International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 6 – APRIL 2017.

OPTIMIZED JOB SCHEDULING BASED ON WORKLOAD IN HADOOP

Rahul Mistry, Student [#1] and M. Suresh Kumar, Associate Professor [*2]

[#] *M.E, Department of Information Technology, Sri Sairam Engineering College, Chennai,India*

[*] *M.E, Department of Information Technology, Sri Sairam Engineering College, Chennai, India*

***Abstract*— MapReduce is an important paradigm for processing very large scale dataset.MapReduce is one of the most popular methods for processing large dataset it can run on large distributed commodity hardware clusters like Clouds. The Map function uses the input Key/value pair for processing and a reduce function takes the input data in the form of Key/value format merge all the key/value and write it into Hadoop Distributed File System (HDFS).Hadoop uses FIFO scheduling by default. When a flock of jobs is concurrently submitted to a MapReduce cluster, the overall system performance in terms of job response times might be seriously degraded.The challenging issue is that ability of efficient scheduling mechanism for MapReduce. However, traditional scheduling algorithms used by Hadoop not always guarantee good average response times under different workloads. The objective of the research is to study MapReduce analyze different scheduling algorithms that can be used to achieve better performance and also provided some guidelines on how to improve the scheduling in Hadoop environments.**

Keywords*— *MapReduce,Hadoop,Job Scheduling,HDFS

I. INTRODUCTION

Hadoop [1] is an open source platform developed by Doug Cutting and Mike Mike Cafarella under the project called “Nutch” which run on java-based programming framework it used for processing and storing extremely large datasets in a distributed environment.Hadoop developed under Apache project and it sponsored by the Apache Software Foundation. Hadoop framework allows distributed processing of very large data set across the different cluster of computers using MapRedue programming model. It designed to scale-up from a single server to thousand of independent clusters which can run on commodity hardware. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so that it can provide highly available service on the top of the cluster.

a) *Hadoop*

Hadoop makes possible to run any MapReduce [2] application on thousand of system's with the help of commodity hardware, facilities and it has the ability to handle the terabytes of data.It provides fast and reliable analysis of structured data, unstructured data and semi-structured data. Given its capabilities to handle large data. Apache Hadoop is a distributed file system[3] facilitates the brisk transfer of data on the different nodes and it allows to process the data in case of any node failure [4] in a cluster.It prevents from unexpected of data loss, Its even if more than one no of nodes become inoperatiIt's a foundation of big data processing such as scientific analytics, business, and sales planning and processing it is essentially a framework that allows for the distributed processing of large datasets across clusters of computers using a simple programming model. It can scale up from the single node to thousands of nodes, each node will offer local computation and storage.

Hadoop was created by two computer scientists Doung Cutting and Mike Cafarella in the year 2006 under the project Nutch.It inspired by the good MapReduce software framework which breaks the files into small chunks and process them individually in the different cluster and once the processing is done combine them all together.Hadoop was developed under the OpenPGP (standard as defined by RFC4880 also known as PGP) and its an open source technology, the latest release of Hadoop version is 3.0.0-alpha2 on 25 January 2017.

b) *MapReduce*

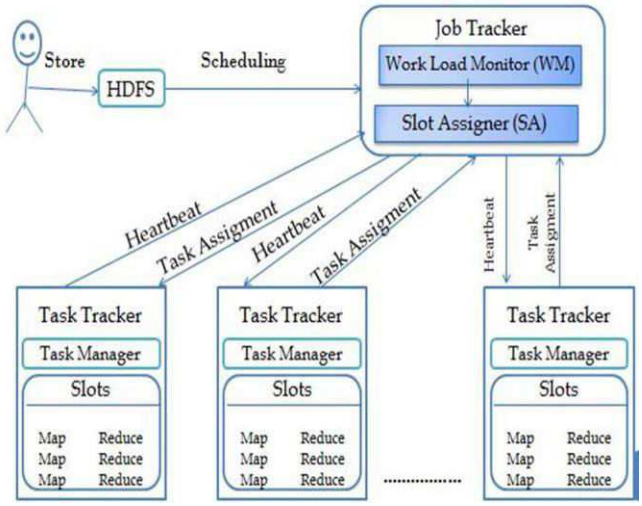
Defin MapReduce is an important high performance computing paradigm software framework used for easy writing of applications which process large-scale data processing [5] in various clusters.Apache Hadoop is an opensource implementation of MapReduce programming model it has been deployed in a large cluster containing thousand of machines. A MapReduce job usually splits the input data-set into two independent chunks which are processed by the mapper and reducer in a completely parallel manner. The MapReduce framework sorts the outputs of the mapper, which are then inputted to the reducer. Typically both the input and the output of the job are stored in a file system called Hadoop distributed file system (HDFS). Hadoop enables resilient, distributed processing of extensive unstructured data sets across commodity computer clusters, The each node in the cluster are used his own storage. MapReduce serves two essential functions: It send out the map output into many clusters, and it organizes and reduces the results from each node into a cohesive answer to a query.

