A Cash-based Approach to Caching Web Documents

Evangelos P. Markatos

Technical Report 230
Institute of Computer Science (ICS)
Foundation for Research & Technology – Hellas (FORTH)

P.O.Box 1385
Heraklio, Crete, GR-711-10 GREECE
tel: +30 81 391 655, fax: +30 81 391 661
markatos@csi.forth.gr

Abstract

As the spread on the World Wide Web increases at alarming rates, caches are placed at strategic places on the Internet to reduce network traffic and its associated costs. Although caches may reduce network traffic and communication costs, they may increase storage costs, sometimes disproportionately so. In this paper we propose a cost-based approach to World Wide Web caching. We view caching as an economic process: caching saves (makes) money by reducing telecommunication costs, while at the same time it costs money by requiring the purchase of caching storage. In this paper we show how to find the optimal cache size that maximizes the net profit of a cache. We show that the relative costs of storage and communication influence the optimal cache size: expensive communication pushes towards the use of larger caches, while expensive storage pushes towards smaller caches. Furthermore, we propose and evaluate an algorithm that finds the optimal cache size for any given communication/storage cost ratio and any given request stream.

1 Introduction

Recent results suggest that traffic on the World Wide Web (hereafter called Web) increases at alarming rates. Some estimates suggest that traffic increases by an order of magnitude each year. Even the most conservative estimates suggest that traffic at least doubles each year [7]. To reduce the amount of traffic traveling over the Web, caches can be (and usually are) placed at strategic places on the Internet. Such caches can easily absorb 40-50% of the requests [1] or even as much as 70% in some cases [12]. The effectiveness of caching depends mostly on the size of the cache, the replacement policy, and the access patterns followed by the cache clients. Several researchers have studied the performance of caches and have proposed efficient cache replacement policies [1, 4, 10, 12, 14, 16].

Although previous work has proposed policies that take advantage of a given cache configuration, little (if any at all) work has been done on finding what is an appropriate cache configuration that will result in cost-effective operation. [1] Currently there are few (if any) documented studies that will guide cache owners in deciding the most appropriate cache configuration. The lack of such guidelines has resulted in contradicting decisions by cache operators. Some cache operators use very small (100-200 Mbytes) cache sizes, while others use very large caches (10-20 Gbytes). Some researchers even propose that proxy caches should cache the entire (cacheable) web [5]. In this paper, we present a cost-based approach to finding the appropriate cache size which will lead to a cost-effective cache operation. Our approach tries to reduce the cost (in currency) of serving a stream of web requests. When using a cache, some of the requests will "hit" in the cache and will be served by the proxy, and the rest will "miss" the cache and will be sent to the appropriate server. Whenever a document is found in a client's/proxy's cache, a request operation to the server is saved, which directly translates to fewer data being transfered over long-distance lines, thus, less bandwidth consumed, and fewer long-distance charges incurred. However, to save on long-distance charges, a proxy must keep the document in a cache (usually a magnetic disk) which will result in storage costs, i.e. more disks need to be purchased. It is obvious that the more documents are kept in the cache, the lower the communication charges will be, but at the same time, storage costs will go up. Depending on the relative costs of storage and communication, different approaches must be followed by different proxies. If communication is significantly more expensive than storage, then keeping large (practically infinite) caches is reasonable, since they will reduce long-distance communication costs. On the other hand, if storage is expensive compared to communication, then medium-sized (or even small-sized) caches will result in reasonable costs. Thus, cache operators should calculate carefully the benefits and costs of each cache configuration and decide which is a appropriate for them. Unfortunately, such a calculation is not easy, since the optimal cache size does not depend only on the relative costs of communication and storage. It may also depend on the cache hit rate, which may also depend on the stream of requests. For example, if clients request small documents, then a small cache may be cost effective. If, on the other hand, clients request large documents (i.e. multimedia information), small caches would be inadequate. To make matters worse even the frequency of requests may change the optimal cache size. For example, busy web caches amortize their storage/operating costs more effectively, than less busy web servers. [2] Given that the the optimal cache size depends (not only) on all the above factors, it is practically impossible for a cache operator to determine the appropriate cache configuration.

[1]To draw an analogy from Virtual Memory, sophisticated page replacement policies make effective use of a *given* main memory size, while the *working set* principle defines the appropriate main memory size with which page replacement policies will perform best. If smaller (than the working set size) main memory sizes are used, sophisticated page replacement policies will still perform better than their counterparts, but the application performance will suffer (a phenomenon called thrashing) [9].

[2]To draw an analogy from real life, suppose that we have a store that sold 1,000 product units. If the units were sold within the same day, the store overheads will be small. If the units are sold within a month, the store will receive the same amount of money, but its operating expenses will be higher than previously, and thus its net profit will be lower.

In this paper, we present a cost-based algorithm that determines the optimal cache size taking into account all the above factors.

