Towards Efficient Web Proxy Caching Algorithms Under Self-Similar Traffic Loads

Pu Huang

February 6, 2001

Web proxy caching has been considered as one of the most important techniques to reduce Web traffic, save bandwidth, and lower network latency times. Both commercial corporations and academic communities have shown a great deal of enthusiasm on proxy caching techniques. In recent years, studies in Web traffic measurement have provided convincing empirical evidence that Web traffic exhibits long-time scale self-similar pattern. However, what this pattern means to proxy caching remains unclear. In this paper, we plan to investigate the proxy performance under the self-similar traffic load through simulations. We hope our study will lead to a more efficient proxy caching algorithm, which takes the Web traffic pattern into consideration and tries to minimize the practically concerned metrics, such as user experienced latency times.

1 Introduction

A Web proxy (proxy, for short) is an application level software, usually runs on a dedicated computer (proxy server) standing at the boundary of an organization's subnets, providing Web access to the end-users inside the subnets. Every time an user sends out a HTTP request, the proxy server first intercepts it and then forwards it to the destination Web server, waits for the response from the destination Web server then sent the response back to the user. Initially designed to provide Internet access to users inside an organization's Intranet, now proxies are widely deployed also because of their ability to cache previously requested documents: it makes logical sense to cache previously requested documents in a proxy to reduce the network traffic. Enormous studies have proved that, if used properly, proxy caching can reduce the network traffic significantly[40]. In fact, proxy caching has been considered as one of the most important techniques to reduce the Web traffic, save bandwidth, balance server load, and lower network latency time. Both commercial corporations and academic communities have shown a great deal of enthusiasm on proxy caching techniques. At least 50 commercial products are currently available[35]; and various aspects of proxy caching have been extensively examined by researchers, including caching algorithms [2][7][29][40][47], trace analysis and model estimations[1][3][5][14][17][32], performance evaluations [10][15][27][23], and system design and implementation issues [6][21][36][39]. In recent year, researchers also begin to pay attention to more complicated proxy mechanisms, such as cooperative proxy caching, dynamic object caching, hierarchy and scalable caching structure, and issues of maintaining strong cache consistency[8][13][20][26][28][46].

However, literature of proxy caching seems to ignore the parallel development in the network traffic research, namely, the finding of self-similar network traffic pattern. Self-similar network traffic pattern means that the measured network traffic exhibits nearly unchanged statistic characteristics in a large scale of time range. Pioneer work of self-similar network traffic begins with Leland et al. [24] [25], where they show that the observed Ethernet LAN traffic exhibits long-range dependence and large time-scale self-similarity. Subsequent works provide convincing empirical evidence that self-similar traffic patterns are prevalent in today's network infrastructure [4][12][22][34]. In a series of papers, Willigner et al. [41][42][43][44] provide more insights into the reason why this is the case by showing that the superposition of traffic between many source/destination pairs exhibits self-similar pattern if the traffic in each pair is generated by a ON/OFF source (the source alternatively generates traffic and falls to sleep) and the time lengths of the ON-period and the OFF-period follow heavy-tailed distributions. A bibliographical guide to self-similar traffic research can be found in Willigner et al. [45], where they provide hundreds of references. With the explosive growth of World Wide Web, Web traffic begins to catch researchers' attention in recent years. Crovella and Bestavros [9] show that Web traffic does exhibit self-similar pattern when the workload is high[1]. Using the model developed by Willigner et al. [44]. They also empirically show that the self-similar Web traffic can be well attributed to two factors: heavy-tailed Web document distribution and heavy-tailed user thinking time distribution (also see [11] [19] [33]). Mahanti et al. examine Web traffic in a proxy hierarchy and finds similar results[32].

Intuitively, traffic information should be useful to proxy caching, but how exactly it can be used remains unclear. In this paper, we investigate this problem. The main objective of our study is to understand what does the observed Web traffic pattern, namely, self-similarity, means to proxies, how does this traffic pattern interact with proxy caching, and consequently proxy performance.

[1]They do not observe self-similar Web traffic in less-busy hours.

