International Journal of Innovative Research in Advanced Engineering (IJIRAE)
Volume 1 Issue 2 (April 2014)

# Survey on Load Rebalancing for Distributed File System in Cloud

Prof. Pranalini S. Ketkar
*IT Department, DCOER,*
*Pune University*
pranalini.ketkar@gmail.com

Ankita Bhimrao Patkure
*PG Scholar, Computer Department DCOER,*
*Pune university*
ankitapatkure@rediffmail.com

***Abstract*— Distributed file system is used as a key building block of cloud computing. In distributed file system, a large file is divided into number of chunks and allocates each chunk to separate node to perform MapReduce function parallel over each node. In cloud, if number of storage nodes, number of files and assesses to that file increases then the central node (master in MapReduce) becomes bottleneck. The load rebalancing task is used to eliminate the load on central node. Using load rebalancing algorithm the load of nodes is balanced as well as the movement cost is reduced. In this survey paper the problem of load imbalancing is overcome.***

***Keywords*— DFS, MapReduce, Load balancing, Distributed Hash Table, cloud**

## I. Introduction

In Cloud computing, the number of computers that are connected using communication network. The notation of cloud indicates that internet is mandatory to perform the various cloud operations i.e. to create delete, append and replace. It is used in IT-companies to share information and resources with the all users. There are various characteristics of cloud i.e. Scalable, on demand service, User centric, Powerful, Versatile, Platform independent etc. In cloud three technologies are included the MapReduce programming, Virtualization and distributed file systems for the data storage purpose.

Distributed file system is classical model of file system that is used in the form of chunks for cloud computing. Cloud computing application is based on the MapReduce programming used in distributed file system. MapReduce is the master-slave architecture in Hadoop. Master act like Namenode and Slave act like Datanode. Master takes large problem, divides it into sub problem and assigns it to worker node i.e. to multiple slaves to solve problem individually. In distributed file system, a large file is divided into number of chunks and allocates each chunk to separate node to perform MapReduce function parallel over each node. For example in word count application it identifies the occurrences of each distinct word in large file. In this application a large file is divided into fixed-size chunks (parts) and assigns each chunk to different cloud storage node. Then each storage node calculates the occurrences of each distinct word by scanning and parsing its own chunk. Then give its result to master to calculate the final result. In distributed file system, the load of each node is directly proportional to number of file chunks that node consists.

As the increase in storage and network, load balancing is the main issue in the large scale distributed systems. Load should be balance over multiple nodes to improve system performance, resource utilization, response time and stability. Load balancing is divided into two categories: static and dynamic. In static load balancing algorithm, it does not consider the previous behaviour of a node while distribute the load. But in case of dynamic load balancing algorithm, it checks the previous behaviour of node while distribute the load. In cloud, if number of storage nodes, number of files and assesses to that file increases then the central node (master in MapReduce) becomes bottleneck. The load rebalancing task is used to eliminate the load on central node. In load balancing algorithm, storage nodes are structured over network based on the distributed hash table (DHT); each file chunk having rapid key lookup in DHTs, in that unique identifier is assign to each file chunk [1]. DHTs enable nodes to self-recognize and repair while it constantly offers lookup functionality in node. Here aim to reduce the movement cost which is caused by load rebalancing of nodes to maximize the network bandwidth. Each Chunkserver first find whether it is light node or heavy node without global knowledge of node. The numbers of file chunks are migrated from heavy node to light node to balance their load. This process repeats until all heavy nodes becomes the light nodes. To overcome this load balancing problem each node perform load rebalancing algorithm independently without global knowledge about load of all nodes.

## II. LITERATURE SURVEY

### MapReduce: Simplified Data Processing On Large Clusters [9]

MapReduce is the programming model used in implementation for processing and generating large scale datasets. It is used at Google for many different purposes. Here map and reduce functions are used. Map function generate set of intermediate key pairs and reduce function merges all intermediate key values associated with same intermediate key. The map and reduce function allows to perform parallelize operation easily and re-execute the mechanism for fault tolerance. At the run-time, system takes care of detail information of partitioning the input data, schedule the program execution across number of available machines, handling failures and managing inter-communication between machines.

