Load Balancing Heterogeneous Request in DHT-based P2P Systems

Mrs. Yogita A. Dalvi
M.TECH.[C.S.E.] (Pursuing), IES, Bhopal
Email: syogita_dalvi@rediffmail.com

Dr. R. Shankar
IES, Bhopal

Mr. Atesh Kumar
M.TECH.[CTA] , IES, Bhopal.
Email: ateshsingh@yahoo.com

Abstract

DHT-based P2P systems have been proven to be a scalable and efficient means of sharing information. With the entrance of quality of services concerns into DHT systems, however, the ability to guarantee that the system will not be overwhelmed due to load imbalance becomes much more significant, especially when factors such as item popularity and skewing are taken into consideration. In this paper, we focus on the problem of load imbalance caused by skewed access distribution. We propose an effective load balancing solution, which takes the peer heterogeneity and access popularity into account to determine the load distribution. Our algorithm achieves load balancing by dynamically balancing the query routing load and query answering load respectively. Experimentations performed over a Pastry-like system illustrate that our balancing algorithms effectively balance the system load and significantly improves performance.

Keywords
Load-balancing, distributed hash tables (DHT), peer-to-peer (P2P) systems, and quality of service (QoS)

1. INTRODUCTION

Distributed Hash Tables (DHTs) provide a reliable, scalable, fault tolerant and efficient way to manage P2P networks. Typically employing some variant of consistent hashing to associate keys with nodes, each node is mapped to a unique ID and owns the set of objects whose IDs are "closest" to it, while object lookup queries consist of following well-defined paths from a querying node to a destination node that holds the index entries pertaining to the query. In theory, DHTs can provide fair and scalable service because they achieve a balanced partition of workload amount the nodes in the system. Given the heterogeneous nature of individual peer capacity in a P2P network (due to non-uniform computational power, storage capacity and network bandwidth difference between peers), even a uniform workload distribution amongst peers can still lead to load imbalance problems, additionally encumbered by the fact that the consistent hash used by DHTs can cause certain peers to have up to O(logN) times as many objects as the average peer in the network , intensifying the imbalance. Furthermore, since

objects and queries within the system tend to be skewed (i.e. certain objects are significantly more popular than others), heavy lookup traffic load is experienced at the peers responsible for popular objects, as well as at the intermediary nodes on the lookup paths to those peers. When subsequent tasks are then obliviously assigned to the already overloaded node, the average response time consequently increases drastically. This paper aims at balancing the highly unbalanced load caused by skewed object and query distribution through the use of a comprehensive balancing mechanism, which includes an adaptive load redistribution scheme as well as a dynamic routing table reconfiguring scheme.

2. RELATED WORK

There have been many load balancing schemes proposed for DHT-based systems. We divide them into four categories:

Virtual server:- The virtual server approach focuses on the imbalance of the key distribution due to the hash function. Each physical node instantiates O(logN) number of virtual servers with random IDs that act as peers in the DHT, which reduces the load imbalance to a constant factor. To address peer heterogeneity, each node selects a number of virtual servers to create proportional to its capacity. Unfortunately, the usage of virtual servers greatly increases the amount of routing metadata needed on each peer and causes more maintenance overhead. In addition, the number of hops per lookup (and latencies) increases. Moreover, it doesn't take object popularity into account.

Dynamic ID: - The dynamic ID approach uses just a single ID per node .The load of a peer can be adjusted by choosing a suitable ID in the namespace. However, all such solutions requires IDs to be reassigned to maintain load balance as nodes dynamically join and leave the system, resulting in a high overhead because it involves transferring objects and updating overlay links.

Multiple hash functions:- This approach uses multiple hash functions to balance the load. The power of two choices uses two or more hash functions to map a key to multiple nodes and store the key on the peer that is the least loaded. In the *k-choice* load balancing algorithm, the node uses multiple hashes to generate a set of IDs and at join time selects an ID in a way to minimize the discrepancies between capacity and load for itself and the nodes that will be affected by its join time. While such a strategy is simple and efficient, it increases the computational overhead for publishing and retrieving content, since multiple hash functions have to be computed each time; in addition, it is a static allocation, and does not change in the case that the workload distribution shifts.

Caching and replication:- Hotspots and dynamic streams are handled by using caches to store popular objects in the network, and lookups are considered resolved whenever cache hits occur along the path. Pastry and Chord replicate an object on the *k* servers whose identifiers are closest to the object key in the namespace to improve the availability, but it also help balance the load of a popular topic. Unfortunately, the last few hops of a lookup are precisely the ones that can least be optimized. Moreover, since the query load is dynamic, a fixed number of replicas do not work well; if the number is chosen too high, then resources may be wasted, and if it is set too low, then these replicas may not be enough to support a high query load.