2 Proposed Effort

We consider the proxy caching problem in the following scenario. A proxy server is deployed to provide Web assess to a large group of end-users. Given limited outbound bandwidth available to this proxy to access the Internet, we want to find an "optimal" caching policy that can maximize the utility of bandwidth and at the same minimize the user experienced latency times.

Three main metrics are used to measure the effectiveness of proxy caching: hit ratio, weighted hit ratio, and latency time. Among these metrics, hit ratio measures the percentage of requests that produces cache hits to all requests; weighted hit ratio is the hit ratio weighted with the size of hit documents. These two metrics are widely used in performance evaluations of proxy caching. However, we argue that latency time should be viewed as a more relevant metric of caching performance. Bottleneck of today's distributed computation environments, such as the Internet, is always the networks connecting individual computation nodes. Therefore, network bandwidth is an expensive resource. By using a proxy, a group of users can share the bandwidth resource. From the point of view of a service provider (ISP, cooperation, universities), latency times experienced by its users is a practical and crucial metric to measure the quality of its service. Thus, a service provider hopes to minimize latency times, of course, under the constraint of how much outbound bandwidth it processes. Therefore, proxy caching mechanisms should be designed to facilitate service providers to achieve this goal. Hit ratio and weighted hit ratio are indirect metrics of users' satisfaction. They are widely used partly because they are easy to obtain, and partly because they are reasonable metrics when considering main memory caching problems, from which proxy caching algorithms stem. To illustrate the problem of only using hit ratio or weight hit ratio as the performance measure of proxy caching, we point to Liu and Fox's study on Web traffic and latency [27], where they find a single proxy cache almost always *increases* latency times, while a majority of studies have reported at least 30% cache hit ratio. Although reasons behind this unusual phenomenon are not so simple as using different measurement metrics, it does, at least partially, reveal the mismatch of academical studies and practical concerns.

To communicate our proposed efforts more accurately and effectively, we first give some notations. We consider a subnet which has m end users. These m users share a proxy server which provide them Web access. Suppose the proxy server processes a limit outbound bandwidth B bytes/second and a cache with size C bytes. Our problem is to find a cache replacement policy which minimizes the total *expected* latency times of all the end-users. One factor effecting latency times is the sizes of documents requested. As show in [9], we can model the size distribution of all the documents in the Web using a heavy-tailed distribution. Thus, we assume there are

n documents in the Web and size distribution of these documents follows a Pareto distribution (see below). We also assume the user's thinking time, the time between the user sending out two requests, follows a Pareto distribution. As shown in [44], these two heavy-tailed distributions, are major contributors to self-similar Web traffic.

We plan to use simulations to find the efficient caching algorithms. The first step of our work is to develop a traffic synthesizer based on the result of previous studies, specifically [9][11]. Our synthesizer will first analyze real Web log data, extract important features and then generate traffic data according to these features.

Once we get the synthesizer, we can run simulations to evaluate the performance of different caching algorithms under various traffic loads. The results will lead us to more efficient proxy caching algorithms which account traffic loads. Here we plan to measure the traffic loads by using the the $Hurst$ parameter H, which is a parameter measuring the degree of self-similarity (see below). Both empirical and theoretical studies have shown that the heavier traffic loads are, the more self-similar the traffic is.

We plan to use Web Polygraph as the simulation test-bed. Web Polygraph is a proxy caching simulation tool developed by IRCache Project, and freely available. However, Web Polygraph does not support self-similar traffic generator. This part will be replaced by our own traffic synthesizer.

In the process of this study, we'll also examine a close-related problem: can a caching algorithm adapts itself in different traffic situation automatically? If we can find an incremental way to estimate various model parameters, we may achieve this goal.

3 Background

In this section, we provide some background on the self-similar network traffic.

