

USAGE PATTERNS IN COOPERATIVE CACHING

A. Bhargava and H. P. Dommel

Department of Computer Engineering

Santa Clara University, Santa Clara

USA

{amitb@yahoo-inc.com, hpdommel@scu.edu}

ABSTRACT

The amount of information requested over the World Wide Web has increased enormously during the past decade. Web caching helps to reduce service times, balance the load to origin servers, and brings content closer to the user. Since duplicating and distributing files amongst proxy caches has proved to be insufficient, cooperative caching aims to ameliorate the shortcomings of basic replication caching through inter-cache communication. This paper categorizes and evaluates cooperative caching schemes with regard to user access patterns from established traffic archives. We use cache statistics at the proxy level and discuss possible causes for cache misses with the aim to bring greater understanding to the interplay of access and caching for web documents.

KEYWORDS

Cooperative proxy caching, distributed information systems, user patterns, traffic analysis.

1. Introduction

The increase in World Wide Web (WWW) [1] traffic has been a mounting strain on the Internet [2]. Various mechanisms have been introduced in Internet infrastructure and end services [3] during the past decade to reduce service time, lower the load and dependency on origin servers, bring content closer to the user and thereby reduce network congestion. Cache-less architectures require the origin server to resend the same object as often as a request is received. Several methods have been proposed to reduce the perceived user latency. Increasing available bandwidth is a straightforward solution, however, this does not decrease the original contact latency perceived by the user, or alleviate server side strain and fault tolerance problems.

Web caching [4][5], as mediated storage of web files for quicker reuse, can reduce user perceived latency, increase object availability, and reduces both congestion and server load. Requesting web objects from a local cache can also reduce cost. Since web servers incur less workload, scalability improves. This paper looks at the interplay of web content and caching, the dynamics of access patterns, and cache cooperation mechanisms.

The remainder of this paper is organized as follows: Section 2 discusses the principles of cooperative caching. Section 3 introduces a simple taxonomy of prominent solutions based on caching patterns. Section 4 discusses performance criteria. Section 5 compares representative usage patterns in web traces from NLANR [6] archives and Section 6 concludes the paper.

2. Cooperative Caching Principles

The main purpose of caching is to improve retrieval latency by providing a web object closer to the client than the original source, although other benefits such as reduction in monetary costs are sometimes mentioned [7]. No matter what metrics are used to quantify success, for a cache to be useful, clients must repeatedly "hit" required web objects in a cache. Traditional cache systems have faced the challenge of limited cache hits due to the restriction to client base and size. Legacy cache servers on the same level do not communicate; requests are strictly forwarded up and down the hierarchy, and incur typical distributed system problems, such as single point failure and scalability.

To overcome this fundamental problem, different schemes have been suggested, from cache farms to cooperative caching [3]. Server farms provide a tight coupling amongst a small group of servers for greater fault tolerance and power. However, the service is topologically clustered, which impacts the average retrieval latency for web objects. By dividing the URL space among a group of servers, the load can be balanced, but even if an optimal hash algorithm is devised, hot spots frequently occurring in the web may lead to a bottleneck. Additionally, unless redundancy is built in, single point failures can still be a problem.

Instead of linearly increasing the 'server power' in conventional proxy caching, cooperative or adaptive caching [8] uses inter-cache communication to intelligently coordinate load among caching peers. Such networked sharing of cache resources by mutual cooperation solves fault tolerance problems and achieves greater hit ratios. We organize existing solutions on cooperative caching in a taxonomy to evaluate their effectiveness for various user access patterns.

3. Taxonomy

Various cooperative web caching schemes have been proposed in recent years [9] [10], with the underlying goal to reduce service latency, server overhead, and network traffic. While impractical, a theoretically infinite cache may never incur a miss. Earlier studies suggest that a few GB may be a pragmatic limit to proxy cache sizes. Caching mechanisms may be located at the server, in the network, and in clients, with the goal to find the "golden cut" to make efficient use of resources and optimize web object delivery. Server-centric schemes can be classified into three paradigms, as shown in Figure 1.

