Proceedings of the Fourth International Web Caching Workshop, San Diego, California
April, 1999, pages 67-77

A Survey of Proxy Cache Evaluation Techniques

Brian D. Davison
Department of Computer Science
Rutgers, The State University of New Jersey
New Brunswick, NJ 08903 USA
davison@cs.rutgers.edu
http://www.cs.rutgers.edu/~davison/

Abstract

Proxy caches are increasingly used around the world to reduce bandwidth requirements and alleviate delays associated with the World-Wide Web. In order to compare proxy cache performances, objective measurements must be made. In this paper, we define a space of proxy evaluation methodologies based on source of workload used and form of algorithm implementation. We then survey recent publications and show their locations within this space.

1 Introduction

Proxy caches are increasingly used around the world to reduce bandwidth and alleviate delays associated with the World-Wide Web. This paper describes the space of proxy cache evaluation methodologies and places current research within that space. The primary contributions of this paper are threefold: 1) definition and description of the space of evaluation techniques; 2) appraisal of the different methods within that space; and 3) a survey of cache evaluation techniques from the research literature.

In the next section we provide background into web caching, including the levels of caching present on the web and and an overview of some of the current research issues in web proxy caching. We then describe current proxy cache evaluation methods and place existing research in this space.

2 Background

Caching has a long history and is a well-studied topic in the design of computer memory systems (e.g. [HP87, Man82, MH99]), in virtual memory management in operating systems (e.g. [Dei90, GC97]), in file systems (e.g. [CFKL95]), Caching on the Internet is also performed for other network services such as DNS [Moc87a, Moc87b], Gopher and FTP [Pet98, Wes98, RH98, Cat92], and in fact much of today's web caching research can be traced back to the effort to reduce the bandwidth used by FTP [DHS93].

2.1 Web caching

Web caching is the temporary storage of web objects (such as HTML documents) for later retrieval.[1] Proponents of web caching claim three significant advantages to web caching: reduced bandwidth consumption (fewer requests and responses that need to go over the network), reduced server load (fewer requests for a server to handle), and reduced latency (since cached responses are available immediately, and closer to the client being served). A fourth is sometimes added: more reliability, as some objects may be retrievable via cache even when the original servers are not reachable. Together, these features can make the World Wide Web less expensive and better performing. One drawback of caching is the potential of using an out-of-date object stored in a cache instead of fetching the current object from the origin server. Another is the lack of logs of client viewings of pages for the purposes of advertising (although this is being addressed, e.g. [ML97]).

[1] Actually, web caches store HTTP responses, but we will use the more generic phrase, web object, as a simplification throughout.

Caching can be performed by the client application, and is built into virtually every web browser. There are a number of products that extend or replace the built-in caches[2] with systems that contain larger storage, more features (such as keeping commonly used pages up-to-date and prefetching likely pages), or better performance (such as faster response times as a result of better caching mechanisms). However, while these systems cache network objects from many servers, they do so for a single user.

Caching can also be utilized between the client and the server as part of a proxy. *Proxy caches* are often located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections. When shared with other users, these systems serve many clients with cached objects from many servers. In fact, much of the usefulness (up to 85% of the in-cache requests [DMF97]) is in caching objects requested by one client for later retrieval by another client. For even greater performance, many proxy caches are part of cache hierarchies, in which a cache can request documents from neighboring caches instead of fetching them directly from the server [CDN+96].

Finally, caches can be placed directly in front of a particular server, to reduce the number of requests the server must handle. Most proxy caches can be used in this fashion, but this form is often called a *reverse cache* or sometimes an *httpd accelerator* to reflect the fact that it caches objects for many clients but usually from only one server [CDN+96, Wes98].

2.2 Proxy caching research

A shared proxy cache serves a population of clients. When a proxy cache receives a request for a web object, it checks to see if the response is available in memory or disk, and if so, returns the response to the client without disturbing the upstream network connection or destination server. If it is not available in the cache, the proxy attempts to fetch the object directly. However, if at some point the space required to store all the objects being cached exceeds the available space, the proxy will need to replace an object from the cache. In general, cache replacement algorithms attempt to maximize the percentage of requests successfully fulfilled by the cache (called the *hit ratio*) by holding onto the items most likely to be requested in the future. However, a number of researchers argue that access cost should be included in replacement calculations [BH96, CI97, WA97, SSV97, SSV98, SSV99].

Even though clients can only retrieve documents from a cache, there is still the issue of cache consistency, as source documents may be updated after the document was loaded into the cache [GS96, CDN+96, SSV98, SSV99]. Therefore, proxy caches generally do not guarantee strong consistency since there is a possibility of returning an out-of-date object to the client.

