International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

Load Rebalancing File Distributed System in Cloud

Prof. Vishal S. Patil, Tarun S. Pilley, Azhar Sheikh, Chetan Patil
Computer Science & Engineering Department, Amravati University / Anuradha Engineering College, Chikhali, Maharashtra, India

ABSTRACT

Load Rebalancing Distributed file systems are the building blocks for cloud computing software based on the one of the Algorithm that is Map Reduce programming paradigm Algorithm. In a such file systems of this Procedures, nodes at the time serve computing and mass storage functions, The file is Subdivided into a number of chunks allocated in nodes so that Map Reduce tasks can be performed in parallel over the nodes. In a cloud computing System, failure is the normal, because of the so many servers are connected and nodes may be upgraded, replaced, and placed in the system of the Load Balancing. Files can be created, deleted, and it will edited. This results in load imbalance in a distributed file system; that is, the file chunks are not Separated as uniformly among the nodes of the systems.

Keywords: Load Management, Algorithm Design and analysis, Cloud Computing Structure.

I. INTRODUCTION

Cloud Computing is a network type technology. In clouds of Load Rebalancing system , there is Server points in Network that will be connected to many Nodes in the System structures , if there is passing of the information of data from server to the clients nodes, in the middle of the servers and the clients node there is a load balancing server to balance the incoming load from the server and it will be the check to the nodes while passing the data if there is having space in nodes to transfer the information of data, if there is no space to adjust incoming data in nodes, then it will be check another one nodes that having in the sequence , if second nodes having that much of space the data will be store there, this is load balancing in the cloud system or any other system structure. Enabling technologies for clouds include the MapReduce programming paradigm , in this concept there is MapReduce Algorithm is used for distributed file systems virtualization, and so forth

Proposed system:

- ✓ The load of each virtual server is stable over the timescale when load balancing is performed.
- ✓ Load balancing is performed in proximity-aware manner, to minimize the overhead of load movement (bandwidth usage) and allow more efficient and fast load balancing.

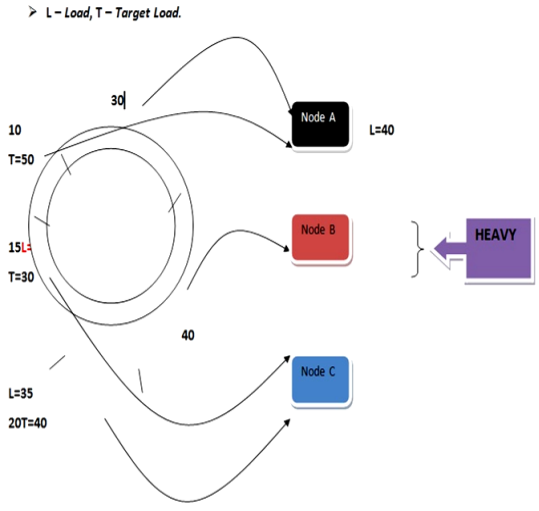


Figure 1

Block Diagram :

Literature Survey:

Existing solutions to balance load in DHTs incur a high overhead either in terms of routing state or in terms of load movement generated by nodes arriving or departing the system. In this paper, we propose a set of general techniques and use them to develop a protocol based on Chord, called Y0, that achieves load balancing with minimal overhead under the typical assumption that the load is uniformly distributed in the identifier space.

In particular, we prove that Y0 can achieve near-optimal load balancing, while moving little load to maintain the balance and increasing the size of the routing tables by at most a constant factor. Using extensive simulations based on real-world and synthetic capacity distributions, we show that Y0 reduces the load imbalance of Chord from O (log n) to a less than 3.6 without increasing the number of links that a node needs to maintain. In addition, we study the effect of heterogeneity on both DHTs, demonstrating significantly reduced average route length as node capacities become increasingly heterogeneous. For a real-world distribution of node capacities, the route length in Y0 is asymptotically less than half the route length in the case of a homogeneous system.

1- Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications

A problem that peer-to-peer applications is the efficient location of the node that stores in a data item. This paper presents Chord, a distributed lookup protocol that addresses this problem. Chord provides support for just one operation: given a key, it maps the key onto a node.