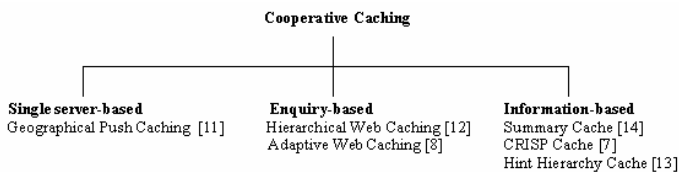


Fig. 1. Cooperative Caching Taxonomy

3.1 Single Server-based Caching

In single server-based cooperative caching, a primary server is responsible for all decisions that contribute to the goals of reduced network traffic and load balancing amongst the servers. An example is Geographical Push Caching [11], where every client accesses a primary server for a request that in turn directs to a local server with a cached copy of the document. The local server also makes a consistency check with the home server.

3.2 Enquiry-based Caching

In enquiry-based cooperative caching, when a server misses a request in its local cache, it forwards the enquiry for the object to a number of cooperating caches. This request forwarding continues until the requested object is found. If none of the caches respond, then the objects is requested from the home server. Representatives are Hierarchical Web Caching [12], where an unsuccessful query is sent to its siblings, parent and the original server after which a multicast message is sent; and Adaptive Web Caching [8], where an unsuccessful query can span the entire diameter of the network.

3.3 Information-based Caching

In information-based cooperative caching, a cache miss is handled by looking into a directory or similar information source to find an appropriate cache to provide the requested object. If the object cannot be retrieved from the listed cache, it is retrieved from the origin server and then added to the cache for future requests. The cache's notification set is occasionally broadcast to neighboring caches to indicate the update of a cache's contents. Hint Hierarchy [13], CRISP [7] [9] and Summary Cache [14] are examples of information-based

caching.

4. Performance Parameters

In our comparison we consider these three main classes and evaluate them with regard to reducing request latency, network traffic and server load.

4.1 Request Latency

Request Latency is a measure of the number of hops required to fetch the requested data. The hop count is indirectly determined by the request routing mechanism used. Single server schemes do not minimize the latency, as the primary server is responsible for routing all requests to an appropriate cache. Enquiry-based schemes focus on maximizing the hit rate by searching more caches which may also increase latency.

Information-based schemes focus on reducing the request latency, serving the request from either the cache or the origin server. Based on its "neighborhood architecture", CRISP stands out among the information-based systems, requiring lower overhead in cache handling than the other information-based methods.

4.2 Server Load

Organization of caches, consistency updates mechanism and managing the processing of requests affect the server load of a caching system.

Single server schemes achieve load balancing by using a threshold parameter and file request history to decide when and where an object is replicated to another web server. The advantage is that the objects are distributed throughout the network based on their geographic popularity, thus spreading the load of the system. However object consistency checks cause added loads on the servers. Enquiry-based approaches reduce server load by using their hierarchical organization. This reduces load depending on where the requested document is cached. However, this system performs poorly in cases where there are many siblings and a query needs to be propagated across several levels.

Information-based schemes reduce server load by a compact data structure containing information to directly locate an object in the system. The Summary Cache system [14] uses Bloom Filters to reduce the memory used in each cache. The CRISP system [9] attempts to improve on this idea by limiting the number of entries to only the neighboring caches. By limiting the information in the table, the amount of work required by the server to maintain consistency is reduced.

4.3 Network Traffic

Network caching traffic is generated by consistency updates and routing requests. Frequent updates of objects and meta-data to maintain consistency can lead to a large communication overhead. The key issue with request routing is propagation of a single request through the network in a cascading fashion.

The single server system has the potential of incurring a traffic bottleneck at the primary server, since all requests must be routed through the server. Extra network traffic is generated by updates because every request is followed by the local server checking with the home server for validation. In enquiry-based systems, a query may be multicast in case of a cache miss at the local and home server levels. Such queries can also cause overhead as the system grows with caches.

In an information-based system, network traffic is decreased by the single-hop mechanism used to route requests. These systems also reduce traffic by maintaining compact data structures and updating them periodically. The small size of the data structure makes updates less expensive, and a weak consistency mechanism reduces traffic overhead. In summary caching, the use of periodic updates on a growing system is less expensive than keeping the tables up to date, thus increasing scalability. The inherent neighborhood architecture of the CRISP system helps to minimize the number of entries stored in each cache. Use of the BGP routing protocol to construct the communication graphs imparts the ability for automatic reconfiguration on cache failures. Because the CRISP system imposes a fixed distance on the set of caches that can be searched on a request, network traffic is lower compared to other information-based systems. The only potential disadvantage of this system is that it stores only limited information about its neighboring nodes, and as a result, caches are not knowledgeable of the entire network topology. This decreases the overall hit rate in the cache. However, this might be a reasonable tradeoff since the many advantages of this system overcome its sole potential disadvantage.

