Efficient Equalization Work in Peer-to-Peer System

I. Abinaya and S. Faizal Mukthar Hussian

Department of Computer Science, Mohamed Sathak Engineering College, Kilakarai, Tamil Nadu, India

abiilango@gmail.com

Abstract **- Peer-to-peer (P2P) architectures present benefits like scalability, load balancing and fault tolerance when compared to Client/Server architectures. Structured P2P systems furthermore feature efficient lookup mechanisms: an exact search is usually performed with logarithmic complexity relative to the number of peers in the system. The notion of virtual servers, peers participating in a heterogeneous, structured peer-to-peer (P2P) network may host different numbers of virtual servers, and by migrating virtual servers, peers can balance their loads proportional to their capacities close by in the address space. Load balancing is a critical issue for the efficient operation of peer-to-peer networks.This project include a simple protocol that balances load by moving nodes to arbitrary locations "where they are needed."**

Keywords **- Peer-to-peer systems, load balance, heterogeneity, Distributed Hash Table**

I. Introduction

Client/Server organization is an asymmetrical structure because there is a clear distinction between server and client roles: a server is a centralized computer system which offers its resources or services to a group of clients. The network address of a server has to be well-known and it must be reachable anytime. These restrictions do not apply for client systems, which do not have to be accessible all the time and which address must not be fixed. On the opposite side, P2P systems have symmetry in roles , where each host can be, at the same time, a server and a client, producer and consumer, of services and/or resources of other hosts in the network. By literature definition a P2P system is a "self-organizing system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services.

A. DHT

A Distributed Hash Table (DHT) is a distributed and often decentralized mechanism for associating Hash values (keys) with some kind of content. Participants in the DHT each store a small section of the contents of the hashtable. The main advantage of DHTs is their scalability. DHTs obviated the server from P2P networks. A DHT is a hash table that partitions the keyspace and distributes the parts across a set of nodes. They can be used to build complex services such as distributed file systems, peer-to-peer file sharing systems, cooperative web caching, multicast, anycast, and domain name services. Several DHT implementations like Chord, Pastry and Tapestry are available. Open DHT is a publicly accessible DHT service whose clients do not need to run a DHT node.

II. Literature Review

Load balancing algorithms for example CHORD and PASTRY involved handling the load of the peers in network. All the algorithms participated in centralized manner which means the server only holds all the load balancing information about the peers

A. Existing System

Previous work on consistent hashing assumes that each node is aware of most of the other nodes in the system, an approach that does not scale well to large numbers of nodes.In contrast, each Chord node needs "routing" information about only a few other nodes. Because the routing table is distributed.

B. Limitations of the Existing System

It is not static, small-scale systems and/or homogeneous environments. The centralized algorithms may introduce the performance bottleneck and the single point of failure. Load imbalance factor in a typical distributed hash tables or DHTs. Our solution attacking the load balancing problem need not rely on any auxiliary tree networks. Chord does not provide nonymity, but its lookup operation runs in predictable time and always results in success or definitive failure.

C. Proposed System

The load balancing is fully based on the decentralized manner. Each participant in the peer salvation to peer network know about the load(capacity) of the all other participating node in that network.If overload happens in any node then sender node recollects its loads. The reallocation of a virtual server from a source peer to a destination peer can be simply done by simulating the leave and join operations offered by a typical DHT.

D. Advantages of Proposed System

It is static, small-scale systems and/or homogeneous environments.Load balance factor in a typical distributed hash tables or DHTs.Many-to-many framework essentially reduces the load balancing problem to a centralized algorithmic problem.It is fully distributed solutions to the load balancing problem.

III. Methodology

A. Peer-to-Peer (P2P) Architecture

P2P computing or networking is a distributed application architecture that partitions tasks or workloads among peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes. Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to for central coordination by servers or stable hosts.