- MapReduce is composed of several components, including:

• JobTracker -- the master node that manages all jobs and resources in a cluster)

International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) ISSN: 0976-1353 Volume 24 Issue 6 – APRIL 2017.

• TaskTrackers -- agents deployed to each machine in the cluster to run the map and reduce tasks



c) *Job Tracker*

Job tracker runs on Apache MapReduce engine ,it is responsible for accepting the client request.Once the requested has been accepted it assign to the TaskTracker where the data locally reside.JobTracker [6] fails to assign the request to the TaskTracker then it will assign the task within same rack where the data locally present. If the TaskTracker fails then JobTracker assign a new Task where the replica of the data exits by default it makes three replica.

- Job Tracker Process

i) Accept Job request from client.
ii) JobTracker consults with NameNode in order to determine the location of the data.
iii) JobTracker locate the TaskTracker where the data reside and submit the task.
iv) The TaskTracker perform the execution of task and send the heartbeat signal to JobTracker.If task tracker fails then JobTracker will again restart the task in some different TaskTracker.
v) One the job has been completed it update its status Job completes.
vi) After completion of task client can pull information from TaskTracker.

d) *Task Tracker*

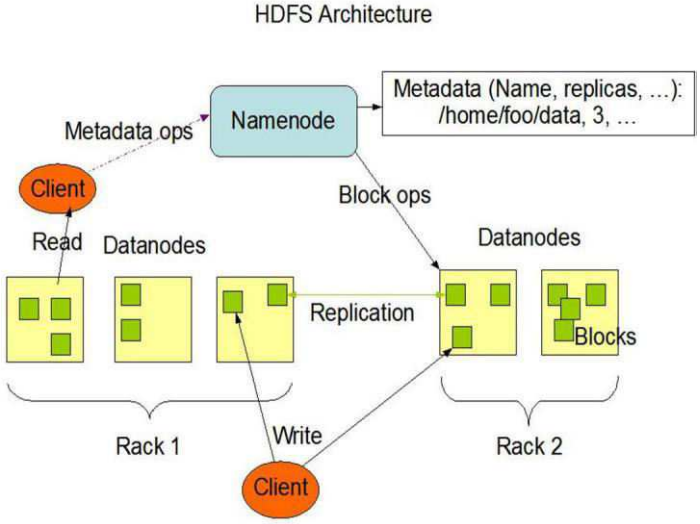
TaskTracker is a cluster node which accept the task from (Map,Reduce and Shuffle) from JobTracker.It keep on sending the heartbeat to the JobTracker to notify that it is alive.Along with this information it is also send the free slots available within it to process tasks.

- Task Tracker Process includes

i) The JobTracker will collect information from the NameNode about the data which is reside in DataNodes. The JobTracker places the client program in the HDFS. Once it placed, JobTracker will assign tasks to TaskTrackers on the DataNodes based on data locality.
ii) TaskTracker handling the starting the Map task on the DataNode by picking up the client program from HDFS.
iii) After completion of Map task an intermediate file created on HDFS.
iv) Result of Map task given as input to Reduce task.
v) Reduce task will works on all data received from map tasks and writes the final output to HDFS.
vi) Once the task executed the intermediate data generated by the TaskTracker will be deleted.

e) *Hadoop Distributed File System (HDFS)*

Hadoop Distributes File sytem [7] is a distributed storage which is design to run on a cluster of commodity hardware. The HDFS id pretty much similar to the existing distributed file system, but the differences are significant. It has highly fault-tolerant freature and low cost feature which makes it different from distributed file system.HDFS are originally build by the Apache under the project called NUTCH web search engine project. It has two main component NameNode and DataNode which manages all these operation



f) *Name Node*

NameNode [8] is the central component of HDFS file system, it keeps the metadata and the directory tree of all files in the file system, and tracks all the cluster where the data file is kept. It does not store the data of these files itself.

