A Structured Architecture P2P Systems Using Load Balancing With Imperfect Information

N. HARISH, C. RAMA MOHAN, U. SHESADRI

M.Tech, Student of Vaagdevi Institute of Technology & Science, Proddatur, A. P. INDIA.

harish.pdtr@gmail.com

Associate Professor of Vaagdevi Institute of Technology & Science, Proddatur, A. P. INDIA.

ramamohanchinnem@gmail.com

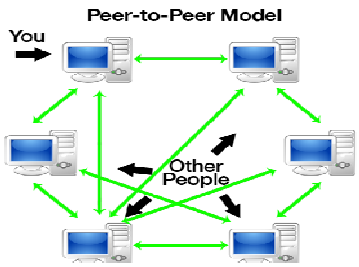
Associate Professor of Vaagdevi Institute of Technology & Science, Proddatur, A. P. INDIA

Abstract:

In this paper, with the concept of virtual servers, and peers participating in a different, structured peer-to-peer (P2P) networks may host different numbers of virtual servers, and by migrating virtual servers, peers can balance their loads proportional to their capacities. we present, a modern load balancing algorithm that mainly get the imperfect information based on the probability estimations we can get the peer and virtual server information based on that information we can divide the total work into sub work that is to allocate into real servers that must summed in the virtual servers by this information we can provide the load in peer to peer systems(p2p) .

Index Terms— P2P, DHT's, Load Balancing, imperfect Information, Virtual Server

I. INTRODUCTION



Peer-to-peer (**P2P**) computing or networking is a distributed application architecture that partitions tasks or workloads between 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 other network participants, without the need for central coordination by servers or stable hosts. Peers are both suppliers and consumers of resources, in contrast to the traditional client–server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers.

The first P2P distributed system platform was Pipes Platform by Peer Logic. One of Peer Logic's first licensees was Texas Instruments in 1993. While P2P systems were used in many

application domains, the architecture was popularized by the file sharing system Napster, originally released in 1999. The concept has inspired new structures and philosophies in many areas of human interaction. In such social contexts, peer-to-peer as a meme refers to the enabled by Internet technologies in general.

A peer-to-peer network is designed around the notion of equal nodes simultaneously functioning as both "clients" and "servers" to the other nodes on the network. This model of network arrangement differs from the model where communication is usually to and from a central server. A typical example of a file transfer that uses the client-server model is the (FTP) service in which the client and server programs are distinct: the clients initiate the transfer, and the servers satisfy these requests. Structured networks

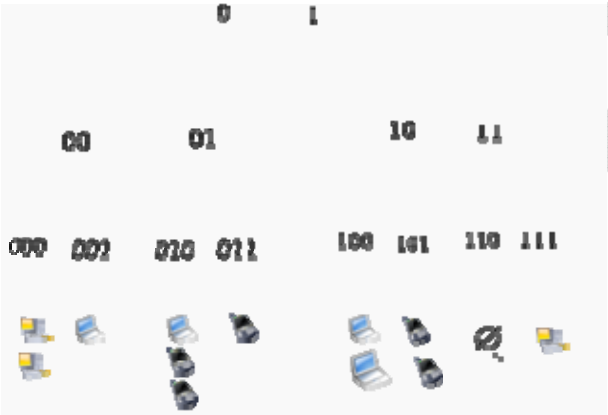


Fig: a structured P2P network.

In *structured peer-to-peer networks* the overlay is organized into a specific topology, and the protocol ensures that any node can efficiently search the network for a file or resource, even if the resource is extremely rare. The most common type of structured P2P networks implement a (DHT), in which a variant of consistent is used to assign ownership of each file to a particular peer. This enables peers to search for resources on the network using a that is, (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key.

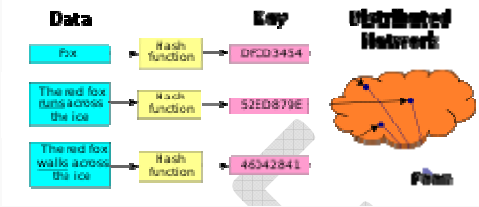


Fig: Distributed hash tables

However, in order to route traffic efficiently through the network, nodes in a structured overlay must maintain lists of neighbors that satisfy specific criteria. This makes them less robust in networks with a high rate of *churn* (i.e. with large numbers of nodes frequently joining and leaving the network). More recent evaluation of P2P resource discovery solutions under real workloads have pointed out several issues in DHT-based solutions such as high cost of advertising/discovering resources and static and dynamic load imbalance.

summetion of v€Vi Li=A---------- (1)

Notable distributed networks that use DHTs include distributed tracker. Some prominent research projects include DHT-based networks have also been widely utilized for accomplishing efficient resource discovery For systems, as it aids in resource management and scheduling of applications.