Data location can be freely and easily implemented top of Chord by associating a key with each data item, and storing the key/data pair at the node to which the key maps. Chord adapts efficiently as nodes join and leave the system, and can answer queries even if the system is continuously changing. Results from theoretical analysis and simulations show that Chord is scalable .

MapReduce: Simplified Data Processing on Large Clusters :

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems

We give two new load balancing protocols whose provable performance guarantees are within a constant factor of optimal. Our first protocol balances the distribution of the keyaddress space to nodes, which yields a load-balanced system when the DHT maps items “randomly” into the address space. To our knowledge, this yields the first P2P scheme simultaneously achieving O(logn) degree, O(logn) look-up cost, and constant-factor load balance (previous schemes settled for any two of the three).Our second protocol aims to directly balance the distribution of items among the nodes

DATAFLOW DIAGRAM:

LEVEL 1:

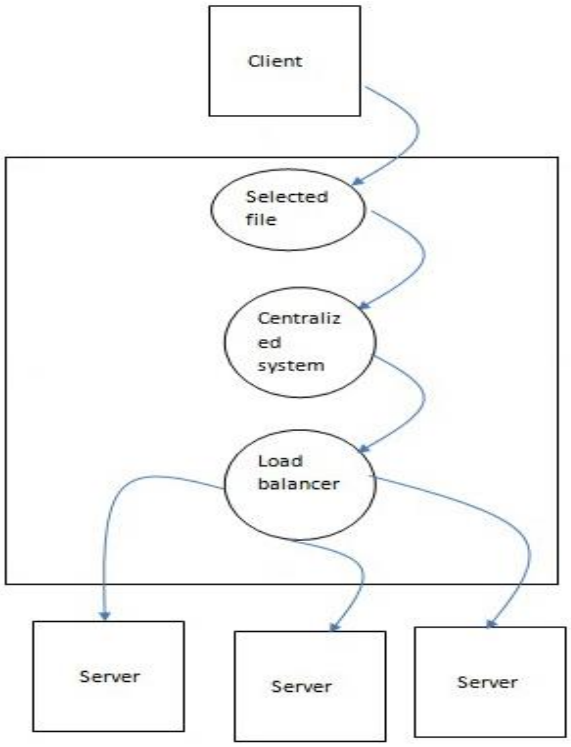


Figure 2

LEVEL 2:

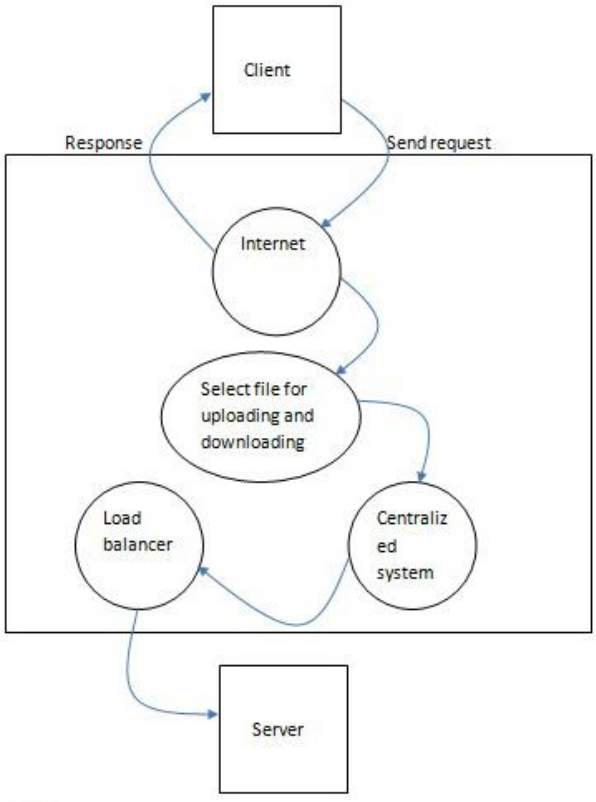


Figure 3

LEVEL 3:

