International Journal of Advanced Research in

Computer Science Engineering and Information Technology

Volume: 3 Issue: 1 26-Jun-2014,ISSN_NO: 2321-3337

LOAD BALANCING FOR OPTIMAL SHARING OF NETWORK BANDWIDTH

S.Hilda Thabitha[1], S.Pallavi[2] , P.Jesu Jayarin[3]

[1]PG Scholar,,Dept of CSE,Jeppiaar Engineering College,Chennai,

[2]Research Scholar ,Sathyabama University,Chennai-119.

[3]Associate Professor,Jeppiaar Engineering College.

hildathabitha@gmail.com.

ABSTRACT— *Peers participating in a DHT are able to balance their loads in the virtual servers. In decentralized load balance algorithm in DHT the peers which are participating should be Asymmetric which introduces another load imbalance problem. In our paper, the symmetric load balancing algorithm where each peers independently reallocates. Our proposal exhibits analytical performance in terms of load balance factor and the algorithmic convergence rate and it will not introduce any load imbalance problem due to algorithmic workload.*

1, INTRODUCTION

DHT are key building blocks in the distributed application. Applications based on DHT include storage clouds, file-sharing network and distributed file systems. Load balancing algorithm designed for DHT's based on virtual servers need to take following into consideration. All load balancers are capable of making traffic decisions based on traditional OSI layer 2 and 3 information. More advanced load balancers, however, can make intelligent traffic management decisions based on specific layer 4 – 7 information contained within the request issued by the client. Such application-layer intelligence is required in many application environments, including those in which a request for application data can only be met by a specific server or set of servers. Load balancing decisions are made quickly, usually in less than one millisecond, and high-performance load balancers can make millions of decisions per second. Load balancers also typically incorporate network address translation (NAT) to obfuscate the IP address of the back-end application server. For example, application clients connect directly to a "virtual" IP address on the load balancer, rather than to the IP address of an individual server. The load balancer then relays the client request to the right application server. This entire operation is transparent to the client, for whom it appears they are connecting directly to the application server. An administrator-selected algorithm implemented by the load balancer determines the physical or virtual server and sends the request. Once the request is received and processed, the application server sends its response to the client via the load balancer. The load balancer manages all bi-directional traffic between the client and the server. It maps each application response to the right client connection, ensuring that each user receives the proper response. The efficient load balancer depends upon these factors.

2, CAUSES FOR LOAD IMBALANCE

1) ***Overlay Name space:*** Every node has an identifier in an address space. An inappropriate no distribution over the identifier space can lead to load in balance where unequal portion of name space assigned to notes.
2) ***Request:*** A node which is responsible for popular keys at given time is susceptible to become over load the request load is expressed as number of processed requests per time unit**.**
3) ***Routing:*** A node selects of its neighbors as next hop for a given lookup message. This neighbor and its communication links become heavily loaded in comparison to others. Routing load is express as number of forwarded request per time unit

Underlying topology: When the overlay is agnostic of the underlying topology, the request may go along paths with a huge stretch in comparison to the shortest paths in the underlay. This is a reason for traffic overhead.

3, SYSTEM ANALYSIS

The transmission time is high to send the data from source to receiver. The sensor nodes in the networks worked in the batteries only. So the time delay uses more battery power. So the lifetime of the sensor nodes is also reduced. Another important thing is existing routing algorithm send message to all the nodes to identify the receiver. It consumes more energy. So lot of battery power is used. And the routing is also affected by deadlock and overload problems. Load balancing is such a difficult task in shortest path routing. The transmission of the real time data requires quality of service such as less delay and efficiency.Load imbalance problem is tackled by centralized algorithm. Centralized-DHT's based on virtual servers requires the participating peers to be asymmetric Organize the Rendezvous nodes in a hierarchical manner Virtual server matched with peers through rendezvous node in the lower layer For unpaired virtual server rendezvous peers relay them to the rendezvous in the upper layer to seek reallocation Until an unpaired virtual server reaches a rendezvous in a highest layer.Considering large-scale and dynamic DHT networks, the centralized algorithms may introduce the performance bottleneck and the single point of failure. Consequently, the rendezvous nodes may experience skewed workloads, introducing another load imbalance problem. They may also become the performance bottleneck and the single point of failure. Moreover, the hierarchical network facilitating the load balancing algorithms is prone to node/communication failure, thus demanding sophisticated maintenance for the networks.

