International Journal of Grid and Distributed Computing
Vol. 13, No. 2, (2020), pp. 567 – 573

Load Rebalancing for Heterogeneous Distributed System

Manoj N. Rathod
manoj97rathod@gmail.com

Dr. Karveer. B. Manwade
mkarveer@gmail.com

D.Y. Patil College of Engineering and Technology, Kolhapur
Shivaji University, Kolhapur
Sanjeevan Engineering and Technology, Institute,Panhala
Shivaji University, Kolhapur

Abstract

Distributed file systems are basic building block for cloud environment based on the Map Reduce programming model. In distributed file systems nodes simultaneously serve storage and computing function. When the file upload, it is partitioned in to number of chunks and allocate to distinct nodes so that map reduce can be performed in parallel among the nodes. In cloud atmosphere disaster can arise, nodes can be added, deleted, upgraded and replaced in the system. In accumulation to this files can be dynamically created, deleted and appended. It might result in load disproportion problem in distributed file system, means file chunk are not distributed uniformly between the nodes. To overcome this problematic a distributed load rebalancing algorithm is proposed. The aim is to assign file chunk as uniformly as possible among the existing node in the Hadoop cluster. The distributed load rebalancing algorithm is incorporated in Hadoop Distributed File System (HDFS) by using cluster environment. The proposed Load rebalancing algorithm decrease the movement cost and reallocate the file chunk uniformly between the nodes. This process is repeated until all node balance.

Keywords: *Hadoop, Load Balancing, HDFS, Overloaded, Under loaded.*

1. Introduction

Failure is the standard in distributed computing atmosphere and the file chunk may be improved, substituted and included in the system, which leads to load imbalance in the distributed file system. It means that the file chunks are not distributed regularly among the nodes. In distributed file system, the load of a node is typically proportional to the number of file chunks the node store/possesses. The files in distributed system can be randomly created, deleted, appended and nodes can be upgraded, replaced and included in the file system. The file chunks are not distributed as uniformly as possible among the nodes. Load balance among storage nodes is a critical problem in distributed system. In a load balanced the resources can be well utilized and exploiting the performance of distributed system application. The distributed file systems HDFS is running on large clusters of hardware and is like the GFS of Google, HDFS. The architecture of HDFS is master/slave and HDFS cluster has a Namenode and multiple Datanodes. HDFS architectures consist of Namenode which is the central server, equivalent to master in GFS. Datanode is similar to chunk server of GFS which is responsible for managing storage on Datanode, creating block, deleting block, copying block, and etc. HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. An application can specify the number of replicas of a file. Hadoop is best known for MapReduce and its HDFS. When the Hadoop cluster goes for a period of time, the data load balancing will be broken since nodes are added and deleted dynamically, become a performance blockage, as they are unable to accommodate a large number of file accesses or load files due to clients and MapReduce applications. Our proposed

ISSN: 2005-4262 IJGDC

Load Rebalancing Algorithm is to assign the chunks of files as uniformly and perform well whenever the file created and nodes can be added, replaced, upgraded in the system. Here we are allowing for different types of files format and dissimilar types of file size, which we are going to balancing among the data nodes. Our proposed Load Rebalancing algorithm can be integrated with hadoop multimode cluster.

2. Literature Survey

Hung-Chang et.al. proposed a novel load-balancing algorithm by choosing random nodes for migration to deal with the load rebalancing problem in extensive, dynamic and distributed file systems in clouds. Developing distributed file system in construction system strongly be contingent on a central node for chunk restructuring [1].

Hung-Chang et.al evaluated load balancing algorithm relies on fractional knowledge of chunk distribution in the system. Peer-to-Peer (P2P) network structures each peer independently estimates the probability distributions for the capacities of participating peers and the loads of virtual servers based on fractional knowledge of the system. With the approximated probability distributions, each peer identifies whether it is under loaded and then reallocates its loads if it is overloaded [2].

