

International Journal of Computer Science and Mobile Computing



A Monthly Journal of Computer Science and Information Technology

ISSN 2320-088X

IJCSMC, Vol. 5, Issue. 1, January 2016, pg.170 – 177

Balancing of Load for Distributed File Systems in Clouds Using Load Rebalancing

Jadhav Varsha
P.G. Student,

SSVPS's B.S. Deore College of Engineering,
Dhule, 424005, India

varshajadhav1989@yahoo.in

Mandre B.R.

Associate Professor,

SSVPS's B.S. Deore College of Engineering,
Dhule, 424005, India

bmandre@gmail.com

Abstract

One of the most significant and important concepts in cloud computing based application is Distributed file systems. The storage functions and computation of the nodes are performed simultaneous or parallel in the distributed file systems; the files are partitioned into a "number of chunks" which are allocated in "different nodes" so that the tasks can be served "simultaneous" over the different nodes. Computing function takes place in clouds, failures take place frequently, and because of the addition, replacement, upgrading of the nodes of the system. The dynamic creation, deletion, appending of files of the system can occur, this causes load imbalance in the distributed file system; that is, the files are split into chunks, and chunks are not distributed uniformly among the nodes. In many of the production systems having distributed file systems are mainly reliable on a centralized approach for reallocation of chunks among the nodes. If the system is centralized and the distribution of files is large then there is lot of workload on the central node which causes single point of failure and may thus become performance bottleneck. This paper present, a "Load Rebalancing Algorithm" which is fully distributed is offered to deal with the load imbalance problem. The performance of Load rebalancing algorithm is implemented using "Cloudsim".

Keywords: Distributed File System, Cloud, Rebalancing, DHT.

1. Introduction

Cloud computing, which involves virtualization, networking, software distributed computing, and web services etc. A cloud consists of elements like datacenter clients, and distributed servers. It includes scalability, fault tolerance, high availability, reduced cost of ownership, reduced overhead for users, flexibility, on-demand services [13] etc.

One of the key enabling technologies for clouds is distributed file systems. In distributed file system, clients can allocate their resources dynamically on-demand without sophisticated any management of resources. Clouds can be large in scale, so these techniques emphasize scalability, and resources can arbitrarily join and fail while maintaining system reliability [9].

Cloud computing provides benefits like: it results in cost saving because there is no need of initial installation of much resource. One of the problems in cloud computing is the problem of load balancing. While balancing the load certain types of information such as the number of jobs waiting in queue, CPU processing rate,

job arrival rate, and at each processor, as well as at neighboring processors, for improving the overall performance, it may be exchanged among the processors [9].

1.1 Load Balancing

Load Balancing is the idea of migration of excess load from heavily loaded (Overloaded) node to lightly loaded (Underloaded) node.

In the large scale distributed systems load should be balance over multiple nodes to improve response time, stability, system performance and resource utilization.

Load balancing categorized into following types:

- Static load balancing
- Dynamic load balancing.

The previous behavior of a node is not considered while distribute the load among nodes in case of static load balancing algorithm. The previous behavior of node is checked while distribute the load among various load in case of dynamic load balancing algorithm.

In Distributed file systems, load of a node is proportional to the number of file chunks the node possesses. In clouds, files can be arbitrarily created, deleted and appended, and node can also be replaced, added, and upgraded, so distribution of file chunks uniformly among storage nodes is difficult task.

To implement Distributed file systems there are different approaches, one of them is centralized approach. Centralized approach simplifies the implementation and design of distributed file systems. But, when number of storage nodes, and number of accesses to files increase linearly, the central node become performance bottleneck because of the central nodes may still be overloaded.

Therefore load rebalancing task is used to eliminate the load on central node. Load rebalancing algorithm, based on the distributed hash table (DHT) and storage nodes are structured over network; rapid key lookup in DHTs for each file chunk, in that unique identifier is assign to each file chunk [1]. DHTs enable nodes to repair and self-recognize while it constantly offers lookup functionality in node. The main goal is to reduce the movement cost which is caused by load rebalancing of nodes to maximize the network bandwidth. Light node or heavy node is found by each Chunkserver without acquiring global knowledge of node. To balance their load the numbers of file chunks are migrated from heavy node to light node. All heavy nodes become the light nodes by repeating this process. Each node perform load rebalancing algorithm independently without global knowledge about load of all nodes, to overcome this load balancing problem. In distributed file systems studying *the load rebalancing problem* specialized for dynamic, large-scale and data intensive clouds [1].

