ISSN 2348 – 9928
IJAICT Volume 7, Issue 2, February 2020 Doi: 10.46532/ijaict-2020006 Published on 05 (2) 2020

SERVER LOAD BALANCING USING LOAD REBALANCING ALGORITHM

D. Suganthi **N.Banuchanthiran** **V. Sridhar** **T. Rajeswari** **G. Sabareesh**

Department of Computer Science and Engineering,
RVS College of Engineering and Technology,
Coimbatore, Tamilnadu, India

Abstract— Distributed systems are the building block for cloud computing applications. In which the data is stored on a server and accessed as if it is stored on the local client machine. Our proposal is to allocate files uniformly as possible. Such that node has larger number of chunks by load rebalancing algorithm.

Keywords— Load Rebalance, Distributed File System, Load Balance.

I. INTRODUCTION

Distributed systems are specialized for large scale, dynamic and data intensive applications. There are larger number of files that are imbalanced. Load rebalancing provides a mechanism in which the nodes are allocated uniformly as possible to the files. the storage nodes are structured as network based on distributed hash table.

Distributed hash table has the unique handle or identifier in which they are assigned to the file.in which it has a additional functionality that they are self-organize and self-repairable that simplifies the system management.

This additionally reduces the network traffic and moment cost and also exploiting the capable node helps in increasing the performance .it maximizes the network bandwidth available in normal applications. thus this method balances the node. These file systems the node simultaneously serve computing and storage function.

II. LITERATURE SURVEY

In this section we review a few related work on load rebalancing algorithm and the working of distributed table uniformly to other nodes .

In john byers research paper he proposed Distributed hash tables have recently become a useful building block for a variety of distributed applications. However, current schemes based upon consistent hashing require both implementation complexity and substantial storage overhead to achieve desired load balancing goals.

He also defined that these goals can be achieved more simply and more cost-effectively.

In david r.karger he used two protocols to refine the consistent hashing data structure that underlies the Chord (and Koorde) P2P network. Both preserve Chord’s logarithmic query time and near-optimal data migration cost. Our first protocol balances the distribution of the key Address space to nodes, which yields a load-balanced system when the DHT maps items “randomly” into the address space. The second protocol aims to directly balance the distribution of items among the nodes.

As the result of the two analysis they balanced the load efficiently and the distributed hash table mainly uses the balancing distribution among the nodes .

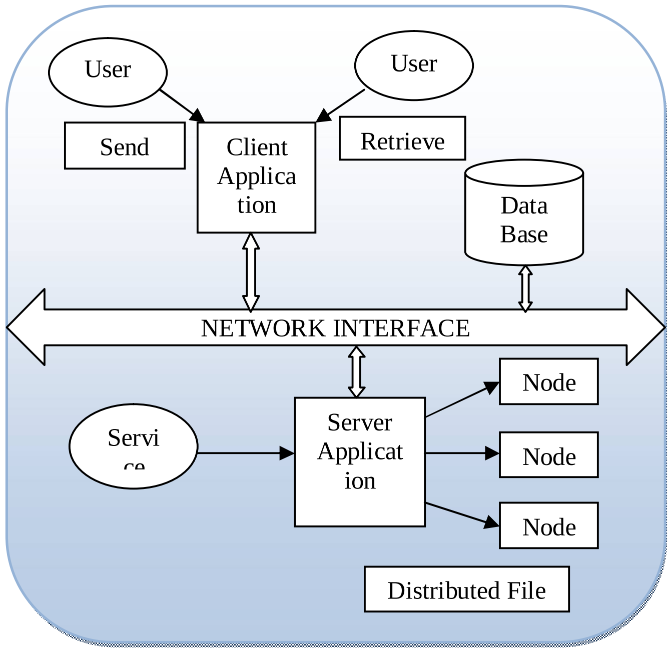
III. PROBLEM DEFINITION

The storage nodes are structured as a network based on distributed hash tables (dhts), discovering a file chunk can simply refer to rapid key lookup in dhts, given that a unique handle (or identifier) is assigned to each file chunk. Dhts enable nodes to self-organize and -repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management.

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. Additionally, we aim to reduce network traffic (or movement cost) caused by rebalancing the loads of nodes as much as possible to maximize the network bandwidth available to normal applications. Exploiting capable nodes to improve the system performance is, thus, demanded.

IV. SYSTEM ARCHITECTURE

System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system.