Konstantin Shvachko et.al. identified characteristic of Hadoop. The HDFS provides distributed file system and a context for the analysis and alteration of very large data sets using the MapReduce programming paradigm. Hadoop is the separating of data and computation across several hosts. The balancer is a tool that balances disk space usage on an HDFS cluster. It takes a threshold value as an input parameter. Data might not always be placed consistently across data nodes [3].

Quang Hieu Vu et.al. discovered global load balance Peer-to-Peer (P2P) structures have rapidly grown in popularity and have become a dominant means for sharing resources. In these systems, load balancing is a key challenge. For structured P2P systems. Each node in HiGLOB conserves the load information of nodes in the systems using histograms. This enables the system to have a global view of the load distribution and henceforth facilitates global load balancing. They have separated the system into no overlying groups of nodes and maintain the average load of them in the histogram at a node. Maintaining the load information of each node increased overhead of system [4].

Kun Liu et.al. presented the enhanced algorithm for balancing the overload racks. This improved algorithm paradigms prior balance list which includes overload machines and next for balance list, experiments show that the improved method can balance the overload racks in time and reduce the chance of breakdown of these racks [5].

3. Load rebalancing algorithm

A client uploads a file, a file is separated in to number of fix size chunks. The load balancing algorithm first collects the status of existing nodes and existing load on the node. According to status of node, the load balancing algorithm assigns the chunks uniformly among the existing nodes (data node). The Name node can preserve the metadata i.e. which chunk is kept on which data node etc. If chunks are deleted, created dynamically in any of the node or if any node added or deleted from the cluster to the cluster. This results into load imbalance in the node. So we migrate the chunks from overloaded node to under loaded node by applying the load balancing algorithm.

Below Figure 1 shows that the proposed architecture of load rebalancing algorithm.

ISSN: 2005-4262 IJGDC

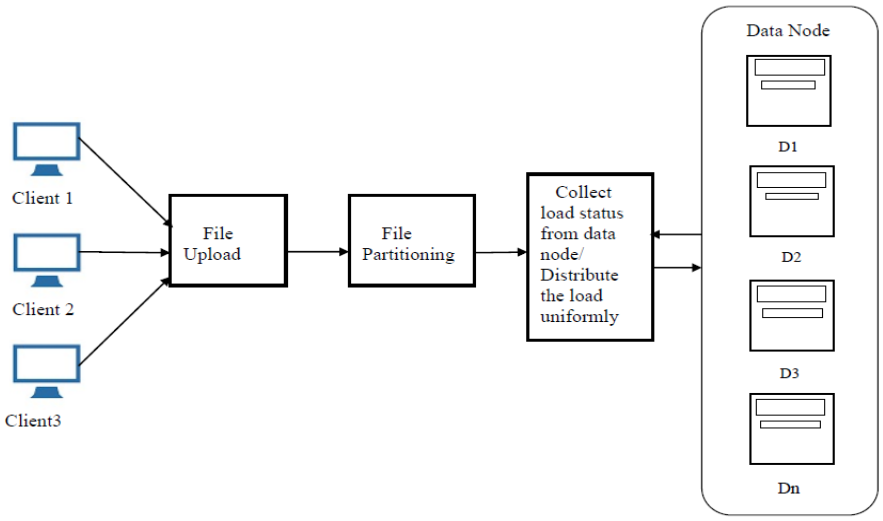

Figure 1. Architecture of Load Rebalancing Algorithm

The system is divided into four different phases.
1. To upload file from local to HDFS
2. File Partitioned into equal size of chunk
3. Collect Load Status from node
4. Distribute the load

The load rebalancing algorithm will transfer data from one data node to another, if the free space on data node drops below a threshold value -t. In the occurrence of file are created, deleted dynamically in the node or if any node added or deleted from the cluster. The algorithm can make the cluster balance with the threshold value -t 10 % or 15% (the threshold value can be set by user). If the threshold value of each data node is less than 10% or 15%, then the cluster balance is finished. According to the typical rate of all the data nodes storage space, the nodes are divided into two type light loaded node and heavy loaded node .The load balancing policy is that first heavy loaded node migrate the file chunk to the light loaded nodes.

