ICACT 2000 Paper Submission
Contact Information

Title :

A Statistical Prefetching Web Caching Scheme for
Efficient Internet Bandwidth Usage

Name :

Jae-Young Kim

Mailing Address :

Distributed Processing & Network Management Lab.
Dept. of Computer Science and Engineering
Pohang University of Science and Technology
San 31, Hyoja-dong, Nam-gu
Pohang, Korea
790-784

E-mail Address :

jay@postech.ac.kr

Fax :

+82-562-279-5699

Telephone :

+82-562-279-5659

Choice of Topic :

Multimedia, Internet Systems, Services and Standards

A Statistical Prefetching Web Caching Scheme for Efficient Internet Bandwidth Usage

Sook-Hyang Kim, Jae-Young Kim and Won-Ki Hong
Department of Computer Science and Engineering
Pohang University of Science and Technology
{shk, jay, jwkhong}@postech.ac.kr

***Abstract* - As the number of World-Wide Web (Web) users grows, Web traffic continues to increase at an exponential rate and becomes one of the major components of Internet traffic. Also, high bandwidth usage due to the Web traffic is observed during peak periods while leaving bandwidth usage idle during off-peak periods. One of the solutions to reduce Web traffic and speed up Web access is through the use of Web caching. However, Web caching has limitations in reducing network bandwidth usage especially during peak periods. In this paper, we focus our attention on the use of a prefetching Web caching scheme for reducing bandwidth during peak periods by using off-peak period bandwidth. We propose a statistical, batch, proxy-side prefetching scheme that improves cache hit rate with a small amount of additional storage space. We have simulated our scheme based on real Web proxy trace logs and the results indicate that the proposed prefetching scheme can reduce peak period bandwidth using off-peak period bandwidth.**

***Keywords* - Web caching, prefetching scheme, Web access pattern, Internet Bandwidth**

1. Introduction

As the number of World Wide Web (Web) users grows, Web traffic continues to increase at an exponential rate. The rapid growth of Internet is due to the fact that users can easily access a variety of information from remote locations. Currently, Web traffic is one of the major components of Internet traffic. Figure 1 shows Web traffic bandwidth usage of student dormitories for 24 hours at POSTECH (a university in Korea). Figure 1 shows that high bandwidth usage due to Web traffic is required during peak periods, while leaving bandwidth usage idle during off-peak periods. It is desirable to effect a balance between bandwidth usage during peak periods and off-peak periods. This will reduce Web access time and make more efficient use of Internet links.

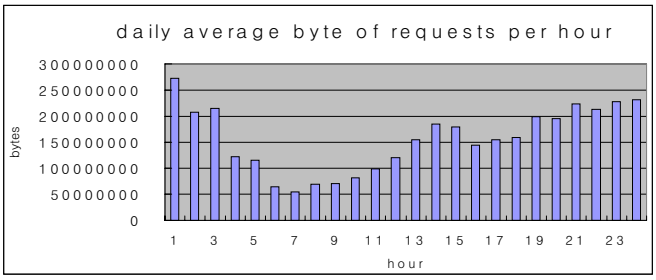

Figure 1. The bandwidth usage of student dormitories at POSTECH

One of the solutions to reduce Web traffic and speed up Web access is through the use of Web caching [10, 7]. However, Web caching is limited for reducing network bandwidth usage during peak periods [4, 5, 11, 12]. In this paper, we focus our attention on the use of prefetching, based on a caching server, for reducing bandwidth during peak periods using off-peak period bandwidth. We have developed a statistical, batch, proxy-side prefetching scheme that improves cache hit rate, while only requiring a small amount of additional storage space. This prefetching scheme uses Web access patterns and Web traffic traces. In fact, this scheme may increase total bandwidth usage slightly in comparison with naive Web caching. However, the proposed scheme can efficiently reduce Web traffic bandwidth usage during peak periods by accessing unused bandwidth during off-peak periods.