5. Usage Patterns

In this section, our goal is to pinpoint important criteria that influence a cache design. We analyze a set of traces from eight different proxy sites that are a part of the NLANR cache system (bo1, bo2, pa, pb, rtp, sd, sj, startap). We evaluate two different types of traces: access log traces (user access to these proxy caches), and store log traces (history for managing objects out of memory, swapping them out from the disk or even releasing them from the proxy cache.) We analyze traces from these caches for the total number of accesses and the size of files of a specific type. We also analyze file access distributions for the total number of requests for the top ten URLs and sites, and the hit rates in the cache hierarchy over a full day at different sites and over a week at a specific site. Our

goal is confirm the viability of cooperative caching in a high client population vs. a low client population, looking at the hit ratio of main memory vs. total proxy hits. We also look at TCP connections to content servers.

5.1 File Type Analysis

Over time, there seems to be a change in the structure of the files being transferred over the web. Our traces show over 100 different file types with an average size of 11Kbytes and a tapering distribution of sizes. For verification of requested content type patterns, we analyze the traces from two different caches for one day. The analysis shows that files have different caching characteristic based their types. Some files are associated with traffic that does not use the same channel, e.g., Real Audio traffic uses multiple protocols. We first look at a mapping of content type with the number of accesses at bo1. Then to verify our findings we will look at the same mapping over all eight NLANR caches. Table 1 and 2 show the results from trace analysis, similar to the findings in [15]. Although the web traces are not current, they are useful for forensic pattern analysis. Our analysis show that 42% of the accesses are for image files and 27% are for text files for the traces from bo1. Traces taken from all eight caches show that 50% of the accesses are for image files and 24% are for text files.

Content Type	Number of Accesses	% of Total	Average Size
Image/gif	25581	42	5510
Text/html	16396	27	6302
Application	18110	30	4132
Video	234	0.4	120707
Audio	45	0.07	175237
Other	139	0.23	220244

Table 1. Characteristics of traces at bo1 on Feb 23, 2002

Content Type	Number of Accesses	% of Total	Average Size
Image/gif	85210	50	7425
Text/html	41315	24	8987
Application	36006	23	8317
Video	173	0.1	746953
Audio	162	0.1	5314
Other	683	.41	298676

Table 2. Characteristics of all eight traces on Feb 23, 2002

5.2 URLs and Sites Distribution Analysis

Previous research [16] has shown that the relative frequency of web page accesses follow Zipf's Law, which states that the relative probability of the request to the i -th most popular page is proportional to with $(1/i)^\alpha$ where α in the Web ranges from 0.6 to 0.8. We ask whether request patterns for URLs follow a Zipf-like distribution and if the asymptotic behavior of cache-hit rates is directly related to Zipf-like nature of request patterns.

Looking at access patterns for both URLs and sites, we analyze the popularity of a document over several days to find out which document is popular on a given day and continues to be popular thereafter. Figure 2 and 3 verify that the URL and site distributions follow a Zipf-like pattern. It shows that a popular document continues to remain popular over multiple consecutive days. This strongly justifies the use of a LFU replacement algorithm for caching and prefetching to improve the hit rate for web data.

5.3 Cache Statistics

It is intuitively clear that user access patterns impact cache design. Earlier work [17] [18] has shown that the hit rate of a cache is directly proportional to user access and that the hit rate is proportional to cache size. The hit rate of a local cache has influence on the network bandwidth available to user requests and the latency observed by a user. Numbers of accesses at different proxies over a full day are shown in Figure 4, with a peak during mid-day.