PEER-TO-PEER (P2P) networking is a new technique for next-generation network applications. P2P networks are application-level networks built on top of end systems, which provide message routing and delivery. Popular

(P2P) applications include file sharing, distributed computing, media streaming, and others, which rely on their P2P network infrastructures for message routing, information searches, and/or content delivery. P2P network infrastructures are key building blocks in the

A= **summetion of v€Vi Li** /summation of i€N,Ci

design and implementation of successful P2P applications. Potential P2P substrates are based on distributed hash tables or DHTs for short. As peers participating in a DHT are often heterogeneous, the work introduces the notion of virtual servers to cope with the heterogeneity of peers. Participating peers in a DHT can host different numbers of virtual servers, thus taking advantage of peer heterogeneity. Let V be the set of virtual servers in the system, and N be the set of participating peers. By reallocating virtual servers in V to peers in N Lv is the load value of a virtual server v€vi ; Ci is the capacity value of peer i; and is the expected load value per unit capacity. We denote LBFi= summetion of v€Vi Lv/Ci as the load imbalance factor of peer i. Minimizing |LBFi-A| More precisely, we attempt to determine a subset S,<V , denoting the virtual servers selected to be moved, such that the movement cost, MC, defined in the following is minimized: Accordingly, by the minimization of and , we intend to distribute loads to participating peers evenly with the minimum movement cost. Accordingly, by the minimization of 1 and 2, we intend to distribute loads to participating peers evenly with the minimum movement cost.

We note, that 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. Second, Lv of any virtual server v at a particular time is the summation of the loads of the objects (data items) stored in v at that time. Possible metrics for measuring the load of an object may include the storage size of the object, the mean bandwidth required for serving the object, and so on. Third, Ci represents the maximum load that peer i is willing to hold, which may denote the available disk space, processor speed, and the bandwidth of peer i, for example. We do not assume a particular resource.

$$MC = \sum_{v \in S} L_v.$$

Unlike the participating peers in our proposal operate independently, and they need not rely on

A= **summetion of v€Vi Lv** /summation of i€N,Ci

$\tilde{A} \leftarrow \ln n \cdot \frac{\int_{y=0}^{C_{max}} (yF_Y(y)dy)}{\int_{x=0}^{C_{max}} (xF_X(x)dx)}$;
$T_i \leftarrow \tilde{A} \times C_i + \epsilon$;
switch LOAD(i) **do**
  case $> T_i$
    $U_i \leftarrow \emptyset$;
    while LOAD(i) $> T_i$ and $V_i \neq U_i$ **do**
      $v \leftarrow \arg\min\{L_v | v \in V_i - U_i\}$;
      find $j \in \mathcal{I}$ satisfing Eq. (4) to accommodate v;
      if j accepts v **then**
        $V_i \leftarrow V_i - \{v\}$;
      $U_i \leftarrow U_i \cup \{v\}$;
    break;
  case $\leq T_i$
    while LOAD(i) $< T_i$ **do**
      receive v to host;
      $V_i \leftarrow V_i \cup \{v\}$;
    break;

Algorithm 1: REALLOCATION(i). Peer i computes the reallocation of its local virtual servers, where LOAD(i) $= \sum_{v \in V_i} L_v$.

Fig: Basic algorithm [15]

dedicated nodes to pair virtual servers and participating peers, eliminating the performance bottleneck and single point of failure. On the other hand, the decentralized algorithm not only depends on global knowledge (i.e., A) but also requires coordination among the participating peers to transfer their virtual servers; this may introduce extra workload to the peers responsible for the coordination. In contrast, each peer in our proposal independently and solely manipulates partial information of the system and then reassigns its virtual servers to other peers based on the approximated system state. While it can be compared with, our proposal is independent of the geometry of DHTs.

II. SYSTEM MODEL AND PROBLEM STATEMENT

For Load Balance with imperfect information in structured p2p by using the probability estimations we are going evaluate the load and capacities of peers and virtual servers , so this is the previous project implementation in this paper mainly we are going to sum the value that must be divided according to the values are coming from the probability ratios we are taken the values that must be represented in the form of algorithm we are going to show the detailed view of project details by this way we are going populate the load balance in structured p2p systems.