Client application directly interacts with the NameNode when they need to locate any file or when they want to copy or delete any file, interact Namenode will redirect the client directly to the Datanode where the real files are stored so that client can directly interact with DataNode.

NameNode is a single point failure for HDFS, where the NameNode fail that time entire filesystem goes offline.The optional is to keep a secondary NameNode on a seperate

machine to create checkpoints if NameNode fails then files can be retrieved through this check points.

g) *Data Node*

A is DataNode [9] where the actual data reside, A functional filesystem has more that one DataNode distributed over multiple clusters.To keep track of DataNode it sends a heartbeat signal to the NameNode to say that DataNode is alive, if NameNode doesn't send any heartbeat then it is considered to be dead and initiate replication of blocks which stored on the DataNode.

At the time of storing data block, it maintains a checksum for that data block. The DataNode keep on updating the NameNode with the block of information periodically, The NameNode will verify the checksum if it matches with the DataNode then it will update that checksum if it doesn't match that means data is corrupted. In this way, NameNode is always aware of any data corruption on DataNode.

II. Hadoop Scheduling

Hadoop scheduler mainly developed to run the very large batch job like web indexing and log mining.User submits their job to the scheduler in a queue, and cluster run this jobs in order.By default, Hadoop uses the FIFO scheduling in order to run the jobs. Since Hadoop became pluggable, several scheduling algorithms developed for it. The result was two scheduling algorithm was developed for multi-user workload approach. The Fair Scheduling which is developed by Facebook and capacity scheduling which is developed at Yahoo.

a) FIFO scheduling

By default Hadoop is using FIFO scheduling, in these, all the user submits their jobs to a queue.Once the job is divided into independent tasks among all the cluster, once the task has been divided the JobTracker allocate the free slot which is available in TaskTracker.One of the better way to implement the FIFO scheduling with the help of parity based. The FIFO scheduling doesn't give the guarantee good response time so it is not useful for an interactive user.

The problem with the FIFO scheduling approach is that the small jobs are stuck behind the big jobs, it causes of poor utilization of the cluster.

b) Fair Scheduling

Fair scheduling approach [10] is developed bt Facebook to manage their cluster.In Fair scheduling approach the resource is divided into jobs in a such a way that all the job get the average and equal share of a viable resource.If there is a single job running in then all the resource allocated to that single job.The Fair scheduling divide the jobs into different jobs pools, and divide the resource among all these pools.The Fair scheduling allow grantee minimum share to pools, so that it is easily restrict the user to allocate resource based on their jobs or group. if the pool share minimum and it is not meeting for the period of time, then scheduler optionally support preemption of jobs which is running in other job pools. It will make room to run the high parity jobs with preemption. When Fair scheduling preemption it will choose the task which launches most recently. Fair scheduling can limit the concurrent task running in the per user and per pool. It will be useful when any user will submit the 100 of the task at a time and it will also ensure the intermediate data not full the cluster disk when the several others task is running off the same cluster.

c) Fair Share with Delay Scheduling

In [11], however, there is always a conflict between fairness and data locality(launching reduce task one the node where input data is present).To address this problem we propose an algorithmic delay scheduling in this first slot we are giving to a task is unlike to have data for it finish, once the finishes so quickly that some slot with data will be free within few second.

In this section, we consider the delay scheduling it allows the job upto T times.

Pseudocode for this algorithm is shown below:

Algorithm 1 Fair Sharing with Simple Delay Scheduling

```
Initialize jb.skip_count to 0 for all jobs jb.
While receiving heartbeat from node N:
  If node N has free slot then
      sort jobs increasing order by no of running task
      for j in jobs do
        if jb has un-launched task t with data on N node then
          launch t on N
          set j.skip_count = 0
       else if jb has un-launched task t
          then if jb.skip_count[] T then
            launch t on N
          else
              set j.skip_count = jb.skip_count +1
          end if
        end if
        end for
    end if
```

Once the job has been skipped by T times, we will launch the arbitary non local task without resetting skip_count.If the job manage to launch the local task again we will set the skip_count back to 0.

The Fair scheduling can be useful for the job with different size and it also supports the preemption.It has the ability to limit the no of concurrent task running per user or per pool. This can be very effective when jobs have their dependency on external factors like web services or Database.

In fair scheduling approach[12][13], there is always a conflict between the fairness and data locality (placing the task on a node where the input data actually reside).To achieve the fair sharing goal two methods are there.