The rest of the paper is organized as follows. Section 2 places our work in the appropriate context and presents related work. Section 3 quantifies the tradeoff between communication and storage, presents a novel algorithm that finds the cache size the optimally balances this tradeoff and evaluates its effectiveness. Finally, section 4 concludes the paper.

2 Previous Work

Caching documents to reduce access latency is being extensively used on the web. Most web browsers cache documents in main memory or in local disk. Although this is the most widely used form of web caching, it is the least effective, since they rarely result in large hit rates [1]. To improve cache hit rates, caching proxies are used. Proxies employ large caches which they use to serve stream of requests coming from a large number of users. Since even large caches may eventually fill up, cache replacement policies have been the subject of intensive recent research. The first cache replacement policies were LRU (Least Recently Used) and its variations. LRU is based on the heuristic that the documents not used recently will probably not be requested in the near future. After studying the access patterns of several web clients, it was realized that small documents were requested more often than large documents. Thus, LRU was extended with heuristics that favor caching of small documents, by precluding caching of large documents [1, 14], or by eagerly removing large documents [1, 16, 20]. Given that clients experience different latency to different servers cache replacement policies may also take network latency into account, in order to avoid replacing documents that may take a lot of time to download [17, 21]. Some policies aggregate all the above factors (access recency, size, latency) into a "weight" or "value" and try to keep in cache the documents with the largest values [6, 12]. Some approaches propose the development of hierarchical caches that cooperate in order to provide a higher hoΦit rate capitalizing on a larger number of clients and a larger cumulative amount of storage [8, 13]. Finally, some caches employ intelligent *prefetching* methods to improve the hit rate even further [3, 2, 15, 11, 19, 18].

We view our work as complimentary to previous approaches. While most of the previous research has focused on finding which documents to replace from a full cache, we focus on finding how large should the cache be in order to result in minimum cost (measured in currency). Our approach works with any of the replacement policies proposed.

It may seem that our approach is at odds with system performance, because we strive to achieve a cost-based balance between local storage and remote communication. This balance may result in lower hit-rates than other approaches that use very large caches, and thus although our approach may result in optimum cost, it may also result in sub-optimum performance. However, we do not believe that our approach results in noticeable performance degradation:

- As we will see in the next section, our approach results in reasonably large caches (for most practical values) and reasonable high hit rates.

- The cost of serving a document from a proxy cache is usually not significantly smaller than the cost of serving the document from the original server. Serving a document from a proxy (disk) cache located in the same city/state with the client may take several tens (or even hundreds) of milliseconds, which is in most cases comparable (within the same order of magnitude) to serving the document from the original server. Thus, small differences in cache hit rates will usually not result in significant performance differences. [3]

3 Experiments

3.1 The traces

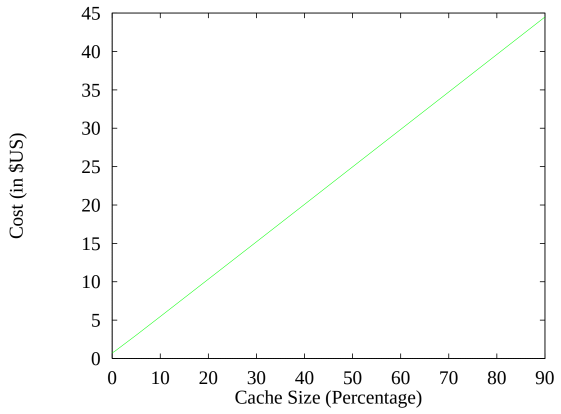
To evaluate our approach we use traces taken from a proxy at the UC Berkeley (http://www.cs.berkeley.edu/gribble/traces/index.html) and at Digital (ftp://ftp.digital.com/pub/DEC/traces/ proxy/webtraces.html). Each trace includes 5 million requests, which represents the proxy activity over several days.

3.2 The Cost of Caching

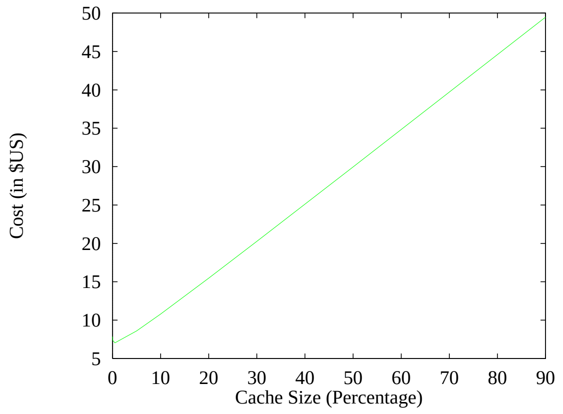
In our first set of experiments we set out to demonstrate the tradeoff between communication and storage in web proxy caches. To do so, we need to use specific values for communication and storage costs. Unfortunately, these costs vary with time and place. For example, a T1 line in the USA costs around $1,000.00 per month. The same line in Europe may easily cost 2-3 times more, and in Australia/New Zealand may cost an order of magnitude higher. Similarly, the cost of storage depends on the actual packaging of the magnetic disks and usually drops with time. However, reasonable prices for storage are between $100.00 and $200.00 per Gbyte. As a starting point, we will assume that the cost of storage (magnetic disk) is about $160.00 per Gbyte, and that the average life of a disk is about two years, which implies that the cost of storing 1 Gbyte of information for 1 hour is about 1 cent. Communication cost calculation is trickier. According to published prices, a T1 line carrying 1.5 Mbit per second costs around $1,000.00 per month Working at full speed, a T1 line can download a little less than 500 Gbytes per month, which implies that the cost of downloading 1 Gbyte is close to $2.00 . Although it can be convincingly argued that the actual communication and storage costs vary from the typical values we calculated, we believe that our calculated costs are close to the actual costs incurred. However, to make sure that we evaluate our methodology over a wide range of cost values we use four different cost sets where the relative cost of communication to storage is varied by three orders of magnitude (end-to-end). That is, we fixed the cost of storing 1 Gbyte for 1 hour to 1 cent, and varied the cost of downloading 1 Gbyte from 2 cents to 2,000.00 cents. [4]