Some caching proxies exist as extensions or options of HTTP servers, such as the publicly available servers Apache [Apa99] and Jigsaw [Wor99]. Proxies are also available as stand-alone systems, such as the publicly available Squid [Wes98] and the commercial CacheFlow [Cac99b], Cisco Cache Engine [Cis99], and the Microsoft Proxy Server [Mic99]. While somewhat dated, Cormack [Cor96] provides a good introduction to web caching and includes an overview of many of these caching systems and a few others.

Much like the memory hierarchy of today's hardware systems, many researchers claim benefits of building hierarchies of web caches [CDN+96, Nat99, Dan98]. Difficulties include finding a nearby cache [CC95, CC96, ZFJ97], knowing what is in the neighboring caches [GRC97, GCR97, FCAB98, RCG98], and communication between caches [WC97b, WC97a, RW98, VR98, Vix98].

3 Evaluating Proxy Cache Performance

It is useful to be able to assess the performance of proxy caches, both for consumers selecting the appropriate system for a particular situation and also for developers working on alternative caching mechanisms. By evaluating performance across all measures of interest, it is possible to recognize drawbacks with particular implementations. For example, one can imagine a proxy with a very large disk cache that provides a high hit rate, but because of poor disk management it noticeably increases the client-perceived latency. One can also imagine a proxy that prefetches inline images but releases those images from the cache shortly afterward (to make room for other objects). This would allow it to report high request hit rates, but not perform well in terms of bandwidth savings. Finally, it is possible to select a proxy based on its performance on a workload of requests generated by dialup users and have it perform unsatisfactorily as a parent proxy for a large corporation with other proxies as clients. These examples illustrate the importance of appropriate evaluation

[2] See `http://www.web-caching.com/` for lists of proxies and browser extensions.

	Workload Source	
Algorithm Implementation	artificial	captured logs
simulated systems/network	A1	A2
real systems/isolated network	B1	B2

Table 1: *The space of traditional evaluation methodologies for web systems.*

	Workload Source		
Algorithm Implementation	artificial	captured logs	current requests
simulated systems/network	A1	A2	A3
real systems/isolated network	B1	B2	B3
real systems/real network	C1	C2	C3

Table 2: *The expanded space of evaluation methodologies for web systems.*

mechanisms for proxy systems.

3.1 The Space of Cache Evaluation Methods

The most commonly used cache evaluation method is that of simulation on a benchmark log of object requests. The byte and page hit rate savings can then be calculated as well as estimates of latency improvements. In a few cases, an artificial dataset with the necessary characteristics (appropriate average and median object sizes, or similar long-tail distributions of object sizes and object repetitions) is used. More commonly, actual client request logs are used since they arguably better represent likely request patterns and include exact information about object sizes and retrieval times.

We characterize web/proxy evaluation architectures along two dimensions: the source of the workload used and form of algorithm implementation. Table 1 shows the traditional space with artificial vs. captured logs and simulations vs. implemented systems. With the introduction of prefetching proxies, the space needs to be expanded to include implemented systems on a real network connection, and in fact, evaluation of an implemented proxy in the field would necessitate another column, that of live requests. Thus, the expanded view of the space of evaluation methodologies can be found in Table 2 and shows that there are at least three categories of data sources for testing purposes, and that there are at least three forms of algorithm evaluation. While it may be possible to make finer distinctions within each area, such as the representativeness of the data workload or the fidelity of simulations, the broader definitions used in the expanded table form a simple yet useful characterization of the space of evaluation methodologies.

For the systems represented in each area of the space, we find it worthwhile to consider how the desired qualities of web caching (reduced bandwidth/server load and improved response time) are measured. As will be seen below, each of the areas of the space represents a trade-off of desirable and undesirable features. In the rest of this section we point out the characteristic qualities of systems in each area of the space and list work in web research that can be so categorized.

3.2 Methodological Appraisal

In general, both realism and complexity increase as you move diagonally downward and to the right in the evaluation methodology space. Note that only methods in areas A1, A2, B1 and B2 are truly replicable, since live request stream samples change over time, as do the availability and access characteristics of hosts via a live network connection.

As can be seen, caching mechanisms are most commonly evaluated by simulation on captured client request logs like the sample log shown in Figure 1. These logs are generally recorded as the requests pass through a proxy, but it is also possible to collect logs by packet sniffing (as in [WWB96, MDFK97, GB97]) or by appropriately modifying the browser to perform the logging (e.g. [TG97, CBC95, CP95]).