1) Firstly, whenever user wants to upload file in cluster, the file is separated in to number of fix size chunks 64Mbytes.
2) Identify whether the node i is under loaded node (light) or overloaded node (heavy). A node is under loaded if the number of chunk hosted it is smaller than threshold value -t, in contrast a node is overloaded if the number of chunk hosted it is larger than threshold value –t.
3) Data is move from overloaded data node to under loaded data node.

3.1 Related to Algorithm:

File F= [f1, f2, f3, .fn]
Node Ni= [n1, n2, n3….np]
Capacity of node = ti
Total number of nodes in the system= m
The number of chunk in node= ui
Files divided into number of chunks/ block= bk
Number of chunk = Ck
Flag=0= false
Flag=1= true

3.2 Migrating chunk from overloaded node to under loaded node:

ISSN: 2005-4262 IJGDC

```
For (i=1; i<=m; i++)
{
        If (Ni≠ ti)
        {
                for (j=1; j<= ui; j++)
                {
                        for (k=1; k<=bk; k++)
                        {
                                Ni[np]=f[Ck]
                        }
                        Else
                        {
                                return Ni as a overloaded node}
                                for (k=1; k<= bk; k++)
                                {
                                        If (Ck = =0)
                                        {flag= false}
                                        else
                                        flag= true;
                                        {
                                                if (Ck= = 1)
                                                Overloaded node move its load to
                                                undeloaded node
```

4. Experimental Setup

4.1 Experimentation Setup:

The proposed system is implemented in java programming language. The experiment is carried out by setting up multi node Hadoop Cluster Package 2.7.0 on Ubuntu 16.02 32 bit Operating System. The master node is created on host machine with 8GB RAM, 500GB Hard Disk and Intel(R) Core (TM) (2) Duo CPU @ 2.2 GHz. Slave node is created on Oracle Virtual box. The latest version of eclipse IDE. Hadoop is based on only java programming language.

5. Experimental Result- Analysis

Before balance, we can see the data load of the three machines in the following Table 1. All nodes are in imbalance state.

Sr. No.	Node	Available Space (GB)	Before Balance	
			Used Space (GB)	Used Percentage
01	N1	2.0	1.4	70
02	N2	1.5	0.75	50
03	N3	1.9	0.85	45

Table1. Initial Configuration

In Figure 1 the dash line represents after balance, the load is uniformly distributed as much as possible across the data node. Our algorithm finished the balance in 13.26 minute, with threshold value 10%. In Figure 2 dash line represent after balance with threshold value 15%

