ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

# Implementation of Load Re-Balancing Technique and Security Measures for Distributed File System in Cloud

Juhi Shah, Prof. Dhara Kurian

M.E. Student, Dept. of Computer Science, RMD Sinhgad, Savitribai Phule Pune University, Pune, India

Assistant Professor, Dept. of Computer Science, RMD Sinhgad, Savitribai Phule Pune University, Pune, India

**ABSTRACT:** Here, this paper examines the load rebalancing problem in cloud computing. The main purpose of the paper is to enhance distributed load rebalancing algorithm to cope with the load imbalance factor, movement cost (network traffic), and algorithmic overhead. The load rebalance algorithm is compared against a centralized approach in a production system and the performance of the proposal implemented in the Hadoop distributed file system (HDFS) for cloud computing applications. But HDFS is not suitable for small data it can be applicable for huge data only as well as it don't provide audit ability of data stored by end user also there is no security for data like data encryption. So I decide to follow Hadoop Architecture and create my own architecture which will demonstrate like hadoop architecture by overcoming limitation of HDFS as well as rebalancing the data stored on distributed data server (chunk servers). and also investigate to implement security provided for cloud computing and evaluate the quality of Service-QOS like a Response Time of whole system. In cloud computing one server controls number of sub servers, files, it can be add, delete, and append dynamically. Any file sharing like uploading or downloading are stored in sub-servers and retrieved from sub-servers. Before uploading and downloading that files are stored in encrypted format and retrieve it in a decrypted format. In our implementation the key is sent to the user's email id and user uses that key to view and download the files from sub-server. For encryption and decryption schemes we are using AES algorithm.

**KEYWORDS**: Hadoop distributed file systems, cloud computing, load rebalancing, Distributed File System, AES.

## I. INTRODUCTION

Cloud computing refers to delivery of computer resources from a remote place based on user needs and for that network connections are necessary to access information and utilize resources in cloud. cloud computing can be defined as style of computing where massively scalable IT- enabled capabilities are delivered to external customers using Internet. According to the Seccombe and National Institute of Standards & Technology [1] [2], guidelines for cloud computing, it has three different types of models namely private, public and hybrid. Performance, security and data locality to both cloud architecture and end users are the key features of public model. Nowadays, increase in the challenges on how to transfer, where to store and compute data are the issues caused by large distributed file systems in cloud computing. Cloud Computing Technologies such as Map Reduce paradigm [3], virtualization [4] and Distributed File Systems [5] are used to achieve scalability and reliability in clouds. Here, Hadoop File Systems (HDFS) and Google File Systems (GFS) are used to overcome the issues which are arise in achieving those factors.

Load balancing is the main issue in large scale distributed file system. It is the process of distributing tasks to all the nodes involved in cloud computing. Adequate allocation of resources to every computing task helps to achieve a resource utilization ratio and high user satisfaction. Minimizing resource consumption, avoiding bottlenecks, implementing fail-over, enabling scalability, reducing network inconsistencies, maximize the network bandwidth and solving network traffic are the main goals of load computing. Whole cloud gets fail while analyzing that existing system clouds performance bottleneck due to failure of central node because of that Functional and technical difficulties are caused because of those failures. Cloud computing allocate a resources dynamically, which connects and add thousands of nodes together. The main aim is to allocate files to these nodes, for avoiding heavy nodes that files are uniformly distributed to these nodes. Load balancing techniques provides maximization of network bandwidth, reduction of network traffic and network inconsistencies. Here, we can add, delete and update nodes dynamically for

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

heterogeneity of the nodes and heterogeneity of the nodes will increase the scalability and system performance. In Distributed File System (DFS) the main functionalities of nodes is to serve computing and storage functions. In this paper, we introduced new load rebalancing techniques in distributed file systems and its aim is to reduce network traffic (or *movement cost*) caused by rebalancing the loads of nodes as much as possible to increase the network bandwidth to cloud applications. For security in cloud computing, we maintained those files in encrypted format using cryptographic algorithms such as AES.