Simulated systems.
Simulation is the simplest mechanism for evaluation as it does not require full implementation. However, simulating the caching algorithm requires detailed knowledge of the algorithms which is not always possible (especially for commercial implementations). Even then, simulation cannot accurately

```
893252015.307 14 <client-ip> TCP_HIT/200 227 GET
  http://images.go2net.com/metacrawler/images/transparent.gif - NONE/- image/gif
893252015.312 23 <client-ip> TCP_HIT/200 4170 GET
  http://images.go2net.com/metacrawler/images/head.gif - NONE/- image/gif
893252015.318 38 <client-ip> TCP_HIT/200 406 GET
  http://images.go2net.com/metacrawler/images/bg2.gif - NONE/- image/gif
893252015.636 800 <client-ip> TCP_REFRESH_MISS/200 8872 GET
  http://www.metacrawler.com/ - DIRECT/www.metacrawler.com text/html
893252015.728 355 <client-ip> TCP_HIT/200 5691 GET
  http://images.go2net.com/metacrawler/images/market2.gif - NONE/- image/gif
893252016.138 465 <client-ip> TCP_HIT/200 219 GET
  http://images.go2net.com/metacrawler/templates/tips/../../images/pixel.gif -  NONE/- image/gif
893252016.430 757 <client-ip> TCP_REFRESH_HIT/200 2106 GET
  http://images.go2net.com/metacrawler/templates/tips/../../images/ultimate.jpg -
  DIRECT/images.go2net.com image/jpeg
```

Figure 1: *This excerpt of a proxy log generated by Squid 1.1 records the timestamp, elapsed-time, client, code/status, bytes, method, URL, client-username, peerstatus/peerhost and objecttype for each request. It is also is an example of how requests can be logged in an order inappropriate for replaying in later experiments.*

assess document retrieval delays [WA97].

It is also impossible to accurately simulate caching mechanisms that prefetch on the basis of the contents of the pages being served (termed *content-based prefetching*), such as those in CacheFlow [Cac99b] and Wcol [CY97], since they need at least to have access to the links within web pages — something that is not available from server logs. Even if page contents were logged (as in [CBC95, MDFK97]), caches that perform prefetching may prefetch objects that are not on the user request logs and thus have unknown characteristics such as size and web server response times.

Real systems/isolated networks.
In order to combat the difficulties associated with a live network connection, measurement techniques often eliminate the variability of the Internet by using local servers and isolated networks, which may generate unrealistic expectations of performance. In addition to not taking into consideration current web server and network conditions, isolated networks do not allow for retrieval of current content (updated web pages).

Real systems and networks.
Because the state of the Internet changes continuously (i.e. web servers and intermediate network connections may be responsive at one time, and sluggish the next), tests on a live network are generally not replicable. Additionally, to perform such tests the experiment requires reliable hardware, robust software, and a robust network connection to handle the workload applied. Finally, connection to a real network requires compatibility with, and no abuse of, existing systems (e.g. one cannot run experiments that require changes to standard httpd servers or experimental extensions to TCP/IP).

Artificial workloads.
Synthetic traces are often used to generate workloads that have characteristics that do not currently exist to help answer questions like whether the system can handle twice as many requests per second, or whether caching of non-web objects is feasible. However, artificial workloads often make significant assumptions (e.g. that all objects are cachable, or that requests follow a particular distribution) which are necessary for testing, but not necessarily known to be true.

Captured logs.
Using actual client request logs can be more realistic than artificial workloads, but since on average the lifetime of a web page is short (less than two months [Wor94, GS96, Kah97, DFKM97]), any captured log loses its value quickly as more references within it are no longer valid, either by becoming inaccessible or by changing content (i.e., looking at a page a short time later may not give the same content). In addition, unless the logs are recorded and processed carefully, it is possible for the logs to reflect an inaccurate request ordering as the sample Squid logs show in Figure 1. Note that the request for the main page follows requests for three subitems of that main page. This is because entries in the log are recorded when the request is completed, and the timestamp records the time at which the socket is closed. Finally, proxy cache trace logs are inaccurate when they return stale objects since they may

not have the same characteristics as current objects. Note that logs generated from non-caching proxies or from HTTP packet sniffing may not have this drawback. In other work [Dav99b], we further examine the drawbacks of using standard trace logs and investigate what can be learned from more complete logs that include additional information such as page content.

Current request stream workloads.
Using a live request stream produces experiments that are not reproducible (especially when paired with live hardware/networks). Additionally, the test hardware and software may have difficulties handling a high real load. On the other hand, if the samples are large enough and have similar characteristics, an argument for comparability might be made.

3.3 Placement of current work in the space of methodologies

In this section, we place work from the research literature into the evaluation space described above. Note that inclusion/citation in a particular area does not necessarily indicate the main thrust of the paper mentioned.