[3]Small differences in hit rates may result in significant performance improvement in environments where cache misses cost several orders of magnitude more than cache hits. For example, in virtual memory systems hits (main memory accesses) cost 4-6 orders of magnitude less than misses (disk accesses).

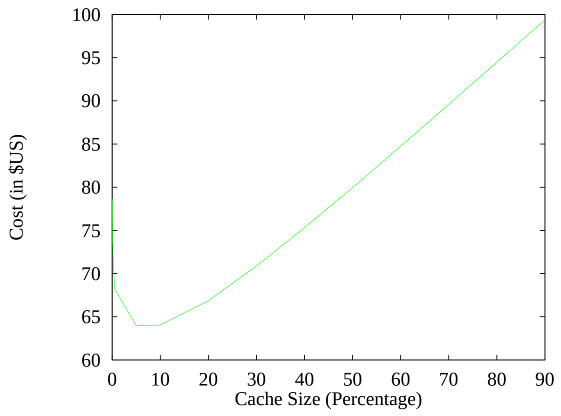
[4]the interested reader should note that actual costs of storage and communication do not alter the results as long as their ratio remains constant. That is, if a given cache size in optimal for a given set of communication and storage cost values, the same cache size would be optimal if *both* the communication and storage cost values were doubled.



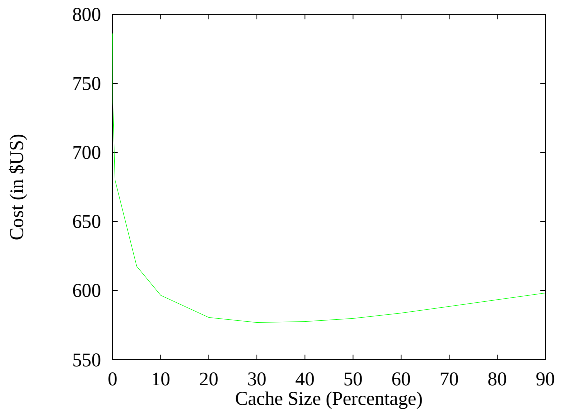
(a)



(b)



(c)



(d)

Figure	Cost of Storing 1 GByte for 1 hour (in US cents)	Cost of downloading 1 GByte (in US cents)
(a)	1	2
(b)	1	20
(c)	1	200
(d)	1	2000

Figure 1: **Cost of delivering the requested URLs to clients.** The various figures represent a different balance between communication and storage costs. We see that when communication costs are low (as in (a) and (b) caching data increases the cost. However, when communication costs are medium to high (as in (c) and (d)), caching data reduces the cost (up to some optimal cache size).