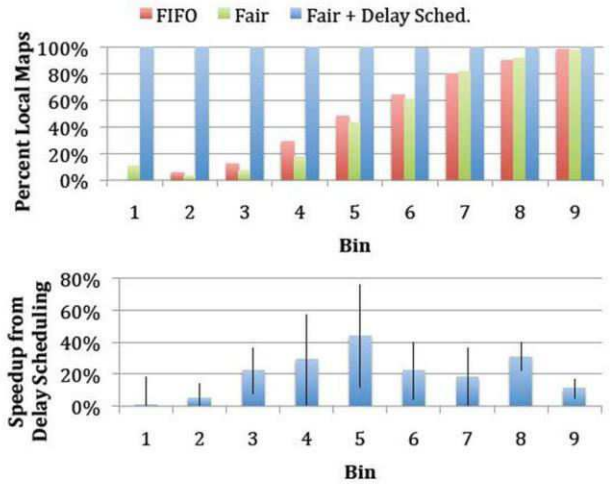
(a) Kill the currently running task and make the slot available for a new task.

(a) Kill the currently running task and make the slot available for a new task.

Killing the currently running task has a disadvantage wasting of work of killing the task.On other hand waiting for currently running job to complete have negative effects on fairness, and it has to wait for an unending amount of time and the new job may not have any input data on that node.The Delay scheduling helps to resolve all these issues by statistically multiplex clusters while the minimum impact on fairness (giving the fair share to new jobs quickly) and archiving the data locality. To reassign the resource to new job reassigning resources approach (killing task from the previous job and make room for the new job) is used and waiting for tasks to finish to assign slots to new jobs. In data, the locality has two problem head-of-line scheduling and sticky slots.

Scan jobs in order given by queuing policy and picking first that is permitted to launch a task, Jobs must wait before being permitted to launch non-local tasks.

If wait < T1, only allow node-local tasks
If T1 < wait < T2, also allow rack-local
If wait > T2, also allow off-rack



The Delay scheduling used temporarily relaxes fairness to improve locality, I t will wait for until the local data node is free.

III. RELATED WORK

Scheduling in Hadoop system has already received lots of attention.Early research by Matei Zaharia et al [14] proposed fair scheduling algorithm to assign resource equally to each user and provide better performance isolation among the user, however, the main objective is not to optimize the system performance.In delay, scheduling was proposed to improving the fair scheduling by increasing the data locality.The [15] quincy scheduler fair assignment is done by considering the data locality by solving minimum network flow problem however, this is very limited because it includes a huge amount of computation complexity.There are several other researchers has done to enhance the MapReduce framework.The Sawzall programming language [16] aim is automatically analyzing huge distributed data files.The difference between Sawzall and MapReduce is that it distributes the reduction in a hierarchical topology-based manner.The Matchmaking scheduling [17] approach makes scheduling decision in round manner, non-local the data local task will be in the first round and non-localtask will be in the second round to avoid the wasting of computation resource. Finally, Condie et al. [18] recently extended the MapReduce architecture to work efficiently for online jobs in addition to batch jobs. Instead of materializing the intermediate data of key/value pairs within every map task, they have proposed pipelining these data directly to the reduce tasks. They further extended this pipeline MapReduce to support interactive data analysis through online aggregation and continuous query processing.Finally to conclude that our proposed algorithm are orthogonal among all the above algorithms are sophisticated and scalable, they do not deal with data locality, as they share only one resource.

IV. EXPERIMENTAL SETUP

In this section, we present a rigorous performance evaluation of our proposed techniques. The details of the system configurations are given along with the conducted experiments with dual 2.20GHz Intel(R) Core(TM)i5-5200U processors, 4GB RAM, and 80GB hard drives.To compute the performance we have install Hadoop 1.1.2 version.The HDFS block size is 128 MB, by default replication of 3.Each TaskTracker has 2 Map slots and one Reduce slot.Our scheduler operates one this parameter: set the size S for both map and reduce slot and the time out the estimate of the size is 10sec.

Workload: we have performed two operation wordcount and maximum and minimum temperature.First, we aim to study the scalability of the different reducing approach in terms of counter and the size of the Map and Reduce.In order to evaluate the system performance, we have compared our scheduling algorithm with native Fair scheduling.We have generated four datasets of 1GB,2GB, and 3GB.