Area A1: Simulations using artificial workloads.
It can be difficult to characterize a workload sufficiently to be able to generate a credible artificial workload. This, and the wide availability of a number of proxy traces, means that only a few researchers attempt this kind of research:

- Jiang and Kleinrock [JK97, JK98] evaluate prefetching mechanisms theoretically (area A1) but also on a limited trace set (area A2).
- Manley, Seltzer and Courage [MSC98] propose a new web benchmark which generates realistic loads to focus on measuring latency. This tool would be used to perform experiments somewhere between areas A1 and A2 as it uses captured logs to build particular loads on request.
- Tewari et. al. [TVDS98] use synthetic traces for their simulations of caching continuous media traffic.

Area A2: Simulations using captured logs.
Simulating proxy performance is much more popular than one might expect from the list of research in area A1. In fact, the most common mechanism for evaluating an algorithm's performance by far is simulation over one or more captured traces.

- Cunha et. al. [CBC95], Partl [Par96], Williams et. al. [WAS+96], Gwertzman and Seltzer [GS96], Cao and Irani [CI97], Bolot and Hoschka [BH96], Gribble and Brewer [GB97], Duska, Marwood and Feeley [DMF97], Caceres et. al. [CDF+98], Niclausse, Liu and Nain [NLN98], Kurcewicz, Sylwestrzak and Wierzbicki [KSW98] and Scheuermann, Shim and Vingralek [SSV97, SSV98, SSV99] all utilize trace-based simulation to evaluate different cache management algorithms.
- Trace-based simulation is also used in evaluating approaches to prefetching, such as Bestavros's server-side speculation [Bes95], Padmanabhan and Mogul's persistent HTTP protocol along with prefetching [PM96], Kroeger, Long and Mogul's calculations on limits to latency improvement from caching and prefetching [KLM97], Markatos and Chronaki's object popularity-based method [MC98], Fan, Cao and Jacobson's latency reduction to low-bandwidth clients via prefetching [FJCL99] and Crovella and Barford's prefetching with simulated network effects [CB98].
- Markatos [Mar96] simulates performance of a web server augmented with a main memory cache on a number of public web server traces.
- Mogul et. al. [MDFK97] use two large, full content traces to evaluate delta-encoding and compression methods for HTTP via calculated savings (area A2) but also performed some experiments to include computational costs on real hardware with representative request samples (something between areas B1 and B2).

Area A3: Simulation based on current requests.
We are aware of no published research that could be categorized in this area. The algorithms examined in areas A1 and A2 do not need the characteristics of live request streams (such as contents rather than just headers of HTTP requests and replies). Those researchers who use live request streams all use real systems of some sort (as will be seen below).

Area B1: Real systems on an isolated network using an artificial dataset.
A number of researchers have built tools to generate workloads for web systems in a closed environment.

Such systems make it possible to generate workloads that are uncommon in practice (or impossible to capture) to illuminate implementation problems. These include:

- Almeida, Almeida and Yates [AAY97] propose a web server measurement tool (Webmonitor), and describe experiments driven by an HTTP load generator.
- Banga, Douglis and Rabinovich [BDR97] use an artificial workload with custom client and server software to test the use of transmitting only page changes from a server proxy to a client proxy over a slow link.
- Almeida and Cao's Wisconsin Proxy Benchmark [AC98b, AC98a] uses a combination of web client and web server processes on an isolated network to evaluate proxy performance.
- While originally designed to exercise web servers, both Barford and Crovella's SURGE [BC98] and Mosberger and Jin's httperf [MJ98] generate particular workloads useful for server and proxy evaluation.
- The CacheFlow [Cac99a] measurement tool was designed specifically for areas C1 and C2, but could also be used on an isolated network with an artificial dataset.

Area B2: Real systems on an isolated network using captured trace logs.
Some of the research projects listed in area B1 may be capable of using captured trace logs. In addition, we place the following here:

- In evaluating their Crispy Squid, Gadde, Chase and Rabinovich [GCR98] describe the tools and libraries called Proxycizer. These tools provide a trace-driven client and a characterized web server that surround the proxy under evaluation, much like the Wisconsin Proxy Benchmark.

Area B3: Real systems on an isolated network using current requests.
Like area A3, we found no research applicable to this area. Since live requests can attempt to fetch objects from around the globe, it is unlikely to be useful within an isolated network.

Area C1: Real systems on a live network using an artificial dataset.
Some of the research projects in area B1 may also be extensible to the use of a live network connection. The primary restriction is the use of real, valid URLs that fetch over the Internet rather than particular files on a local server.

Area C2: Real systems on a live network using captured logs.
With captured logs, the systems being evaluated are as realistically operated without involving real users as clients. Additionally, some of the tools from area B1 may be usable in this type of experiment.

