Efficient, Proximity-Aware Load Balancing for Structured P2P Systems

Yingwu Zhu
Department of ECECS
University of Cincinnati
Cincinnati, OH 45221, USA
zhuy@ececs.uc.edu

Yiming Hu
Department of ECECS
University of Cincinnati
Cincinnati, OH 45221, USA
yhu@ececs.uc.edu

Abstract

Structured peer-to-peer (P2P) systems address the load balancing issue in a rather naive way, by simply resorting to the uniformity of the hash function utilized to generate object IDs. Such a random choice of object IDs could result in $O(\log N)$ load imbalance. In this position paper, we propose an efficient, proximity-aware load balancing algorithm for such systems. To our knowledge, this is the first work to utilize the proximity information to guide load balancing. In particular, our main contributions are: (1) A self-organized, fully distributed K-nary tree structure is constructed on top of a DHT for load balancing information collection/dissemination and load reassignment. (2) Load balancing is achieved by aligning those two skews in both load distribution and node capacity inhere in P2P systems — that is, have higher capacity nodes carry more loads. (3) Proximity information is utilized to guide load balancing such that virtual servers are assigned and transferred between physically close heavy nodes and light nodes, thereby minimizing the load transferring overhead and making load balancing fast and efficient.

1. Introduction

Structured P2P systems (e.g., [2, 1]) offer a distributed hash table (DHT) abstraction for object storage and retrieval. By providing such a simple and homogeneous abstraction, while theoretically elegant, these DHTs have two main limitations. First, simply resorting to the uniformity of the hash function used to generate object IDs in DHTs does not produce perfect load balance. It could result in an $O(\log N)$ load imbalance. Second, they build a homogeneous structure overlay network, ignoring the heterogeneity nature of P2P systems. Recent measurement studies have shown that node capabilities (in terms of bandwidth, storage and CPU) are very skewed in deployed P2P systems.

The primary goal of P2P systems is to harness all available resources (e.g., storage, bandwidth and CPU) in the P2P network so that users can access all available objects *efficiently*. From the P2P system perspective, "efficiently" is interpreted that it strives to ensure fair load distribution among all peer nodes. We therefore argue that achieving load balancing is of fundamental importance in a P2P system, due to the assumption that nodes are supposed to be uniform in resources, the resulting $O(\log N)$ load imbalance by a random choice of object IDs, and the fact that heterogeneous capabilities prevail among the nodes.

Many solutions have been proposed to tackle the problem of load balancing in structured P2P systems. However, all these solutions either ignore the heterogeneity nature of the system, or reassign loads among nodes without considering proximity relationships, or both.

Therefore, we propose a proximity-aware load balancing approach by using the concept of virtual servers. The goal of our approach is to not only ensure fair load distribution over nodes proportional to their capacity, but also minimize the load-balancing cost by transferring virtual servers between heavy nodes and light nodes in a proximity-aware fashion. That is, heavy nodes are trying to assign their virtual servers to physically close light nodes so that we might end up with an efficient and fast load balancing.

2. Proximity-Aware Load Balancing

The proximity-aware load balancing is based on the concept of virtual servers. A virtual server looks like a single DHT node, responsible for a contiguous region of the DHT's identifier space. A physical DHT node can own multiple non-contiguous regions of the DHT's identifier space by hosting multiple virtual servers. The key advantage of using virtual servers for load balancing is that it has flexibility in being able to move loads between DHT nodes in the unit of virtual servers, and the movement of virtual servers can be visioned as a *join* operation followed by a *leave* operation, both of which are supported by all DHTs.

Our load balancing approach consists of four phases:

1. *Load balancing information (LBI) aggregation.* Aggregate load and capacity information in the whole system.
2. *Node classification.* Classify nodes into overloaded (heavy) nodes, underloaded (light) nodes, or neutral nodes according to their loads and capacity.
3. *Virtual server assignment (VSA).* Determine virtual server assignment from heavy nodes to light nodes in order to have heavy nodes become light.
4. *Virtual server transferring (VST).* Transfer assigned virtual servers from heavy nodes to light nodes.