**EXISTING SYSTEM**

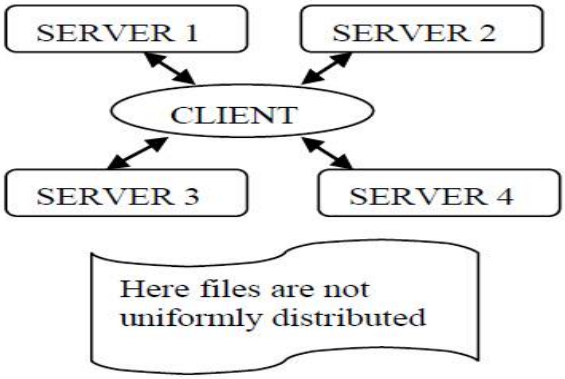



Fig1. Existing system Diagram

When we examine the existing system cloud computing depends on centralized nodes because of centralized node performance bottleneck occurs because the failure of central nodes causes the failure of the whole system And also load imbalance exists, since the loads are not uniformly distributed to these nodes. This dependence is inefficient and not achieving maximum productivity of whole system. It conducts technical and functional problems like,

***Limitations of Existing System***

- High Movement Cost
- High Network Traffic
- Algorithm Overhead
- Load Imbalance
- Relying on Central Node
- Security difficulties

## II. RELATED WORK

Game-theoretic static load balancing for distributed systems [7] discussed on static load balancing strategy based on game theory for distributed file systems (DFS). Here, In this work they provides us with a new review of the load balance problem in the cloud environment. As an implementation of distributed system, the load balancing in the cloud environment can be viewed as a game.

Load Balancing in Structured P2P Systems [8] based on the concept of virtual servers the many-to-many framework is to survive with the load imbalance in a DHT. In the many-to-many framework light and heavy nodes are register their loads with some dedicated nodes name as the directories. This directories compute matches between heavy and light nodes. Then request the heavy and light nodes to transfer and to receive designated virtual servers.

Load Balancing in Dynamic Structured P2P Systems [9] discussed on the many-to-many framework essentially reduces the load balancing problem to a centralized algorithmic problem. Here, the entire system heavily depends on the directory nodes or center node so the directory nodes may thus become the performance bottleneck and single point of failure.

Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications [10] has proposed the notion of virtual servers as a means of improving load balance. By allocating (log *N)* virtual servers per physical node, Chord ensures

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

that with the high probability the number of objects on any node is within an average. So, these schemes assume that nodes are homogeneous and that objects have the same size, and object IDS are uniformly distributed.

Simple Load Balancing for Distributed Hash Tables [11] has proposed the use of the power of two choices paradigm for achieving better load balance. This scheme was not identify or simulated for the case of heterogeneous node capacities and object sizes. And in any case is not prepared to handle a dynamic system of any kind which we have described. And this is largely complementary to the work presented in this paper.

Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System [12] have proposed algorithms for replica placement in distributed file systems (DFS) which are similar with our algorithms. However, their primary aim is to place object replicas to maximize the availability in an un-trusted P2P system, while we consider the load balancing problem in a cooperative system.

## III. PROPOSED ALGORITHM

### A. *Design Considerations:*

The Method is works in four major steps (Modules):

- **Data Owner or Client Registration :** If a data owner want to store data on a cloud server, he/she should register their details with our application. These details are maintained in a Database for further use. Then he has to upload the file on name node (Application server). The files which are stored in a database are in an encrypted form and only authorized users can decode it and download.
- **Trusted Third Party Login:** TTP who monitors the data owners file by verifying the data owner's file and balance file in a database on distributed server.TTP balances load on cloud server and also maintain security of data by providing verification service using DHT (Distributed Hash Table)
- **DHT Devising:** Load rebalancing charge to storage nodes by having the storage nodes/file blocks balance their loads un-promptly. This ends the addiction on central nodes. The storage nodes are structured as a network based on distributed hash tables (DHT). It authorizes nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management..
- **Load Rebalancing Algorithm Implementation:** Here ,we will implement the Load Rebalancing algorithm to relieve the load of name node; data nodes perform the load rebalancing task spontaneously without name node and each file chunk is assigned with a unique ID (i.e. Hash Code using SHA-1). This algorithm distribute file on various chunk servers (nodes) with equal size. This algorithm basically perform "Load balance" by having data nodes maintain the same number of chunks.
- **Finally, Application deployment and security:** we will deploy our application on private network to access in particular private premises. We can provide security to our user database by providing of the encryption technique using AES algorithm and also provide audit ability.