- Wooster and Abrams [Woo96, WA97] report on evaluating multiple cache replacement algorithms in parallel within the Harvest cache, both using URL traces and online (grid areas C2 and C3, respectively) but the multiple replacement algorithms are within a single proxy. Wooster also describes experiments in which a client replayed logs to multiple separate proxies running on a single multiprocessor machine.
- Maltzahn and Richardson [MR97] evaluate proxies with the goal of finding enterprise-class systems. They test real systems with a real network connection and used carefully selected high load-generating trace logs.
- Liu et. al. test the effect of network connection speed and proxy caching on response time using public traces [LAJF98] on what appears to be a live network connection.
- Lee et. al. [LHC+98] evaluate different cache-to-cache relationships using trace logs on a real network connection.

Area C3: Real systems on a live network using the current request stream.
In many respects, this area represents the strongest of all evaluation methodologies. However, most real installations are not used for the comparison of alternate systems or configurations, and so we report on only a few research efforts here:

- Chinen and Yamaguchi [CY97] described and evaluated the performance of the Wcol proxy cache on a live network using live data, but do not compare it to any other caching systems.
- Rousskov and Soloviev [RS98] evaluated performance of seven different proxies in seven different real-world request streams.

- Cormack [Cor98] described performance of different configurations at different times of a live web cache on a live network.
- Elsewhere [Dav99a] we describe the Simultaneous Proxy Evaluation (SPE) architecture that compares multiple proxies on the same request stream. While originally designed for this area with a live request stream and live network connection, it can also be used for replaying captured or artificial logs on isolated or live networks (areas B1, B2, C1, and C2).

4 Discussion and Conclusions

Evaluation is a significant concern, both for consumers and for researchers. Objective measurements are essential, as are comparability of measurements from system to system. Furthermore, it is important to eliminate variables that can affect evaluation.

Selection of an appropriate evaluation methodology depends on the technology being tested and the desired level of evaluation. Some technologies, such as protocol extensions, are often impossible to test over a live network because other systems do not support it. Similarly, if the goal is to find bottlenecks in an implementation, one may want to stress the system with very high synthetic loads since such loads are unlikely to be available in captured request traces. In fact, this reveals another instance in which a live network and request stream should not be used — when the goal of the test is to drive the system into a failure mode to find its limits.

In general, we argue for the increased believability of methodologies in the evaluation space as you move down and to the right when objectively testing the successful performance of a web caching system. If the tested systems make decisions based on content, a method from the bottom row is likely to be required. When looking for failure modes, it will be more useful to consider methodologies near the upper left of the evaluation space.

This survey provides a sampling of the published web-caching research and presents one of potentially many spaces of evaluation methodologies. In particular, we haven't really considered aspects of testing inter-cache communication, cache consistency, or low-level protocol issues such as connection caching which are significant in practice [CDF+98].

In this paper we have described a space of evaluation methodologies and shown where current research efforts fall within it. By considering the space of appropriate evaluation methodologies, one can select the best trade-off of information provided, implementation costs, comparability, and relevance to the target environment.

Acknowledgments

Thanks are due to Haym Hirsh, Brett Vickers, and anonymous reviewers for their helpful comments on earlier drafts of this paper.

References

[AAY97] Jussara Almeida, Virgílio A. F. Almeida, and David J. Yates. Measuring the behavior of a world wide web server. In *Proceedings of the Seventh Conference on High Performance Networking (HPN)*, pages 57–72. IFIP, April 1997.

[AC98a] Jussara Almeida and Pei Cao. Measuring Proxy Performance with the Wisconsin Proxy Benchmark. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998. Also available as Technical Report 1373, Computer Sciences Dept., Univ. of Wisconsin-Madison, April, 1998.

[AC98b] Jussara Almeida and Pei Cao. Wisconsin proxy benchmark 1.0. Available from `http://www.cs.wisc.edu/~cao/wpb1.0.html`, 1998.

[Apa99] Apache Group. Apache HTTP server documentation. Available at `http://www.apache.org/docs/`, 1999.

[BC98] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'98/ PERFORMANCE'98)*, pages 151–160, Madison, WI, June 1998.

[BDR97] Gaurav Banga, Fred Douglis, and Michael Rabinovich. Optimistic Deltas for WWW Latency Reduction. In *Proceedings of the USENIX Technical Conference*, 1997.

[Bes95] Azer Bestavros. Using Speculation to Reduce Server Load and Service Time on the WWW. In *Proceedings of CIKM'95: The Fourth ACM International Conference on Information and Knowledge Management*, Baltimore, MD, November 1995. Also available as Technical Report TR-95-006, Computer Science Department, Boston University.

[BH96] Jean-Chrysostome Bolot and Philipp Hoschka. Performance engineering of the world wide web: Application to dimensioning

and cache design. In *Proceedings of the Fifth International World Wide Web Conference*, Paris, France, May 1996.