3. ADAPTIVE LOAD BALANCING SCHEME

We propose a comprehensive load balancing strategy, which address this problem by dynamically re-distributing the load of hot spots to other 'cold spots'. Particularly, we distinguish two types of load: query answering load and query forwarding load (query load and routing load for short). Aiming at balancing these two kinds of load, three balancing strategies have been proposed:

(1) Adaptive object replication scheme, which targets balancing the query load; and

(2) Adaptive routing replication and

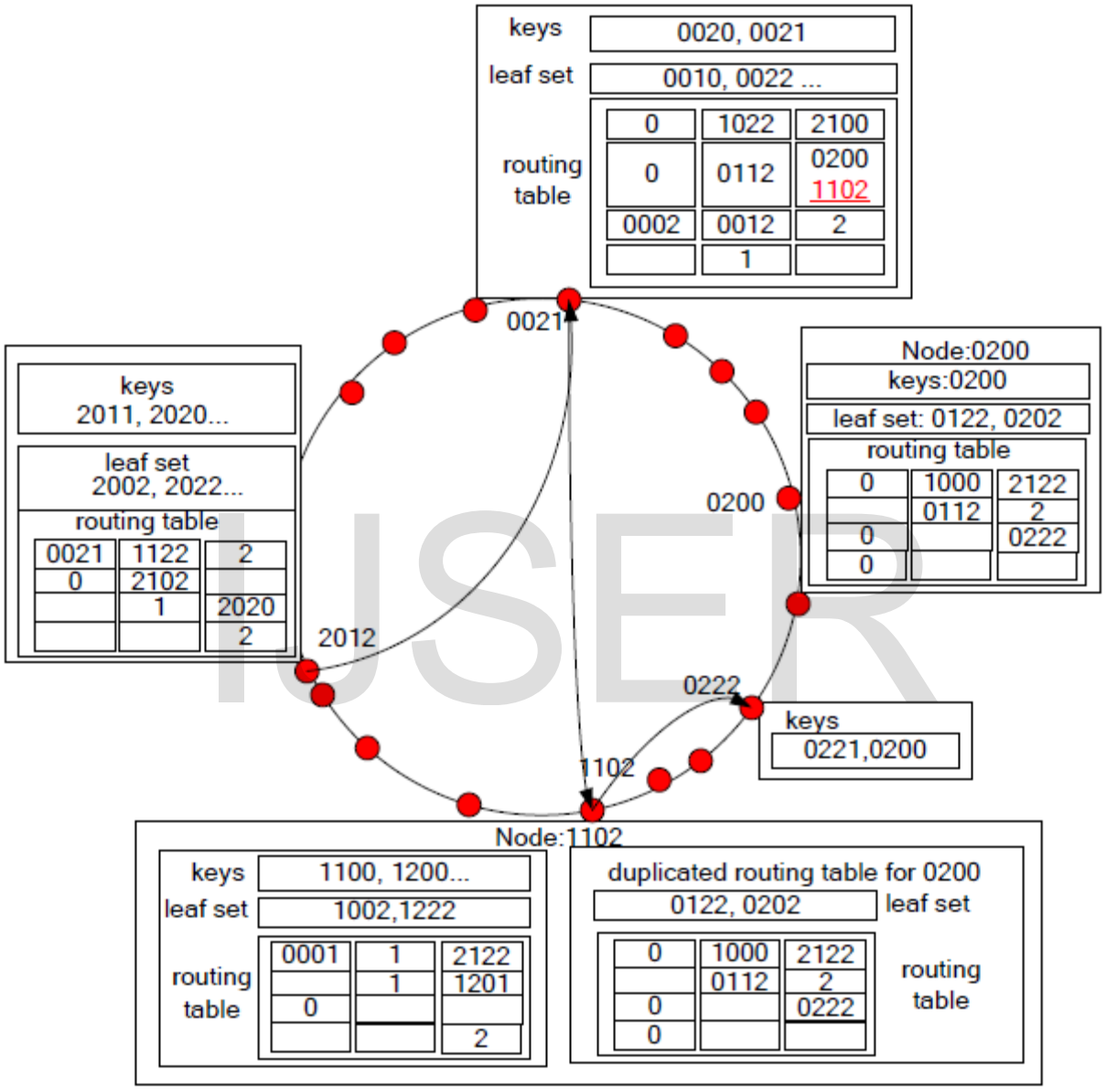
(3) Dynamic routing table reconfiguration, both aimed at balancing the system's routing load. Each node analyzes the main cause of its overloading and uses a particular balancing algorithm to correct its situation.