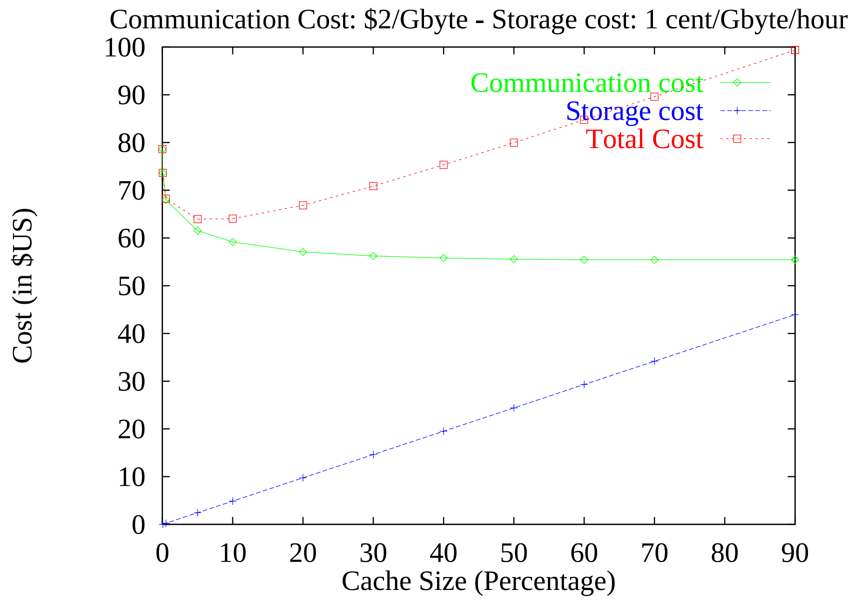


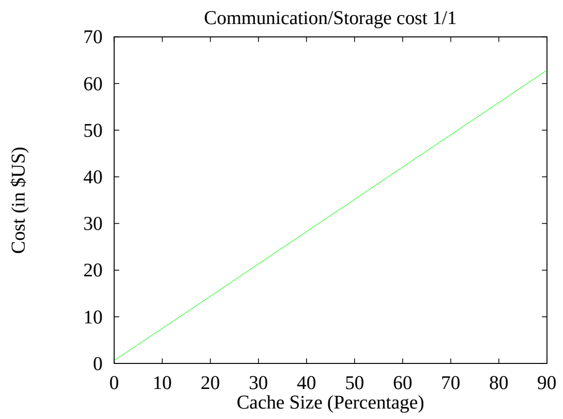
Figure 2: **Cost of delivering the requested URLs to clients.** The total cost is further subdivided into communication cost (that is the cost of downloading documents) and storage cost (that is the cost of storing documents in a cache). We see that when the cache is very small ($< 1\%$), the communication costs decreases sharply, while the storage cost is hardly noticeable. However, when the cache exceeds 10%, then the storage costs become high, while the communication costs improve very little (since the hit rate does not improve significantly).

Figure 1 shows the cost of serving the DEC trace as a function of the cache size for four different cost value sets. When communication is inexpensive compared to storage (as in (a) and (b)), the cost increases (almost) linearly with cache size. However, when the cost of communication is increased to the values currently charged by ISPs (figure 1(c)), then the tradeoff between communication and storage is clear. When the cache is very small, the communication cost dominates. When the cache is very large, the storage cost dominates. The cache size that results in minimum cost is around 2 Gbytes (10%). Similarly, when the cost of communication is even higher (figure 1(d)), the cache size that results in minimum cost is around 6 Gbytes (30%).

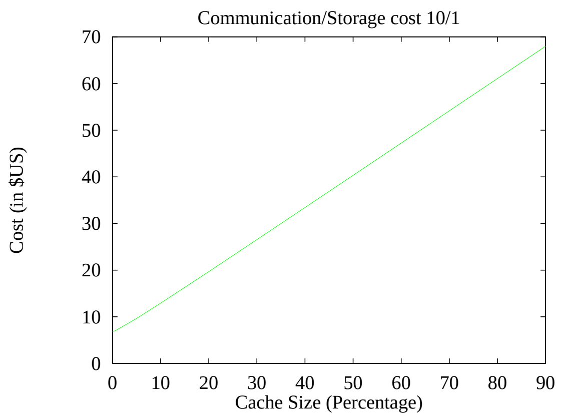
Figure 2 shows the communication and storage costs for the experiment shown in Figure 1(c). As the cache size increases, the communication cost drops sharply at the beginning. As the cache size exceeds 20%, there is no noticeable decrease anymore. On the contrary, storage costs rise linearly with cache size, since the cost of storage depends only on its size and not on the hit-rate of the cache. Therefore, for small cache sizes, the total cost is dominated by the communication cost, while for large cache sizes, it increases linearly (just like the storage cost).

Fgiure 3 shows the cost of serving the UCB traces. The results are qualitatively similar to those shown in figure 1.

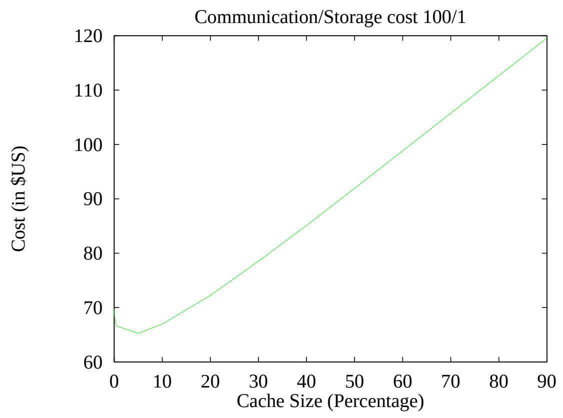
UCB traces



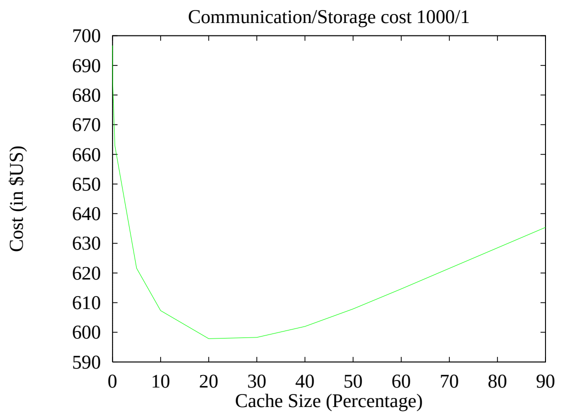
(a)



