Exhibit 1010

A Survey of Cooperative Caching

Mohammad Salimullah Raunak

December 15, 1999

Abstract

The last decade has seen a super-exponential growth of the World Wide Web. The demand of bandwidth has been consistently increasing faster than the resource. Researchers have looked for alternative solutions to improve response time and reduce bandwidth consumption in the Internet. Caching has been well accepted as a viable method for ease the ever growing bandwidth need and also to improve the speed of information delivery. However, single point caching has limitation regarding scalability. Cooperative caching, where cache servers support each other in serving requests for cached objects, has emerged as an approach to overcome the limitation. This survey looks at different studies done on cooperative caching. The studies have been grouped together according to their approaches to achieve cooperation. Cooperative caching architectures can be divided into two major categories, hierarchical and distributive. Both approaches have their advantages and disadvantages. There is also a third category of cooperative caching where cache servers are clustered together to use multicast for communication. This report also provides descriptions of all the known cache to cache communication protocols. Most of the work regarding cooperative caching involves designing the architecture and communication protocols. More work is needed to find out the effectiveness of cooperative caching for bandwidth intensive applications. Also, maintaining consistency in cooperative environment is an important research area that needs more attention.

1 Introduction

Due to its continuous exponential growth, the World Wide Web is increasingly experiencing several problems, such as "hot spots", increased network bandwidth usage, and excessive document retrieval latency known as the World Wide Wait problem. The popular solution to these problems is to use a caching proxy [8, 1, 4, 6]. Over the last few years, caching has become very popular and it is well accepted as the method to ameliorate the situation caused by the mentioned problems.

A lot of research has been done based on using a proxy cache to save bandwidth and reduce latency and congestion in the Internet. However, as researchers have found out there are certain limitations in using a single proxy [6, 14]. The single proxy can be a bottleneck. It is also difficult to scale with a single cache server architecture. As the client population

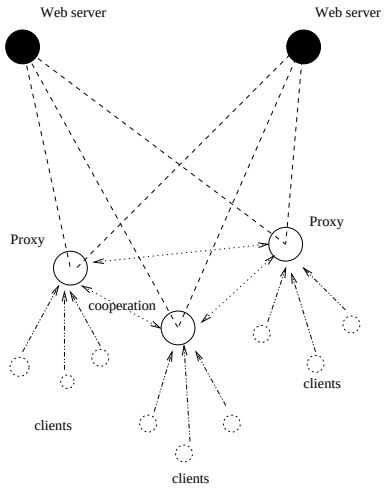

Figure 1: A general caching architecture

increases, it produces a lot of load on the proxy servers which eventually degrades the service quality and thus reduces the effectiveness of caching. Moreover, a caching proxy is a potential single point of failure. These ideas propelled the design of cooperative caching. In a cooperative environment, a number of cache servers work together to serve a set of clients. This helps to cache more objects collectively without overloading a single proxy. The architecture also builds a better system in terms of fault tolerance as the proxies can share the burden when a particular proxy is down or unavailable. A general cooperative caching structure is illustrated in figure 1.

Caching systems are generally evaluated according to three metrics: speed, scalability, and reliability. There are variety of design techniques which many commercial and academic systems use to improve performance in these respects. Among them are: caching architecture (hierarchical, distributive etc.), cache-to-cache communication, prefetching, consistency methods, optimized disk I/O and dedicated microkernel operating system. This survey is going to focus on the first two.

There are basically two types of cooperative architectures that have been studied by the researchers - Hierarchical and Distributive. With hierarchical caching, caches are placed at multiple levels of the network. With distributed caching, caches are only placed at the bottom levels of the network and there are no intermediate caches [20]. There are also some hybrid architectures that goes in between.

The rest of the report is organized as follows. Section 2 provides a brief background about caching and cooperative caching. Section 3 looks at the effectiveness of cooperative caching in file servers. Section 4 describes the major hierarchical caching architectures. Section 5 groups the cooperative caching techniques that are based on clusters and multicast communication. Section 6 describes distributed caching architectures. Section 7 deals with

different cache to cache communication protocols and section 8 provides some comparative analysis between hierarchical and distributed caching. Section 9 concludes the survey.

2 Background

One of the earliest efforts to support caching in a wide-area network environment was the Domain Naming System [16]. While not a general file or object cache, the DNS supports caching of name lookup results from server to server and also from client to server using timeouts for cache consistency.