3.1 Proposed System

Load balance algorithms overcome the Performance bottleneck and single point of failure. The server does not get overloaded. Movement cost does not occur. System consists of light peers and heavy peers. Light peers queries the virtual servers Heavy peers register the load value to the virtual server. Our paper proposes a dependable and load-balanced P2P system. We view a P2P system as comprising clusters of peers and present techniques for both intra-cluster and inter-cluster load-balancing. Notably, load balancing facilitates reduced query response times. We analyze the trade-offs between the options of migration and replication and formulate a strategy based on which the system decides at runtime which option to use. Incidentally, analysis of trade-offs between migration and replication is expected to facilitate load-balancing. We propose an effective strategy aimed towards automatic self-evolving clusters of peers. This is important since P2P environments are inherently dynamic. The advantages of this approach are that the search operation is expected to require less time and only the peers containing the data items will be involved in answering the query. However, a serious drawback of this strategy is that the overhead required for keeping the meta-information updated may become prohibitively high owing to many reasons. A very large number of data items may be added or updated or deleted within a very short time interval. Nodes may enter or leave the system frequently, thereby introducing significant amount of redundancies to the meta-information or making some of the meta-information obsolete.

4, LOAD BALANCING ALGORITHM

In our proposal, as each heavy peer selects its virtual servers with small sizes to migrate, the resultant movement cost is small. Thus. Analyzing the load balance factor for each peer suffices. The load balance factor of peer i (demoted by LBF_i) is defined as follows:

$$LBF_i \triangleq \frac{\sum_{v \in V_i^{\mathcal{A}}} L_v}{C_i^{max}}$$

Where $\mathcal{A}$ represents our load balancing algorithm. Consequently, due to Eq. (1), we have

$$\left| \sum_{v \in V i^{\mathcal{A}}} L_v - \frac{\sum_{v \in V} L_v}{\sum_{k \in N} C_k^{max}} \times C_i^{max} \right|$$

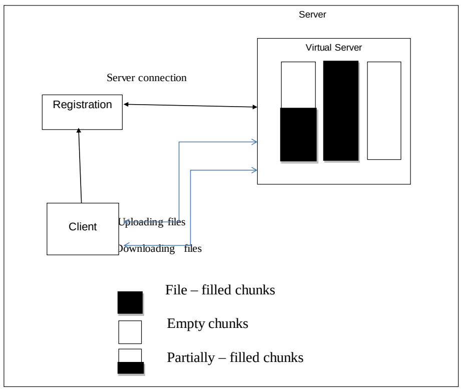
$$= \left| C_i^{max} \left(\frac{\sum_{v \in V_i^{\mathcal{A}}} L_v}{C_i^{max}} - \mu \right) \right|$$

Volume: 3 Issue: 1 26-Jun-2014,ISSN_NO: 2321-3337

$$= |C_i^{max}(LBF_i - \mu)|.$$

5, LOAD BALANCE IMPLEMENTATION

The proposed scheme is made up of client registration, server design, client upload and download, load balancing algorithm phases. Server farms achieve high scalability and high availability through server load balancing, a technique that makes the server farm appear to clients as a single server. Server load balancing distributes service requests across a group of real servers and makes those servers look like a single big server to the clients. The incoming request is directed to a dedicated server load balancer that is transparent to the client. Based on parameters such as availability or current server load, the load balancer decides which server should handle the request and forwards it to the selected server. To provide the load balancing algorithm with the required input data, the load balancer also retrieves information about the servers' health and load to verify that they can respond to traffic. To decide which load balancing solution is the best for the infrastructure, we need to consider availability and scalability.



ARCHITECTURE

Availability is defined by the time between failures. High availability, basically, is redundancy in the system: if one server fails, the others take over the failed server's load transparently. The failure of an individual server is invisible to the client. Scalability means that the system can serve a single client, as well as thousands of simultaneous clients, by meeting quality-of-service requirements such as response time. Under an increased load, a high scalable system can increase the throughput almost linearly in proportion to the power of added hardware resources

6, DYNAMIC LOAD BALANCING IN CHANNEL ALLOCATION:

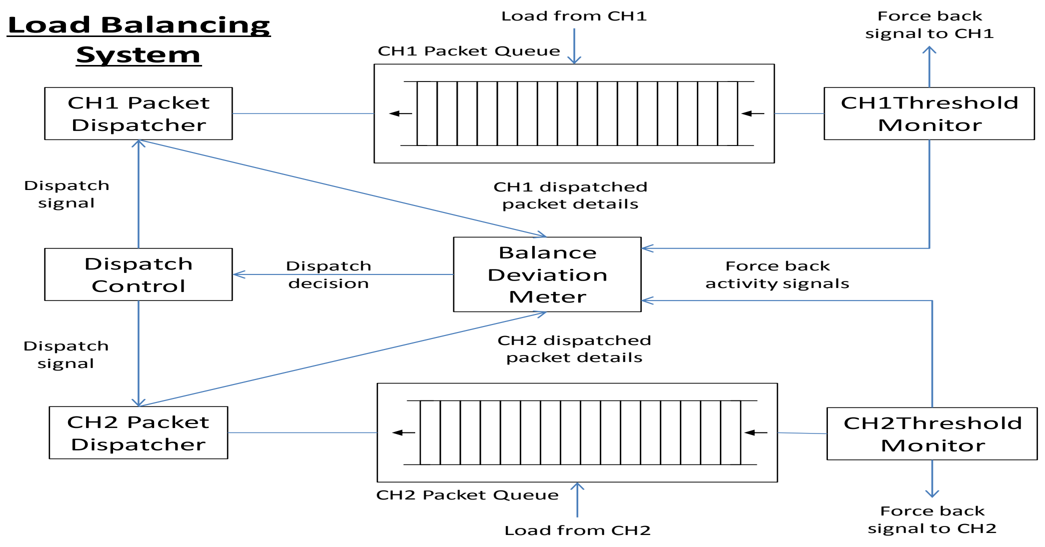
At the same time when the channel get overloaded then they are allocated by the load balancer which as a minimum load. Hence it take less number of time.The components used in dynamic load balancing are