(b)



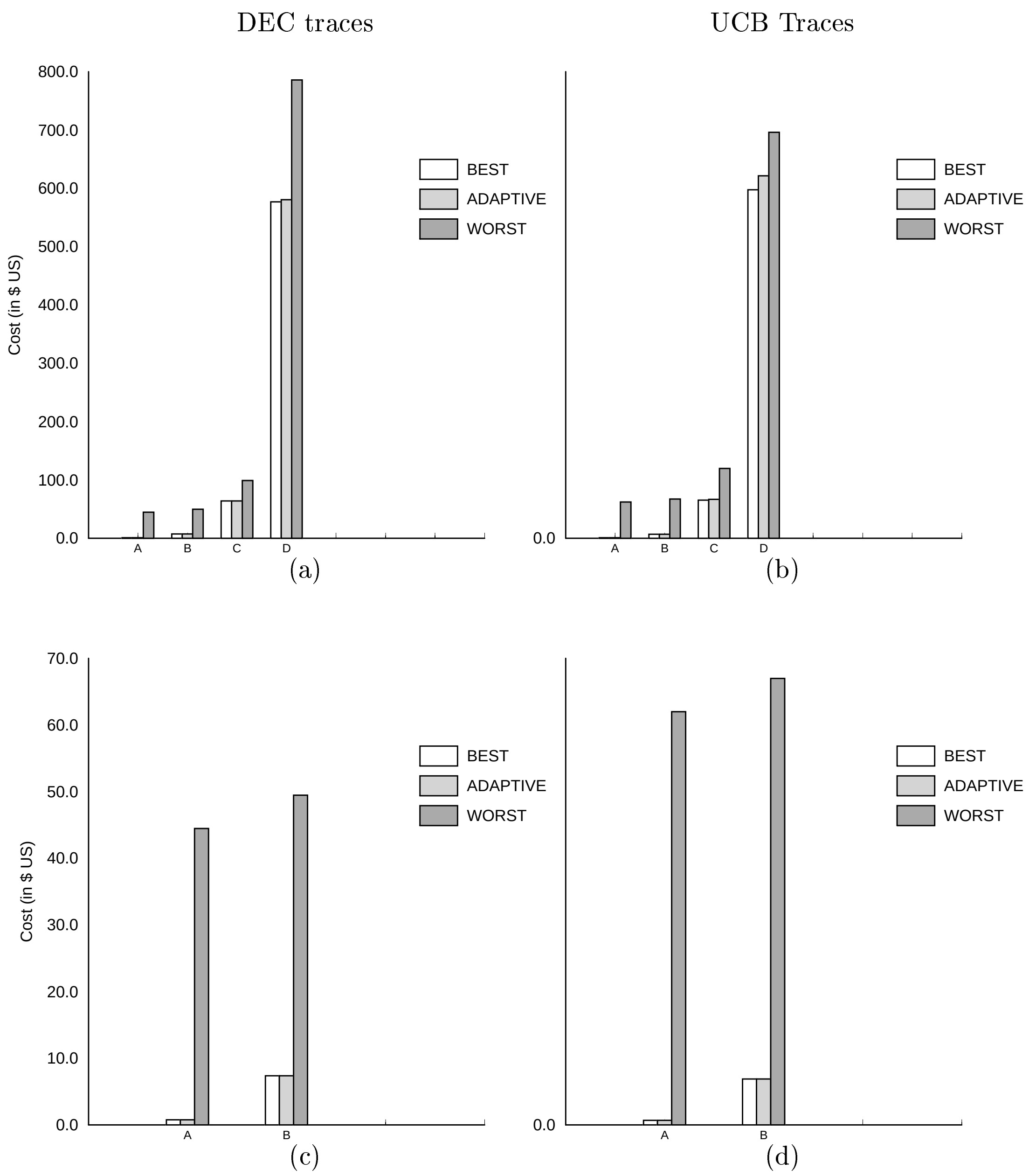
(c)



(d)

Figure	Cost of Storing 1 GByte for 1 hour (in US cents)	Cost of downloading 1 GByte (in US cents)
(a)	1	2
(b)	1	20
(c)	1	200
(d)	1	2000

Figure 3: **Cost of delivering the requested URLs to clients.** The various figures represent a different balance between communication and storage costs. We see that when communication costs are low (as in (a) and (b) caching data increases the cost. However, when communication costs are medium to high (as in (c) and (d)), caching data reduces the cost (up to some optimal cache size).