ISSN: 2005-4262 IJGDC

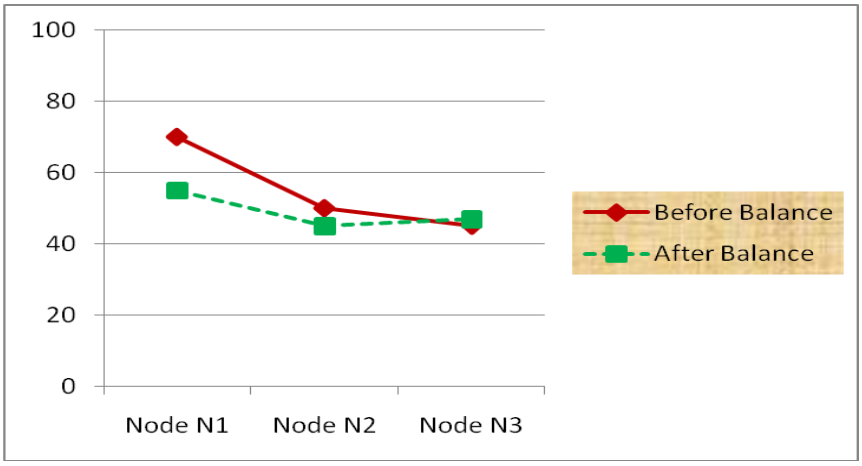

Figure 2. Threshold value = 10%

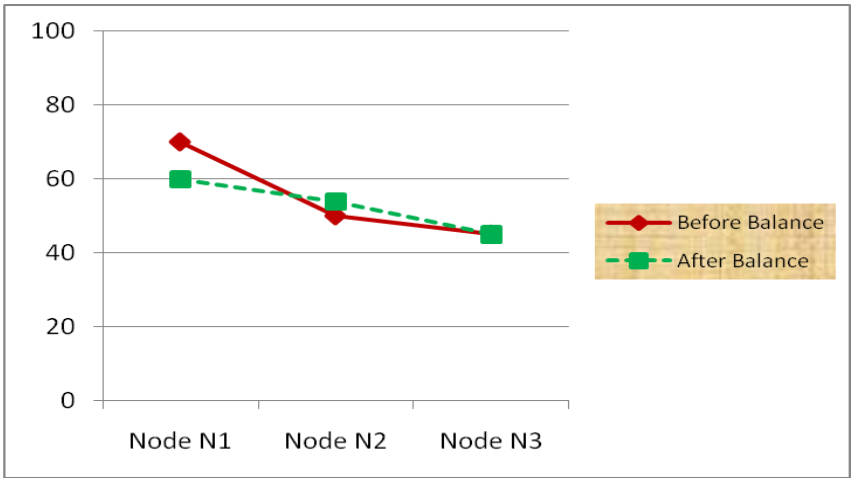

Figure 3. Threshold value = 15%

5.1 Storing 2.5 GB Data (Text Data):

Sr. No.	Node	Available Space (GB)	Before Balance	
			Used Space (GB)	Used Percentage
01	N1	3.2	2.5	78
02	N2	3.0	1.4	47
03	N3	2.8	1.1	39

Table 2. Initial Configuration when storing 2.5 GB Data

When storing 2.5 GB data in node N1, the initial capacity of node N1. N2, N3 and space usage percentage after storing 2.5 GB data are shown in the Table 2. While storing 2.5 GB data in node N1, the usage percentage of N1 becomes 78% and the usage percentage of N2 and N3 are 47%, 39% respectively, it means load among these node is not balance. After load balance the result is shown in Figure 3. These loads have been balance as much as possible, almost same

ISSN: 2005-4262 IJGDC

load on each node. The algorithm finished the balance in 16.22 minute, with threshold value 10%.

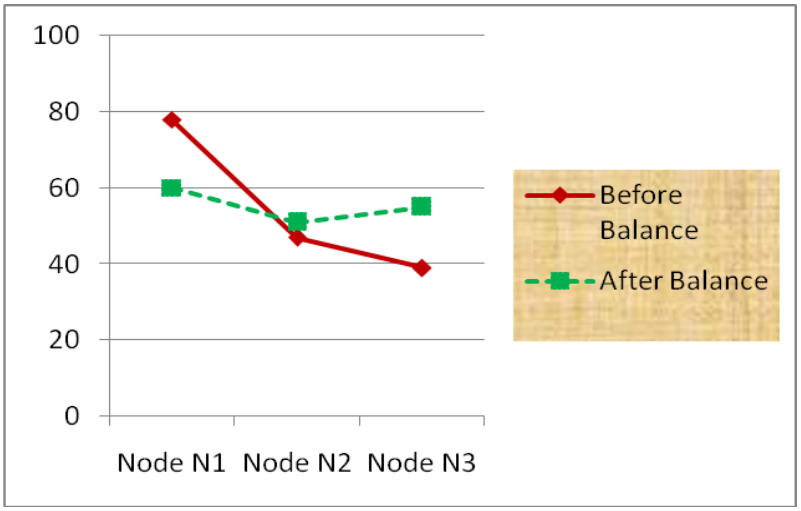

Figure 4. Storing 2.5 GB Data (Text Data)

5.2 Storing 2.0 GB Data (Mix data such as text, image, and video):

Table 3. Initial Configuration when storing 2.0 GB Mix Data

Sr. No.	Node	Available Space (GB)	Before Balance	
			Used Space (GB)	Used Percentage %
01	N1	3.0	2.0	67
02	N2	2.1	0.9	43
03	N3	1.9	1.1	58