2. Related work

Several papers have been studied for load rebalancing for distributed file systems in clouds and few of them summarized as follows,

Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, I. Stoica *et al*. [6], uses the idea of virtual server. The main problem of P2P application is that, the efficient location of the node which stored the desired data item. A distributed lookup protocol, Chord, addressed this problem. In this load is splitting among virtual servers. Even though splitting load into virtual servers will increase the path length on the overlay, the flexibility to move load from one node to another node is crucial to any load-balancing technique over DHTs. These techniques assume that objects have the same size, object ID's are uniformly distributed and nodes are homogeneous.

Load Balancing in Structured P2P system, A. Rao *et al*. [2], defining the problem of load balancing in heterogeneous p2p systems that provides DHT abstraction to distribute data among different peer nodes. Three load balancing techniques are presented in this paper that are, one to one technique, in this 2 nodes picked at random. One node consider as light node and another node consider as heavy node. Light node picks random ID and then performs lookup operation to find node which is responsible for that node. It is easy to implement. Second technique is one to many, consideration of one heavy node and number of light node, this technique maintain directory that store load information about set of light node. Third technique is many to many; it is logical extension of first and second technique. In the many to many technique light and heavy nodes register their load with dedicated nodes called as directories. Matches between heavy and light node computed by directories and then, request the heavy and light nodes to transfer as well as to receive designated virtual server, respectively.

Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured P2P Networks H. Shen *et al*. [11] took both proximity [4] and dynamic features of DHTs [16] to present the locality aware randomized load balancing algorithm. Randomized matching between heavy and light nodes can be done for dynamic feature. But physical proximity of node do not consider. Locality-aware methods in load balancing deal with this problem but it is costly in terms of construction and maintenance of network. According to node proximity information and node capacities, in topology-aware DHTs, algorithms distribute load among the nodes by “moving items”. In this paper key value cache system in cloud environment is presented, which is a new technique of load balancing in consideration with the effect of load balancing and the scope of invalid cache. To improve the effect of load balancing randomized factor in the searching process to deal with proximity Cache-invalidation-scope model is established. Also improve the efficiency of load balancing by d-way probing.

Simple Load Balancing for Distributed Hash Tables, John Byers *et al*. [14], has done the use of the power of two choices paradigm that is, in this paper two new load balancing protocols are discussed to achieve better load balance. This scheme was not analyzed or simulated for case of heterogeneous object sizes and node capacities.

A comparative study into Distributed Load Balancing Algorithm, D. Lamb *et al*. [10] discussed three distributed load balancing algorithm, which are, Honeybee Foraging Behavior, Active clustering, Biased Random Sampling.

Load balancing in cloud computing system, Ram Prasad padhy *et al*. [12] discussed on basic concept of cloud computing and load balancing and also some existing load balancing algorithms are studied, which can be applied to cloud. In addition, minimum measurement for single level tree networks and the closed form solutions for reporting time with different load balancing strategies were also studied.

3. Existing System

In existing system load balancing takes place as follows [3],

Table 1: File and its respective chunks

File	Chunks
f1	2
f2	5
f3	5
f4	3
Total Chunks	15

Table 2: File Chunks allocated to Nodes

N1	N2	N3	N4	N5
f1-c1	f1-c2			
f2-c1	f2-c2	f2-c3	f2-c4	f2-c5
f3-c1	f3-c2	f3-c3	f3-c4	f3-c5
f4-c1	f4-c2	f4-c3		

Table 1 shows files and its respective chunks. File f1 is partitioned into 2 chunks. File f2 is partitioned into 5 chunks. File f3 is partitioned into 5 chunks. File f4 is partitioned into 3 chunks. Therefore files are divided into total 15 chunks.

Table 2 shows the allocation of chunks to nodes. Table 2 shows that the existing system is in imbalance state. It shows that there are 5 nodes and distribution of these chunks take place among these nodes but not in as uniform as thus causing load imbalance. This affects the resource utilization, load imbalance factor movement cost, and convergence to an extended rate.