In distributed file system nodes simultaneously perform computing and storage operations. The large file in partitioned into number of chunks and allocate it to distinct nodes to perform MapReduce task parallel over nodes. Typically, MapReduce task processes on many terabytes of data on thousands of machines. This model is easy to use; it hides the details of parallelization, optimization, fault-tolerance and load balancing. MapReduce is used for Google's production Web search service, machine learning, data mining, etc. Using this programming model, redundant execution used to reduce the impact of slow machines, handle machine failure as well as data loss.

**Load Balancing Algorithm for DHT based structured Peer to Peer System [10]**

Peer to peer system have an emerging application in distributed environment. As compared to client-server architecture, peer to peer system improved resource utilization by making use of unused resources over network. Peer to peer system uses Distributed Hash Table (DHTs) as an allocation mechanism. It perform join, leave and update operations. Here load balancing algorithm uses the concept of virtual server to temporary storage of data. Using the heterogeneous indexing, peers balanced their loads proportional to their capacities. In this, decentralized load balance algorithm construct network to manipulate global information and organized in tree shape fashion. Each peer can independently compute probability distribution capacities of participating peers and reallocate their load in parallel.

**Optimized Cloud Partitioning Technique to Simplify Load Balancing [5]**

Cloud computing has some issue regarding resource management and load balancing. In this paper, Cloud environment is divided into number of parts by making use of cloud cluster technique which helps for the process of load balancing. The cloud consists of number of nodes and it partitioned into n cluster based on cloud cluster technique. In this, it consists of main controller which maintains all information regarding all load balancer in cluster and index table. Initial step is to choose the correct cluster and it follows algorithm as

1) In cloud environment, the nodes connected to central controller are initialized as 0 in index table.
2) When controller receives new request, it queries the load balancer of each cluster for job allocation.
3) Then controller pass index table to find next available node having less weight. If found then continue the processing otherwise index table reinitialized to 0 and in an increment manner then again controller passes table to find next available node.
4) After completing the process the load balancer update the status in allocation table.

Cloud partitioning method consist 2 steps:-

1) Visit each node randomly to match it with neighbor node. If it having same characteristics and shares similar data with minimal cost then two nodes are combined into new node with share same details. Repeat until there is no neighbor node having similar characteristics. Subsequently update the cost between neighbor two nodes and current neighbor node.
2) After joining two nodes into new node having similar characteristics visited node send the information to new node instead of sending it twice.

It gives the high performance, stability, minimum response time and optimal resource utilization.

**Histogram-Based Global Load Balancing in Structured Peer to Peer System [11]**

Peer to peer system having solution for sharing and locating resources over internet. In this paper there are two key components. First is histogram manager that maintain histogram that reflect global view of distribution of load. Histogram stores statistical information about average load of no overlapping groups of nodes. It is used to check whether node is normally loaded, Light or heavily loaded. Second component is load balance manager that take the action of load redistribution if node becomes light or heavy. Load balancing manager balance the load statically when new node joins and dynamically when existing node become light or heavily loaded. The cost of constructing histogram and maintaining it may be expensive in dynamic system. To reduce the maintaining cost two techniques are used. Constructing and maintaining histogram is expensive if node join and leave system frequently. Every new node in peer to peer system find its neighbour node and these neighbour nodes need to share its information with new node to setup connection. Now the cost of histogram is totally based on histogram update message caused by changing the load of nodes in the system .To reduce the cost approximate value of histogram is taken.

## III. ALGORITHMS FOR LOAD REBALANCE [1]

It contains three different modules.

1) File Allocation

2) DHT Devision
3) Load Rebalancing
4) Advantage of Node heterogeneity

## File Allocation

In this, large file is partitioned into number of chunks (C1, C2, C3…..Cn) and it allocates to sub servers (Chunkserver). Here the files can be added, deleted or appended dynamically to sub server. It will help to avoid the data loss. Fig. [1] Shows that, given large file is dived into number of parts and that parts are distributed over different Chunkserver.

