Proactive Caching for Better than Single-Hop Lookup Performance

Venugopalan Ramasubramanian and Emin Gün Sirer
Cornell University, Ithaca NY 14853
{ramasv, egs}@cs.cornell.edu

Abstract

High lookup latencies prohibit peer-to-peer overlays from being used in many performance intensive applications, even though they provide self-organization, scalability, and failure resilience. In this paper, we show that lookup performance of structured DHTs can be improved to any desired constant, even under a single hop, by controlled proactive replication. By exploiting the popularity distribution of objects, we can minimize the number of replicas and reduce the storage and bandwidth cost of replication. This enables structured DHTs to efficiently support a wide variety of latency sensitive applications. We describe three different applications, namely DNS, web access, and content distribution, and show how they can derive significant performance gains by using DHTs.

1 Introduction

Structured overlays provide a framework for storing and retrieving objects with guaranteed lookup performance in both the worst and the average case, at the same time providing self-organization, scalability, and high resilience to failures. These properties make them ideal building blocks for a number of large scale distributed applications including archival file storage, cooperative web caching, and application level multicast. However, recent studies have shown that the lookup performance provided by most structured distributed hash tables (DHTs) is inadequate to support latency sensitive applications such as the domain name service (DNS) and web content distribution [3]. Table 1 shows a comparison of the mean and median latencies for a DNS request with the mean and median round trip times between two hosts in Planetlab. The average latency of a DNS request is comparable to the average round trip time in the Internet. Therefore, distributed overlays need to provide lookup latency of one hop or lower in order to support performance intensive applications such as DNS.

One distributed approach for improving lookup latency is *passive caching*. Passive caching refers to an opportunistic scheme that stores a replica of the object at each intermediate node traversed by the request. While caching is known to be very effective in many systems applications, we show that the improvement in lookup performance provided by caching in DHTs is small and insufficient to support DNS and web content distribution. Our studies identify two reasons for the limited effectiveness of caching for these applications. First, DNS and web requests follow Zipf-like or power-law popularity distributions that are characterized by heavy-tails [1, 9]. In such distributions, objects of very low popularity cumulatively get a predominant number of requests. Consequently, the benefit of passively caching the most popular objects does not compensate for the high cost of finding the less popular objects. Second, the contents of DNS resource records and web pages are not static but changing. In order to support mutable objects, caching schemes generally associate a lifetime with the stored objects and remove the object from the cache upon expiry of this lifetime. Guaranteeing the consistency of the objects upon updates imposes a conservative choice for the lifetime. For example, 95% of the DNS records have a lifetime of less than 1 day, whereas less than 0.8% of the records change in 24 hours. Consequently, caching systems have to frequently refetch the objects, significantly reducing the improvement in lookup performance as well as substantially increasing the load on the Internet.

	median	mean
DNS latency	112 ms	256 ms
Internet RTT	81.9 ms	202 ms

Table 1: **This table compares the latency of DNS requests with the round trip time between two hosts in Planetlab. It shows that DHTs need to provide better than single-hop lookup performance to compete with latency sensitive applications.**

In this paper, we propose the use of *proactive replication* and show that controlled replication of objects can provide very low, even less than a single hop, lookup performance in structured overlays with minimal overhead. By proactive replication, we mean that copies of the object are replicated on some nodes that requests are likely to traverse even before the requests are routed through those nodes. This enables the proactive replication scheme to achieve considerable improvement in the lookup performance even for heavy tailed distributions. By controlled replication, we mean that replicas are distributed in the network strategically, exploiting the underlying structure of the DHT, in order to achieve a target lookup performance. In fact, we show that by suitably varying the extent of replication based on the popularity of the objects, we can tune the amortized lookup performance of the system to any desirable constant. Of course, any improvement in lookup performance must incur some cost. Our replication scheme is based on an analytical model that minimizes the number of replicas required to achieve the desired performance. Minimizing the number of replicas enables us to optimize the per node storage consumption, reduce the total bandwidth required to replicate and maintain the objects, and impose a low network load at each node. Controlled placement of the replicas also facilitates an efficient mechanism to handle object mutability. Since the locations of the replicas can be easily tracked, we can proactively disseminate updates to objects, supporting coherent updates and obviating conservative timeouts.