The remainder of this paper is organized as follows. Section 2 examines current Web prefetching methods. In Section 3, we discuss the issues associated with the existing prefetching schemes, analyze Web traffic access patterns and present our prefetching scheme. Section 4 describes the design of our prefetching system. In Section 5, we describe the simulation framework used to test our system, and the results of the simulation are discussed in Section 6. Section 7 summarizes our work and discusses possible future research.

2. Prefetching Methods

Prefetching is a method to anticipate future Web object requests and prefetch the Web objects in a local cache. A prefetching approach can be categorized according to the following three aspects: from where prefetching shoud be initiated, whether a Web object should be prefetched, and when a Web object should be prefetched.

2.1 Server-Side, Client-Side and Proxy-Side Prefetching

Prefetching can be initiated by Web servers, by clients or by cache servers (also called proxies). When a client demands a Web object, a Web server may anticipate the next request and immediately preload the corresponding Web objects to the client [2, 14, 15]. A client can also initiate the prefetching of Web objects by changing the user's configuration. A client can also use a Web access pattern monitoring system, which observes the past access patterns for particular Web objects and prefetches Web objects on the behalf on the user [1, 2, 13]. Also, a proxy can initiate the prefetching of Web objects. A

proxy can analyze Web access patterns of clients, anticipate future requests and prefetch them [3, 17, 18].

2.2 Statistical and Deterministic Prefetching

The decision for whether a Web object should be prefetched can be either statistical [5, 8, 9] or deterministic [1]. In statistical prefetching, the prefetching system uses the Web access patterns of users. In a deterministic scheme, prefetching can be configured statically by clients or by proxies. For example, a user may configure some Web objects such as the home page of a popular newspaper with prefetching. These Web objects are immediately prefetched when the newspaper's home page is accessed.

2.3 Prediction and Batch Prefetching

The criteria for deciding when a Web object should be prefetched can either be by prediction or by batch. In the prediction scheme, when a client requests a Web object, a Web server, proxy server or client may predict the next request and immediately preload the predicted Web objects to the client [2, 5, 13, 16, 18]. The gole of most prediction prefetching is to reduce delay in Web access time. However, if the prediction is not accurate, network resources are wasted and network bottlenecks are generated during peak periods. In the batch scheme, Web objects that the user is likely to access in the near future are prefetched during off-peak periods [6].

3. Proposed Prefetching Scheme

We described several prefetching methods in the previous section. In the server-side prefetching scheme, the client needs to be aware of the prefetching, and both client and Web server software must change their configurations. Client-side prefetching cannot share the Web access patterns of all users in an Intranet, because this scheme relies on only one user's Web access pattern. Deterministic prefetching has limited expansion to all Web objects. In the prediction prefetching scheme, network traffic may be generated during peak periods so this scheme is not appropriate for reducing peak bandwidth usage. Therefore, we decided to develop a batch, statistical and proxy-side prefetching scheme.

3.1 Issues

There are several issues associated with a batch, statistical and proxy-side prefetching scheme.

- For the statistical prefetching scheme to work efficiently, Web access patterns of users should be accurately reflected in prefetching. However, Web access patterns of users can change frequently.
- After prefetching, if prefetched objects are not requested, network resources are wasted. So, a prefetched Web object's hit rate should be high.
- The traditional batch prefetching scheme demands a large amount of disk space. The total amount of disk space for prefetching should be adjusted flexibly.

3.2 Solutions

The followings are solutions which address the above issues.

- To reflect Web access patterns in prefetching, the prefetching system periodically analyzes the Web object access patterns, based on the most recent "access log", then generates a list of Web objects that are to be prefetched.
- To increase the prefetched object hit rate we need to restrict to selecting prefetchable objects. To begin, we choose prefetchable objects that are stored in the cache and are past their expiration times. Then, we consider the reference count of each prefetchable object.
- Applying the first and second solutions, we can reflect Web access patterns in prefetching, increase the prefetched object hit rate and diminish the total amount of disk space required for prefetching.

3.3 Prefetching Parameters

The prefetching parameters for determining prefetchable objects are periods for gathering input data (logs of HTTP request), reference count of objects in the cache, total bytes of prefetchable objects, and prefetching execution time.