Channel Packet Queue: Packets from a channel that is to be dispatched waits in a queue of the respective channel.

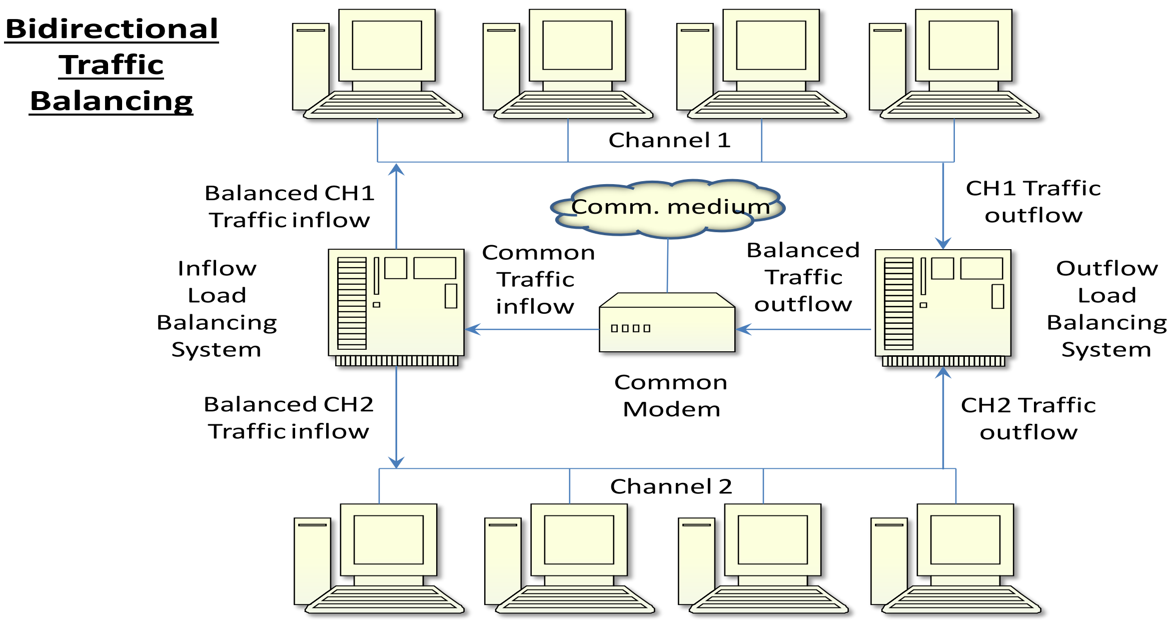
Channel Packet Dispatcher: Given a signal from the dispatch control, the channel packet dispatcher sends the packet from the respective channel queue to the communication medium. It also sends the details of the dispatched packet such as the size, lifetime and waiting time to the balance deviation meter.

Channel Threshold Monitor: This component constantly monitors the packet queue of the specified channel. If the queue length exceeds the set threshold, a force back signal is sent to the respective channel to slow down the speed of requests from that channel.

Dispatch Control: Given the dispatch decision from the balance deviation meter, the dispatch control sends a signal to the appropriate channel packet dispatcher to send out a packet from its channel queue to the communication medium.



Balance Deviation Meter: The balance deviation meter stores a history of events from which it makes its decision as to which channel packet is next to be dispatched. The channel to dispatch the first packet can be selected in random or in some priority. Then the incoming packets are dispatched according to the set load balance deviation.It may take in account the amount of traffic in each channel by considering the queue size and the number of force backs encountered in that queue from the respective threshold monitor.E.g., consider the set load balance deviation to be 100 kB. This means that the load balancer balances the load between the allocated channels with the tolerance of size 100 kb between the channels.