[Cac99a] CacheFlow Inc. CacheFlow performance testing tool. `http://www.cacheflow.com/technology/tool/`, 1999.

[Cac99b] CacheFlow Inc. CacheFlow products Web page. `http://www.cacheflow.com/products/`, 1999.

[Cat92] V. Cate. Alex— A global filesystem. In *Proceedings of the USENIX File System Workshop*, pages 1–12, Ann Arbor, MI, May 1992.

[CB98] Mark Crovella and Paul Barford. The Network Effects of Prefetching. In *Proceedings of IEEE Infocom '98*, San Francisco, CA, 1998. More detailed version available as Boston University Computer Science Department Technical Report, TR-97-002, February 1997.

[CBC95] Carlos R. Cunha, Azer Bestavros, and Mark E. Crovella. Characteristics of WWW Client-based Traces. Technical Report TR-95-010, Computer Science Department, Boston University, July 1995.

[CC95] Mark E. Crovella and Robert L. Carter. Dynamic server selection in the Internet. In *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS'95)*, August 1995. Also available as TR-95-014, Computer Science Department, Boston University.

[CC96] Robert L. Carter and Mark E. Crovella. Dynamic Server Selection Using Bandwidth Probing in Wide-Area Networks. Technical Report TR-96-007, Computer Science Department, Boston University, March 1996.

[CDF+98] Ramón Cáceres, Fred Douglis, Anja Feldmann, Gideon Glass, and Michael Rabinovich. Web proxy caching: the devil is in the details. In *Workshop on Internet Server Performance (WISP'98)*, Madison, WI, June 1998.

[CDN+96] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A Hierarchical Internet Object Cache. In *Proceedings of the USENIX Technical Conference*, San Diego, CA, January 1996.

[CFKL95] Pei Cao, Edward W. Felten, Anna Karlin, and Kai Li. A Study of Integrated Prefetching and Caching Strategies. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1995.

[CI97] Pei Cao and Sandy Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the USENIX Symposium on Internet Technoloy and Systems*, pages 193–206, December 1997.

[Cis99] Cisco Systems, Inc. Cisco cache engine. Available at `http://www.cisco.com/warp/public/cc/cisco/mkt/scale/cache/`, 1999.

[Cor96] Andrew Cormack. Web caching. Available from `http://www.jisc.ac.uk/acn/caching.html`, 1996.

[Cor98] Andrew Cormack. Experiences with installing and running and institutional cache. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[CP95] Lara D. Catledge and James E. Pitkow. Characterizing Browsing Strategies in the World Wide Web. *Computer Networks and ISDN Systems*, 26(6):1065–1073, 1995.

[CY97] Ken-ichi Chinen and Suguru Yamaguchi. An interactive prefetching proxy server for improvement of WWW latency. In *Proceedings of the Seventh Annual Conference of the Internet Society (INET'97)*, Kuala Lumpur, June 1997.

[Dan98] Peter B. Danzig. NetCache architecture and deployment. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[Dav99a] Brian D. Davison. Simultaneous Proxy Evaluation. In *Proceedings of the Fourth International WWW Caching Workshop (WCW99)*, San Diego, CA, March 1999.

[Dav99b] Brian D. Davison. Web traffic logs: An imperfect resource for evaluation. In *Proceedings of the Ninth Annual Conference of the Internet Society (INET'99)*, June 1999. To appear.

[Dei90] Harvey M. Deitel. *Operating Systems*. Addison-Wesley, Reading, MA, 1990. Second Edition.

[DFKM97] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey C. Mogul. Rate of Change and other Metrics: a Live Study of the World Wide Web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997. An extended version is available as AT&T Labs - Research Technical Report 97.24.2.

[DHS93] Peter B. Danzig, Richard S. Hall, and Michael F. Schwartz. A case for caching file objects inside internetworks. *Computer and Communications Reviews*, 23(4):239–248, October 1993.

[DMF97] Bradley M. Duska, David Marwood, and Michael J. Feely. The Measured Access Characteristics of World-Wide-Web Client Proxy Caches. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997.

[FCAB98] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *Proceedings of ACM SIGCOMM*, September 1998.

[FJCL99] Li Fan, Quinn Jacobson, Pei Cao, and Wei Lin. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '99)*, Atlanta, GA, May 1999. To appear.

[GB97] Steven D. Gribble and Eric A. Brewer. System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997.

[GC97] Gideon Glass and Pei Cao. Adaptive Page Replacement Based on Memory Reference Behavior. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 115–126, Seattle, WA, June 1997. ACM Press.

[GCR97] Syam Gadde, Jeff Chase, and Michael Rabinovich. Directory Structures for Scalable Internet Caches. Technical Report CS-1997-18, Department of Computer Science, Duke University, November 1997.