First, the periods for gathering input data (logs) is intended to reflect Web access patterns of the users, so these periods are determined by the characteristics of each network in question (e.g., the day before, several days before, etc).

The second parameter is the reference count of objects in cache. To reduce the unnecessary use of network resources and increase the hit ratio of prefetched Web objects, we must set a limit on the minimum reference count of prefetchable objects. In this way, we obtain greater results with the highly-referenced Web objects.

Again, the third parameter involves the total bytes of the Web objects that will be prefetched. This can be determined by taking reference count into consideration. For example, if prefetching storage is limited to 100 Mbytes of disk space, the Web object with a high reference count will be given a high priority for being selected to be prefetched.

The fourth parameter is prefetching execution time. Our goal is to use prefetching to access the unused bandwidth during off-peak periods. Thus, the applied time of prefetching is determined by off-peak periods. We decide that off-peak period is the period with lower than 80% of total Web traffic. Consequently, from Figure 1, the off-peak period is from 04:00 to 13:00 (9 hours) and the peak period is from 13:00 to 04:00 (15 hours).

3.4 Web Traffic Trace at POSTECH

We have monitored Web traffic traces of 16 subnets that are used by student dormitories at POSTECH for two weeks from October 15, 1999 to October 28, 1999. The average object hit ratio was 54.96%, the average byte hit ratio was 31.42%, and the total number of Web object requests was about 4.07 million. The number of clients making requests was 447. The total bytes to Clients was 50.2 Gbyte. Table 1 displays the number of Web

objects and the bytes of Web objects related to each reference count. Among the Web objects which were stored in the cache server, 70.75% of them had only 1 reference count and 29.25% of them had a reference count of two or more. The total bytes of Web objects which were accessed only once is 68.12% and 31.88% of them had a reference count of more than 2.

Table 1. Web Traffic Categorized by Reference Count

Reference count	Daily average # of objects (%)	Daily total average bytes of objects (Mbytes) (%)
1	86,227 (70.75%)	1411.2 (68.12%)
2	16,690 (13.69%)	296.39 (14.31%)
3	6,462 (5.3%)	112.71 (5.44%)
4	3,451 (2.83%)	70.74 (3.41%)
over 5	9,053 (7.43%)	180.74 (8.72%)
Total	121,883 (100%)	2071.62 (100%)

4. Design of a Prefetching System

Based on the prefetching scheme discussed in the previous section, we have designed a prefetching system. The design architecture of our system is illustrated in Figure 2.

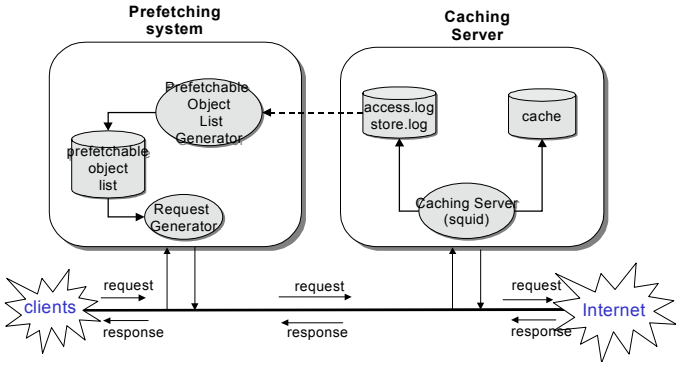

Figure 2. Design Architecture of a Prefetching System

The overall system consists of two parts: a prefetch list generator and a request generator. This system uses an "access log" and a "store log" from Squid [7] and operates independent of Squid as an add-on.

4.1 Prefetch List Generator

The prefetch list generator is used for deciding whether a Web object should be prefetched. It uses an "access log", which reports on the information of HTTP request from clients, and a "store log", which reports information of Web objects in the cache. It extracts a stale list that consists of Web objects that are past their expiration time using the refresh scheme of Squid. If a Web object does not have the expiration time information in the "store log", we set the expiration time to the default value of three days in Squid. Thus, the prefetch list generator makes a prefetch list using the conditions of prefetchable objects described in the previous section. Figure 3 shows the prefetch list format.