Configuration	Cost of Storing 1 GByte for 1 hour (in US cents)	Cost of downloading 1 GByte (in US cents)
A	1	2
B	1	20
C	1	200
D	1	200

Figure 4: **Performance of the adaptive policy.** Both in DEC traces (a) and in UCB traces (b) our ADAPTIVE policy results in practically the same cost as the OPTIMAL cache size. Figures (c) and (d) zoom in the results reported for configurations A and B.

3.3 Optimal Cache Size Calculation

Our experiments so far suggest that there exists a tradeoff between communication and storage, and that there exists a cache size that balances this tradeoff to achieve the minimum cost. We will now propose and evaluate an algorithm that dynamically finds the optimal cache size. The algorithm continually increases/decreases the cache size, measures the cost achieved by these changes, and based on that cost keeps changing the cache size towards the optimal value. That is, if we increase (decrease) the cache size and the cost keeps getting small, we continue increasing (decreasing) it until the cost increases. At that point, we start decreasing (increasing) the cache size and continue to change the cache size in the same direction so as long as the cost keeps getting smaller.

Our algorithm works as follows:

- We define a set of discrete cache sizes that the algorithm will use. In our experiments we calculate the cache size as a percentage of the maximum cache size needed by the traces. The percentages used are: 0.005%, 0.05%, 0.5%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70% and 90%.

- We start the execution of the algorithm at 0.005%.

- Periodically (every 50,000 references in our experiments) we measure the cost of serving the entire request stream so far. If the cost (per request) is lower than what it was during the previous period we keep changing the cache size to the same direction. That is, if were increasing the cache size, we keep increasing it. If the cost (per request) is higher than what it was during the previous period we reverse the direction of change. That is, if were increasing the cache size, we start decreasing it.

Figure 4 shows the performance of our algorithm for the four system configurations (A-D). Figure 4(a) shows the results for the DEC traces and 4(b) shows the results for the UCB traces. We see that in all cases our adaptive algorithm performs close or exactly the same as the best. Figures 4(c) and 4(a) show the same information as (a) and (b), but magnified to show the details for configurations A and B. Both in DEC traces and in UCB traces our ADAPTIVE policy results in practically the same cost as the OPTIMAL cache size.

To study the sensitivity of the optimal cache size in the length of the trace used, we calculated the optimal cache size for traces of varying length ranging from 1 day to 2 weeks (corresponding roughly from 1 to 14 million URL requests). The optimal cache size as a function of the trace length is shown in figure 5. We see that the optimal cache size for the DEC traces is in the range of 1.5 to 2.5 Gbytes. It is surprising to see that even at the end of the very first day the optimal cache size has reached a high value (1.5 Gbytes) suggesting that calculating the optimal cache size can be done even from small traces. As the length of the trace increases the optimal cache size increases as well, although rather slowly. During the last week the optimal cache size stays practically constant, suggesting that decisions made about the optimal cache size of a proxy server will stay valid for several days or even weeks. Figure 6 shows the optimal

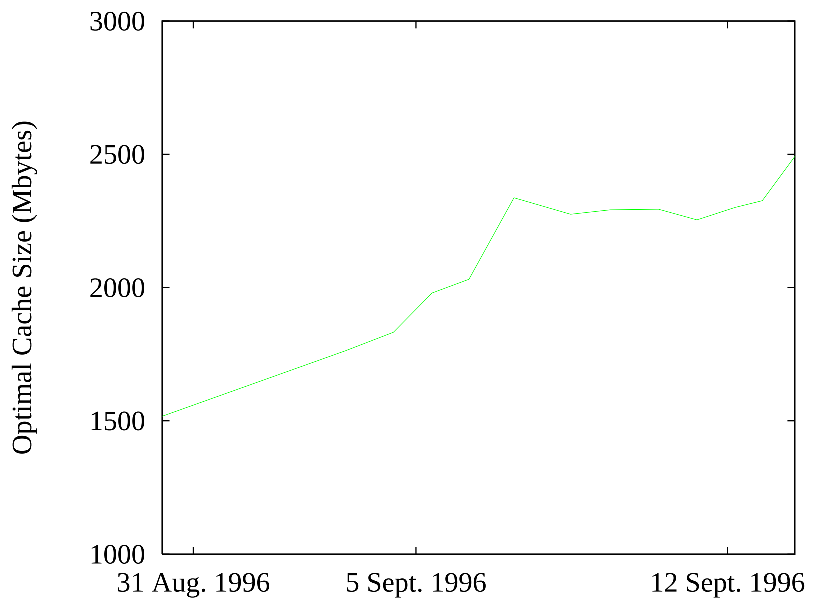


Figure 5: **Optimal Cache Size as a function of the trace size simulated.** We see that the optimal cache size remains stable over several days, or even weeks. As soon as the first day, the optimal cache size reaches up to 1.5 Gbytes. Over the next few days, the optimal cache size grows slowly, reaching a stable range of 2-2.5 Gbytes.