Popularity based replication helps structured overlays to achieve high performance at low cost in addition to improved failure resilience and availability. These properties enable them to support highly demanding latency sensitive applications. In this paper, we survey three widely different applications that are particularly well suited for overlay networks. The DNS, web access, and multimedia content distribution are all performance intensive applications that suffer from similar problems. First, they are all slow; accessing information currently takes much longer time in these systems than what users desire. Second, they are highly unreliable; sudden increases in load due to flash-crowd effect or denial of service attacks easily bring down these systems. We show that implementing these applications on distributed overlays using proactive caching can substantially improve the lookup performance, while enabling them to effectively manage rapid changes in popularity of objects.

A few existing overlays also provide constant lookup performance by utilizing various techniques such as d-dimensional hypercube [4], gossip based fixed replication [7] and fully replicated routing tables [8]. We take an orthogonal approach and propose a general replication scheme that can be applied to improve the performance of many existing structured DHTs, including [13] and [12]. Our scheme allows the system to tune its amortized lookup performance to any desired constant, even fractions under one hop, instead of binding the lookup performance to the design of the system. Finally, our scheme exploits the popularity distribution of the requests to provide high performance with minimal overhead.

2 Improving on Single-Hop Lookups

The regular underlying structure of a DHT enables us to analyze the impact of different replica placement strategies on lookup performance. Our controlled proactive replication scheme exploits the structure of the underlying DHT to tune the lookup latency, and utilizes the popularity distribution of the objects to achieve the desired target performance with low overhead. We will illustrate our replication scheme by considering Pastry [12] as the underlying DHT. The general replication scheme is applicable to all structured DHTs with a uniform fanout.

In Pastry, both objects and nodes have randomly assigned identifiers from the same circular space, and each object is stored at the nearest node in the identifier space, called the *home node*. Each Pastry node routes a request for an object, say 0121, by successively matching prefixes; that is, by routing the request to a node that matches one more digit with the object until the home node, say 0122, is reached. This process takes O(logN) hops to reach the home node. By placing copies of the object at all nodes one hop prior to the home node in the request path, the lookup latency can be reduced by one hop. In the above example, the lookup latency can be reduced from 3 hops to 2 hops by replicating the object at all nodes that start with 01. Similarly, the lookup latency can be reduced to 1 hop by replicating the object at all nodes that start with 0. Thus, we can vary the lookup latency of the object between 0 and logN hops by systematically replicating the object to different levels. We say an object is replicated at *level* i, if it is replicated on nodes with i matching prefixes.

The central insight behind our scheme is that by judiciously choosing different levels of replication for different objects, the amortized lookup performance of the system can be tuned to any desired constant. Naturally, an efficient approach to determine the appropriate replication level for each object must incorporate the popularity distribution of the objects, that is, objects with greater popularity should be replicated to a greater extent than objects with lesser popularity. We can obtain the most efficient replication strategy by solving the following optimization problem: minimize the total number of replicas subject to the constraint that the aggregate lookup latency is less than a desired constant C.

We earlier noted that several interesting applications including DNS and the web are characterized by power-law or Zipf-like popularity distributions [1, 9]. For Zipf-like query distributions, we can solve the above optimization problem analytically and obtain the closed form optimal solution [11]. The following expression gives the optimal replication level for each object for Zipf-like distributions with parameter $\alpha \leq 1$. Here x_i is the fraction of most popular objects to be replicated at level i or lower, and b is the fanout or base of the DHT.

$$x_i = [\frac{d^i(logN - C)}{1 + d + \cdots + d^{logN-1}}]^{\frac{1}{1-\alpha}}, \text{where } d = b^{\frac{1-\alpha}{\alpha}}$$

Our analysis clearly captures the trade off between lookup latency and cost of replication by using the number of replicas as an indication of the storage and bandwidth overhead. Zipf-like distributions exhibit an exponential trade off between performance and overhead; that is, the optimal number of replicas grows exponentially as the target latency decreases. The analytical model facilitates the overlay system to optimally distribute the replicas of the objects and reduce the storage and bandwidth consumption, while also enabling the deployer to understand the overhead required to meet the desired lookup performance.