1. Adaptive Routing Replication Algorithm

Replicating popular keys relieves the query answering load of the nodes responsible for the keys. However, another major source of workload in DHT overlays is caused by relaying queries among nodes. A node may be overwhelmed simply by the traffic of forwarding incoming routing queries. For example, the last hop neighbors of a popular key can be overloaded by forwarding queries to the popular node. While this problem can be partially solved by the aforementioned duplication of popular keys to disperse the traffic, it cannot completely alleviate the problem since certain nodes in the system might still be functioning effectively as traffic hubs for popular sections of the network. To address this problem, we propose a balancing scheme which actively redistributes the routing load of an overloaded node by duplicating its routing table to other nodes, thereby sharing its routing load. When a node is overloaded by routing loads, it will pick a light loaded node to replicate its routing table, so that the replica node can share its routing load. As with the object replication algorithm, the routing replica information should be propagated to other related nodes. These nodes subsequently update their respective routing tables by adding a replica entry to the entry of the original node so that future queries can be routed to either the original node or the new node, all the while maintaining system network correctness. Besides load balancing, replication approach can also improve the routing resiliency in the face of network failures.

Figure 1 shows an example of the Pastry structure with the replication of routing tables. The query for item ID 0221, which is actually served by node 0222, is initiated at node 2012. According to its routing table, node 2012 chooses 0021 as the next hop. Node 0021 determines that node 0200 should be the right node to forward the query. Since node 0200 has a replica at node 1102, node 0021 may choose 1102 as the next hop. When the query is sent to 1102, it uses the duplicated routing table for 0200 to serve the query and send the query to the destination node 0222. When node 0200 is exposed to a high load, the replicas will share some of the traffic, preventing overload.

Figure 1. An example of adaptive routing replication algorithm



2. Dynamic routing load adjusting algorithm

In addition to the use of replication, another scheme to balance the routing load is by dynamically reconfiguring the routing table. In the previously mentioned methods, an overloaded node actively redistributes its own load, but in cases where external policies or the network environment prevents the redistribution effort, replacing routing table content can help relieve highly loaded nodes.

This algorithm is tailored specifically for DHTs like Pastry or Tapestry. In those systems, many nodes with the same prefix can be potentially filled in a node's routing table; the one in the table is the one the node knows, and with topological metric considered, it will 'pick' the one closest to itself. We propose changing the strategy of choosing nodes in the routing table to balance routing load (especially to relieve heavily loaded nodes). In lieu of simply choosing according to a proximity metric, we choose with lower routing load instead. Whenever an overloaded

node receives a querying message from its neighbour, it will reply with a message indicating its overloaded status. This neighbour, receiving the message, will, at the earliest opportunity possible, replace the entry of the overloaded node in its routing table with another node of the similar prefix. The light-loaded candidate nodes are learned from forwarded query messages which include IDs of passed nodes. By doing so, traffic is alleviated from the overloaded load as long as it is not the actual 'end target' of the query request, as the replacement node will be able to direct any queries the original node could've, and forwarding traffic is spread out more evenly.

Continuing from our example in Figure 1, in node 1102's routing table, let us assume that a neighbor node, 2122, (1^{st} row 3^{rd} column) is heavily-loaded. When a query passes through node 2012 to 0021 and then comes to node 1102, since 2012 shares the identical first digital prefix (2) as the overloaded neighbor 2122 in 1102's routing table, the entry of 2122 will be replaced with 2012. This way, the traffic to the more heavily loaded 2122 will be redirected to the more free 2012.

4. CONCLUSION

We have presented an effective approach to balance load in DHT systems. Our work distinguishes routing load and retrieval load, and deals them separately. By dynamically replicating different potions of the overloaded node based on source of the overloading (replicating either its routing table or its keys), we overcome the restrictive nature of traditional balancing schemes that assumes homogeneity among peers and the type of load they incur. This approach enables the system good load balance even when demand is heavily skewed. Extensive simulation results indicate significant improvements in maintaining a more balanced system, leading to improved scalability and performance.

5. REFERENCES

[1] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in dynamic structured P2P systems," in *Proceedings of the 23rd Conference of the IEEE Communications Society (Infocom 2004)*, 2004.

[2] D. Karger and M. Ruhl, "Simple efficient load balancing algorithms for peer-to-peer systems," in *Proceedings of 16th ACM Symposium on Parallelism in Algorithms and Architectures*, 2004, pp. 36–43.

[3] A. R. Karthik, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in structured P2P systems," in *Proceedings of 2ndInternational Workshop on Peer-to-Peer Systems, 2003*, pp. 68–79.

[4] G. Manku. "Balanced binary trees for ID management and load balance in distributed hash tables." In *Proceedings of Twenty-Third Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2004)*, 2004.

[5] J. Byers, J. Considine, and M. Mitzenmacher, "Simple load balancing for distributed hash tables," in *Proceedings of 2nd International Workshop on Peer-to-Peer Systems, 2003*, pp. 80–87.

[6] J. Ledlie and M. Seltzer, "Distributed, secure load balancing with skew, heterogeneity, and churn," in *Proceedings of the 24rd Conference of the IEEE Communications Society, (Infocom 2005)*, 2005.