V. ROPOSED METHODOLOGY

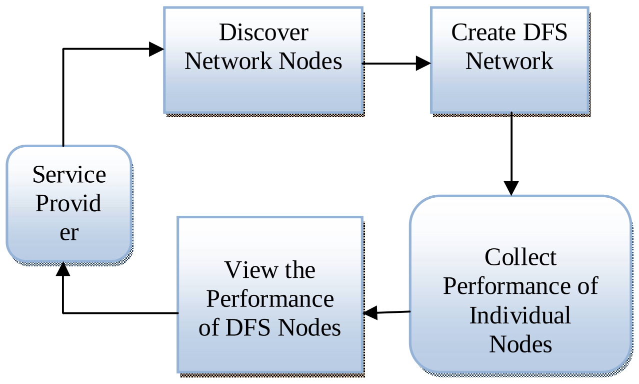
- Client server authentication
- Analysis of network nodes
- Load Rebalancing

5.1 Client Server Authentication

If you are the new user going to consume the service then they have to register first by providing necessary details. After successful completion of sign up process, the user has to login into the application by providing username and exact password. The user has to provide exact username and password which was provided at the time of registration, if login success means it will take up to main page else it will remain in the login page itself., and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

5.2 Analysis Of Network Nodes

In this module, we discover all network nodes name that are connected through local area network. And we can create customized distributed file system by selecting the nodes from network computers. Once if you selected chunk nodes, then the performance of all chunk nodes will be collected for analysis.


Discover Network Nodes
Create DFS Network
Service Provid er
View the Performance of DFS Nodes
Collect Performance of Individual Nodes

5.3 Load Rebalancing

Load Rebalancing module helps to balance the loads of node while searching for requested file chunk. When the requested file chunk is not presented in the central node, then load rebalance helps to balance the load by seeking in chunk nodes.

VI. GIVEN INPUT EXPECTED OUTPUT

Modules	Input	Output
Authentication	User identities such as username, password	Granting access privilege
Analysis of network nodes	LAN communication	Network nodes list
Load rebalancing	Imbalance loads	Load rebalance

VII. LOAD REBALANCING 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 of (1-Δl)a (where 0<=Δl< 1). In contrast, a heavy node manages the number of chunks greater than(1+Δu)a , where 0<=Δl<1. Δl and Δu are system parameters. In the following discussion, if a node i departs and rejoins as a successor of another node j, then we represent node i as node j +1, node j's original successor as node j + 2, the successor of node j's original successor as node j + 3, and so on. For each node i v, if node i is light, then it seeks a heavy node and takes over at most a chunks from the heavy node.

Step 1: we first present a load-balancing algorithm, in which each node has global knowledge regarding the system that leads to low movement cost and fast convergence.

ISSN 2348 – 9928
Doi: 10.46532/ijaict-2020006 Published on 05 (2) 2020

Step 2: we then extend this algorithm for the situation that the global knowledge is not available to each node without degrading its performance.

Step 3: based on the global knowledge, if node i finds it is the least-loaded node in the system, i leaves the system by migrating its locally hosted chunks to its successor i + 1 and then rejoins instantly as the successor of the heaviest node (say, node j).

Step 4: to immediately relieve node j's load, node i requests min {lj _a, a} chunks from j. That is, node i requests a chunks from the heaviest node j if j's load exceeds 2a; otherwise, i requests a load of lj _a from j to relieve j's load.

VIII. CONCLUSION

Our proposal strives to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of physical network locality and node heterogeneity. In the absence of representative real workloads (i.e., the distributions of file chunks in a large scale storage system) in the public domain, we have investigated the performance of our proposal and compared it against competing algorithms through synthesized probabilistic distributions of file chunks.

References

[1] Rao, k. Lakshminarayanan, 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.

[2] D. Karger 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.

[3] P. Ganesan, m. Bawa, and h. Garcia-molina, "online balancing of range-partitioned data with applications to peer-to-peer systems," proc. 13th int'l conf. Very large data bases (vldb '04), pp. 444-455, sept. 2004.

[4] J.w. byers, j. Considine, and m. Mitzenmacher, "simple load balancing for distributed hash tables," proc. First int'l workshop peer-to-peer systems (iptps '03), pp. 80-87, feb. 2003.

[5] G.s. manku, "balanced binary trees for id management and load balance in distributed hash tables," proc. 23rd acm symp. Principles distributed computing (podc '04), pp. 197-205, july 2004.

[6] Y. Zhu and y. Hu, "efficient, proximity-aware load balancing for dht-based p2p systems," ieee trans. Parallel and distributed systems, vol. 16, no. 4, pp. 349-361, apr. 2005.

[7] H. Shen and c.-z. Xu, "locality-aware and churn-resilient load balancing algorithms in structured p2p networks," ieee trans. Parallel and distributed systems, vol. 18, no. 6, pp. 849-862, june 2007.

[8] Q.h. vu, b.c. ooi, m. Rinard, and k.-l. Tan, "histogram-based global load balancing in structured peer-to-peer systems," ieee trans. Knowledge data eng., vol. 21, no. 4, pp. 595-608, apr. 2009.

[9] H.-c. Hsiao, h. Liao, s.-s. Chen, and k.-c. Huang, "load balance with imperfect information in structured peer-to-peer systems," ieee trans. Parallel distributed systems, vol. 22, no. 4, pp. 634-649,apr.2011.