Experimental Results

Beehive [11] implements this model-driven proactive caching scheme to provide constant lookup performance in Pastry [12]. While the design and implementation of Beehive is beyond the scope of this paper, we merely provide experimental results from the Beehive system to demonstrate that better than single-hop lookup performance is efficiently achievable in practice.

We ran Beehive in the simulation mode for a network of 1024 nodes for Pastry with base$= 16$. We studied the performance of this system by issuing requests from traces collected for three different performance intensive applications, namely DNS, Web, and Gnutella to 40960 distinct objects. We issued queries from MIT DNS traces, UC Berkeley Home IP Web traces, and CMU Gnutella traces. The DNS and Web traces are characterized by Zipf distributions of $\alpha = 0.91$ and 0.78, while the Gnutella trace does not exhibit Zipf-like behavior. We compared the lookup performance of Beehive with that of pure Pastry, as well as, PC-Pastry, which performs passive caching using an unlimited cache for the DNS application.

Figure 1 shows the average number of hops taken by requests in Pastry, Beehive, and PC-Pastry for different applications as the systems evolve with time. Pastry incurs an average lookup cost of about 2.34 hops independent of the popularity distribution. The lookup performance of Beehive quickly converges to reach the targeted constant target lookup performance of 0.5 hops for all applications. In contrast, PC-Pastry provides about 1.54 hops for DNS, a limited improvement over the lookup performance of Pastry. Beehive is also significantly more efficient than passive caching. PC-Pastry caches 420 objects

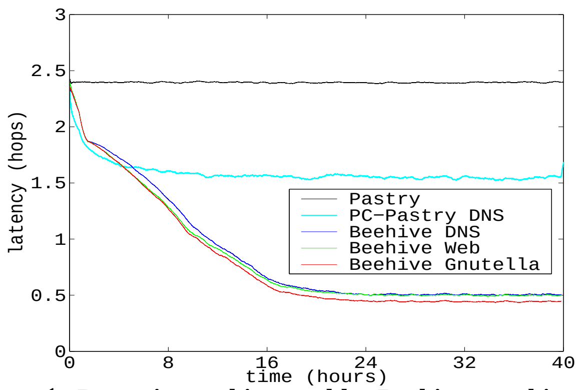

Figure 1: **Proactive caching enables Beehive to achieve the target lookup performance of 0.5 hops. In contrast, passive caching provides only a small improvement in lookup performance.**

per node on average, while Beehive can provide the same lookup performance by replicating only 100 objects per node. Overall, these experimental results substantiate our earlier claim that passive caching is unsuited for demanding applications such as DNS, and indicate that controlled proactive replication can provide better than single-hop lookup performance efficiently for a wide variety of applications.

3 Performance Intensive Applications

We have shown that lookup performance in structured DHTs can be improved to any extent through controlled replication exploiting the DHT structure and popularity distribution. In addition to the performance improvement, proactive replication also substantially increases the availability of the content. Continuous adaptation of replication to changes in popularity enables prompt response to flash crowds, while stipulating a minimum level of replication for all objects easily provides tolerance against failures. High performance combined with self-organization and resilience make distributed overlays ideal building blocks for a wide class of highly demanding and performance sensitive applications. In this section, we describe three important applications that can benefit greatly from our proactive replication scheme.

Domain Name System

The DNS is a legacy system that has existed for more than 15 years with very little change to its core design. The current use and scale of the Internet has exposed several of its shortcomings. First, DNS is a latency sensitive service; even though the current DNS response times (see Table 1) compare to average round trip times in the Internet, DNS is a huge performance bottleneck in providing fast access to web content, since access to every web page generates one or more DNS queries. Second, the hierarchical design of DNS poses improper load balance at the root servers and makes them a significant source of vulnerability to the whole system. Distributed denial of service attacks targeting some or all of the root servers can easily bring down the whole DNS. Finally, DNS imposes a considerable overhead to administer and secure; misconfigurations and faulty implementation of name servers are a significant cause for high DNS query failure rate [9, 10].