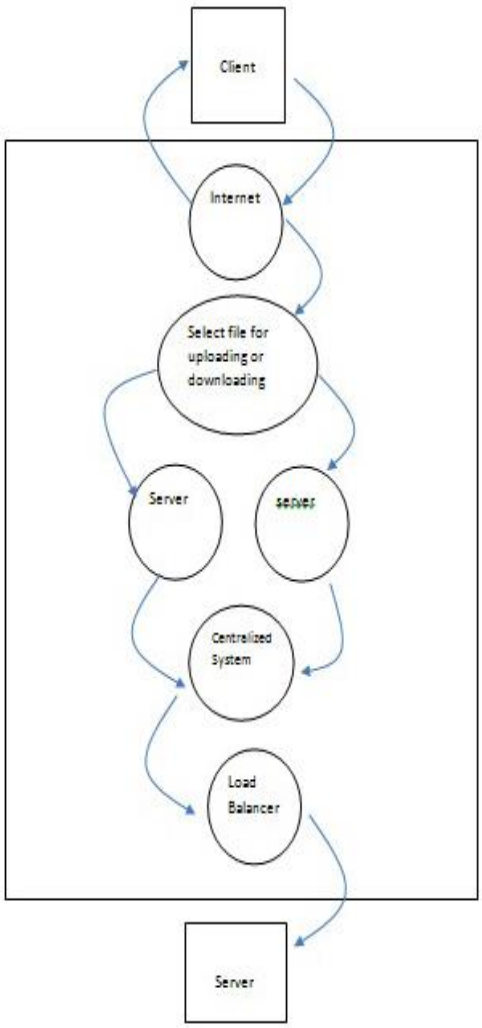


Figure 3

USER DIAGRAM:

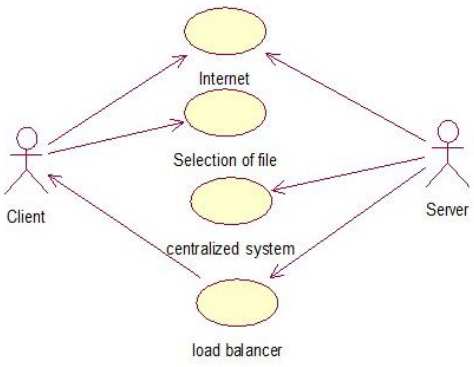
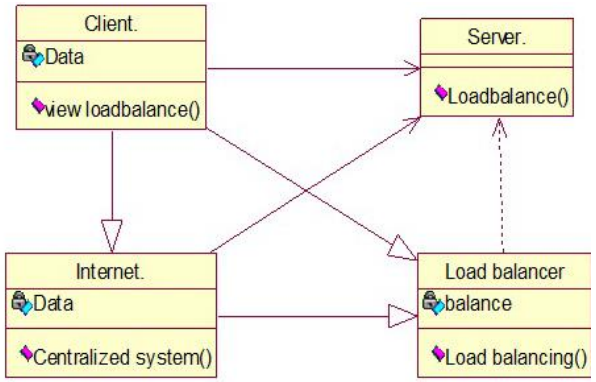


Figure 4

CCLASS DIAGRAM:



Figrue 5

MODULES:

1) Chunk creation
2) DHT formulation
3) Load balancing algorithm

CHUNK CREATION

A file is partitioned into a number of chunks allocated in distinct nodes so that Map Reduce Tasks can be performed in parallel over the nodes. The load of a node is typically proportional to the number of file chunks the node possesses.

Because the files in a cloud can be arbitrarily created, deleted, and appended, and nodes can be upgraded, replaced and added in the file system, the file chunks are not distributed as uniformly as possible among the nodes. Our objective is to allocate the chunks of files as uniformly as possible among the nodes such that no node manages an excessive number of chunks.