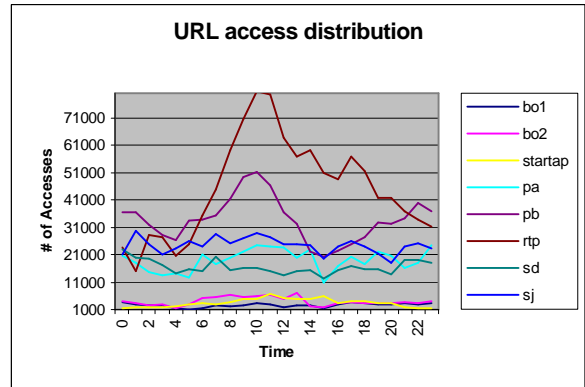


Fig. 4. URL access distribution over a full day at different proxy cache sites

We investigate the dependency between user population and cache hit rates to understand better the effectiveness of cooperative caching. Wolfman et. al. [15] argue that hit rates may reach a saturation level as the number of users exceeds a threshold of 2500. Hit rate improvements are more pronounced for smaller proxies and virtually non-existent for a much larger proxy. Our study differs in terms of locality of access, with focus on regional proxy caches. A proxy cache in the NLANR cache system has a user population of about 100-170 regional proxy caches. We assume that there is unlimited memory space on each cache and do not take into account the expiration time of documents. Figure 5 shows that with the increase in user population from 100 to 700, the hit ratio increases by 20%.

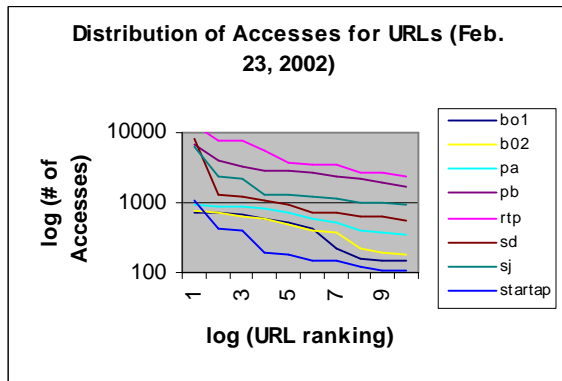


Fig. 2. Distribution of Accesses per URL

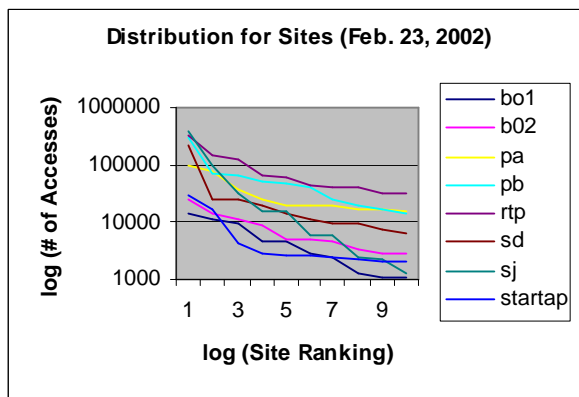


Fig. 3. Distribution of Accesses per Site

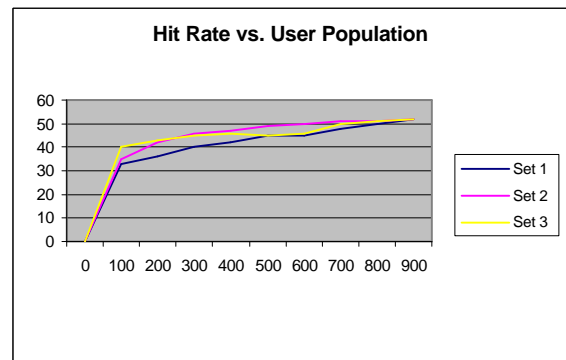


Fig. 5. Hit rate vs. User Population

It shows that the average time to retrieve one byte of data from a sibling cache is 5 to 10 times faster than retrieving the data from the original server. Thus if there is a miss in the local cache, it is beneficial that the document be retrieved from a sibling cache instead of the original server. Figure 5 also shows that in NLANR, caches achieve a better performance by

querying only three other siblings. This is because the hit rate reaches the saturation level after a user population of 500. This in turn leads to a reduction of inter-proxy ICP traffic.

Krishnan and Sugla [18] showed that the hit rates for cooperative caches are much higher than comparative numbers without cooperation. Hit rates with cooperative caching increase by up to 40% compared to non-cooperative caching. Larger hit rate improvements are seen in the case of warmer caches. Table 3 shows the distribution of accesses at different points in the cache hierarchy. The data show that there is room for improvement in the hit rates of a cooperative cache. By serving a miss in the local cache from the sibling cache versus going to the original server, the hit rate of the overall system can increase by up to 40%.