Proactive caching enables peer-to-peer overlays to meet the performance requirements of DNS. Proactive caching provides lookup performance that can be tuned to provide very low latency. At the same time, updates can be quickly propagated to all replicas in a proactive manner. The decentralized peer-to-peer design obviates the necessity for manual administration, evenly balances load across several peers, and reduces vulnerability to DDoS attacks. Overall, distributed overlays can be used to build a cooperative DNS that outperforms legacy DNS in many aspects including response time, availability, and robustness.

Implementing DNS on a distributed overlay raises several issues. Security is the primary concern, since DNS requests may be answered by any peer node. Clients receiving responses from peer nodes need to be assured of the authenticity of the name mappings. DNS Security Extensions [5] (DNSSEC), an existing standard, offers a solution for authenticating DNS responses. DNSSEC enables authentication by associating a chain of certificates that can be easily verified by any client. By replicating the certificates along with DNS data, distributed overlays can provide secure and robust services.

Some DNS servers dynamically generate responses to DNS requests for different reasons. For example, nameservers perform load balancing by randomly changing the order of IP addresses in DNS replies, and Akamai nameservers provide addresses of nearby web servers by dynamically generating DNS replies. Distributed overlays automatically provide load balancing, and can be easily modified to route queries through nearby nodes. However, changes to the DNS client may be required to implement other specialized services. Alternatively, the clients can incur an extra hop by forwarding the request directly to the nameserver after obtaining the identity of the nameserver using the overlay network.

Web Access

The Web is an extensively used repository of information and services, but access to the Web is still slow and unreliable. Existing approaches to reduce the access latency on the web are predominantly based on caching, while extensive studies on web caching have shown that the performance benefit is very limited [14]. The second significant problem with the Web is that servers can be overwhelmed by sudden increases in load and become extremely slow or unresponsive. This behavior may arise due to flash-crowds or may be intentionally caused by denial of service attacks.

Proactive replication on distributed overlays can efficiently provide lookup performance below a single-hop, and hence is a well-suited approach to bring down the latency of web access and improve its reliability. Fast access to the most popular content at the expense of less popular data substantially brings down the overall traffic. Moreover, serving data replicated at other peers within the same institution can considerably lower the cost paid for the bandwidth consumption. Proactive replication scheme not only thwarts the availability and load imbalance problems caused by flash crowds, but also proactively fetches the suddenly popular content right to the user's computer.

We have to address several problems while implementing replication to support Web access on distributed overlays. First, web objects come in a wide range of sizes. Hence, we need to incorporate object sizes to the replication cost in the analytical model. One simple heuristic to enhance the analytical model is to treat each object as a combination of several fixed-size fragments, but each receiving a fraction of requests to the whole object. Ultimately objects will be replicated at each node as a whole, but with a small extra overhead. Second, the web hosts an increasing amount of content that is dynamically generated by servers. Replicated versions of these servers can be made available to improve the access latency of dynamic content. While distributed overlays do not support replication of services in the same manner as objects, our analytical model provides a good intuition to judge the number of replicated servers that need to be deployed to improve access latency.

Secure authentication of data provided by peer nodes is a significant concern in distributed overlays. Unlike the DNS, web objects are not self-certifying. Establishing a centralized infrastructure to support Internet scale data authentication is not a realistic solution. A practical approach to data authentication is to simultaneously fetch copies of the object (or message-digests if the object is large) from several nodes and check for majority agreement. Secure admission control can limit the distribution of malicious nodes and prevent them from occupying large portions of the identifier space in DHTs [2]. Hence, by querying 2f+1 nodes, where f is an estimate on the number of malicious nodes in a fixed length portion of the identifier space, we can efficiently authenticate the data using few extra nodes.

Content Distribution

Distributed sharing of content, especially multimedia content, is the driving application for peer-to-peer systems in today's Internet. While sharing of copyrighted multimedia content is a contentious issue, there are several legitimate vendors and distributors of multimedia content who would like to reach a vast audience. The existing P2P systems, such as Kazaa, are very inefficient, have expensive and unreliable lookup protocols, and generate huge amount of traffic in the Internet. Further, multimedia servers frequently go down even with small increase in load caused by sudden changes in popularity.

