCPUUtilizatedResources = filterLessCPUComsumpResources(highPriorityResource); // List at least 5 Percent node having least CPU Utilization
Resource resourceHavingLessLoadAverage = findLessLoadAverageResources(resourceHavingLessCPUUtilization); // Find least Load average node from above node. If load average found same among some node then resource having least CPU Utilization will be selected.

C. Result of Scheduling Algorithms
If job of searching Pattern from multiple input file are given to different scheduler then execution time of different schedulers are also different. When Grep job to find particular pattern data bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduceexamples-2.5.1.jar grep input output 'dfs[a-z.]+' is given to different schedulers than outputs of schedulers are shown in below:

D. FIFO Scheduling

E. Fair Scheduling
F. Capacity Scheduling

Improving Job Scheduling in Hadoop

V. CONCLUSION

Hadoop scheduler resource aware is one the emerging research problem that grabs the attention of most of the researchers. FIFO, Fair and Capacity Scheduling Algorithms are some of the algorithms used for scheduling of jobs.

At present, we have executed FIFO, fair, Capacity and TDWS scheduling algorithms and have measured the Turn-around Time for each of the algorithm. Then after we have improved the TDWS algorithm by considering parameters like CPU Utilization, Load average of the resources. And then calculate turnaround time. We see that turnaround time of Customized TDWS algorithm is decreased as compared to Existing TDWS algorithm.

VI. FUTURE WORK

In Future We will apply same algorithm to Multinode cluster, and measure turnaround time of each algorithms.

REFERENCES

Rice University Houston, America, 2013, pp. 47-53.