Location	Bo1	Bo2	pa	Pb	Sd	Sv	Rm	uc
Memory	796	1010	1772	1472	1450	1394	1863	1176
Local	95995	122012	135679	118300	161059	124879	225399	125541
Sibling	15887	16352	0	27718	29832	954	33234	18591
Direct	327801	421709	1229726	616899	948359	1319639	1215513	459773
Deny	1802	1798	11306	1381	64326	506	31494	17095

Table 3: Cache Statistics – Number of Accesses

5.4 Cache Replacement Algorithms

The hit rate for the main memory of a cache is about 1% of the total hit rate of the cache. This indicates that unexpired cached documents are swapped out from the main memory by cache replacement algorithms. The high difference in the documents found in memory and those found on disk indicates that there is space for improvement. Our analysis shows that the documents on the disk are accessed more frequently than the documents in the main memory. We also see that the average time to get a document from disk is higher than the time to get it from the main memory. Hence we find that the hit rate on the main memory is low.

NLANR caches use a Least-Recently-Used (LRU) algorithm. An object is brought into memory only when it is served from the original server. If an object that was moved onto disk from main memory using LRU has a hit, it is not brought back into the main memory. The object is finally removed from the disk after it expires or when there is no space on the disk. It shows [16] that temporal locality is not the dominant factor when analyzing cache performance. This is a reason why LRU is not an efficient algorithm for web caching. Table 4 compares the hit rates when using different replacement algorithms on different memory sizes. We see how not bringing the page from the disk back into memory hurts the hit rate of the overall system. Looking at the consistency of LFU algorithm, we can conclude that it is a preferred replacement algorithm over LRU.

Memory Size	100KB	1MB	10MB	100MB
LRU	2	8	16	24
LFU	0.002	1	6	11

Table 4: Cache Replacement Algorithm – Hit Rate Comparison

Considering the fact that the average time to retrieve a byte of data from a document on memory is far more than from the disk, is a strong justification for using LFU as a replacement algorithm. Another benefit from LFU is prefetching. Earlier in this paper, we saw how a popular document remains popular for several consecutive days. When such a popular document expires, the counter associated with the document can determine if a prefetch for the content makes sense and whether a GET-if-modified-since request is issued to the host server.

5.5 TCP Connections

A TCP connection is established using a three-way handshake. It takes several round trips to establish a TCP connection with the desired speed. Each connection introduces added latency and processing overhead. Keeping a connection open is seen to reduce the latency and processing overhead. However, an open connection will still consume socket and buffer space memory. The minimum size of the buffer needs to be bigger than the largest TCP packet. The number of available sockets is also limited.

A large number of open connections can have detrimental impact on server performance. Figure 6 shows the maximum number of open connections at any given time. We see very high values for the maximum number of open connections at any given time. It also shows that the maximum number of open connections increases with a rise in holding time from 0 min to 1 min and then from 1 min to 10 min. However, with an increase in holding time, the number of instances for a connection reduces and the same instance of connection serves a large number of requests.

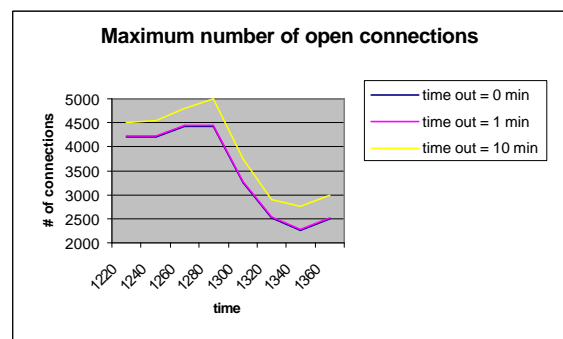


Fig. 6. Maximum number of open connections

6. Conclusion

We introduced a simple systematics for cooperative caching methods and tested the effectiveness of these methods, using off-the-shelf web traces with specific workload patterns. We analyzed the distribution of accesses based on various file types and content, a Zipf-governed URL request distribution, hit rates in different places in the hierarchy, and connection statistics. Among the various caching paradigms, we find that information-based systems are most efficient. Using cooperative caching in large client populations, hit rates increase up to 35%. Frequently requested documents typically remain popular over multiple consecutive days. The LRU algorithm being used in the NLANR caches limits the hit rate of the cache. We confirmed how a LFU algorithm has twice the hit rate of LRU and is hence a preferred replacement technique. Proxy and origin server analysis shows that a large number of open connections negatively affect overall performance. With increasing holding time, the number of instances for a connection reduces and the same instance of connection serves a large number of requests.