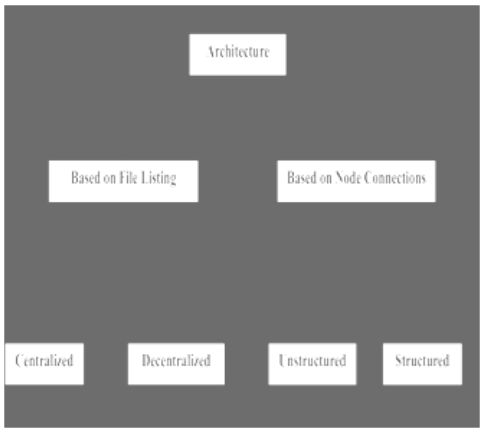

Fig.1 Peer-to-peer (P2P) Architecture

Peers are both suppliers and consumers of resources, in contrast to the traditional (receive). The peer-to-peer application structure was popularized by file sharing systems like Napster. The concept has inspired new structures and philosophies in many areas of human interaction. Peer-to-peer networking is not restricted to technology, but covers also social processes with a peer-to-peer dynamic. In such context, social peer-to-peer processes are currently emerging throughout society.Peer-to-peer systems often implement an abstract overlay network, built at Application Layer, on top of the native or physical network topology. Such overlays are used for indexing and peer discovery and make the P2P system independent from the physical network topology. Content is typically exchanged directly over the underlying Internet Protocol (IP) network. Anonymous peer-to-peer systems are an exception, and implement extra routing layers to obscure the identity of the source or destination of queries.

B .Modules

- Peer Connection Establishment
- Peer Task Assigning
- Load Monitoring
- High Priority Task Assignment
- Result Evaluation

1) Peer Connection Establishment : The interconnection networks can exist in physical or logical dimensions as well as wired and wireless domains. Decentralized and distributed search techniques are required for a network composed of transient populations of nodes having intermittent connectivity and dynamically assigned IP addresses. In this first module has connection of all peers using their IP address.

2) Peer Task Assigning : In this module the server can assigning the task for eash neighbor peer present in the network. Before assigning the task all peers has equal priority. In this network connection any peer can act as server and client .so if the server can assign the task to the client or nighber peer and then this client will forward this task to the server.

3) Load Monitoring : The aim is to reduce the number of DHT lookups per search by mapping related keywords to nearby peers on the overlaySo the server peer will monitor the load for each peer by using this derivation

$$\mathcal{A} = \frac{\sum_{v \in V} l_v}{\sum_{i \in N} C_i},$$

4) High Priority Task assignment : The server peer will search the high priority task in this networked peer then It will assign the task to measure the impact of this phenomena it defined hit rate as the average percentage of matches that are discovered by a query.

5) Result Evalaution : Finally the performanance of the peers are measured by calculation of message overhead and movement cost of each peers.Failure of indexing peers may result into unreachable leaf peers.

C. Algorithm Sketch

The entire hash space provided by a DHT is [0, 1], and each virtual server in the DHT has a unique ID selected independently and uniformly at random from the space [0, 1]. Let N be the set of participating peers, and V be the set of virtual servers hosted by the peers in N in the DHT. Denote the set of virtual servers in peer i by Vi. Each peer i Є N in our proposal estimate the load, which is denoted by Ti, which it should perceive.mm,mmbf.The participating peers in our proposal balance their loads periodically every time period (e.g., T minutes). However, we impose no global synchronization among the peers, and in our performance study to be discussed later on, each peer schedules its load balancing algorithm every T minutes accordin to its local clock.

Psuedo Code

$$\mathcal{A} = \frac{\sum_{v \in V} l_v}{\sum_{i \in N} C_i},$$

```
input : Pr(X), Pr(Y), I, n
Ã ← ln n · ∫_{y=0}^{L_max} (y F_Y(y) dy) / ∫_{x=0}^{C_max} (x F_X(x) dx);
T_i ← Ã × C_i + ε;
switch LOAD(i) do
    case > T_i
        U_i ← ∅;
        while LOAD(i) > T_i and V_i ≠ U_i do
            v ← arg min{L_v | v ∈ V_i − U_i};
            find j ∈ I satisfing Eq. (4) to accommodate v;
            if j accepts v then
                V_i ← V_i − {v};
            U_i ← U_i ∪ {v};
        break;
    case ≤ T_i
        while LOAD(i) < T_i do
            receive v to host;
            V_i ← V_i ∪ {v};
        break;
```

III. Performance Analysis

The performance results of this proposal operating in a dynamic environment. As shown in Fig. 2, tree performs fairly. Even each peer in tree has the accuracy information, and it can exactly determine whether it is underloaded or not. In the tree-based approach, a rendezvous peer implements the best-fit scheme to pair underloaded peers and virtual servers.The probability distributions, each peer identifies whether it is underloaded and then reallocates its loads if it is overloaded. Our proposal is driven by rigorous performance analysis and validated by extensive simulations. The performanance results reveal that that our proposal performs well and that it is comparable with the centralized directory approach and out

performs the tree-based solution in terms of the load imbalance factor, the movement cost of virtual servers, and/ or the protocol message overhead. In particular, while the centralized directory and tree-based approaches introduce hotspots to the system, the participating peers in our proposal perceive the nearly identical workloads in manipulating our load balancing algorithm.

