International Journal of Engineering Applied Sciences and Technology, 2019
Vol. 4, Issue 8, ISSN No. 2455-2143, Pages 119-122
Published Online December 2019 in IJEAST (http://www.ijeast.com)

A DEEP DIVE INTO LOAD BALANCING TOOLS FOR HADOOP APPLICATION MANAGEMENT

Kaveri T Hombal
M. Tech Scholar,
Dept. Of. Computer Science & Engineering,
BMS Institute of Technology & Management
Yelahanka, Bengaluru

Anjan K Koundinya
Associate Professor and PG Coordinator,
Dept. Of. Computer Science & Engineering,
BMS Institute of Technology & Management,
Yelahanka, Bengaluru

Abstract-**Hadoop has become an important tool for the researchers and scientists in order to store and analyze huge amount of data. This huge data is placed in Hadoop with the help of Hadoop Distributed File System (HDFS). Block placement policy is employed in HDFS to split a really huge file into blocks and place these block across the cluster in an exceedingly distributed manner. Basically, Hadoop and HDFS are designed to works expeditiously on the consistent cluster. However during this era of networking, we cannot think about having solely a cluster of consistent nodes. So, there's the necessity of storage policy which will work expeditiously on each consistent still because of the heterogeneous cluster. Thus, the need of applications which will be executing in a time-efficiently manner and supporting consistent still because *the heterogeneous setting will be sufficient. In Hadoop data***1.
MapReduce is programming framework for writing Map-Reduce applications which enables them to run on the distributed platform in parallel. MapReduce permits the applications to run on Hadoop environment.

Hadoop uses HDFS block placement policy to place the data blocks on nodes. Hadoop cluster gets unbalanced every now and then, because of overutilization of few nodes against the less used nodes or recently created other new nodes with no blocks hold on them. To resolve this case, Hadoop encompasses an inherent tool known as HDFS Balancer.

*KeyWords- **Hadoop, Load Balancing, Data Blocks, HDFS, Storage.***

I. INTRODUCTION

Hadoop [1] has been widely used because of its ability to make utilization general computers. Hadoop has essentially 3 most vital constituents: The Hadoop Distributed File system (HDFS), MapReduce and Yet Another Resource Negotiator (YARN) [3]. HDFS permits nodes to store information on the distributed cluster. HDFS is known to be adherent and it can be deployed on any machine.
Hadoop stores the data blocks on the various different nodes of the cluster. The policy of storing the blocks in Hadoop, helps to distribute the data block uniformly amongst the cluster nodes. This policy enables the data blocks to be placed in an consistent manner with the subsequent approach:

- Splitting the large number of files into blocks and replicates the blocks with respect to the replication issue that has been already outlined within the hdfs-site.xml file.
- That data node itself. Otherwise, place it any of the data node of the cluster.
- The next exact copy of the block are going to be placed on different racks of the nodes if offered, or even placed on a similar rack of the initial replica.
- Third copy needs to be placed on any node of the rack where the second copy is already placed.

II. HDFS BLOCK PLACEMENT POLICY

By making use of block placement policy offered in HDFS, block are mapped to a process within the same node by Data Locality, however typically once you're addressing huge data, there is a need of mapping the data block to processes over multiple nodes. To influence this Hadoop has a functionality to copy that data block wherever mappers are running. This creates various performance degradation particularly on heterogeneous cluster because of I/O delay or congestion in network. In order to balance the data block on specific nodes i.e. custom block placement here we use an efficient algorithm, solely by dividing total number of nodes among two classes like: homogenous vs. heterogeneous or high performing vs. low performing nodes. This policy helps to attain load balancing among the nodes and that we will place data blocks truly wherever we would like our data to be placed for the processing.

III. HDFS BALANCER

Sometimes within the cluster, load imbalance might occur whenever the load is not uniformly distributed on the nodes. In case of Hadoop, another reason behind cluster imbalance is because of its elasticity. We can also add data nodes into existing cluster at any time. This might cause cluster imbalance as newly added data nodes won't have any data blocks. It is very important to have a balanced cluster in order to get an efficient results. HDFS Balancer tool is used for balancing the load on Hadoop cluster.

For moving number of blocks it relies on the intended threshold limit that is set on the cluster. When HDFS Balancer tool is made to run with its default threshold limit of 10%, it attempts to keep up an in general 10% contrast of disk utilization from the general cluster usage. That implies if generally speaking cluster utilization is 60% of the cluster limit than every node will have disk utilization approximately in the middle of 50% to 70% [4].

There exist some issues with HDFS Balancer, first is that, it moves block of information without investigating, what is the threshold limit of nodes where information blocks are being moved and second, there exist no method which can be utilized to control the translation behavior of information blocks of determined document/datasets as it were. Also it is imperative to think about that moving data hinders starting with one node then onto the next requires high data transfer capacity, so that cluster balancing can occur quicker.

