Adaptive Load Sensing and Rebalancing for Distributed File System in Cloud Environment

B. Nithya Poojaiah[1] and Suryabahadur[2]
[1]M.Tech (CSE), [2]Assistant professor(CSE), MITS, Madanapalle,Chittoor, India
E-mail: ramanjulu.it@gmail.com, suryabahadur@mits.ac.in

Abstract - **Large scale distributed systems such as cloud computing applications are becoming very common. These applications come with increasing challenges on how to transfer and where to store and compute data. This system presents an innovative idea in cloud computing. In a giant cloud we are able to add thousands of nodes together. The main aim is to allot files to those nodes while not creating significant load to any of the nodes, for that files square measure partitioned off into completely different modules. Another objective is to cut back the network inconsistencies and network traffic attributable to the unbalancing of hundreds. The reduction of network inconsistency can result in maximization of network information measure in order that so many numerous such a big amount of, such a giant amount of, such a lot of large applications will run in it. Because of ability to quantify property, we are able to add, delete, update new nodes in order that it supports heterogeneousness of the system. To enhance the potential of nodes we tend to use Distributed file system in Cloud Computing Applications.**

Keywords: **file system, Cloud, Chunk server, Rebalance**

I.INTRODUCTION

Cloud Computing is a technology that uses the internet and central remote servers to maintain data and applications. Extensive scale disseminated frameworks, for example, distributed computing requisitions are getting to be exceptionally basic. These requisitions accompany expanding tests on the most proficient method to exchange and where to store and register information. This framework introduces a creative thought in distributed computing. In a goliath cloud we have the ability to include many hubs together. The primary point is to assign records to those hubs while not making critical burden to any of the hubs, for that documents square mark divided off into totally diverse modules. An alternate target is to decrease the system inconsistencies and system activity attributable to the unbalancing of hundreds. The lessening of system conflict can bring about expansion of system data measure in place that such a large number of various such an enormous measure of, such a monster measure of, such a ton of extensive requisitions will run in it. Due to capability to quantify property, we have the capacity to include, erase, redesign new hubs in place that it upholds heterogeneousness of the framework. To upgrade the potential of hubs we have a tendency to utilize Distributed record framework as a part of Cloud Computing Applications.

II. SYSTEM OVERVIEW

The basic concepts of cloud computing which are required to do the project, means how clouds will store the data and what are the techniques required to provide security for the data while transferring storing and while retrieving the data from clouds.

The concepts related to the project those are central node manage ment and distributed file systems. And how to provide service to the user of different sectors. This helps how node management will provide service to different users.

A.*Existing System With Disadvantages*

Emerging distributed file systems depends on a central node for module reallocation. Will so become the performance bottleneck and therefore the single purpose of failure. Load balance based on distributed file system which is not based on central node. The rebalancing task would performed by the each node individually. Due to this complexity will occur.

III.PROPOSED ARCHITECTURE

A.Proposed System

In previous systems load balancing task will be based on central node. And based on distributed file systems with considering total cloud . The rebalancing algorithm will be applied for all nodes with in the cloud. There fore the original data will be moved from one node to other which is far away from node to which the user will connect and send his request. At first the original user requested data will be at first node to which the user will be connect. Before getting user request some rebalancing tasks may be performed. At that time the data will be moved to the node which is far away to the user. To give response to the user they would re balance total cloud with out considering the path through which the chunk will move from user connected node to current node. We will consider that path and apply re balancing algorithm for that path only. While rebalancing the chunk will be moved to previous node that is we migrate the file chunks to the previous node without randomly selecting node for migration.

Now-a-days the forceful technology is CLOUD computing. In this technology the patrons have organizing of resources and also without any complicated deployment they assign vigorously their resources. The disseminated file systems, Map Reduce programming paradigm, virtualization are the important technologies used for clouds.

The comprising entity will be gradually not succeeded so that they connect to maintain system consistency. The clouds will be used in huge range because of the emphasizes scalability gives by this technique. In Map Reduce programming paradigm, the disseminated file systems are the important structure blocks in cloud computing application. The node provides storage space and also continuously serves computing in this file system. The Map Reduce problems will be performed in equivalently over the nodes by using the file can be partitioned into many number of chunks are assigned in separate nodes.

