Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Structured Peer-to-Peer Systems using Load Balancing with Imperfect Information

[1]K.Soujanya. [2]T.Swapna, [3]Dr.P.Raja Prakash Rao
[1] M.TECH STUDENT OF TRR ENGINEERING COLLEGE, HYDERABAD, INDIA
[2] ASSISTANT PROFESSOR OF TRR ENGINEERING COLLEGE, HYDERABAD, INDIA
[3] PROFESSOR OF TRR ENGINEERING COLLEGE, HYDERABAD, INDIA

Abstract: ***with 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. The existing and decentralized load balance algorithms designed for the heterogeneous, structured P2P networks either explicitly construct auxiliary networks to manipulate global information or implicitly demand the P2P substrates organized in a hierarchical fashion. Without relying on any auxiliary networks and independent of the geometry of the P2P substrates, we present, a novel load balancing algorithm that is unique in that each participating peer is based on the partial knowledge of the system to estimate the probability distributions of the capacities of peers and the loads of virtual servers, resulting in imperfect knowledge of the system state. With the imperfect system state, peers can compute their expected loads and reallocate their loads in parallel. Through extensive simulations, we compare our proposal to prior load balancing algorithms.***

Indexingterms:P2P,LoadBalancing,imperfectsystem,DHTs

I. INTRODUCTION

PEER-TO-PEER (P2P) networking is an emerging 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 design and implementation of successful P2P applications. Potential P2P substrates are based on distributed hash tables or DHTs for short. Examples of DHTs are Chord [1] and Pastry [2]. 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 [3], [4], [5], suggest that each peer i Є N deal with the load balancing problem by minimizing the following:

summetion of v€Vi Li=A---------- (1)

Where Vi<v represents the set of virtual servers allocated to peer i; L_v 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|

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

implies that peer i manages the loads of virtual servers proportional to its capacity, leading to a load balanced DHT network.

First, 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, L_v of any virtual server v at a particular time is the sum 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, C_i 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. However, similar to prior studies, (e.g., [3], [4], [5]), we assume that there is only one bottleneck resource in the system, leaving multiple-resource load balancing in the future.

For balancing loads among participating peers, we are also concerned with the minimization of the system wide performance metric—movement cost—as much as possible [3], [4], [5]. 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:

---------- (2) Accordingly, by the minimization of (1) and (2), we intend to distribute loads to participating peers evenly with the minimum movement cost.

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

Here, we are interested in solving the load balancing problem in a fully decentralized manner. While pioneer studies presenting load balancing algorithms for distributed systems can be found in the literature (e.g., [6], [7]), these studies either propose centralized load balancing algorithms [6] or often aim at static, small-scale systems and/or homogeneous environments [7]. As the centralized algorithms may introduce the performance bottleneck and the single point of failure, recent proposals presenting centralized algorithms that rely on a few rendezvous nodes to balance the loads of peers in a DHT can be found in [3], [8].

To the best of our knowledge, the existing load balancing algorithms that operate in a distributed manner and aim at dynamic and heterogeneous DHTs include the studies [4], [5]. In [4], [5], each node, k, typically specifies a target load, T_k, and $T_{k \leq} C_{k.}$ Without exceeding T_k, k can only accept virtual servers, such that the total load of the virtual servers in k is no more than T_k. Obviously, the ideal setting for T_k is A x C_k, that is, k is expected to manage the load proportional to its capacity. While Shen and Xu [5] explicitly configure T_k for each peer k independent of the system state, the proposal by Zhu and Hu [4] acquires the global knowledge of the entire system to compute the expected load per unit peer capacity, A, spontaneously and to disseminate the value of A to all participants, allowing each peer k to compute its $T_{k=}$ A x C_k.

In this paper, we present a novel load balancing algorithm to minimize both (1) and (2) as much as possible. The unique feature of our proposal is that each participating peer estimates and represents the "system state" as the probability distributions for the capacities of nodes and the loads of virtual servers. The approximated probability distributions not only help estimate the expected load a peer should perceive but also provide hints for each peer in the system to schedule the transfers of virtual servers. [3], [8], unlike the participating peers in our proposal operate independently, and they need not rely on 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 [5], our proposal is independent of the geometry of DHTs.

II RELATED WORK

Earlier studies (e.g., [6], [7]) 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. As the previous study [9] has provided a survey on the load sharing algorithms in traditional high-performance

computing systems, we refer interested readers to [10] for an interesting survey on the load balancing algorithms in DHTs—the emerging Internet based autonomous network infrastructures.

Consider an object set O and a peer set N. Conventional DHTs (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 l_o =1, earlier studies [1], [11] show that the load imbalance factor in a typical DHT can be up to $O(\log n)$, where n =|N| is the total number of nodes participating in the system. To be more precise, the maximum load of a peer can be $O(\log n)$, times the average. Later, several proposals (e.g., [11]) have reduced the load imbalance factor to a constant. In contrast to [11], the proposals (e.g., [3]) exploit the heterogeneity of peers, such that the number of objects allocated to a peer is proportional to the peer's capacity. Unlike [1], [3], [11] our work presented in this paper does not assume that objects in the system contribute the identical load to peers. That is, our proposal does not aim to balance loads among peers in terms of numbers of objects allocated to peers.