Cooperative caching first came under consideration in the context of distributed file systems and database systems [13]. The early such studies found cooperative caching on distributed file systems somewhat ineffective [17]. With trace driven simulation, Muntz and Honeyman [17] concluded that the sharing amongst the clients of the file system is quite low (less than 10%) for caching-only intermediate servers to be effective. They also found disappointingly low cache hit rates (less than 20%) even with infinite cache at the intermediary servers.

These results led researchers to think for a little while that cooperative caching is probably not going to be effective for Internet environment either. However, these experiments were done in a small LAN environment tracing the workstations' file system traffic. This scenario is completely different than Internet object sharing. While workstation file systems share a large relatively static collection of files, such as *gcc*, the Internet exhibits a high degree of read-only sharing among a rapidly evolving set of popular objects. This was pointed out by the Harvest study [6] - one of the pioneering works on hierarchical caching. Researchers, in the mean time, had established the usefulness of caching in the Internet [8]. Cooperative caching also received attention soon [8, 7, 14]. Along with the idea of cooperative cache, came the issues of cache to cache communication and other protocol issues. With the explosion of the World Wide Web, large scalable cache architecture became a focus of attention both in the academia and in the industry.

3 Cooperative Caching in File Systems

3.1 Caching in Distributed Environment

As mentioned in section 2, cooperative caching was first studied under the distributed file system environment. Muntz and Honeyman [17] and Blaze and Alonso [2] simulated multi-level caching architectures on distributed file systems in the LAN environment. They used traces taken from over a hundred workstations running the Andrew File System at DEC's System Research Center. While Muntz and Honeyman found "disappointingly low" hit rates, Blaze and Alonso reported that caching could reduce file server traffic by a factor of two or more, and thought that a hierarchical set of caches could reduce load by an order of

magnitude. Although these two studies employed similar file workloads, Blaze and Alonso cached only read-mostly files while Muntz and Honeyman cached writes as well. Blaze and Alonso cached reads both to reduce server workload and for improved consistency. This is probably one of the reasons why the two studies found somewhat different results. The file type studied by Blaze and Alonso are similar to today's Internet environment.

3.2 Improving File System Performance

Dahlin et al. performed a study on file system to see the improvement in file system performance through the use of cooperative caching [7]. In this study, they tried to coordinate the file caches of many machines distributed on a LAN to form a more effective overall file cache. With trace driven simulation, the authors studied four cooperative caching algorithms. They found that cooperative caching in file systems can reduce the disk access by half. They also found an improvement of 73% in file system read response time. This study concluded cooperative caching is effective to improve file system read response time. The paper also tried to establish that relatively simple cooperative caching algorithms are sufficient to realize most of the potential performance gain.

The algorithms discussed in this paper were *Direct Client Cooperation*, *Greedy Forwarding*, *Centrally Coordinated Caching* and *N-chance Forwarding*.

- The idea in *direct client cooperation* is to allow an active client to use an idle client's memory as backing store. The active client forwards cache entries that overflow its local cache directly to an idle machine. The active client can then access this private remote cache to satisfy its read requests until the remote machine becomes active and evicts the cooperative cache.

- In *greedy forwarding*, the cache memories of all clients in the system are considered as a global resource that may be accessed to satisfy any client's request, but the algorithm does not attempt to coordinate the contents of these caches. Each client manages its local cache greedily, without regard to the contents of the other caches in the system or the potential needs of other clients. If a client does not find a block in its local cache, it asks the server for the data. If the server has the required data in its memory cache, it supplies the data. Otherwise, the server consults a data structure listing the contents of the client caches. If any client is caching the required data, the server forwards the request to that client. The client receiving the forwarded request sends the data directly to the client that made request.

- *Centrally coordinated caching* adds coordination to the Greedy Forwarding algorithm by statically partitioning each client's cache into a locally managed section, managed greedily by that client, and a globally managed section, coordinated by the server as an extension of its central cache. If a client does not find a block in its locally managed cache, it sends the request to the server. If the server has the desired data in memory, it supplies the data. Otherwise the server checks to see if it has stored the block in centrally coordinated client memory. If it locates the data in client memory, it forwards the request to the client storing the data. If all fails the server supplies the data from disk.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.