DHT Formulation

DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. The chunk servers in our proposal are organized as a DHT network. Typical DHTs guarantee that if a node leaves, then its locally hosted chunks are reliably migrated to its successor; if a node joins, then it allocates the chunks whose IDs immediately precede the joining node from its successor to manage

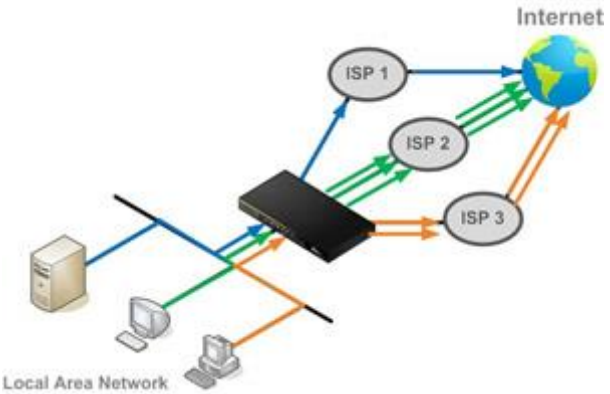
LOAD BALANCING ALGORITHM

In our proposed algorithm, each chunk server node I first estimate whether it is under loaded (light) or overloaded (heavy) without global knowledge. A node is *light* if the number of chunks it hosts is smaller than the threshold.

Load statuses of a sample of randomly selected nodes. Specifically, each node contacts a number of randomly selected nodes in the system and builds a vector denoted by V. A vector consists of entries, and each entry contains the ID, network address and load status of a randomly selected node.

Weighted Balance:

Assign more traffic to a faster link or less traffic to a connection with a bandwidth cap. Set a weight on the scale for each connection and outgoing traffic will be proportionally distributed according to the specified ratio



Figrue 6

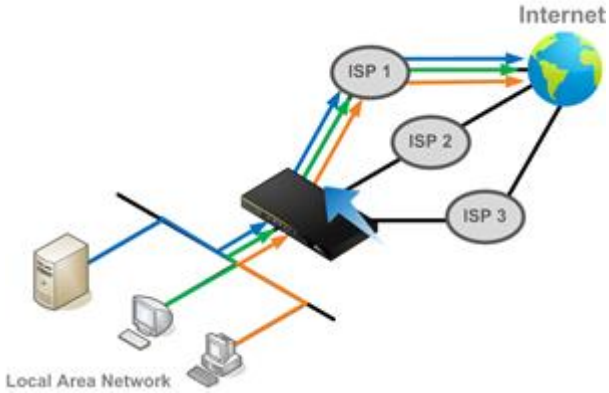
Algorithm:

Load balancing algorithms help you easily fine-tune how traffic is distributed across connections. Each deployment has a unique setup, and Peplink's enterprise grade load balancing features can fulfil all of your special requirements. Create your own rule with the following algorithms and you can sit back and enjoy the high performance routing that Peplink brings to you.

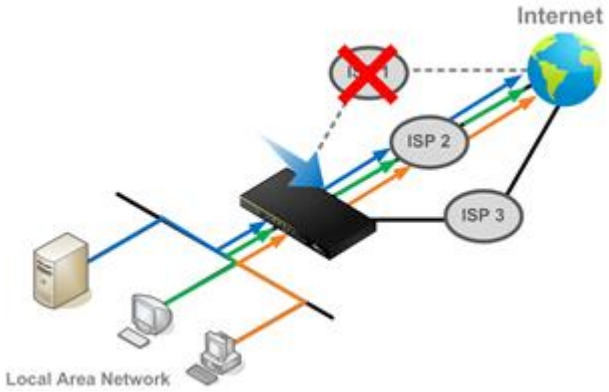
Priority

Route traffic to your preferred link as long as it's available.

- Arrange the connection priority order, and traffic will be routed through the healthy link that has the highest priority in the list. Lower priority links will only be used if the current connection fails.