The distributed hash table (DHT) generalized for providing entity storage space and their repossession by using DHT-based p2p systems like as chord, pastry, Tapestry and CAN. By the hypothetical approach in DHT's they imagined that the nodes in the systems are homogeneous in resources. On condition that the DHT have two restrictions. Primarily, they do not generate perfect load balancing in DHT's the object ID's are used to resorting the strandization of the hash function. The following one is they do not take into account of the heterogeneity behavior of p2p systems, through the uniform structure of overlaid network.

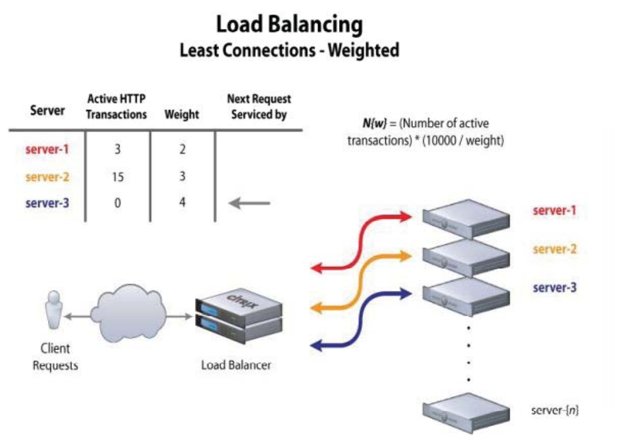
The user can contact with each and every accessible object competently is the main objective of p2p systems is to attach all accessible resources in the p2p network. As the p2p system "competently "is interpreted as determined to make certain fair load disseminated along with the individual peer node. The outcome of achieving load balance in DHT is the essential consequence.

Which answerable for an adjusted allotment of the DHT location space. First and foremost, the regular irregular part of the location space around hubs is not totally adjusted. A few hubs wind up with a bigger part of the locations and along these lines get a bigger parcel of the haphazardly con-veyed things. A critical issue in Dhts is burden adjust the even circulation of things (or other burden measures) to hubs in the DHT. This comes about the uneven conveyance of module servers.

a. Uniformly allocation of chunks with out search the chunk server for user request.
b. We aim to reduce network traffic.
c. Improve the overall system performance.
d. Improve the overall user satisfaction.

B.System Architecture:

The Overall Architecture of the project that shows the workflow of the entire proposed system.


Load Balancing
Least Connections - Weighted
N(w) = (Number of active transactions) * (10000 / weight)
Client Requests
Load Balancer
server-1
server-2
server-3
server-(n)

Server	Active HTTP Transactions	Weight	Next Request Serviced by
server-1	3	2	
server-2	15	3	
server-3	0	4	←

Fig.1 Load Balancing

Advantages

a. Overcome dependency on central node.
b. Reduce movement cost.
c. Reduce load imbalance factor.
d. Overcome performance bottleneck.
e. Consider the path through which chunk will be travel in spite of considering whole cloud. More importantly reduce access time.

IV. MODULES AND IMPLEMENTATION

A. *Modules*

File Partitioning

A file can be upload by a client that file can be partitioned into number of fixed size chunks. Nodes will be allocated by these file chunks. Based on key value pair portioned will occur. Among nodes parallel perform map reduce tasks. For example, each chunk has the same size 64Mbytes.

Collect Load Status

A gossip based aggregation protocol is used to collect the load status of node in the system. This protocol executes a thread to the nodes which allocates the chunks to the node and also contains the load information along with ID and network address. Each node performs the load rebalancing algorithm independently.

Identify Chunk update in Node

The load of the chunk server is proportional to the number of chunks hosted by the server. Hence the number of file that a node handles may increase day by day, also the file chunks which are stored in different node gets deleted dynamically in any of the nodes in the system. Here we identify the chunk deletion of files which are stored in different nodes.

Migrate Chunks to the Previous Node

When there is a file chunk deletion in a node, we migrate the file chunks to the previous node in the system without randomly selecting nodes for file chunk reallocation. Thereby file chunks can be reallocated to the nodes uniformly and also we can reduce the migration time of the file chunks in the system.