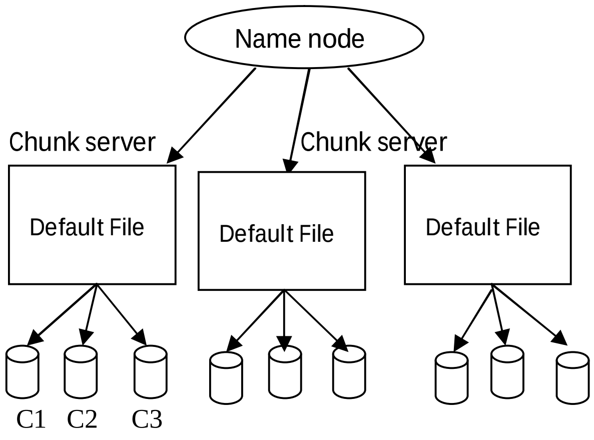


Fig 1: File allocation

## DHT Devising

The storage nodes are structured over network based on the distributed hash table (DHT); each file chunk having rapid key lookup in DHTs, in that unique identifier is assign to each file chunk. DHT guarantees that if any node leaves then allocated chunks are migrated to its successor; if node joins then it allocate the chunks which are stored in successor. DHT network is transparent. It specifies the location of chunks that can be integrated with existing distributed file system where master node manages namespace of file system and mapping of chunks to storage nodes. Fig. 2 shows the structure of Distributed Hash Table which is visible to all nodes.

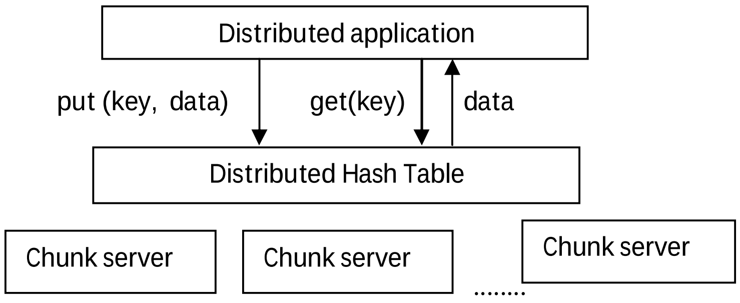


Fig 2: Distributed Hash Table

## Load Rebalancing

First calculate whether loads are light or heavy in each sub server.

A node is Light if, Number of chunk < (1-ΔL) A
And a node is heavy if, Number of chunk > (1-ΔU) A

Where ΔL and ΔU are the system parameters. All heavy nodes shared its load with light node. If node 'i' is the lightly loaded then it contact to its successor 'i+1'and migrate its load to its successor and then join instantly to heavy node.

Here node equalization technique is used to reduce latency, overload and resource utilization. The heavily loaded node transfer requested chunks to lightly loaded node and it traverse through physical network link.

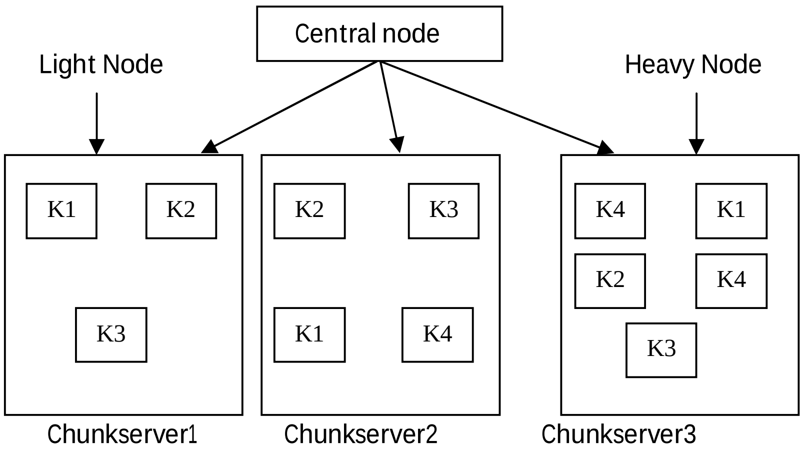
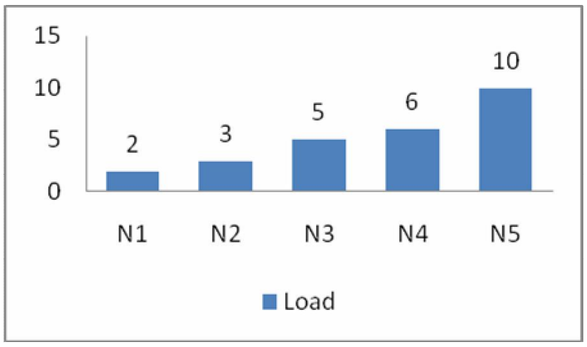


Fig 3: Heavy node and light node

Consider example, the capacity of each Chunkserver is 256MB i.e. 3 chunks (each chunk is of 64 MB) then according to above Fig. 3 and load rebalancing algorithm chunkserver1 is light node having load 192MB (no of chunk < (1-ΔL) A) and Chunkserver 3 is the heavy node having load 320MB (no of chunk > (1-ΔU)A). Using the load equalization technique, heavy node transfers its load to light node i.e. Chunkserver 3 transfer its some load to Chunkserver 1.