A. Movement Cost

In Fig. 2, the movement costs of tree and in this proposal are normalized to that of centralized directory. Centralized directory performs best in terms of the load imbalance factor .However, this is at the expense of a greater movement cost. In contrast, tree and this proposal have a comparable movement cost less than that of centralized directory.

B. Protocol Message Overhead

In Fig. 1, the load balancing algorithms we investigate in this work introduce the protocol message overhead. Let us consider centralized directory. The protocol message overhead includes the messages introduced by each peer, i, to update the directory on the knowledge regarding i's capacity and the load of each i's local virtual servers. In addition, the message overhead comprises those generated by the directory to inform the overloaded peers which of the peers can share loads. each peer issues independent routes each for its local virtual servers in the DHT network to update the directory, while the directory then informs the overloaded peers though end-to-end messages where to migrate their loads.

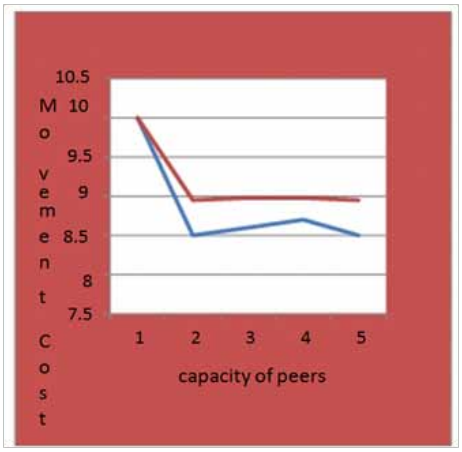

Fig .2 Movement Cost

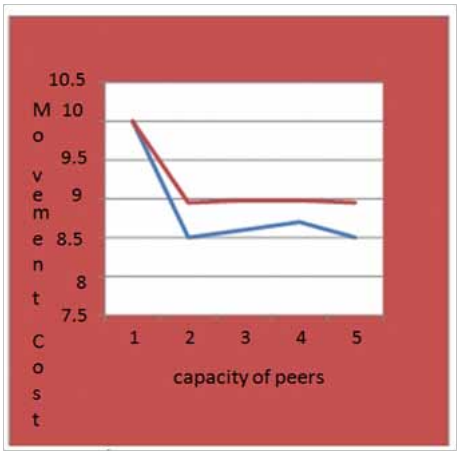

Fig .3 Message Overhead

IV. Conclusion

Peer to peer network need some decentralized algorithm for efficient load balancing technique. These algorithms are simple and easy to implement, so an obvious next research step should be a practical evaluation of these schemes. In addition, three concrete open problems follow from this work. First, it might be possible to further improve the consistent hashing scheme as discussed earlier . Second, it would be interesting to determine whether this item balancing protocol also works for the case where the cost of storing an item is node-dependent, because some nodes

have greater storage capacity or bandwidth than others. And finally, the range search data structure does not easily generalize to more than one order.The effort incurred by this load-balancing approach is low because it requires no extra communication but it gather statistic data from normal interactions and "piggy-back" the load-balancing into the standard information exchanges required by the DHT. And also preserve key ordering, which is vital for semantically rich queries like range queries. In this mechanism allows the access structure to adapt and restructure dynamically, but preserves its structural properties, unlike other mechanisms which require extrinsic mechanisms like redirection pointers, that make queries inefficient.

References

[1] Chen..C and. Tsai K.C, "The Server Reassignment Problem for Load Balancing in Structured P2P Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 12, no. 2, pp. 234-246, Feb. 2008.

[2] Gendreau T.B and L.M. Ni, C.-W. Xu, and, "A Distributed Drafting Algorithm for Load Balancing," *IEEE Trans. Software Eng*., vol. 11, no. 10, pp. 1153-1161, Oct. 1985.

[3] Rowstron .A and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems,"*Lecture Notes in Computer Science*, pp. 161-172, Springer, Nov. 2001.

[4] Shen .H 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.

[5] Zhu.Y and Hu.Y, "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.

[6] 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. Networking,* vol. 11, no. 1, pp. 17-21, Feb. 2003.

[7] T.L. Casavant and J.G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Trans. Software Eng*., vol. 14, no. 2, pp. 141-154, Feb. 1988.

[8] 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.

[9] S. Saroiu, P.K. Gummadi, and S.D. Gribble, "Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. Multimedia Computing and Networking* (MMCN '02), Jan. 2002.

[10] A. 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.