4. Current System

Table 2 shows existing system is in imbalance state.

Table 3: Files and its respective chunks

File	Chunks
f1	2
f2	5
f3	5
f4	3
Total Chunks	15

Table 3 shows files and its respective chunks. File f1 is partitioned into 2 chunks. File f2 is partitioned into 5 chunks. File f3 is partitioned into 5 chunks. File f4 is partitioned into 3 chunks. Therefore files are divided into total 15 chunks. Nodes not balance dimensionally.

Table 4: Chunks allocated to Nodes in proposed system

N1	N2	N3	N4	N5
f1-c1	f1-c2	f4-c1	f4-c2	f4-c3
f2-c1	f2-c2	f2-c3	f2-c4	f2-c5
f3-c1	f3-c2	f3-c3	f3-c4	f3-c5
3	3	3	3	3

Table 4 shows the distribution of these chunks to balance load in the current system. It shows that there are 5 nodes and the distribution of these chunks takes place among them in a uniform manner thus causing “load rebalance”. According to table 4 each node has equal number of chunks thus avoiding excess load on only some nodes [3].

5. Load Rebalancing Algorithm

Algorithm contains three different modules as follows [3]:

- 1) File Allocation
- 2) DHT Formulation
- 3) Load Rebalancing

1) File Allocation

In this module, large file is partitioned into number of chunks (C1, C2,....., Cn) and it allocates to Chunkserver. Here the files can be appended, added or deleted dynamically to chunkserver. It will help to avoid the data loss. Figure-1 Shows that, given large file is partitioned into number of parts or chunks and those chunks are distributed among number of Chunkserver also called sub server.

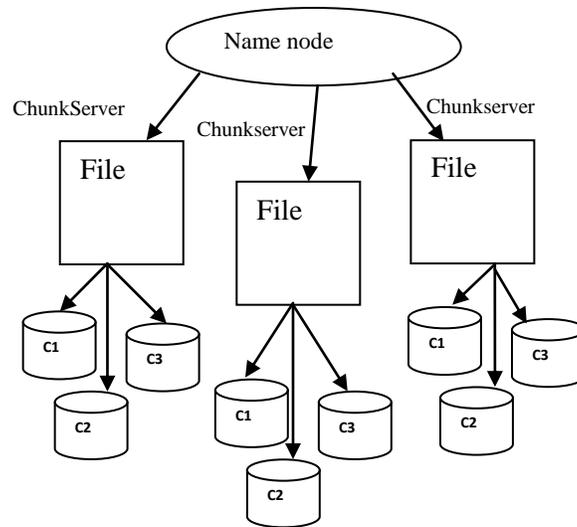


Figure 1: File Allocation

2) DHT Formulation

To structured over network the distributed hash table (DHT) is used for the storage nodes; Each file chunk is assign with Unique identifier, each file chunk having rapid key lookup in DHTs. DHT guarantees that if node joins or add then it allocate the chunks which are stored in successor and if any node leaves or delete then allocated chunks are migrated to its successor. Network specifies the location of chunks that can be integrated with existing distributed file system, due to the transparency of DHT network. In this system master node manages mapping of chunks to storage nodes and namespace of file system. Figure 2 shows the structure of Distributed Hash Table, and DHT is visible to all nodes.

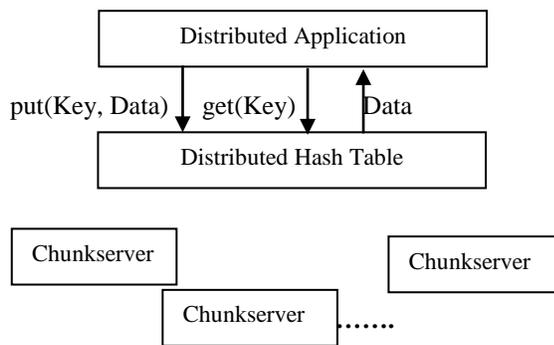


Figure 2: Distributed Hash Table

3) Load Rebalancing

First, to find whether node is lightly loaded (Under loaded) or heavily loaded (Overloaded) in each sub server.