Proactive replication on structured overlays enables content distributors to provide fast and efficient access to their material by providing less than single-hop lookups. In particular, some of the most popular files are proactively fetched and stored in all the computers, thus providing instantaneous access to them. Replication based on popularity allows the system to meet sudden increase or decrease in demand, by increasing or decreasing the number of replicas. Distributed overlays also provide load balance, uniformly spreading requests among several nodes, and find nearby nodes for downloading the content.

Replicating multimedia content imposes different kinds of issues, because multimedia files differ from regular web content in their characteristics. The popularity distribution of multimedia files deviates significantly from Zipf-like distributions [6]. We can easily handle this problem by solving the optimization problem for any popularity distribution numerically instead of analytically. Multimedia content is immutable, and therefore caching can provide reasonable improvement to lookup latency. But as we showed earlier, controlled replication driven by an analytical model incurs considerably lower bandwidth and storage cost than passive caching. This is particularly significant because, multimedia files are very big and unnecessary copying of data can impose substantial overhead in terms of bandwidth. Overall, structured overlays can serve multimedia content providing fast and guaranteed access with minimal network overhead.

4 Conclusions

Even though structured overlays provide self organization, load balance, and failure resilience, high lookup costs make most of them unsuitable for latency sensitive applications such as DNS. Proactive replication of objects reduces lookup cost with a corresponding, but small increase in storage and bandwidth overhead. By exploiting the regular structure of the DHTs and popularity distribution of objects, we can reduce the cost of replication and achieve any desired gain in lookup performance. This enables overlays to serve as ideal building blocks for high performance, scalable, and reliable systems. We have presented a case for three different applications, namely DNS, web access, and multimedia content distribution, which can derive great benefits by using distributes overlays and controlled replication.

Acknowledgments

We are grateful to Robert Morris and his students for providing us with their DNS traces, and to Kunwadee Sripanidkulchai for providing Gnutella traces.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. "Web Caching and Zipf-like Distributions: Evidence and Implications." *IEEE INFOCOM*, New York NY, 1999.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach. "Secure Routing for Structured Peer-to-Peer Overlay Networks." *OSDI*, Boston MA, 2002.

[3] R. Cox, A. Muthitacharoen, and R. Morris. "Serving DNS using a Peer-to-Peer Lookup Service". *IPTPS*, Cambridge MA, 2002.

[4] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, M. Theimer. "Reclaiming Space from Duplicate Files in a Serverless Distributed File System." *ICDCS*, Vienna, Austria, 2002.

[5] D. Eastlake. "Domain Name System Security Extensions". *RFC 2535*, 3^{rd} ed., 1999.

[6] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan. "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload." *ACM SOSP*, Bolton Landing NY, 2003.

[7] I. Gupta, K. Birman, P. Linga, A. Demers, and R. V. Renesse. "Kelips: Building an Effi cient and Stable P2P DHT Through Increased Memory and Background Overhead." *IPTPS*, Berkeley CA, 2003.

[8] A. Gupta, B. Liskov, R. Rodrigues. "One Hop Lookups for Peer-to-Peer Overlays." *HotOS*. Hawaii, 2003.

[9] J. Jung, E. Sit, H. Balakrishnan, and R. Morris. "DNS Performance and Effectiveness of Caching." *ACM SIGCOMM Internet Measurement Workshop*, San Francisco CA, 2001.

[10] P. Mockapetris and K. Dunlop. "Development of the Domain Name System." *ACM SIGCOMM*, Stanford CA, 1988.

[11] V. Ramasubramanian and E. G. Sirer. "Beehive: Exploiting Power Law Query Distributions for O(1) Lookup Performance in Peer to Peer Overlays" (under submission).

[12] A. Rowstorn and P. Druschel. "Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems." *IFIP/ACM Middleware*, Heidelberg, Germany, 2001.

[13] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. "Chord: A scalable Peer-to-peer Lookup Service for Internet Applications." *ACM SIGCOMM*, San Diego CA, 2001.

[14] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. "On the scale and performance of cooperative Web proxy caching." *ACM SOSP*, Kiawah Island SC, 1999.