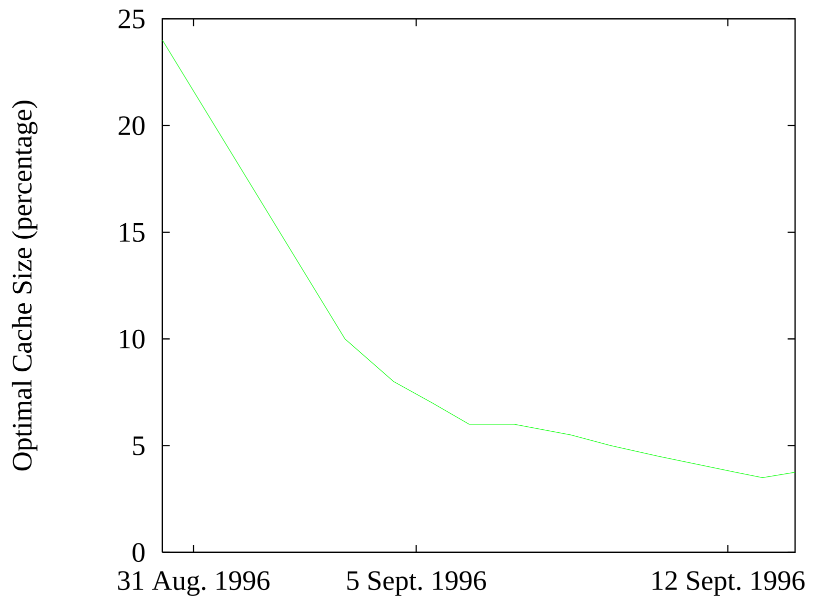


Figure 6: **Optimal Cache Size as a function of the trace size simulated.** The Cache Size is given as a percentage of the max. cache size needed to hold the entire trace and thus have the maximum hit rate.

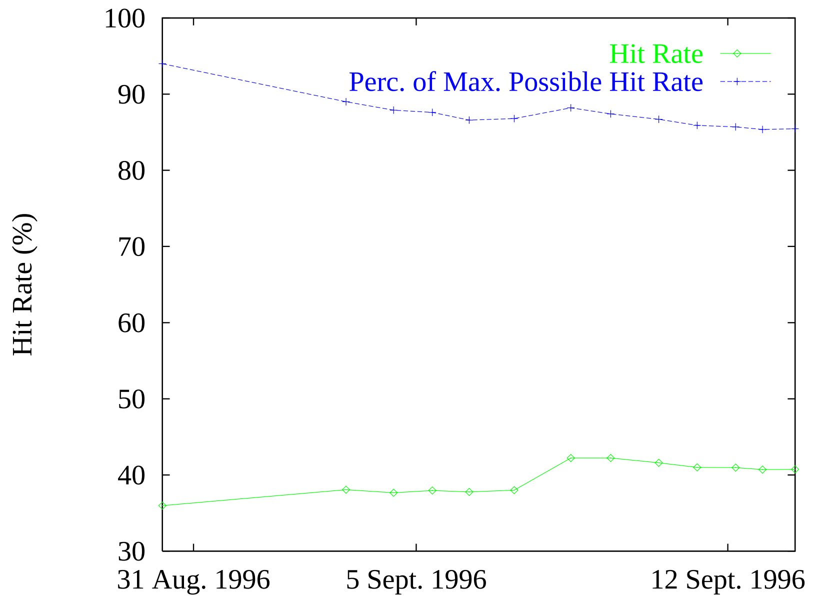


Figure 7: **URL Hit-Rate achieved for the optimal cache size.** The lower curve shows the (absolute) URL hit-rate. We see that during the two-week period it stays in the range of 37%-42%. The upper curve shows the URL hit-rate as the percentage of the maximum URL hit-rate that can be achieved by the maximum-sized cache. We see that the optimal-cost cache size achieves 85%-95% of the max. hit-rate possible, which is a very good ratio considering that it is achieved at the lowest possible cost (in $).

cache size (same as in figure 5) expressed as a percentage of the maximum cache size needed to hold the entire trace used, and achieve the best possible hit rate.

To see if the optimal cache size (calculated in $) delivers a good hit rate figure 7 shows the hit rate achieved by the optimal cache size as a function of the trace length. The lower curve shows the (absolute) URL hit-rate. We see that during the two-week period it stays in the range of 37%-42%. The upper curve shows the URL hit-rate as the percentage of the maximum URL hit-rate that can be achieved by the maximum-sized cache. We see that the optimal-cost cache size achieves 85%-95% of the max. hit-rate possible. In other words, less than 2.5 Gbytes of cache are enough to achieve more than 85% of the maximum hit-rate that can only be achieved by a cache more than 20 times larger. That is, having a cache 20 times larger than 2.5 Gbytes would improve performance by only 15%, while it would drive storage costs more than one order of magnitude higher.