3.1 Heavy-Tailed Distribution

A probability distribution function $F(x) = P\{X < x\}$ is said to be heavy-tailed if its tail approximates a power function,

$$P\{X > x\} \sim x^{-\beta}, \qquad as\ x \to \infty, \tag{1}$$

where $\beta > 0$ is a constant. The main difference between heavy-tailed distribution and other widely used distributions is that the tail of the former converges to 0 much slower than the later does. For example, the tail of an exponential distribution with mean $1/\lambda$ is

$$P\{X > x\} = e^{-\lambda x},$$

which means this tail converges to 0 at an exponential rate $e^{-\lambda x}$. Compared with power converging rate $x^{-\beta}$, an exponential rate is much faster. Give an example, when $\beta = 2$ and $\lambda = 1$, the probability of random variable X to be greater than 10 is about 0.01 in (1); while it is about about 4.54×10^{-5} in the exponential distribution, which is about 220 times lesser than that in (1). With numbers more greater than 10, the difference increases more fast, in fact, exponentially fast. That is the reason why this family of distributions are named as heavy-tailed: they have long tails and thus the probability of some usual events, such as a big X shows up, is much higher than that in other widely used distributions, such as the exponential family.

One classic example of heavy-tailed distribution is Pareto distribution. By definition, a Pareto distribution has the form

$$F(x) = P\{X < x\} = 1 - (a/x)^{\beta},$$

Where $\beta > 0$ and $x > a$. As we can see, its tail has the form

$$a^{\beta} x^{-\beta},$$

which matches (1). The probability density function of Pareto distribution is

$$f(x) = \beta a^{\beta} x^{-\beta-1}.$$

3.2 Self-Similarity and Long-Range Dependence

Consider a stochastic process $\{X_t\}$ with constant mean $\mu = E[X_t]$ and finite variance $\sigma^2 = E[(X_t - \mu)^2]$. We assume the autocorrelation function of this process is *stationary,* i.e.,

$$r(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)}{\sigma^2},$$

where $k = 1, 2, 3, ...$, is only a function of lag parameter k. If as k goes to infinity, the autocorrelation function follows a power law, i.e. has the form

$$r(k) \sim k^{-\beta}, \qquad as\ k \to \infty, \tag{2}$$

where $0 < \beta < 1$, then we say process $\{X_t\}$ is long-range dependent. Intuitively, power law $k^{-\beta}$ means that autocorrelation between X_t and X_{t+k} decreases slowly as k increase, i.e., events in a long time future are dependent on the current event and this dependence decreases by following a power law.

Now, consider an aggregated process $\{X_t^{(m)}\}$ constructed from $\{X_t\}$ by averaging the series $\{X_t\}$ over non-overlapping blocks of size m, i.e., using the following formula

$$X_t^{(m)} = \frac{X_{tm-m+1} + ... + X_{tm}}{m},$$

where $t = 1, 2, 3........$ Since autocorrelation function $r(k)$ of original process $\{X_t\}$ is stationary, the autocorrelation function $r^{(m)}(k)$ of the derivative process $\{X_t^{(m)}\}$ is also stationary. By self-similarity, we mean that

$$r^{(m)}(\cdot) = r(\cdot),$$

for all $m = 1, 2, ...$ If this is the case, then in any time scale order m, the process $\{X_t\}$ just looks the same as its m order average $\{X_t^{(m)}\}$.

Fractional Gaussian noise $\{X_t\}$ is a widely used self-similar process. It has a constant mean μ and a finite variance σ^2. Its autocorrelation function is

$$r(k) = \frac{(k+1)^{2H} - k^{2H} + (k-1)^{2H}}{2},$$

where $k = 1, 2, 3.......$ and H is the $Hurst$ parameter. It can be shown that as k goes to infinity,

$$r(k) \sim H(2H-1)k^{2H-2}.$$

Compared with (2), we can see that

$$H = 1 - \frac{\beta}{2}.$$

For $0 < \beta < 1$, we have $\frac{1}{2} < H < 1$. Also, it can be shown that the aggregate process $\{X_t^{(m)}\}$ has an exactly same autocorrelation function as $\{X_t\}$. As we can see, Hurst parameter H is the only parameter in the autocorrelation function of Fractional Gaussian noise and self-similarity is defined in terms of autocorrelation function $r(\cdot)$, thus, H is also a measure metric of the degree of self-similarity. As $H \to 1$, both the degree of self-similarity and the long-range dependence of Fractional Gaussian noise increase. As $H \to \frac{1}{2}$, Fractional Gaussian noise degenerates to ordinary Brownian Motion.