Referenced Count	URL	Byte

Figure 3. Prefetch List Format

4.1.1 Refresh Algorithm of Squid

Figure 4 shows how Squid decides when to refresh a cached object [7]. The refresh algorithm is checked in the order listed here. The first entry which matches is used. If none of the entries matches, then the default will be used. The OBJ_DATE is the value of the HTTP date [19] denoted by the date and time at which the message originated. The OBJ_LASTMOD is the value of the HTTP Last-Modified indicating the date and time when the sender believes the resource was last modified. The Expires from the HTTP reply header gives the date/time after which the entity should be considered stale. The CLIENT_MAX_AGE is the maximum object age the client will accept as taken from the HTTP/1.1 Cache-Control request header [20]. The OBJ_AGE is how much the object has aged since it was retrieved. The LM_AGE is how old the object was when it was retrieved. The LM_FACTOR is the ratio of OBJ_AGE to LM_AGE. The NOW is current time. AGE is how much the object has aged since it was retrieved: OBJ_AGE = NOW - OBJ_DATE. LM_AGE is how old the object was when it was retrieved: LM_AGE = OBJ_DATE - OBJ_LASTMOD. LM_FACTOR is the ratio of AGE to LM_AGE: LM_FACTOR = OBJ_AGE / LM_AGE. MAX_AGE, MIN_AGE and CONF_PERCENT are compared with the parameters of the refresh_pattern rules. These values are used for the default value of Expires.

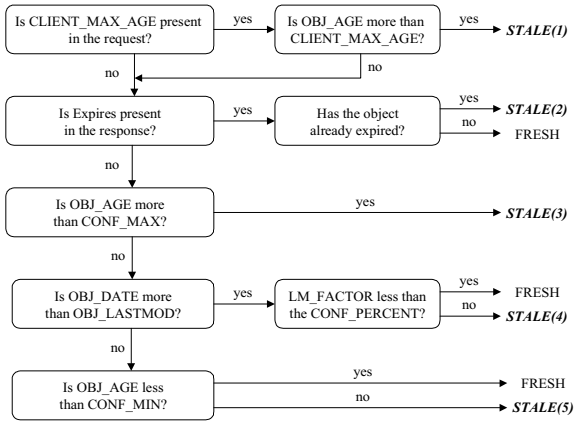

Figure 4. Flow Chart for Refresh Algorithm of Squid

4.1.2 Freshness Decision

Web objects are removed from the cache when they expire. When a Web object enters the cache, Squid checks the freshness requirements when objects are requested, and then records each information in an "access log" and a "store log". If an object is 'FRESH', it is available. If it is 'STALE', then it is not available so it needs to be prefetched. We use the refresh algorithm of Squid to determine whether Web objects are stale or fresh. We use OBJ_DATE, OBJ_LASTMOD and Expires from the "store log" and MAX_AGE, MIN_AGE and CONF_PERCENT from squid.conf. But we cannot get CLIENT_MAX_AGE from any logs, so in our work we do not

consider it. Figure 5 shows the format of "access log" in Squid. Figure 6 shows the format of "store log" in Squid.

Time	Elapsed	Remotehost	Code/Status	Byte	Method	URL

Figure 5. Access Log Format

Time	Action	Status	OBJ_DATE	OBJ_LASTMOD	Expires	Type	Len	Method	URL

Figure 6. Store Log Format

When we decide whether a Web object is stale or not, we must consider both current stale objects and future stale objects until the next prefetching time. Consequently, NOW (Current Prefetching Time + Prefetching Frequency) is calculated by adding current time to prefetching frequency. Figure 7 describes the prefetching freqeuncy.

