ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

DYNAMIC ALLOCATION OF SLAVE NODES IN HADOOP FRAMEWORK

Malarvizhi.P[1], *Dr.K.Amshakala*[2]

1. Department of Computer Science and Engineering and Information technology, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu -641014, INDIA
2. Associate Professor, Department of Computer Science and Engineering and Information technology, Coimbatore Institute of Technology, Coimbatore, Tamil Nadu -641014, INDIA
Email:fragranceflower08@gmail.com, amshakalacit@gmail.com

Abstract— Scheduling in hadoop cluster is the way by which the tasks are assigned to multiple nodes in a cluster for computation. Optimal scheduling strategy aims at assigning tasks to appropriate number of nodes and enhance the overall performance in computation. As real-time scheduling of dynamic and multitasking workload is a challenging issue, scheduling becomes essential to increase the throughput of computing. In this project, dynamic allocation of slave nodes is done on hadoop framework, before executing tasks on hadoop. The allocation is done based on the file size. This eventually impacts the performance of the entire system.

INDEX TERMS— BIG DATA, HADOOP, MAP REDUCE, JOB SCHEDULING ALGORITHM, MULTITASKING WORKLOAD.

I.INTRODUCTION

Hadoop is an open-source distributed framework that helps to store and process big data in a distributed environment across various nodes of computers using simple programming models. It is designed to scale from single servers to thousands of machines, each having its own storage and processing capability. Its distributed nature allows rapid data transfer rates among nodes and also it keeps on working in case of any node failure. In hadoop, computation is not performed by one single powerful computer but by various low cost computers in a distributed and parallel manner. Fluctuating workloads, multitasking, big data are the main ingredient of large scale business and scientific applications. A large volume of data is being generated by various applications. The issue is to handle these large set of data being generated and to process it. Map reduce programming [12] will help to process these large volume of data. It consists of map phase and reduce phase. In map phase, filtering and sorting is done and in reduce phase the summary operation is done. Hadoop framework consists of master nodes and slave nodes for data processing. The data which need to be processed is given to the master node, which splits the data among slave nodes and processes them simultaneously. Master node and slave node will have TaskTracker and DataNode. In addition to this, master node will have JobTracker and NameNode. The Job of the TaskTracker is to process the smaller pieces of tasks, and the job of the DataNode is to manage the piece of data that is given to this particular node. The role of JobTracker component is to break bigger task into smaller pieces of tasks and send each small piece of task to the task tracker running on the slave

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

machine. The NameNode keeps the index of data that are processed on each DataNode. TaskTracker and Job tracker are responsible for MapReduce and the entire data node and name node are responsible for executing the files in HDFS. The execution of task which is running on hadoop will contact the master node .If any slave in the hadoop cluster fails then the built-in fault tolerance capability overcome those failure. By default hadoop maintains three copies of each file and store these copies across various slave nodes in the cluster. If any TaskTracker fails, hadoop has a capacity to detect which TaskTracker has failed and it fetches the tasks which are assigned to the failed node and assign the same tasks to any other TaskTracker of slave node for processing. If the master node fails hadoop has the back up of data index tables that are maintained by the NameNode. Backup copies are maintained in more than one node in the cluster to use as fault-tolerance.

II.RELATED WORK

There are various pluggable scheduling available for hadoop framework. Since they are pluggable they are easily added to hadoop framework. The various scheduling strategy [1-3] that are proposed by various authors, have their own usage according to their implementation. Work proposed in [2] describes dynamic scheduling for I/O intensive jobs and mainly focuses on I/O waiting time during execution of task. If CPU has high I/O wait time, then more tasks are added to newly available free slots. Master node will monitor the state of CPU using the command /proc/stat in a periodic interval; this command will return how much time is spent by each CPU in a cluster. If a particular CPU takes additional time, then free slots are added to process more tasks. Authors in reference [1] propose a multitasking scheduling model which has task classes and VM to execute each task. It will have a set of tasks that are of same type and can be executed at a time, i.e. suppose if it have C task classes then it's index can be represented by c. Each task class has an index to reference it. VMs are built on top of each machine for scheduling and computing a given task and each VM has the ability to support each task class which has unique characteristics. Each VM will support each task class for computation. Author in reference [4] propose two levels of processing to schedule the task. The first one will fill empty slots based on number of hosted map task and its input data replication scheme, and the second one will be based on replicas of the task's input data, i.e. usually hadoop will replicate data for fault tolerant purpose. Their work discusses an idea to use the replicated copies of data to serve slow nodes that are called as stragglers. Authors in reference [3] propose locality aware scheduling which works based on input data location and size. When data is moved repeatedly to distant nodes, then that will become a bottleneck. It gives the idea to schedule reduce task using data locality, In general hadoop will schedule map tasks which are in proximity to input splits. Authors in Reference [5] propose delay technique. Using this technique, if any job which comes first does not have local map task then the method will delay it and tries to assign another job's local map tasks. Maximum delay can be specified to accomplish this task. Authors in reference [6] propose size based scheduling; which needs to know job size information in prior to scheduling. Using this technique the job size information which is not known prior is identified. It uses virtual time and aging function. Using this technique both small and large jobs will not suffer from waiting or starvation.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

III. DYNAMIC ALLOCATION OF SLAVE NODES

In Hadoop framework, the slave node can be added whenever needed, i.e. when slave nodes are added to hadoop then it is fixed, in this paper an idea to dynamically allocate the slave node based on the input file size.