IV. TECHNIQUES USED TO SOLVE A LOAD BALANCING ISSUE IN HADOOP

Hadoop has two common ways for overcoming the load balancing issues: First, cluster Re-balancing by migrating the data or Second is, migrating the Task itself. Researchers are concentrating mainly on the Task Migration instead of Data Migration, as data transfer among the hubs will results in increase of time complexity.

4.1 Hadoop Schedulers:

Can achieve load balancing to reduce the information migration of job or task to some extent [8]. Various kinds of Hadoop programming algorithms are evaluated based on latency time, completion time, and data locality. For experiment purpose, six applications of Big Data are enforced by making use of three different scheduling queue configurations like single queue, multi-queue, and mixed multi-queue. Most of the experiments were conducted by fine-tuning scheduling policy for Hadoop environment. The subsequent conclusion are drawn:

- In single queue, Fair-DRF outperforms in terms of execution time and effective resource usage capability as compared to remaining three. The sole issue that it lacks is that C.P.U. usage time may be a bit high compared to Fair-Fair scheduler.
- In multi-queue, Fair-FIFO is that the most suitable choice if we tend to take into account workload waiting time, completion time, turnaround time, and C.P.U. usage. Fair-Fair is best only if resource utilization is vital.
- In mixed multi-queue, Fair-DRF is that the most applicable alternative with relation to resource utilization and workload execution performance.

4.2 Apart from Built-In Scheduler which is provided by Hadoop ,Researchers have developed Improved Scheduling Algorithm to contribute it for Load Balancing[6,12]:

Distinguishing moderate task is not sufficient. What is important is recognizing the task that will hurt response time the most, and doing as fast as possible.

Distinguishing an undertaking as a slow poke when it has kept running for in excess of two standard deviations than the mean isn't useful for diminishing response time: at this point, the activity could have just run 3x longer than it ought to have! Thus, LATE depends on evaluated time left, and can identify the moderate task at an opportune time. A couple of different components, for example, a top on theoretical assignments, guarantee sensible conduct. Through the inference of heterogeneity on appropriated applications. There are four exercises:

1. Make a choices early as possible, instead of depending upon the base decisions for measurements of mean and variance.
2. Instead of using progress time, make use completing time to make a priority among tasks to speculate.
3. All the Nodes are not same too. Abstain from relegating theoretical assignments to slow nodes.
4. Resources are valuable. Tops ought to be utilized to make preparations for over-burdening the framework.

This work is likewise identified with multiprocessor undertaking booking with processor heterogeneity [8] and with errand duplication when utilizing reliance diagrams [11].Multiprocessor errand booking work centers around conditions where processor speeds, albeit heterogeneous, are known ahead of time, and undertakings are profoundly associated due to entomb task correspondence. This implies, in the multiprocessor setting, it is both conceivable and important to plan task assignments ahead of time, though in MapReduce, the scheduler must respond progressively to conditions in the earth.

In MapReduce, Speculative execution shares some concepts with “speculative execution” in DFS, configuration management, data gathering. However, whereas LATE focuses on approximation that running a tasks may be overtaken to cut back the response time of a distributed computation, strong scheduling algorithm:

- LATE, uses finish times which is estimated to speculatively execute the tasks that hurt the response time foremost.
- LATE performs considerably higher in real workloads on Amazon’s Elastic compute Cloud than Hadoop’s de-fault speculative execution algorithmic.

4.3 Load Balancing Based on Disk Latency And Disk Space Utilization:

In case of comparable disk latencies among the data nodes, balancing the blocks uniformly may be an appropriate selection. Moreover, with respect to time, because of mechanical issues and dangerous sectors the disk latencies will increase. Further, the disks which crash and become non-functional are replaced with newer disks that may be of newer generation and might have larger rate. This results in non-uniformity in terms of disk within the cluster that is otherwise unvaried, and balancing uniformly in line with disk utilization might not offer optimum job run time.

In order to overcome this problem disk latency aware balancer was introduced, that balances the cluster taking each space utilization and disk latency into thought. This strategy for Balancing the disk makes positive that an occasional latency disk gets higher variety of blocks compared to high latency disk.

On the heterogeneous cluster, authors have tested their custom block placement approach and show up to 20% improvement in running time of the job.