Fractional Gaussian noise is an appealing model for network traffic mainly because of two reasons: first, it is simple and easy to characterize, only one parameter H needs state; second, it matches with empirical data reasonably well.

Using the method first developed by Mandelbrot [30][31], then extended by Taqqu and Levy [38], Millinger et al. [44] shows that number of packets passing the networks in any given period is a Fractional Gaussian noise, providing there are a large group of packet sources. This model subsequently is used by Crovella and Bestavros to model the Web traffic, where the number of packets is replaced by the number of URL requests. However, Crovella and Bestavros do find that the Web traffic is not self-similar when the workload is not high. Thus, we directly model the document size distribution and user thinking time distribution. Through simulation, we can examine under what condition, the aggregation of many users' requests will form a self-similar Web traffic.

4 Related Work

The simplest, while most widely used the algorithm is Least-Recently -Used (LRU) algorithm, which evicts the document which was requested the least recently. Least-Frequently-Used (LFU) evicts the documents which is accessed least frequently. These two algorithms have their roots in early studies of operating systems. Since the proxy caching and the main memory caching are not exactly the same problems, various algorithms taking consideration of special characteristics of proxy caching are proposed. A summary and comparison of these algorithms can be found in [7]. Also in [7], Cao and Irani show that their GreedyDualSize algorithm outperforms all other algorithms in many aspects, including hit ratios, latency and network cost reductions. The main idea of this algorithm is to associate a value with each cached page. Every time when a replacement is necessary, the page with the minimal value will be replaced. When assigning a value to a page, the cost of bring this page to the cache is evaluated. With different goal, the cost is evaluated in a different way, for example, if the goal is to maximize the hit ratio, the cost is simply set to 1 every time; if the goal is to minimize average latency, the cost is set to the download latency. This algorithm takes the document size into consideration by setting the final value of each page to be *cost/size.* Surprisedly, GD-Size(1), which does not take latency into account, outperforms other GreedyDualSize algorithms which take latency into consideration. The authors' conclusion of this unexpected result is that the high variance among loading latencies for the same documents reduces the effectiveness of latency-conscious algorithms.

Liu and Fox's work on real-world Web traffic measurements shows that proxy caching "almost always increases response time" [27]. They suggest that this may be due to badly designed HTTP protocol. One of their other findings is that the Web response time is highly correlated with Web traffic, which is consistent with Cao and Irani's finding of the high variance among loading latencies for the same documents, which is the reason why GD-Size(1) outperforms other latency-aware algorithms.

All these studies have clearly shown that traffic is a crucial factor effecting proxy caching performance. But how traffic loads effect the caching performance has not been well understand.

5 Work Plan

1. Feb. 21, (1 week) Analyze the structure of test bed Web Polygraph, figure out how to integrate self-similar synthesizer.
2. Feb. 28, (2 weeks) Web log analysis, develop and validate synthesizer.
3. Mar. 15, (2 weeks) Integrate synthesizer.
4. Apr. 1,(2 weeks) Run simulations.
5. Apr. 15, (2 weeks) Project write-up.

6 Conclusions

We hope our work will lead to more efficient caching algorithms. The importance of our study is clear: given the popularity of Web proxies, even a little improvement of proxy caching algorithms will lead to considerable traffic reduction of the whole Web.

References

[1] G. Abdulla. Analysis and Modeling of World Wide Web Trac. PhD thesis, Virginia Tech, 1998.

[2] M. Abrams, C.R. Standbridge, G.Abdulla, S. Williams and E.A. Fox. Caching Proxies: Limitations and Potentials. WWW-4, Boston Conference, December, 1995.

[3] M. Arlitt and C. Williamson, entitled “Web Server Workload Characterization: The Search for Invariants”, to appear in the proceedings of the 1996 ACM SIGMETRICS Conference on the Measurement and Modeling of Computer Systems, Philadelphia, PA, May 23-26, 1996.