While cooperative caching is well established as a methodology for accelerated content distribution, analyses of the effectiveness of various schemes as a function of different traffic types will be of continued interest [19], in particular given the dynamic nature of Internet traffic patterns and the introduction of new models of data representation as with the semantic web [20]. No single strategy may be sufficient as a one-fit-all solution to perform universally well for all types of documents, nor do sampled caching patterns from one period accurately capture the breadth of real traffic flow. Per-document or per-media replication of caching through adaptive, pattern-driven cooperation is a promising direction for web caching. Future work will focus on more exhaustive caching pattern studies for dynamic cache reconfiguration and transcoding to adaptively fit content onto devices with diverse processing, networking and interface capabilities. Another important direction is the coupling of client and proxy caching with filtering to introduce more relevance into cache utilization.

References

- [1] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, World Wide Web: The information universe. *Electronic Networking Research, Applications and Policy*, 2(1):52-58, 1992.
- [2] V. Jacobson, How to Kill the Internet. Proc. *ACM SIGCOMM*, 1995.
- [3] M. Dahlin, R. Wang, T. E. Anderson, and D. A. Patterson, Cooperative Caching: Using Remote Client Memory to Improve File System Performance. *Operating Systems Design and Implementation*, 267-280, 1994.
- [4] G. Barish and K. Obraczka, World Wide Web caching: Trends and techniques. *IEEE Commun. Mag.*, 38(5):178-184, May 2000.
- [5] J. Wang, A Survey of Web Caching Schemes for the Internet. *ACM Computer Communication Review*, 1999.
- [6] NLANR/IRCache: A Distributed Testbed for National Information Provisioning. <http://ircache.net>.
- [7] M. Rabinovich, J. Chase and S. Gadde, Not All Hits Are Created Equal: Cooperative Proxy Caching Over a Wide-Area Network. *3rd Int. WWW Caching Workshop*, 1998.
- [8] S. Floyd, V. Jacobson, and L. Zhang, Adaptive web caching. *Proc. NLANR Web Cache Workshop*, 1997.
- [9] J. Chase, S. Gadde, and M. Rabinovich, Not all Hits are Created Equal: Cooperative Proxy Caching over a Wide Area Network. *Computer Networks and ISDN Systems*, 30(22-23):2253 - 2259, 1998.
- [10] R. Lancellotti, M. Colajanni, B. Ciciani, A Scalable Architecture for Cooperative Web Caching. Proc. of *Workshop in Web Engineering, Networking 2002*, Pisa, May 2002.
- [11] J. Gwertzman and M. Seltzer, The Case for Geographical Push Caching. *Proc. Workshop on Hot OS*, 1995.
- [12] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, A Hierarchical Internet Object Cache. *Proc. USENIX Tech. Conference*, 153-163, San Diego, CA, Jan.1996.
- [13] M. Dahlin, J. Kay, R. Tewari, and H. Vin, Design Considerations for Distributed Caching on the Internet. *Proc. ICDCS*, May 1999.
- [14] J. Almeida, A. Z. Broder, P. Cao, and L. Fan, Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. *IEEE Transactions on Networking*, 2000.
- [15] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy, On the scale and performance of cooperative Web proxy caching. *Operating Systems Review* 34(5):16-31, Dec. 1999.
- [16] L. Breslau and P. Cao, Web caching and Zipf-like distributions: Evidence and implications. *IEEE INFOCOM*, 1999.
- [17] S. D. Gribble and E. A. Brewer, System design issues for Internet middleware services: Deductions from a large client trace. *Usenix Symposium on Internet Technologies and Systems*, August 1997.
- [18] P. Krishnan and B. Sugla, Utility of Cooperating Web Proxy Caches. *Computer Networks and ISDN Systems*, 30(1-7), 195-203, 1998.
- [19] K.-W. Lee, K. Amiri, S. Sahu, and C. Venkatramani, On the sensitivity of cooperative caching performance to workload and network characteristic. *Proc. ACM SIGMETRICS*, 268-269, Marina Del Rey, 2002.
- [20] L. Brunie, J.-M. Pierson, and D. Coquil, Semantic collaborative web caching. *Proc. Web Information Systems Engineering (WISE'00)*, Dec. 2002, Singapore.