4.4 Dynamic Load Balancing Algorithm For Heterogeneous Clusters:

The algorithmic uses a sliding window-based approach to attain important improvement of the Hadoop job processing 11]. The MapReduce programming framework is one candidate framework for large-scale process, and Hadoop is its open-source implementation. This framework consists of the Hadoop Distributed File System and also the MapReduce for computation capabilities. However, the MapReduce framework doesn't allow data sharing for computation among the computing nodes. Implementation of a sliding-window algorithmic rule for data sharing for computation dependency in MapReduce is meant to facilitate the information processing in a successive order, e.g., moving average. The algorithmic rule utilizes the MapReduce job metadata, e.g., input split size, to organize the shared information between the computing nodes while not violating the MapReduce fault tolerance handling mechanism.

4.4 Hadoop tool for load balancing:

Using custom block placement policy Hadoop [12] tool has been designed for load balancing by considering hardware generation of the nodes. An adaptive approach is proposed combined with a PMK-ELM, prediction model, and a multi-object selective algorithm, TS-NSGA-II. The PMK-ELM can facilitate the prediction of execution time of tasks, whereas for selecting a suitable number of reducers the TS-NSGA-II was designed. The experiment results have shown that both models achieve a good performance. About 47–55 s have been saved during experiments. Only 1.254% of differences on hard disk occupation were made in terms of storage efficiency, among all scheduled reducers, which achieves 26.6% improvement than the original scheme.

4.5 Load Balancing through Block Rearrangement Policy for Hadoop Heterogeneous Cluster:

The processing capacity of CPU is hold by the custom block placement policy, during block placement. This approach will be helpful to minimize the internode and inter-rack transfer in MapReduce. This shows that only data blocks of specific file are placed to specific nodes only. As a result, this approach will not affect the overall load balancing of cluster as rest of the files won't be affected. Experimental results prove that this scheme can be applied to both homogeneous as well as to heterogeneous cluster.

V. CONCLUSION

There are numerous strategies for equalisation of the load in Hadoop. The results that are drawn here are often application dependent and future investigator will take a look at constant with completely different application varieties. Most of the analysis work for load equalisation is either that specialize in load equalisation throughout planning [8, 9] or load equalisation throughout MapReduce [11,12] part solely. These approaches reach the higher results particularly just in case of the heterogeneous cluster.

VI. REFERENCES

[1] "Welcome to Apache™ Hadoop®!",Hadoop.apache.org, 2018. [Online]. Available: http://hadoop.apache.org/. [Accessed: 09- Jul- 2018].

[2] Vavilapalli, Seth.S, Saha.B, Curino.C, O'Malley.O, Radia.S, Reed.B, Baldeschwieler.E, Murthy.A, Douglas.C, Agarwal.S, Konar.M, Evans.R, Graves.T, Lowe.J and Shah.H (2013) "Apache Hadoop YARN", *Proceedings of the 4th annual Symposium on Cloud Computing* SOCC '13.

[3] "HDFS Balancers | 5.7.x | Cloudera Documentation", *Cloudera.com*, 2018. [Online] Availablehttps://www.cloudera.com/documentation/enterprise/57x/topics/admin_hdfs_balancer.html. [Accessed: 07- Jul- 2018]

[4] "Hadoop Fair Scheduler". https://hadoop.apache.org/docs/r2.7.2/hadoopyarn/hadoop-yarn-site/FairScheduler.html. Accessed 20 May 2017.

[5] Pike.R, Dorward.S, Griesemer.R and Quinlan.S (Oct. 2005) "Interpreting the Data: Parallel Analysis with Sawzall", *ScientificProgramming Journal*, 13 (4): 227298.

[6] Olston.C, Reed.B, Srivastava.U, Kumar.R and Tomkins.A(June 2008) "Pig Latin: A Not-So-Foreign Language for Data Processing". *ACM SIGMOD 2008*.

[7] Ucar.B, Aykanat.C,Kaya.K, and Ikinci.M (Jan 2006) "Task assignment in heterogeneous computing systems". *J. of Parallel and Distributed Computing*, 66 (1): 32-46.

[8] Manoharan.S. "Effect of task duplication on the assignment of dependency graphs". *Parallel Comput.*,

27 (3):257-268, 2001

[9] Dharanipragada, Padala.S, Kammili.B and Kumar.V (*2017 IEEE)* "Tula: A disk latency aware balancing and block placement strategy for Hadoop".*International Conference on Big Data (BigData).*

[10] Liu.Y, Jing.W,Liu.Y, Lv.L,Qi.M and Xiang.Y(2016) "A sliding window-based dynamic load balancing for heterogeneous Hadoop clusters", *Concurrency and Computation: Practice and Experience*, vol. 29, no. 3, p. e3763.

[11] Liu.Q, Cai.W, Shen.J, Fu.Z, Liu.X and Linge.N (2016) "A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment", *Security and Communication Networks*, vol. 9, no. 17, pp. 4002-
4012.

[12] Shah.A and Padole.M, "Load Balancing through Block Rearrangement Policy for Hadoop Heterogeneous Cluster"