The basic idea behind our proximity-aware load balancing is to utilize proximity information (represented by *Hilbert numbers* [3] which are derived from *landmark vectors* using the Hilbert curve, e.g., two physically close DHT nodes are supposed to have close Hilbert numbers) to guide VSA such that the virtual servers are assigned and transferred between physically close heavy nodes and light nodes. In other words, the proximity-aware load balancing is "greedy" in the sense that it tries at each step to reduce the load transferring cost by making appropriate VSA among physically close nodes.

3. Experimental Evaluation

We evaluated our experiments on a Chord overlay consisting of 4096 nodes each with 5 virtual servers in the beginning. The load of a virtual server was simulated by two distributions: *Gaussian distribution* and *Pareto distribution*. We used a *Gnutella-like* capacity file to account for the heterogeneity of node capacity [3]. In addition, we used two Internet topologies with approximately 5,000 nodes each: "ts5k-large" and "ts5k-small". For more detail of experimental setup, please refer to [3].

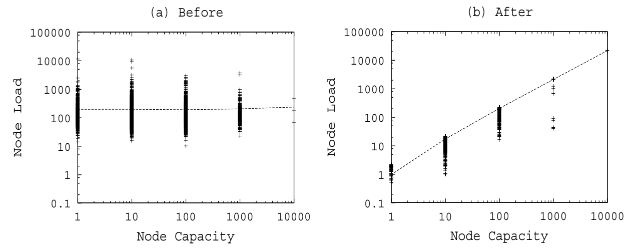

Figure 1. Results for the Pareto distribution

Figure 1 shows the scatterplot of loads according to node capacity for Pareto distribution. Note that our load balancing scheme is able to align those two skews in load distribution and node capacity, by having higher capacity nodes take more loads.

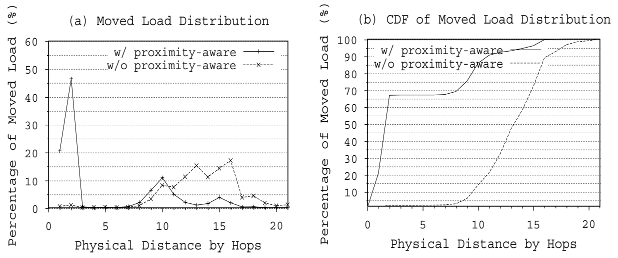

Figure 2. Results for "ts5k-large".

Figure 2 shows the results of load balancing scheme with/without the proximity-aware mechanism for "ts5k-large". We note that, compared to the proximity-ignorant load balancing scheme, the proximity-aware load balancing scheme is able to transfer most loads (about 67%) between physcially close nodes, thereby reducing the load balancing cost and making load balancing fast and efficient. More results are presented in [3].

4. Summary

In this position paper we propose an efficient, proximity-aware load balancing scheme to tackle the issue of load balancing in structured P2P systems. The first goal of our load balancing scheme is to align those two skews in both load distribution and node capacity inherent in P2P systems to ensure fair load distribution among nodes — that is, have higher capacity nodes carry more loads. The second goal is to use proximity-aware information to guide load assignment and transferring, thereby minimizing the cost of load balancing and making load balancing fast and efficient. The experimental results show that our proximity-aware load balancing scheme can not only ensure fair load distribution but also effectively reduce the load transferring overhead.

References

[1] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proceedings of the 18th IFIP/ACM International Conference on Distributed System Platforms (Middleware 2001)*, Heidelberg, Germany, Nov. 2001.
[2] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, pages 149–160, San Diego, CA, Aug. 2001.
[3] Y. Zhu and Y. Hu. Efficient, proximity-aware load balancing for structured peer-to-peer systems. Technical Report TR-270/04/03/ECECS, University of Cincinnati.