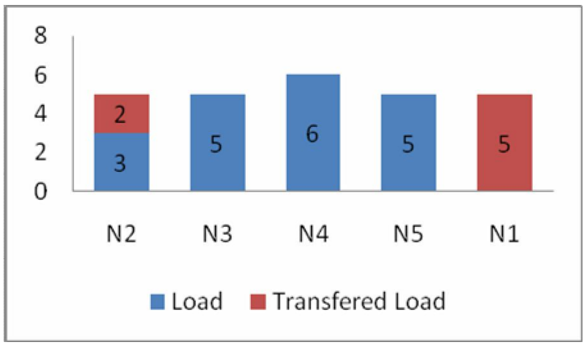


Fig4 (a): Initial load of chunks

Fig4 (b): Rebalancing the load

Consider example, in Fig. 4(a) capacity of each node is 5 chunks (A) but here N1 is the light node having 2 chunks and N5 is heavy node consist 10 chunks. Then N1 identifies that N5 is the heavy node then it contact to its successor i.e. to N2 and transfer its own load to N2. And request to heavy node for some load. Min (Lj-A),A) that much load is transfer from heavy node to light node. According to formulae 5 chunks are transfer from N5 to N1 (Fig 4(b)).

**Advantage of Node Heterogeneity**

Nodes on which file is distributed are heterogeneous in nature. According to nodes capacity there is one bottleneck resource. Consider, capacity of nodes (Cp1 , Cp2, Cp3,...., Cpn). Each node consist approximate number of file chunks. The load on that node needs to be balanced as follows:

$$A_i = \gamma \, Cp_i$$

where γ is the load per unit capacity of node. And

$$\gamma = m / \Sigma_{k=1} Cp_k$$

where m is the number of file chunks stored on system.

## IV. CONCLUSIONS

Cloud computing application is based on the MapReduce programming used in distributed file system. Load imbalancing Problem mostly occur in dynamic, large-scale and distributed file system. Load should be balance over multiple nodes to improve system performance, resource utilization, response time and stability.

In the load rebalancing algorithm the load of node is balanced as well as the movement cost also reduced. The load balancing task performs independently without global knowledge of system. This load balancing algorithm has fast concurrency rate. Load balancing reduce the load and movement cost while it uses physical network locality and node heterogeneity.

## REFERENCES

1. Hung-Chang Hsiao, Member, IEEE Computer Society, Hsueh-Yi Chung, HaiyingShen, Member, IEEE, and Yu-Chang Chao, proposed a “Load Rebalancing for Distributed File Systems in Clouds” IEEE transaction on parallel and distributed systems, vol. 24, no. 5, May 2013
2. J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Proc. Sixth Symp. Operating System Designed Implementation (OSDI ’04), pp. 137-150, Dec. 2004.
3. S. Mohammad, S. Breß, and E. Schallehn, “Cloud Data Management: A Short Overview and Comparison of Current Approaches,” Proc. 24th GI-Workshop Foundations of Databases (Grundlagen von Datenbanken), 2012.
4. Apache Hadoop, http://hadoop.apache.org/, 2013
5. P.Jamuna and R.Anand Kumar “Optimized Cloud Computing Technique To Simplify Load Balancing”International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 11,November 2013.
6. Kokilavani .K, Department Of Pervasive Computing Technology, Kings College Of Engineering, Punalkulam, Tamil nadu “Enhance load balancing algorithm for distributed file system in cloud” International Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 6, December 2013
7. Hadoop Distributed File System, http://hadoop.apache.org/hdfs/, 2012.
8. Hadoop Distributed File System “Rebalancing Blocks,” http:// developer.yahoo.com/hadoop/tutorial/module2.html#rebalancing,2012
9. J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters,” Proc. Sixth Symp. Operating System Design and Implementation (OSDI ’04), pp. 137-150, Dec. 2004.
10. ChahitaTanak, Rajesh Bharati “Load Balancing Algorithm for DHT Based Structured Peer to Peer System”International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 1, January 2013)
11. QuangHieu Vu, Member, IEEE, 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,vol. 21, no. 4, April2009.