[GCR98] Syam Gadde, Jeff Chase, and Michael Rabinovich. A taste of crispy squid. In *Workshop on Internet Server Performance (WISP'98)*, Madison, WI, June 1998.

[GRC97] Syam Gadde, Michael Rabinovich, and Jeff Chase. Reduce, Reuse, Recycle: An Approach to Building Large Internet Caches. In *The Sixth Workshop on Hot Topics in Operating Systems (HotOS-VI)*, pages 93–98, May 1997.

[GS96] James Gwertzman and Margo Seltzer. World-Wide Web Cache Consistency. In *Proceedings of the USENIX Technical Conference*, San Diego, CA, January 1996.

[HP87] Fredrick J. Hill and Gerald R. Peterson. *Digital Systems: Hardware Organization and Design*. John Wiley & Sons, New York, 1987. Third Edition.

[JK97] Zhimei Jiang and Leonard Keinrock. Prefetching Links on the WWW. In *ICC '97*, pages 483–489, Montreal, Canada, June 1997.

[JK98] Zhimei Jiang and Leonard Kleinrock. An adaptive network prefetch scheme. *IEEE Journal on Selected Areas in Communications*, 16(3):358–368, April 1998.

[Kah97] Brewster Kahle. Preserving the Internet. *Scientific American*, 276(3):82–83, March 1997.

[KLM97] Thomas M. Kroeger, Darrell D. E. Long, and Jeffrey C. Mogul. Exploring the bounds of web latency reduction from caching and prefetching. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS '97)*, December 1997.

[KSW98] Michal Kurcewicz, Wojtek Sylwestrzak, and Adam Wierzbicki. A filtering algorithm for proxy caches. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[LAJF98] Binzhang Lui, Ghaleb Abdulla, Tommy Johnson, and Edward A. Fox. Web Response Time and Proxy Caching. In *Proceedings of WebNet98*, Orlando, FL, November 1998.

[LHC+98] Jieun Lee, Hyojung Hwang, Youngmin Chin, Hyunchul Kim, and Kilnam Chon. Report on the costs and benefits of cache hierarchy in Korea. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[Man82] M. Morris Mano. *Computer System Architecture*. Prentice-Hall, Englewood Cliffs, NJ, 1982. Second Edition.

[Mar96] Evangelos P. Markatos. Main memory caching of web documents. *Computer Networks and ISDN Systems*, 28(7-11):893–905, May 1996.

[MC98] Evangelos P. Markatos and Catherine E. Chronaki. A top–10 approach for prefetching the web. In *Proceedings of the Eighth Annual Conference of the Internet Society (INET'98)*, Geneva, Switzerland, July 1998. Also available as ICS-FORTH Technical Report 173.

[MDFK97] Jeffrey Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. Potential Benefits of Delta-encoding and Data Compression for HTTP. In *Proceedings of ACM SIGCOMM*, pages 181–194, September 1997. An extended and corrected version appears as Research Report 97/4a, Digital Equipment Corporation Western Research Laboratory, December, 1997.

[MH99] Miles J. Murdocca and Vincent P. Heuring. *Principles of Computer Architecture*. Addison Wesley Longman, To appear, 1999.

[Mic99] Microsoft Corporation. Microsoft Proxy Server. Available at `http://www.microsoft.com/proxy/`, 1999.

[MJ98] David Mosberger and Tai Jin. httperf—A tool for measuring web server performance. In *Workshop on Internet Server Performance (WISP'98)*, Madison, WI, June 1998.

[ML97] Jeffrey Mogul and P. Leach. Simple hit-metering and usage-limiting for HTTP. RFC 2227, `http://nic.ddn.mil/ftp/rfc/rfc2227.txt`, October 1997.

[Moc87a] P. V. Mockapetris. Domain names — concepts and facilities. RFC 1034, `http://nic.ddn.mil/ftp/rfc/rfc1034.txt`, November 1987.

[Moc87b] P. V. Mockapetris. Domain names — implementation and specification. RFC 1035, `http://nic.ddn.mil/ftp/rfc/rfc1035.txt`, November 1987.

[MR97] Carlos Maltzahn and Kathy J. Richardson. Performance issues of enterprise level web proxies. In *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pages 13–23, Seattle, WA, June 1997. ACM Press.

[MSC98] Stephen Manley, Margo Seltzer, and Michael Courage. A self-scaling and self-configuring benchmark for web servers. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '98/ PERFORMANCE '98)*, pages 270–271, Madison, WI, June 1998.

[Nat99] National Laboratory for Applied Network Research. A distributed testbed for national information provisioning. Home page: http://www.ircache.net/, 1999.