[4] J. Beran, R. Sherman, M.S. Taqqu and W. Willinger, "Long-Range Dependence in Variable-Bit-Rate Video Traffic", IEEE Trans. on Communications, Vol. 43, pp. 1566-1579, 1995.

[5] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. Technical Report 1371, Computer Sciences Dept, Univ. of Wisconsin-Madison, April, 1998.

[6] Ramon Caceres, Fred Douglis, Anja Feldmann, Gideon Glass, and Michael Rabinovich. Web proxy caching: the devil is in the details. In Workshop on Internet Server Performance, Madison, Wisconsin USA, June 1998.

[7] P. Cao and S. Irani, "Cost-aware WWW proxy caching algorithms," in Proc. USENIX Symp. on Internet Technologies and Systems, pp. 193–206, December 1997.

[8] Pei Cao, Jin Zhang, and Kevin Beach. Active Cache: Caching Dynamic Contents on the Web. In Proceedings of Middleware, 1998.

[9] M. E. Crovella and A. Bestavros. Self-similarity in world wide web traffic evidence and possible causes. In Proceedings of the ACM SIGMETRICS 96, pages 160–169, Philadelphia, PA, May 1996.

[10] B. Davison. A survey of proxy cache evaluation techniques. In Proceedings of 4th International Web Caching Workshop, March 1999.

[11] S. Deng. Empirical Model of WWW Document Arrivals at Access Link. In Proc. International Communications Conference – ICC'96, Dallas, Texas, USA, 23-17 June 1996.

[12] D.E. Duffy, A.A. McIntosh, M. Rosenstein and W. Willinger, "Statistical Analysis of CCSN/SS7 Traffic Data from Working Subnetworks", IEEE J. Select. Areas in Communications, Vol. 12, No. 3, pp. 544-551, 1994.

[13] S. G. Dykes and K. A. Robbins. A viability analysis of cooperative proxy caching. Available online at http://www.cs.utsa.edu/sdykes/.

[14] Bradley Duska, David Marwood, and Michael J. Feeley. The measured access characteristics of World Wide Web client proxy caches. In USENIX Symposium on Internet Technology and Systems, Monterey, California, USA, December 1997.

[15] Anja Feldmann, Ramon Caceres, Fred Douglis, Gideon Glass, and Michael Rabinovich. Performance of web proxy caching in heterogeneous bandwidth environments. In Proceedings of the IEEE Infocom '99 Conference, New York, NY, March 1999.

[16] S. Gadde, J. Chase, and M. Rabinovich. Web caching and content distribution: A view from the interior. In Proc. of the Fifth Int. Web Caching and Content Delivery Workshop, May 2000.

[17] S. Glassman. A caching relay for the world wide web. In First International Conference on the World Wide Web. CERN, Geneva, Switzerland, May 1994.

[18] S. D. Gribble and E. 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, 1997.

[19] S. D. Gribble, G. S. Manku, D. Roselli, E. A. Brewer, T. J. Gibson, & E. L. Miller. Self-Similarity in File Systems, in 'Proceedings of ACM SIGMETRICS '98', 1998.

[20] V. Holmedahl, B. Smith, and T. Yang. Cooperative Caching of Dynamic Content on a Distributed Web Server . In Proc. of Seventh IEEE International Symposium on High Performance Distributed Computing, pages 243–250, July 1998.

[21] S. J. Kelly and Jeffrey K. MacKie-Mason. Variable QoS from shared Web caches: User-centered design and value-sensitive replacement. In Proceedings of the MIT Workshop on Internet Service Quality Economics (ISQE 99), Cambridge, MA, December 1999.

[22] S. Klivansky, A. Mukherjee and C. Song, "Factors Contributing to Self-Similarity over NSFNet", preprint, Georgia Institute of Technology, 1994.

[23] B. Krishnamurthy and C. E. Wills. Proxy cache coherency and replacement, towards a more complete picture. In Proceedings of the 19th IEEE ICDCS'99, Austin, TX, June, 1999.

[24] W. E. Leland, M. Taqqu, W. Willinger, D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," Proc. SIGCOM93, 1993, San Francisco, California, pp. 183-193.