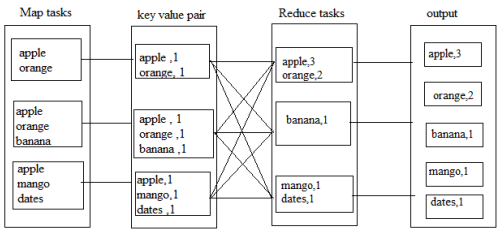

FIG 1: Scheduling behind MapReduce

To scale the slave node, frequently update the etc/hosts file; this will update the slave node according to the scaling factor. This scaling factor should be determined well to enhance the performance, considering various factors into account we can determine this scaling factor. This etc/host file will be referenced by the hadoop to reach its slave node in the network, so the proper definition of the slave node's name and it's IP address must be defined in this file, only then master can able to allocate the task to the slave nodes, so by frequently updating this file can help to scale the slave node. Doing this kind of pre scheduling task will have impact on scheduling performance. In case of fixed slave node the input data will split among the fixed slave node, not considering whether the task is small or big. Whereas in dynamic scaling, it is vice versa, i.e. if the task to be executed is very small the number of slave node can be limited, in the same way if the task is very large the number of slave node can be increased dynamically. So that proper resource can be allocated to execute the task in the hadoop framework. Assume that hadoop framework has fixed slave node say 5 for example. The large task also gets executed in that 5 nodes and small task will also gets executed in that 5 nodes. But in dynamic scaling method if the task is small the slave node can minimized to 2 and then the task can be submitted to the scheduler, so that task gets executed only in that 2 nodes, in the same way if the task is large then we can increase the slave node to 8 and then execute that job in that 8 node. So that larger task will take less time to compute the task. To accomplish this task we need to define the scaling factor. This scaling factor can be based on the file size in the particular batch. Hadoop itself follows the batch processing mechanism so that jobs that are similar can be grouped and the file analysis should be done for that particular batch and scaling factor is determined for that particular batch. The scaling factor can be identified by the following,

Let us assume that there are 'n' batch of files, each batch of files will contain 'm' files of same types, so we need to find the average file size which is consider to be scaling factor. So the scaling factor can be determined using the following formula,

$$\text{Scaling factor} = \sum_{i=1}^{n} \sum_{m=1}^{j} \frac{m_j}{i}$$

To find out scaling factor it is not enough to consider average file size of the particular batch,but also several factors like the total availabity of node, maximum file size, timing constraints etc. After finding the scaling factor, the number of slave nodes must be determined to execute the batch of files. So it can be determined by the

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

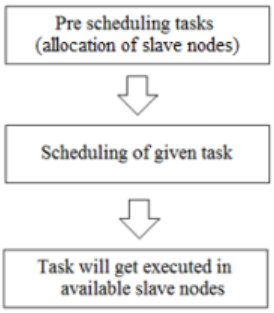

Fig 2: Scaling slave nodes

$$\#\text{ of Slave nodes} = \frac{Input\ file\ size}{scaling\ factor}$$

Input file size is the current input file size that is going to get processed by the hadoop framework. Using this we will be able to allocate slave nodes dynamically before actual scheduling work begins.

IV ANALYSIS

For analysis we considered two file of size 700 MB and 800 MB and scaling factor is 250. when considering 700MB file size slave nodes allocated is 2, and when considering 800MB file, the slave nodes allocated is 3. The scaling factor should be determined so that they enhance the overall performance. After having various analysis this scaling factor can be fixed. This scaling of node has an impact on scheduling performance. A scaling of node can be analyzed with the default scheduler like FIFO, Fair and Capacity to show improvements and the impact of dynamic allocation of nodes can be studied.

V. CONCLUSION

Advantage of using dynamic allocation of slave nodes improves the overall performance of the system by allocating appropriate number of slave nodes based on the size of the input file of the executing task. Increasing the throughput of the cluster benefits with the following things i.e. It able to increases number of the slave nodes in the cluster if larger file needs to be executed and It can helps in improving the throughput and with balanced cpu utilization in all the slave nodes.

REFERENCES

[1] Zhang, Fan, et al. "Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud." Emerging Topics in Computing, IEEE Transactions on 2.3 (2014): 338-351.

[2] Kurazumi, Shiori, et al. "Dynamic processing slots scheduling for I/O intensive jobs of Hadoop MapReduce." Networking and Computing (ICNC), 2012 Third International Conference on. IEEE, 2012.

[3] Hammoud, Mohammad, and Majd F. Sakr. "Locality-aware reduce task scheduling for MapReduce." Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. IEEE, 2011.

[4] Ibrahim, Shadi, et al. "Maestro: Replica-aware map scheduling for mapreduce." Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. IEEE, 2012.

[5] He, Chen, Ying Lu, and David Swanson. "Matchmaking: A new mapreduce scheduling technique." Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. IEEE, 2011.

[6] Pastorelli, Michele, et al. "HFSP: Bringing Size-Based Scheduling To Hadoop." (2015).

[7] Rao, B. Thirumala, and L. S. S. Reddy."Survey on improved scheduling in Hadoop Map Reduce in cloud environment"arXivpreprint arXiv:1207.0780 (2012).

[8] Guo, Yingjie, et al."The Improved Job Scheduling Algorithm of Hadoop Platform." arXiv preprint arXiv:1506.03004(2015).

[9] Patil, Mr AU, Mr TI Bagban, and Mr AP Pande.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

"Recent Job Scheduling Algorithms in Hadoop Cluster Environments: A Survey." International Journal of Advanced Research in Computer and Communication Engineering 4.2 (2015).

[10] Wang, Dan, Jilan Chen, and Wenbing Zhao. "A task scheduling algorithm for Hadoop platform." Journal of Computers 8.4 (2013): 929-936.

[11] Nanduri, Radheshyam, et al. "Job aware scheduling algorithm for mapreduce framework." Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. IEEE, 2011.