### B. *Description of the Proposed Algorithm:*

Figure 2 show System Architecture of proposed system.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

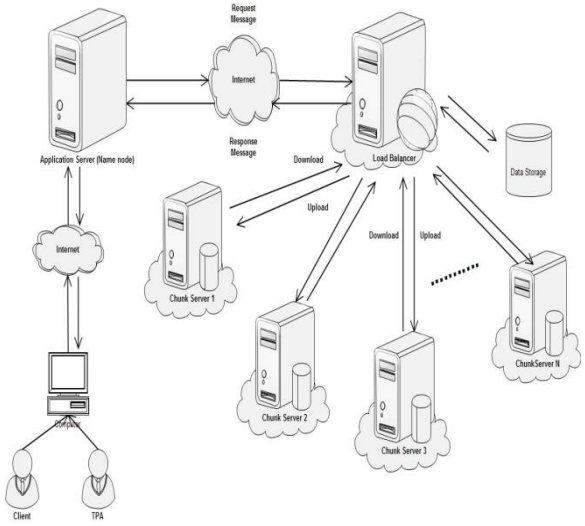



Fig. 2. System Architecture

Let the set of files be F which is uploaded by client via laptop, desktop, mobiles. Files in F may be dynamically created, opened, deleted. Our proposal maintains Load Balancer and chunk servers. Chunk servers are denoted by S. Nodes are divided into two types based on their load and they are

- Light node
- Heavy node

The proposed load rebalancing algorithm first evaluates whether the loads are light (under loaded) or heavy (overloaded) in each chunk servers without global knowledge. All heavy loads are converts into light nodes. F are downloading or uploading with the help of the centralized system. Load equalization technique used to distribute the F uniformly into chunk servers. The advantage of the method is to reduce latency, isolated overload, and great utilization of resource provident outcome. Finally examine the QOS like, bandwidth, response time which owing to enhance system performance. For security reason, these files F are encrypted before sharing (uploading & downloading) the files F to chunk servers using cryptographic techniques and decrypted while downloading. Our proposal is eliminates the dependence on central nodes because we are going to used DFS. The storage nodes or files are structured as a network based on distributed hash tables. Here, DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism and it simplifying the system provision and management. Our algorithm is compared against a centralized approach in a production system which uniformly distributes across chunk servers. So that centralized approach, outperforms the previous distributed algorithmic rule in terms of load imbalance issue, movement value, and recursive overhead.

***Advantages of Proposed System***

- Deploy wide-range and failure error domain
- Reduce Network Traffic or Movement Cost
- Maximize the Network Bandwidth
- Improve overall System Performance
- Utilizes Physical Network Locality
- Better throughput and response time
- System consistency (i.e., avoid data loss)
- Excellent security
- Picking lead of node heterogeneity
- Extends resource utilization
- Speedy & Diminishes time consistency

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

### C. *Block Diagram*

This presents the proposed system framework. As shown in Figure 3, the framework defines the different steps to perform load rebalancing techniques and security measures in distributed file system. First, client have to register on cloud and cloud will response to client. After Successfully login client can upload their file in encrypted formatted form via laptop, desktop, smart phone. File metadata is store on name node like File name, File unique id, file size, SHA-1 and also maintain DHT. TTP will migrate chunk data and verify data when client request comes for verification. If any modification occurs in client data at that time TTP Sent a notification to client.