V. CONCLUSION

Apache Hadoop software framework is used for processing large data set organization like Facebook Yahoo are processing Petabytes of data in a day.For processing, these dataset MapReduce paradigm is used.In this paper, we have seen different types of job scheduling like FIFO, this scheduling method is not suitable for small jobs and average job response time is significantly decrease under the case of high variance and this scheduling doesn't consideration other factors like resource, locality and job latency.To overcome these drawback the various pluggable scheduler was interduce. Our work is to improvise the Hadoop Fair scheduling to get

the better result and improve the data locality where the large cluster are used for wide range of workload which includes interactive data processing task.As the consequence, we have witnessed the rise of deployment best practice resources among competing jobs.In addition, we have seen the best practice in which fair sharing resource among competing jobs.It also demonstrated the best cluster utilization involving creation os resource pool to accommodate workload and important tuning efforts.We showed Fair with delay scheduling performs well, with widely and that precise job information not essential for scheduler function properly.Our experiment result we have compared the Fair with the delay with widely used native Fair scheduling algorithm, it indicates that both intractive and efficiency for samll and large job size with good data locality, which have not considered in native Fair scheduling method.

REFERENCES

[1] Apache Hadoop. (2014). [Online]. Available: http://hadoop.apache.org/

[2] Mahwish Arif,Hans Vandierendonck and Dimitrios S. Nikolopoulos "A scalable and composable map-reduce system"DOI: 10.1109/BigData.2016.7840854,Date Added to IEEE Xplore: 06 February 2017.

[3] Kaiyang Qu,Luoming Meng and Yang Yang "A dynamic replica strategy based on Markov model for hadoop distributed file system (HDFS)"DOI: 10.1109/CCIS.2016.7790280,IEEE Xplore.

[4] Kapil Pandey, Anand Gadwal and Prashant Lakkadwala "Hadoop multi node cluster resource analysis" DOI: 10.1109/CDAN.2016.7570925, IEEE Xplore.

[5] Srinath Perera "Large scale data processing in real world: From analytics to predictions" DOI: 10.1109/ICTER.2014.7083870 ,IEEE Xplore.

[6] Jisha S. Manjaly,Varghese S. Chooralil "TaskTracker Aware Scheduling for Hadoop MapReduce" DOI: 10.1109/ICACC.2013.103 ,Publisher: IEEE.

[7] Farag Azzedin "Towards a scalable HDFS architecture" DOI: 10.1109/CTS.2013.6567222,Publisher: IEEE.

[8] https://wiki.apache.org/hadoop/NameNode

[9] https://wiki.apache.org/hadoop/DataNode

[10] Yi Yao,Jianzhe Tai and Bo Sheng "LsPS: A Job Size-Based Scheduler for Efficient Task Assignments in Hadoop" DOI: 10.1109/TCC.2014.2338291 ,Publisher: IEEE.

[11] Matei Zaharia,Dhruba Borthakur,Joydeep Sen Sarma "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in ClusterScheduling" ISBN:978-1-2344-577-2.

[12] Matei Zaharia,Dhruba Borthakur,Joydeep Sen Sarma "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling" ISBN: 978 -1-60558-577-2 ,doi10.1145/1755913.1755940.

[13] Asmath Fahad Thaha,Anang Hudaya Muhamad Amin and Subarmaniam Kannan "Data location aware scheduling for virtual Hadoop cluster deployment on private cloud computing environment" DOI: 10.1109/APCC,Publisher: IEEE.2016.7581422.

[14] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker,and I. Stoica, “Job scheduling for multi-user mapreduce clusters,” University of California, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-55 , Apr. 2009.

[15] M. Isard, Vijayan Prabhakaran, J. Currey, U. Wieder, K. Talwar,and A. Goldberg, “Quincy: Fair scheduling for distributed computing clusters,” in Proc. 5th Eur. Conf. Comput. Syst., 2009, pp. 261–276.

[16] Rob Pike, Sean Dorward, Robert Griesemer, and Sean Quinlan. Interpreting the data: Parallel analysis with sawzall. Sci. Program.,13(4):277–298, 2005.

[17] C. He, Y. Lu, and D. Swanson, “Matchmaking: A new mapreduce scheduling technique,” in IEEE Cloudcom. IEEE, 2011.

[18] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein,Khaled Elmeleegy, and Russell Sears. Mapreduce online. Technical Report UCB/EECS-2009-136, University of California at Berkeley,2009.

International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 24 Issue 6 – APRIL 2017.