When storing 2.0 GB blends data in node N1, the initial capacity of node N1. N2, N3 and space usage percentage after storing 2.0 GB data are shown in the table 3. While storing 2.0 GB data in node N1, the usage percentage of N1 being 67 % and the usage percentage of N2 and N3 are 43 %, 58 % respectively, it means load among these node is not balance. After load balance the result shown in figure 4. These loads have been balance as much as possible, almost same load on each node. The algorithm finished the balance in 18.36 minute, with threshold value 15%. However the text 2.5 GB data balance in 16. 22 minute with threshold value 10% i.e. mix data such as text, image, video take greater time balancing as compare to balancing text data.

ISSN: 2005-4262 IJGDC

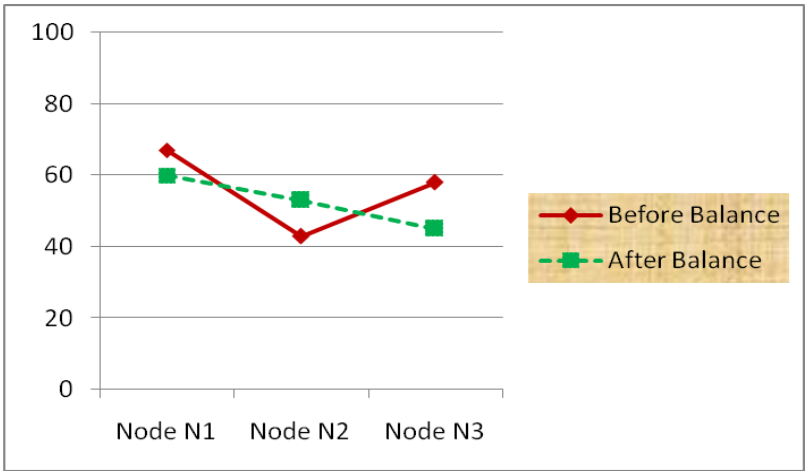

Figure 5. Storing 2.0 GB Data (Mix data such as text, image, and video)

6. Conclusion

A novel load balancing approach to deal with rebalancing problem in large scale, dynamic and distributed system. This proposed algorithm strives to balance the nodes and reduce the movement cost as much as possible. The load rebalancing algorithm is incorporated in multimode cluster environment. It distributes the load consistently amongst the data node. The heavy loaded node quickly transfer the data to light loaded node based on the threshold value. The proposed algorithm operates in a distributed manner. In load balance the resource can be well used and provisioned, maximizing the performance of MapReduce based application. This system accomplishes the challenging distributed in terms of load imbalance factor, also performs well on different data set.

7. References

[1] Hung-Chang Hasio, Hsueh-Yi Chung, HaiyingShen and Yu-Chang Chao, "Load Rebalancing for Distributed File Systems in Clouds", IEEE Transaction on parallel and distributed system, May 2013.
[2] Hung-Chang Hsiao, Hao Liao, Ssu-Ta Chen and Kuo-Chan Huang, "Load Balance with Imperfect Information in Structured Peer-to-Peer Systems", IEEE Transaction on parallel and distributed system, April 2011.
[3] Konstantin Shvachko, HairongKuang, Sanjay Radia, Robert Chansler, "The Hadoop Distributed File System", IEEE Conference, 2010.
[4] QuangHieu Vu, Beng Chin Ooi, Martin Rinard and Kian-Lee Tan, "Histogram-Based Global Load Balancing in Structured Peer-to-Peer Systems", IEEE Transaction on knowledge and data engineering, April 2009.
[5] Kun Liu, Gaochao Xu, and June e Yuvan,"An Improved Data Load Balancing Algorithm", Journal of Network Vol. 8, No. 12 December 2013.

Web References

[1] Installation of Hadoop, http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster.
[2] Hadoop Distributed File System, http://Hadoop.apache.org/hdfs.
[3] Ubuntu, http://www.ubuntu.com

ISSN: 2005-4262 IJGDC