Figrue 7

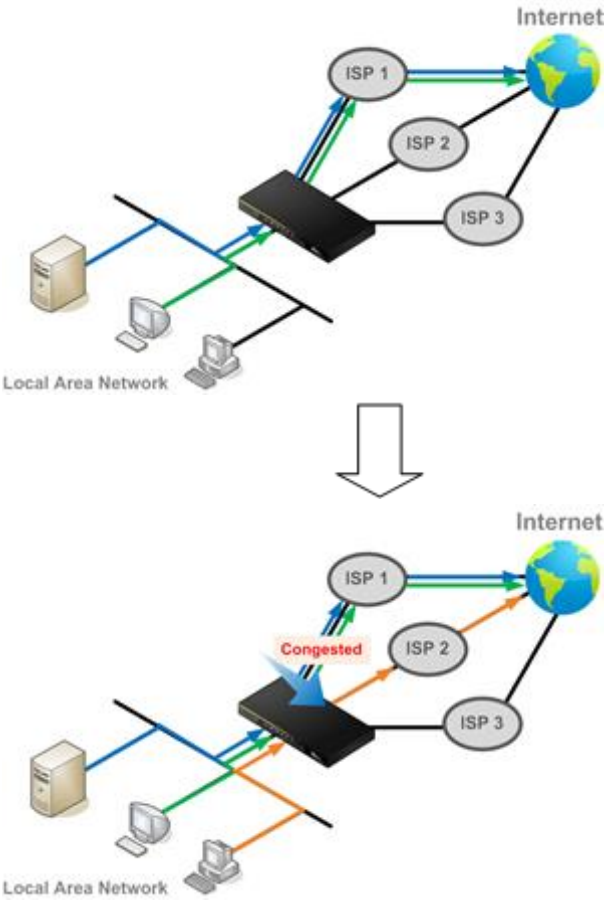


Figrue 8

Overflow

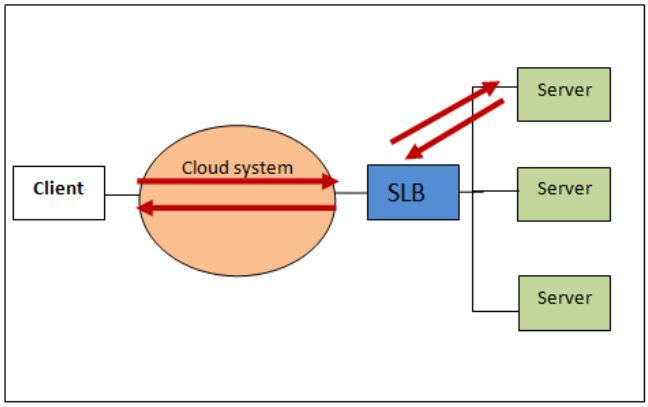
Prevent traffic flow from slowing down when the connection runs out of available bandwidth.

- Drag and drop to arrange the connection overflow order and the highest priority link will route traffic as long as it has not been congested. Once it saturates, the lower priority links will start routing traffic.



Figrue 9

SLB - Server Load Balance - Centralised



Figrue 10

SLB: Server Load Balancing; Server points in Network that will be connected to many Nodes in the System structures , if there is passing of the information of data from server to the clients nodes, in the middle of the servers and the clients node there is a load balancing server to balance the incoming load from the server and it will be the check to the nodes while passing the data if there is having space in nodes to transfer the information of

data, if there is no space to adjust incoming data in nodes, then it will be check another one nodes that having in the sequence , if second nodes having that much of space the data will be store there, this is load balancing in the cloud system or any other system structure.

Advantage:

- ✓ Nodes take more loads. Its benefits is its Node take more Load to Connections
- ✓ Main the consistency and speed.

Disadvantage:

- ✓ Emerging distributed file systems in production systems strongly depend on a central node for chunk reallocation. This dependence is clearly inadequate in a large-scale, failure-prone environment because the central load balancer is put under considerable workload that is linearly scaled with the system size, and may thus become the performance bottleneck and the single point of failure.

II. CONCLUSION

Based on the study of load rebalance and conclusion of load rebalance through distributed File System. If we conclude the whole Scenario, here is nodes are nothing but is Storage and system delivering high performance and balance the maintenance in the network traffic system. Load balancing define that , there is not overloading of data in system nodes, if there is no space in Node, then it will be the choose the another node system in structure. Load rebalance algorithm and Replica management concepts are greatly enhanced and significantly performs well than existing system.

III. REFERENCES

[1]. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processingon Large Clusters," in Proc. 6th Symp. Operating System Design and Implementation (OSDI'04), Dec. 2004, pp. 137–150.

[2]. S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in Proc. 19th ACM Symp. Operating Systems Principles (SOSP'03), Oct. 2003, pp. 29–43.

[3]. Hadoop Distributed File System, ttp://hadoop.apache.org /hdfs/.

[4]. VMware, http://www.vmware.com/.

[5]. Xen, http://www.xen.org/

[6]. Apache Hadoop, http://hadoop.apache.org/.