[25] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson. On the Self-Similar Nature of Ethernet Traffic (Extended Version). IEEE/ACM Transactions on Networking, Vol. 2, pp. 1–15, 1994.

[26] D. Li and D. R. Cheriton. Scalable web caching of frequently updated objects using reliable multicast. In Proceedings of the Second USENIX Symposium on Internet Technologies and Systems, pages 92–103, Oct. 1999.

[27] Binzhang Liu and Edward Fox. Web traffic latency: Characteristics and implications. In WebNet98, Orlando, FL, November 1998.

[28] Chengjie Liu and Pei Cao. Maintaining strong cache consistency for the world-wide web. In The 17th International Conference on Distributed Computing Systems, May 1997.

[29] P.Lorenzetti, L.Rizzo, L.Vicisano, "Replacement policies for a proxy cache", Technical report LR960731, DEIT, Univ. di Pisa, July 1996.

[30] B. Mandelbrot, Self-similar Error Clusters in Communication Systems and the Concept of Conditional Stationarity. IEEE Transactions on Communication Technology COM-13 (1965).

[31] B. Mandelbrot, The Fractal Nature of Geometry. W. H. Freeman and Company, New York, NY, USA, 1983.

[32] A. Mahanti, D. Eager, and C. Williamson. Temporal locality and its impact on web proxy cache performance. Performance Evaluation, September 2000.

[33] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. In Proc. SPIE International Conference on Performance and Control of Network Systems, November 1997.

[34] V. Paxson and S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling," IEEE/ACM Transactions on Networking, pp.226-244, June 1995.

[35] Proxy Caching. http://www.web-caching.com/proxy-caches.html. 2001.

[36] P. Rodriguez, S. Sibal, and O. Spatscheck, "TPOT: Translucent Proxying of TCP", Technical report TR 00.4.1, AT&T Research Labs, 2000.

[37] Squid Internet Object Cache. http://www.squid-cache.org. 2001.

[38] M. S. Taqqu and J. Levy. Using Renewal Processes to Generate Long-Range Dependence and High Variability. In E. Eberlein and M. S. Taqqu, editors, Dependence in Probability and Statistics, pp. 73–89, Boston, 1986. Birkhauser.

[39] C. Wills and M. Mikhailov, Examining the Cacheability of User-Requested Web Resources ", Proceedings of the 4th International Web Caching Workshop, San Diego, CA, March/April 1999.

[40] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox. Removal policies in network caches for World-Wide Web documents. In Proceedings of ACM SIGCOMM'96 Conference, pages 293–305, August 1996.

[41] W. Willinger, M. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through high variability: statistical analysis of Ethernet LAN traffic at the source level. In ACM Sigcomm '95, pages 100–113, 1995.

[42] W. Willinger, M. S. Taqqu, W. E. Leland, and D. V. Wilson. Self-Similarity in High-Speed Packet Traffic: Analysis and Modeling of Ethernet Traffic Measurements. Statistical Science, 10:67–85, 1995.

[43] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson. Self-similarity through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level. Computer Communications Review, 25, pp. 100–113, 1995. Proceedings of the ACM/SIGCOMM'95, Cambridge, MA, August 1995.

[44] W. Willinger, M. S. Taqqu, W. E. Leland, R. Sherman and D. V. Wilson. Self-Similarity in High-Speed Packet Traffic: Analysis and Modeling of Ethernet Traffic Measurements. IEEE/ACM Transactions on Networking, pp.71-86, Vol. 5, No. 1, 1997.

[45] W. Willinger, M. S. Taqqu and A. Erramilli, A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks, in Stochastic Networks, F. P. Kelly, S. Zachary and I. Ziedins (eds.), Oxford University Press, Oxford, 1996, 339–366.

[46] Alec Wolman, Geoffrey M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. In Proceedings of the 17th ACM Symposium on Operating Systems Principles, December 1999.

[47] R. Wooster and M. Abrams. Proxy Caching the Estimates Page Load Delays. In the 6th International World Wide Web Conference, April, 1997. Santa Clara, CA.