Bidirectional Traffic Balancing
Channel 1
Balanced CH1 Traffic inflow
Comm. medium
CH1 Traffic outflow
Inflow Load Balancing System
Common Traffic inflow
Balanced Traffic outflow
Outflow Load Balancing System
Common Modem
Balanced CH2 Traffic inflow
CH2 Traffic outflow
Channel 2

7, MODES OF OPERATION

There are three modes of operation in dynamic load balancing allocation

Inflow traffic balancing: Each traffic outflow from each channel is given to the modem and it is given to the load balancing system. The loads are dispatched from the queue and send it to the channel where the load is minimum. Hence the traffic is reduced.

Outflow traffic balancing: The traffic outflow is initially given to the load balancing system and the loads are dispatched from the queue and send it to the modem.

Bidirectional flow traffic balancing: The load balancing system is used in inflow traffic balancing as well as outflow traffic balancing

S. No	Channel to dispatch packet	Packet size	Cumulative load		Load Difference (CH1–CH2)	Next channel to dispatch packet
			CH1	CH2		
1.	CH1	110	110	0	110	CH2
2.	CH2	210	110	210	-100	CH1
3.	CH1	90	200	210	-10	CH1
4.	CH1	130	330	210	120	CH2
5.	CH2	250	330	460	-140	CH1
6.	CH1	160	490	460	30	CH1

CONCLUSION AND FUTURE ENHANCEMENTS

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. The computer simulation results are encouraging, indicating that our proposed algorithm performs very well.

Our proposal is comparable to the centralized algorithm and dramatically outperforms the competing distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead.

REFERENCES

[1] S. Ajmani, D. Clarke, C.-H. Moh, and S. Richman, “ConChord: Cooperative SDSI certificate storage and name resolution,” in Proc. 1st Int. Workshop Peer-to-Peer Systems, Cambridge, MA, Mar. 2002.

[2] A. Bakker, E. Amade, G. Ballintijn, I. Kuz, P. Verkaik, W. I. van der, M. van Steen, and A. Tanenbaum, “The globe distribution network,” in Proc. 2000 USENIX Annu. Conf. (FREENIX Track), San Diego, CA, June 2000, pp. 141–152. STOICA et al.: CHORD: SCALABLE PEER-TO-PEER LOOKUP PROTOCOL 31

[3] Y. Chen, J. Edler, A. Goldberg, A. Gottlieb, S. Sobti, and P. Yianilos, “A prototype implementation of archival intermemory,” in Proc. 4th ACM Conf. Digital Libraries, Berkeley, CA, Aug. 1999, pp. 28–37.

[4] I. Clarke, “A distributed decentralised information storage and retrievalsystem,” Master’s thesis, Univ. Edinburgh, Edinburgh, U.K., 1999.

[5] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” in Proc. ICSI Workshop Design Issues in Anonymity and Unobservability, Berkeley, CA, June 2000, [Online]. Available: http://freenet.sourceforge. net.

[6] R. Cox, A. Muthitacharoen, and R. Morris, “Serving DNS using Chord,”in Proc. 1st Int. Workshop Peer-to-Peer Systems, Cambridge, MA, Mar2002.

[7] F. Dabek, F. Kaashoek, D. R. Karger, R. Morris, and I. Stoica, “Wide-area cooperative storage with CFS,” in Proc. ACM Symp. Operating Systems Principles, Banff, Canada, 2001, pp. 202–215.

[8] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, “Dynamo: Amazon’s Highly Available Key-value Store,” in Proc 21st ACM Symp. Operating Systems Principles (SOSP’07), Oct. 2007, pp. 205–220.

[9] Gnutella. [Online]. Available: http://gnutella.wego.com/

Volume: 3 Issue: 1 26-Jun-2014,ISSN_NO: 2321-3337

[10] J. Li, J. Jannotti, D. De Couto, D. R. Karger, and R. Morris, "A scalable

location service for geographic ad hoc routing," in Proc. 6th ACM Int. Conf. Mobile Computing and Networking, Boston, MA, Aug. 2000, pp. 120–130.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in Proc. ACM SIGCOMM, San Diego, CA, Aug. 2001, pp. 161–172.

[12] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," LNCS 2218, pp. 161–172, Nov. 2001.

[13] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger,M. F. Kaashoek,F. Dabek, and H. Balakrishnan, "Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Trans. Netw., vol. 11, no. 1, pp. 17–21, Feb. 2003.

[14] J. Stribling, E. Sit, M. F. Kaashoek, J. Li, and R. Morris, "Don't Give Up on Distributed File Systems," in Proc. 6th Int'l Workshop Peer-to-Peer Systems (IPTPS'07), Feb. 2007.

[15] A Symmetric Load Balancing Algorithm with

Performance guarantees with Distributed Hash Table by Hung-Chang Hsiao†, Che-Wei Chang