[NLN98] Nicolas Niclausse, Zhen Liu, and Philippe Nain. A new and efficient caching policy for the world wide web. In *Workshop on Internet Server Performance (WISP'98)*, Madison, WI, June 1998.

[Par96] Tomas Partl. A comparison of WWW caching algorithm efficiency. In *ICM Workshop on Web Caching*, Warsaw, Poland, 1996.

[Pet98] Arthur Petrosyan. Development of the Armenian regional academical cache service. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[PM96] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using predictive prefetching to improve world wide web latency. *Computer Communication Review*, 26(3):22–36, July 1996.

[RCG98] Michael Rabinovich, Jeff Chase, and Syan Gadde. Not all hits are created equal: Cooperative proxy caching over a wide area network. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[RH98] Mark Russell and Tim Hopkins. CFTP — A caching FTP server. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[RS98] Alex Rousskov and Valery Soloviev. On performance of caching proxies. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '98/PERFORMANCE '98)*, pages 272–273, Madison, WI, June 1998.

[RW98] Alex Rousskov and Duane Wessels. Cache digests. In *Proceedings of the Third International WWW Caching Workshop*, Manchester, England, June 1998.

[SSV96] Peter Scheuermann, Junho Shim, and Radek Vingralek. WATCHMAN: A Data Warehouse Intelligent Cache Manager. In *Proceedings of the 22nd International Conference on Very Large Databases*, 1996.

[SSV97] Peter Scheuermann, Junho Shim, and Radek Vingralek. A case for delay-conscious caching of web documents. In *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, CA, April 1997.

[SSV98] Junho Shim, Peter Scheuermann, and Radek Vingralek. A Unified Algorithm for Cache Raplacement and Consistency in Web Proxy Servers. In *Proceedings of the Workshop on the Web and Data Bases (WebDB98)*, 1998. Extended version will appear in Lecture Notes in Computer Science, Springer Verlag, 1998.

[SSV99] Junho Shim, Peter Scheuermann, and Radek Vingralek. Proxy Cache Design: Algorithms, Implementation and Performance. *IEEE Transactions on Knowledge and Data Engineering*, 1999. To appear.

[TG97] Linda Tauscher and Saul Greenberg. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human Computer Studies*, 47(1):97–138, 1997.

[TVDS98] Renu Tewari, Harrick M. Vin, Asit Dan, and Dinkar Sitaram. Resource-based Caching for Web Servers. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, San Jose, CA, January 1998.

[Vix98] Paul Vixie. Hyper Text Caching Protocol — HTCP/0.0. Internet Draft, March 1998.

[VR98] Vinod Valloppillil and Keith W. Ross. Cache Array Routing Protocol v1.0. Internet Draft, February 1998.

[WA97] Roland P. Wooster and Marc Abrams. Proxy caching that estimates page load delays. In *Proceedings of the Sixth International World Wide Web Conference*, pages 325–334, Santa Clara, CA, April 1997.

[WAS+96] Stephen Williams, Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, and Edward A. Fox. Removal Policies in Network Caches for World-Wide Web Documents. In *Proceedings of ACM SIGCOMM*, pages 293–305, Stanford, CA, 1996. Revised March 1997.

[WC97a] Duane Wessels and Kimberly Claffy. Application of Internet Cache Protocol (ICP), version 2. RFC 2187, `http://nic.ddn.mil/ftp/rfc/rfc2187.txt`, September 1997.

[WC97b] Duane Wessels and Kimberly Claffy. Internet Cache Protocol (ICP), version 2. RFC 2186, `http://nic.ddn.mil/ftp/rfc/rfc2186.txt`, September 1997.

[Wes98] Duane Wessels. Squid Internet object cache documentation. Available at `http://squid.nlanr.net/Squid/`, 1998.

[Woo96] Roland Peter Wooster. Optimizing response time, rather than hit rates, of WWW proxy caches. Master's thesis, Virginia Polytechnic Institute, December 1996. `http://scholar.lib.vt.edu/theses/materials/public/etd-34131420119653540/etd-title.html`.

[Wor94] Kurt Worrell. Invalidation in large scale network object caches. Master's thesis, University of Colorado, Boulder, CO, 1994.

[Wor99] World Wide Web Consortium. Jigsaw HTTP server documentation. Available at `http://www.w3c.org/Jigsaw/`, 1999.

[WWB96] Roland O. Wooster, Stephen Williams, and Patrick Brooks. HTTPDUMP: Network HTTP packet snooper. Working paper available at `http://www.cs.vt.edu/~chitra/work.html`, April 1996.

[ZFJ97] Lixia Zhang, Sally Floyd, and Van Jacobson. Adaptive web caching. In *Proceedings of the NLANR Web Cache Workshop*, Boulder, CO, June 1997.