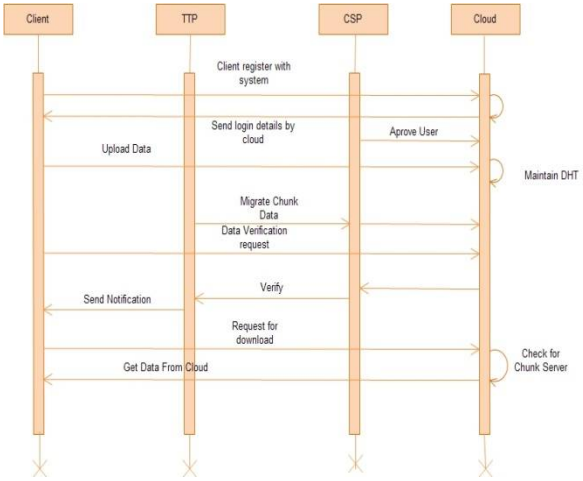



Fig. 3. Overview of system

## IV. PSEUDO CODE

**Algorithm :** Proposed Algorithm
**Required:** File Data, Users Info

1. User will get registered to system along with secret key.
2. Users can Upload/Download files on Name Node which is secured by AES Encryption Algorithm.
3. Calculate HASH of file block using SHA-1.
4. Name node maintains DHT and Stores metadata of file.
5. Balancer Load Rebalances and Migrate data if required.
6. Managing Replicas on Different sub-Server.
7. TTP verifies data using DHT form name node.

### A. *Mathematical model of system*

Let S be defined as,

$S = \sum(U, P, F(x))$

U: User Id

P: Password

F(x): It provides set of functions that performs on the client registration, defined as,

$F(x) = \{K, Hash\_Comp, DHT, LRB, REP, V\_Data\}$

**Step 1 :** K : Secrete Key Generated, and it consist of following parameters,

$K = \{ A, DG, SC, RG\}$

A: A set of alphabet

DG: A set of Digits

SC: A set of Special Characters

RG: Random number generator using

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

$$X_{n+1} = (aX_n + b) \bmod m$$

**Step 2 :** It generate a 20 byte length hash of input blocks using SHA-1 for chunk servers, and it consist of following parameters,

Hash_Comp = {B, K}
B: A set of encrypted file block
K: Secret key (K1, K2,….)

**Step 3 :** Distributed Hash Table (DHT) generated when hash generate, and it consist of following parameters,

DHT = {F, MF, K}
F: A File (.txt, .pdf, .doc)
MF: Meta Data of File (F)
K: Secret key(K1, K2,….)

**Step 4 :** Name node (Application Server) stores DHT.

**Step 5:** Load rebalancing algorithm will balance the node on Distributed file system, and it consist of following parameters,

LRB = {IN, LN, HN}
IN: Idle node
LN: Low node
HN: High node

**Step 6 :** Managing replicas (Data Availability): It provides replicas on different chunk server if one of the chunk server down or fails using,

$\sum_{i=2}^{k} \frac{1}{n}$
K = typically a small constant and (k=3) in Google file system
N = total no of storage nodes

**Step 7 :** Verify data (V_Data) gives True/False when client sends request to TTP when their data misbehaves, and it consist of following parameters,

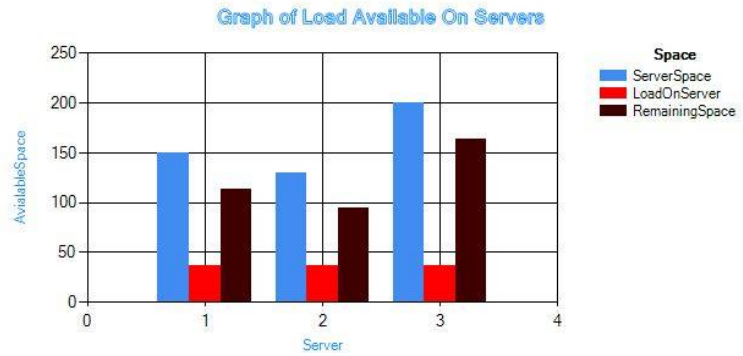
V_Data = {H, SF, P, MF}
H: A set of hash of data block
SF: signature of file
P: A set of proof
MF: metadata of file

**Step 8 :** AES: Advanced Encryption Standard algorithm provide a secure data on cloud.

Finally, we get File Security and Data Server Load Balance.