A node is *light* if, number of chunk $< (1 - \Delta L).A$ and a node is *heavy* if, Number of chunk $> (1 + \Delta U).A$ Where ΔL ($0 \leq \Delta L < 1$) and ΔU ($0 \leq \Delta U < 1$) are the system parameters. All heavily loaded nodes shared its load with lightly loaded node. If node 'i' is the lightly loaded then it transfer its load to its successor 'i+1' and then join instantly as successor to heavy node. The lightly loaded node request for chunks from heavily loaded node and it traverse through physical network link.

Consider example, each chunkserver having the capacity 512 MB i.e. 3 chunks (each chunk is of 128 MB) then according to above Figure 3 and load rebalancing algorithm, chunkserver1 is light node having load 384 MB (no. of

chunk $< (1-\Delta L) A$) and Chunkserver 3 is the heavy node having load 640 MB (no. of chunk $> (1-\Delta U) A$). By using this technique, heavy node transfers its load to light node i.e. Chunkserver 3 transfer its some load to Chunkserver 1 [1], [3].

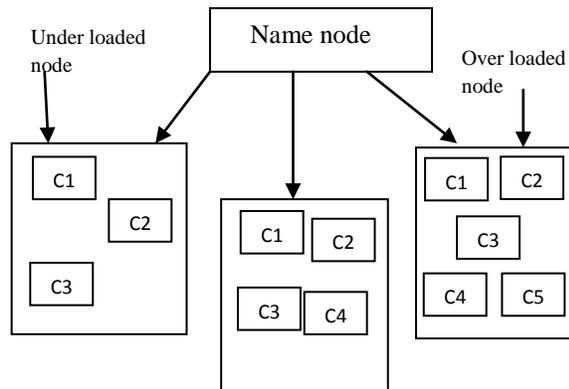


Figure 3: Under loaded and overloaded node

6. Simulation Results

Simulation is used due to increase in complexity of networks. In terms of the time, memory and hardware requirement simulation time often decreases the complexity in the network. CloudSim can be used for the simulating load balancing problem. To simplify the installment of real machines in the system for testing the balancing problem occurring in the network the virtual Machine components are being used. The performance of an algorithm is evaluated through simulation. Simulation implemented with threads [8].

Simulation setup:

The simulation takes place by using a simulation toolkit workflowsim, which is extended version of cloudsim simulator [17]. In this setup, a single datacenter and 5 virtual machines are consideration. Table 5 and table 6 shows, considered properties for Datacenter and virtual machine respectively.

Table 5: Datacenter properties

Arch	OS	Vmm	Time zone	C o s t	Cost per Mem	Cost Per Storage	Cost per Bw
X86	Linux	Xen	10.0	0.3	0.05	0.1	0.1

Table 6: Virtual Machine properties

Image size(MB)	VM Mem(MB)RAM	Mips	BW	pesNo.	VMM
10000	512	10000	10000	1	Xen

For simulation purpose dataset log files are used as load. Each file F contains number of files f and f partitioned into file chunks. File chunks are distributed among virtual machines. Load is distributed as uniform as possible among this virtual machine.

Simulation results carried on dataset log files, Figure 4 shows, Load distribution among considered virtual machines for different algorithms, that is, RB (Rebalancing algorithm), RR (Round Robin Algorithm), and MaxMin algorithm.

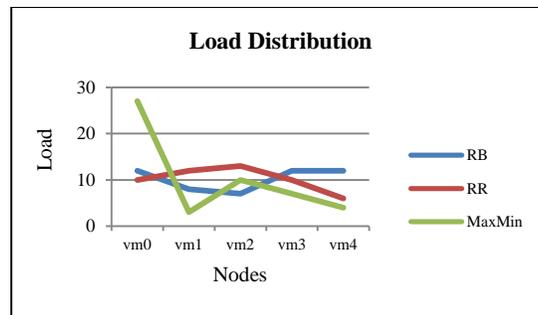


Figure 4. Load Distribution among nodes

In Figure 4, as compare to MaxMin algorithm, RB algorithm distribute load uniformly in some extend. In vm1 and vm2, also in vm3 and vm4 the load distributions are nearly equal in amount.

Figure 5 shows the execution time required for these algorithms.