Instead of simply assuming l_o =1for all o Є O and evenly partitioning the key space into each peer, Chord [1] suggests the notion of virtual servers. Different virtual servers in a system manage disjoint key subspaces, with a 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 [12] to cope with the load imbalance in a DHT. In the many-to-many framework, light and heavy nodes register their loads with some dedicated nodes, namely, the directories. The directories compute matches between heavy and light nodes and then, respectively, request the heavy and light nodes to transfer and to receive designated virtual servers. As noted by the authors in [3], 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 study most relevant to ours is perhaps the work by Zhu and Hu [4]. Zhu and Hu [4] organize virtual servers in a DHT network into an auxiliary tree-shaped overlay network on top of the DHT. Such tree-shaped overlay is used to compute and disseminate the global aggregate, namely, the average allowing each peer k in the system to be based on A to compute exactly its target load threshold T_k and then to identify whether it is heavy or light. Additionally, the tree overlay facilitates the reallocation of virtual servers. The reallocation is performed in a bottom-up fashion toward the root of the tree. As we discussed above, while [4] simulation strives to distribute loads evenly due to the virtual servers, the nodes in the tree experience skew the workload for coordinating the reallocation of virtual servers, thus introducing another load imbalance issue.In contrast to [4], our solution attacking the load balancing problem need not rely on any auxiliary tree networks, which are clearly failure-prone and thus require sophisticated maintenance. In addition, while the solution requires aggregates global information to compute A and then to reassign virtual servers, each peer in our proposal is independently based on partial knowledge of the system to reallocate its virtual servers. Specifically, the nodes in our proposal need not coordinate the reallocation of their virtual servers, leading to a design more appropriate for large-scale, dynamic environments.

Shen and Xu [5] also present a fully decentralized method for migrating objects (i.e., data items) stored in a DHT. Akin to the solution in [4], Shen and Xu suggest organizing a DHT into a two-level hierarchical network, where the higher level of the network consists of local clusters. Objects with excess loads are moved in a local cluster to balance the loads of the peers located in the same cluster. For those objects that cannot be moved in a local cluster, they can be transferred to peers in some foreign clusters. Unlike [5], our proposal does not assume any specific geometry of DHTs and is thus applicable to any DHT network.

Algorithm Sketch

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 T_i, that it should perceive, where $T_i = \tilde{\mathcal{A}} \times C_i + \epsilon$, $\tilde{\mathcal{A}}$ is an estimation for the expected load per unit capacity, i.e.

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

$\tilde{\mathcal{A}} \leftarrow \ln n \cdot \frac{\int_{y=0}^{y_{max}} (yF_Y(y)dy)}{\int_{x=0}^{C_{max}} (xF_X(x)dx)}$;
$T_i \leftarrow \tilde{\mathcal{A}} \times C_i + \epsilon$;
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]

and Є is a predefined system parameter. If the current total load of i is greater than T_i (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 load each time until the hosting peer becomes under-loaded.

As we can see in Algorithm 1, the challenges of implementing the algorithm are 1) how a peer precisely and timely estimates A and 2) how an overloaded peer seeks the peers to receive its migrated virtual servers for balancing the loads among peers. To deal with these issues, our idea is to represent the capacities of participating peers and the loads of virtual servers as the probability distributions, which are denoted by Pr(X<x) and Pr(Y<y), respectively. Both Pr(X<x) and Pr(Y<y)

provide valuable information to help the participating peers estimate A, and the overloaded peers discover the under-loaded peers to share their excess loads[15].

III .SUMMARY

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. Unlike prior solutions[15] that often rely on global knowledge of the system, each peer in our proposal independently estimates the probability distributions for the capacities of participating peers and the loads of virtual servers based on partial knowledge of the system. 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.[15]

REFERENCES

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

[2] A. Rowstron 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.

[3] 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. 2006.

[4] 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.

[5] 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.

[6] L.M. Ni and K. Hwang, "Optimal Load Balancing in a Multiple Processor System with Many Job Classes," IEEE Trans. Software Eng., vol. 11, no. 5, pp. 491-496, May 1985.

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

[8] C. Chen and K.-C. Tsai, "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.

[9] 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.

[10] Y. Zhu, "Load Balancing in Structured P2P Networks," Handbook of Peer-to-Peer Networking, Springer, July 2009.

[11] 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.

[12] 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.

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

[14] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area Cooperative Storage with CFS," Proc. 18th ACM Symp. Operating Systems Principles (SOSP '01), pp. 202-215, Oct. 2001.

[15] 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, OL. 22, NO. 4, APRIL 2011