## V. SIMULATION RESULT

Below Graph Shows that Loads available on Server. After implementation it is concluded that load is optimally distributed to three different server. Here three server are used where blue colour show the total server space, red colour show Load on the server and brown colour show remaining space on server. Server space for server1 is 150MB, server2 is 130MB, and server3 is 200MB.



ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**

## VI. CONCLUSION AND FUTURE WORK

Here, balancing the loads of nodes and reducing the demanded movement cost as much as possible. While taking advantage of physical network locality and node heterogeneity is done by using proposed idea. Competing algorithms through synthesized probabilistic distributions of file chunks are compared with proposed idea. Emerging distributed file systems (DFS) in production systems strongly depend on a central node for chunk reallocation and this dependence is clearly inadequate in a large-scale and failure-prone environment because the central load balancer is put under considerable workload that is linearly scaled with the system size, and because of that reason it become the performance is bottleneck and the single point of failure. Here, centralized approach in a production system and a competing distributed method are compared with proposed algorithm and load imbalance factor, movement cost, and algorithmic overhead are handled by developed algorithm efficiently. To securing the data, implemented the AES algorithm.

Future work, Use any other balancing algorithm strategy in cloud computing environment. In this paper only file sharing (file download & upload). In future sharing multimedia files (audio & video streaming).This algorithm used between two various cloud computing environments. Use various cryptographic algorithms for security purposes.

## REFERENCES

1. Seccombe A.., Hutton A, Meisel A, Windel A, Mohammed A, Licciardi A, (2009). Security guidance for critical areas of focus in cloud computing, v2.1. Cloud Security Alliance, 25
2. Mell P, Grance T (2011) The NIST definition of Cloud Computing. NIST, Special Publication 800–145, Gaithersburg, MD.
3. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," in Proc. 6th Symp,Operating System Design and Implementation (OSDI"04), Dec. 2004, pp. 137–150
4. J. Byers, J. Considine, and M. Mitzenmacher, Simple load balancing for distributed hash tables, in Proceedings of IPTPS, Berkeley, CA, Feb. 2003.
5. Hadoop Distributed File System, http://hadoop.apache.org/hdfs/.
6. Tripti Saxena, IT Security Issue in Cloud Computing Models , "https://www.rgpv.ac.in/iccbdt/Papers/CL-139.pdf"
7. S.Penmatsa and T.Chronopoulos, Game-theoretic static load balancing for distributed systems, Journal of Parallel and Distributed Computing, vol.71, no.4, pp.537-555, Apr. 2011.
8. A.Rao, K.Lakshmi narayanan, S.Surana, R.Karp and I.Stoica, Load Balancing in Structured P2P Systems‖, Proc. Second Int,„l Workshop Peer-to-Peer Systems (IPTPS „02), pp. 68-79, Feb. 2003.
9. S.Surana, B.Godfrey, K.Lakshminarayanan, R.Karp and I.Stoica,―Load Balancing in Dynamic Structured P2P Systems,‖ Performance Evaluation,vol.63, no. 6, pp. 217-240, Mar. 2006.
10. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.‖ in Pmc. ACM SIGCOMM. San Diego, 2001, pp. 149-160.
11. John Byers, Jeffrey Considine, and Michael Mituenmacher, ―Simple Load Balancing for Distributed Hash Tables._„in Pmc.IPTfS, Feb. 2003.
12. J. R. Douceur and R. P. Watlenhofer, "Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System,‖ in Pmc. DISC,2001.
13. D. N. Rewadkar, and Juhi Shah, "A Survey On Load Rebalancing for Distributed File System," in Proc. Int. Jou. Sci. Research, IJSR, vol. 3, Issue 11, Nov. 2014.

## BIOGRAPHY

**Juhi Shah** is a Final year ME Student in the Computer Science Department, College of RMD Sinhgad, Savitribai Phule PuneUniversity. She received Bachelor of Computer Engg. degree in 2009 from SP Uni., Aanad, Gujarat, India. Her research interests are Cloud computing , Load Balancing and AES, Algorithms.