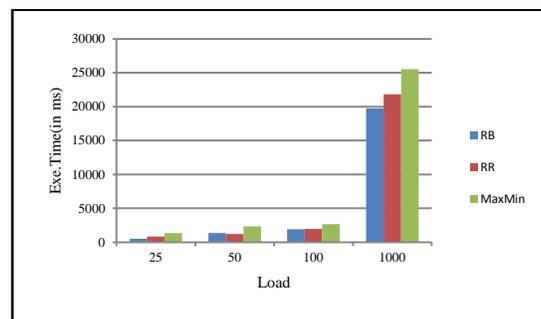


Figure 5. Required Execution time

7. Conclusion

Load balancing is one of the challenges in the cloud environment [13]. Balancing of load by using load rebalancing algorithm in dynamic, large scale, and distributed file systems in clouds is described over here. This algorithm distribute load evenly across the nodes. Resource utilization takes place. But Cloud Computing covers a very vast area; it is applicable to both large and small scale area. In this paper, it can be simulated for small scale cloud environment that is consideration of single datacenter with 5 virtual machines, by using the CloudSim toolkit. It can be programmed by using java programming language.

References

- [1] Hsueh Yi Chung, Haiying Shen and Yu Chang Chao Hung Chang Hsiao, "Load Rebalancing for Distributed File Systems in Clouds," *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, vol. 24(5), pp. 951-962, May 2013.
- [2] K. Lakshmi narayanan, S. Surana, R. Karp and I. Stoica A. Rao, "Load Balancing in Structured P2P Systems," *Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '02)*, pp. 68-79, Feb. 2003.
- [3] Smita Salunkhe, S. S. Sannakki, "Load Balancing in Clouds," *International Journal of Emerging Trends in Electrical and Electronics*, vol. 11(2), pp. 126-131, June 2015.
- [4] Y. Zhu and Y. Hu, "Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 16(4), pp. 349-361, Apr. 2005.
- [5] Hadoop Distributed File System. [Online]. <http://hadoop.apache.org/hdfs/>, 2012.
- [6] Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan Ion Stoica, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in *Pmc. ACM SIGCOMM. San Diego*, pp. 149-160, 2001.
- [7] Jeffrey Considine and Michael Mituenmacher John Byers, "Simple Load Balancing for Distributed Hash Tables," *Pmc.IPTFS*, Feb. 2003.
- [8] PrittoPaul.P2 Karthick Smiline Britto.J1, "Decentralized Approach for Balancing Load in Dynamic Cloud Environment,"

International Journal of Innovative Research in Computer and Communication Engineering, vol. 2(1), pp. 2603-2610, March 2014.

- [9] K. S. Sundaram, "Load Balancing for Distributed File System," *International Journal of Advanced Computational Engineering and Networking*, ISSN Vol. 1(1), pp. 25-30, Mar. 2013.
- [10] D. Lamb and A. Taleb Bendiab M. Randles, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia*, pp. 551-556, April 2010.
- [11] H. Shen and C.-Z. Xu, "Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured P2P Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18(6), 849–862, June 2007.
- [12] Ram Prasad Padhy and PGoutam Prasad, "Load balancing in cloud computing system," *Department of Computer Science and Engineering National Institute of Technology, Rourkela Rourkela-769 008, Orissa*, May 2011.
- [13] Suriya Begum and Dr. Prashanth, "Investigational Study of 7 Effective Schemes of Load Balancing in Cloud Computing," *International Journal of Computer Science Issues*, Vol. 10(1), pp. 276-287, Nov. 2013.
- [14] D. Kargar and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," *Proc. 16th ACM Symp. Parallel Algorithms and Architectures (SPAA '04)*, pp. 36-43, June 2004.
- [15] H. Gobiuff and S. T. Leung S. Ghemawat, "The Google File System," *Proc. 19th ACM Symp. Operating Systems Principles (SOSP)*, pp. 29-43, Oct. 2003.
- [16] B. Godfrey, K. Lakshminarayanan, R. Karp and I. Stoica S. Surana, "Load Balancing in Dynamic Structured P2P Systems," *Performance Evaluation*, vol. 63(6), pp. 217-240, 2006 Mar.
- [17] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *IEEE 8th International Conference on (e- Science)*, 2012.