4 Conclusions

In this paper, we present a cost-based approach to finding the appropriate cache size which will lead to a cost-effective cache operation. Our approach tries to reduce the cost (in currency) of serving a stream of web requests. Different Telecom Operators charge substantially different rates for their long-distance traffic. The tradeoff between telecommunication charges and storage costs may lead to different optimal proxy cache configurations. For example, when telecommunication charges are high, employing very large proxy caches makes sense (in order to reduce cost). On the other hand, when telecommunication charges are low, even small proxy caches may achieve a near-optimal performance. We propose an ADAPTIVE algorithm which approximates the optimal cache size for any given communication-to-storage ratio and any given request stream. We use trace driven simulation to evaluate our approach. Based on our simulations we conclude:

- Our ADAPTIVE algorithm effectively approximates the optimal cache size as shown in figure 4.
- The optimal cache size increases (very) slowly with the duration of the cache operation. For example, in a two-week operation based on the DEC's proxy traces, the optimal cache size increased from 1.5 Gbytes to 2.5 Gbytes (for a request stream totaling more than 75 Gbytes of unique data).
- Cache configurations that do not take into account the communication-to-storage cost ratio are likely to perform at a larger cost than what they could.

Acknowledgments

This work was supported in part by PENED project "Exploitation of idle memory in a workstation cluster" (2041 2270/1-2-95). We deeply appreciate this financial support.

Some of the experiments were run at the Center for High-Performance Computing of the University of Crete (http://www.csd.uch.gr/hpcc/). Pei Cao provided the inittal version of the simulator used in this paper. Digital quipment Corporation and the University of the California at Berkeley provided the traces used. We thank all of them.

References

[1] M. Abrams, C.R.Standridge, G. Abdulla, S. Williams, and E.A. Fox. Caching Proxies: Limitations and Potentials. In *Proceedings of the Fourth International WWW Conference*, 1995.

[2] Azer Bestavros. Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems. In *Proceedings of ICDE'96: The 1996 International Conference on Data Engineering*, March 1996.

[3] Azer Bestavros. Using speculation to reduce server load and service time on the WWW. In *roceedings of CIKM'95: The Fourth ACM International Conference on Information and Knowledge Management*, November 1995.

[4] P.M.E. De Bra and R.D.J. Post. Information Retrieval in the World-Wide Web: Making Client-based searching feasible. In *Proceedings of the First International WWW Conference*, May 1994.

[5] Pei Cao. Disks Solve Web Scalability Problems, 1997. Panel Presentation, ICDCS 97.

[6] Pei Cao and Sandy Irani. Cost-Aware WWW Proxy Caching Algorithms. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[7] Viton Cerf. Keynote Address at INET 98, 1998.

[8] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A Hierarchical Internet Object Cache. Technical Report 95-611, Computer Science Department, University of Southern California, Los Angeles, California, March 1995.

[9] P. J. Denning. The Working Set Model for Program Behavior. *Communications of the ACM*, 11(5):323–333, May 1968.

[10] Fred Douglis, Antonio Haro, and Michael Rabinovich. HPP: HTML Macro-Preprocessing to Support Dynamic Document Caching. In *USENIX Symposium on Internet Technologies and Systems*, pages 83–94, 1997.

[11] J. Gwertzman and M. Seltzer. The Case for Geographical Pushcaching. In *Proceedings of the 1995 Workshop on Hot Operating Systems*, 1995.

[12] P. Lorenzetti, L. Rizzo, and L. Vicisano. Replacement Policies for a Proxy Cache, 1998. http://www.iet.unipi.it/ luigi/research.html.

[13] Radhika Malpani, Jacob Lorch, and David Berge. Making World Wide Web Caching Servers cooperate. In *Proceedings of the Fourth International WWW Conference*, 1995.

[14] E.P. Markatos. Main Memory Caching of Web Documents. *Computer Networks and ISDN Systems*, 28(7-11):893–906, 1996.

[15] E.P. Markatos and C. Chronaki. A Top-10 Approach to Prefetching on the Web. In *Proceedings of the INET 98 Conference*, 1998.

[16] J.E. Pitkow and M. Recker. A Simple, Yet Robust Caching Algorithm Based on Dynamic Access Patterns. In *Proceedings of the Second International WWW Conference*, 1994.

[17] P. Scheuearmann, J. Shim, and R. Vingralek. A Case for Delay-Conscious Caching of Web Documents. In *6th International World Wide Web Conference*, 1997.

[18] Joe Touch. Defining High Speed Protocols : Five Challenges and an Example That Survives the Challenges. *IEEE JSAC*, 13(5):828–835, June 1995.

[19] Stuart Wachsberg, Thomas Kunz, and Johnny Wong. Fast World-Wide Web Browsing Over Low-Bandwidth Links, 1996. http://ccnga.uwaterloo.ca/ sbwachsb/paper.html.

[20] S. Williams, M. Abrams, C.R. Standbridge, G. Abdulla, and E.A. Fox. Removal Policies in Network Caches for World-Wide Web Documents. In *Proc. of the ACM SIGCOMM 96*, 1996.

[21] Roland P. Wooster and Marc Abrams. Proxy Caching that Estimates Page Load Delays. In *6th International World Wide Web Conference*, 1997.