Algorithm:

Optimal Path Load Rebalancing Algorithm
Here we assume the entire node have identical capacity and node can handles equal number of chunks.
Assumptions:

F= {f_1, f_2 ,f_3, ….. f_r}
N_i={n_1,n_2,n_3……….n_s}
G= defines capacity of node
Boolean flag=false
Boolean full=true
m =defines the total number of nodes in the system
s =defines the number of chunks in nodes
b= files splitted into number of chunks based on file size
c_k=defines the number of chunks

```
Input : Set of nodes /chunk servers
Output: Migrating chunks to previous node
for (i=1;1<=m; i++) {
if (Ni!=G) {
for (j=1; j<=s; j++) {
for (k=1; k<=b: k++) { Ni[ns] = f[ck] } }
else { return Ni as heavy node} for ( k=1; k<=b; k++) {
if( ck==NULL)
{flag= false; }
else
{ flag=true; } if (ck==flag)
{  Ni+1 migrate its
load ck to Ni  } }
}
```

V. CONCLUSION AND FUTURE WORK

A load rebalancing algorithm to deal with rebalancing problem in large scale, dynamic and distributed file systems in cloud has been presented in this system. Our proposal strives to balance the load of nodes and reduce the demanded movement cost as much as possible. Our proposal is comparable to the centralized algorithm in HDFS and DFS can be incorporated in Single Node or Multi Node cluster environment. The load balancing tasks can be performed by the nodes independently, without synchronization or global knowledge regarding the system. In a load balance cloud the resources can be well utilized and provisioned, maximizing the performance of Map Reduce based applications. The algorithm also outperforms the competing distributed in terms of movement cost and load imbalance factor.

REFERENCES

[1] Andersen, D. Resilient overlay networks. Master's thesis, Department of EECS, MIT, May 2001. http://nms.lcs.mit.edu/projects/ron/.

[2] Bakker, A., Amade, E., Ballintijn, G., Kuz, I., Verkaik,P., Van Der Wijk, I., Van Steen, M., and Tanenbaum., A.The Globe distribution network. In Proc. 2000 USENIXAnnual Conf. (FREENIX Track) (San Diego, CA, June 2000), pp. 141-152.

[3] Chen, Y., Edler, J., Goldberg, A., Gottlieb, A., Sobti, S., AND YIANILOS, P. A prototype implementation of archival intermemory. In Proceedings of the 4th ACM Conference on Digital libraries (Berkeley, CA, Aug. 1999), pp. 28-37.

[4] CLARKE, I. A distributed decentralised information storage and retrieval system. Master's thesis, University of Edinburgh, 1999.

[5] Clarke, I., Sandberg, O., Wiley, B., and Hong, T. W. Freenet: A distributed anonymous information storage and retrieval system. In Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability (Berkeley, California, June 2000). http://freenet.sourceforge.net.

[6] Dabek, F., Brunskill, E., Kaashoek, M. F., Karger, D., Morris, R., Stoica, I., and Balakrishnan, H. Building peer-to-peer systems with Chord, a distributed location service. In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII) (Elmau/Oberbayern, Germany, May 2001), pp. 71-76.

[7] Dabek, F., Kaashoek, M. F., Karger, D., Morris, R., and STOICA, I. Wide-area cooperative storage with CFS. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01) (To appear; Banff, Canada, Oct. 2001).

[8] DRUSCHEL, P., AND ROWSTRON, A. Past: Persistent and anonymous storage in a peer-to-peer networking environment. In Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS 2001) (Elmau/Oberbayern, Germany, May 2001), pp. 65-70.

[9] FIPS 180-1. Secure Hash Standard. U.S. Department of Commerce/NIST, National Technical Information Service, Springfield, VA, Apr. 1995.

[10] Gnutella. http://gnutella.wego.com/.

[11] Karger, D., Lehman, E., Leighton, F., Levine, M., Lewin, D., AND PANIGRAHY, R. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing (El Paso, TX, May 1997), pp. 654-663.

[12] Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, s., Eaton, P., Geels, D., Gummadi, R., Rhea, s., Weatherspoon, H., Weimer, W., Wells, C., and Zhao, B. OceanStore: An architecture for global-scale persistent storage. In Proceeedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000) (Boston, MA, November 2000), pp. 190-201.

[13] LEWIN, D. Consistent hashing and random trees: Algorithms for caching in distributed networks. Master's thesis, Department of EECS, MIT, 1998. Available at the MIT Library, http://thesis.mit.edu/.