Proposed system:

In this paper, we assume that 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 estimates the load, which is denoted by Ti, that it should perceive, where is an estimation for the expected load per unit capacity, i.e.and Є is a predefined system parameter. If the current total load of i is greater than Ti (i.e., i is overloaded), then i migrates some of its virtual servers to other peers. Otherwise, i is under-loaded, which does nothing but waits to receive the migrated virtual servers. For an overloaded peer (e.g., peer i), i picks those virtual servers for migration, such that 1) i becomes under-loaded, and 2) the total movement cost, MC, in (2) is minimized due to the reallocation. If i is an under-loaded peer, then i may be requested to receive a migrated virtual server, and i accepts such a virtual server if the added load due to the virtual server will not overload itself; otherwise, i rejects such virtual server. Algorithm 1 (REALLOCATION (i)), which given as above illustrates our idea. Notably, similar to studies in [3], [4], [5], 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; each peer schedules its load balancing algorithm every T minutes according to its local clock. Next, Algorithm 1 intends to minimize MC by simply choosing a virtual server v with the minimum

$$T_i = \tilde{\mathcal{A}} \times C_i + \epsilon,\ \tilde{\mathcal{A}}$$

load each time until the hosting peer becomes under-loaded. Provide valuable information to help the participating peers estimate A, and the overloaded peers discover the under-loaded peers to share their excess loads server will not overload itself; otherwise, otherwise, i rejects such virtual server. Algorithm 1 (REALLOCATION (i)), which given as above illustrates our idea. Notably, similar to studies in [3], [4], [5], 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; each peer schedules its load balancing algorithm every T minutes according to its local clock. Next, Algorithm 1 intends to minimize MC by simply choosing a virtual server v with the minimum load each time until the hosting peer becomes under-loaded.

provide valuable information to help the participating peers estimate A, and the overloaded peers discover the under-loaded peers to share their excess loads

III. RELATED WORK

Earlier studies have proposed load balancing algorithms, targeting at static, small-scale, and/or homogeneous environments. Due to space limitation, we provide a concise review of the load balancing techniques designed for DHTs in this section. (e.g., Chord [1] and Pastry [2]) assume that the loads of objects in O are identical. The load of a peer can thus be estimated as the number of objects hosted by the peer. Assuming that the load of each object o Є O is lo =1, earlier studies, show that the load imbalance factor in a typical DHT can be up to , where n =|N| is the total number of nodes participating in the system virtual server serving as an elementary entity for balancing loads among peers. We note that if a virtual server v manages the key subspace S, then the objects, which may have unequal loads and whose keys are within S, contribute their loads to v. The idea of virtual servers enables a DHT to reallocate the virtual servers, such that the resultant load of a peer is proportional to the peer's capacity. Based on the concept of virtual servers, the many-to-many framework is presented in to cope with the load imbalance in a DHT. the many-to-many framework essentially reduces the load balancing problem to a centralized algorithmic problem. As the entire system heavily depends on the directory nodes, the directory nodes may thus become the performance bottleneck and single point of failure. In contrast, in this paper, we are particularly interested in fully distributed solutions to the load balancing problem. the tree overlay facilitates the reallocation of virtual servers.

IV. CONCLUSION

In this paper, we have presented a novel load balancing algorithm for DHTs with virtual severs. Our proposal is unique in that we represent the system state with probability distributions. With the approximated probability distributions, each peer identifies whether it is under loaded and then reallocates its loads if it is overloaded. Our proposal is driven by rigorous performance analysis and validated by extensive simulations. The simulation results reveal that

that our proposal performs well and that it is comparable with the centralized directory approach and outperforms 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

V. REFERENCES

[1] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," Performance Evaluation, vol. 63, no. 6, pp. 217-240, Mar. 2009.

[2] 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. 2007.

[3] Y. Zhu, "Load Balancing in Structured P2P Networks," Handbook of Peer-to-Peer Networking, Springer, July 2006.

[4] 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.

[5] 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.

[5] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., 1979.

[7] Hung-Chang Hsiao, Hao Liao, Ssu-Ta Chen, and Kuo-Chan Huang "Load Balance with Imperfect Information in Structured Peer-to-Peer Systems" IEEE transactions on parallel and distributed systems

BIOGRAPHY

Author Details: N. HARISH, a student of M.Tech, Vaagdevi Institute of Technology & Science. Proddatur, A.P.

Email: harish.pdtr@gmail.com

Guide Details: C. RAMA MOHAN, Associate Professor of Vaagdevi Institute of Technology & Science, Proddatur, A. P. INDIA. Email: ramamohanchinnem@gmail.com

Guide Details: U. SHESADRI, Associate Professor of Vaagdevi Institute of Technology & Science, Proddatur, A. P. INDIA