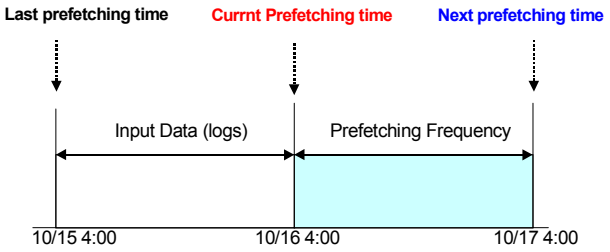

Figure 7. Prefetching Frequency

4.2 Request Generator

The request generator operates as a Web client. The request generator will generate HTTP requests from the prefetch list and send the HTTP requests to the Web server during off-peak periods. We have developed the request generator using "wget", which is command-line Web client that supports HTTP [21].

5. Simulation

We have designed a simulation model to estimate the performance of our prefetching scheme. The simulation is trace-driven and a real cache server (Squid) log files, such as "access log" and "store log", are employed to emulate the requests of the clients and to access the operational behavior of the prefetching system. We have also verified our scheme with the simulation system illustrated in Figure 8.

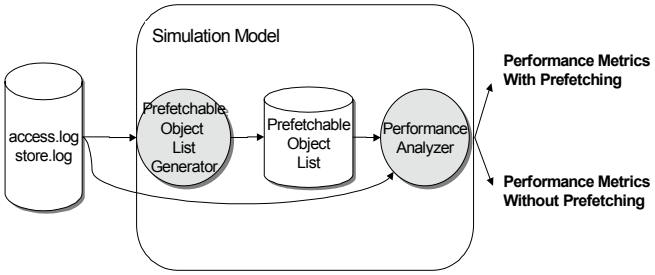

Figure 8. Simulation System

We have used "access log" and "store log" that were obtained from the Web traffic trace in Section 3.4. The simulation procedure is as follows:

- The prefetch list generator extracts the prefetch list from the "access log" using the prefeching parameters determined in Section 5.1.
- If a Web object in the prefetch list exists in the "access log" of next day, the performance of prefetching becomes higher.
- The performance of the cache server with prefetching is compared to the performance of the cache server without prefetching. The factors of comparison are performance metrics of proposed prefetching in Section 5.2.

5.1 Prefetchable Web Objects

Using the prefetching parameters described in Section 3 and the results from the Web traffic trace, we have selected the conditions of prefetchable objects for simulation.

The period for gathering input data (log files) is 24 hours. Analyzing the results of the web traffic traces at POSTECH, we found that the access pattern variation is repeated every day.

To evaluate the performance of our prefetching approach per reference count, we chose the prefetchable Web objects that are referenced more than ones, more than twice, more than three times, more than four times and more than five times. In this section, we did not select particular reference count for prefetching because we should decide appropriate reference count for prefetching from the results of simulation.

The total size of prefetchable objects is less than the surplus bandwidth of off-peak periods and the prefetching execution time off-peak period is from 04:00 to 13:00 (9 hours).

5.2 Performance Metrics

Four quantities, expressed in percentages, are used for evaluating performance of the prefetching system:

- Request Saving: The number of prefetched Web objects that the users requested for the first time divided by the total number of requested pages. When the request saving increases, the cache object hit rate also increases.
- Bandwidth Saving: The amount of bytes of prefetched Web objects that the users requested for the first time divided by the total amount of bytes of requested Web objects. The bandwidth saving prompts the cache bytes hit rate to increase.
- Accuracy: It consists of the prefetched object hit rate and the prefetched byte hit rate.
  - ✓ prefetched object hit rate: The number of prefetched Web objects that the users requested divided by the total number of prefetched Web objects.
  - ✓ prefetched byte hit rate : The amount of bytes of prefetched Web objects that the users requested divided by the total amount of bytes of prefetched Web objects.
- Wasted Bandwidth: The amount of bytes of Web objects that are prefetched but are not accessed by the client.

6. Results

In this section, we discuss the results obtained from our simulation experiments based on performance metrics.

6.1 Request Saving

Figure 9 shows the performance of the prefetching strategies related to cache object hit rate with and without prefetching per reference count. The cache object hit rate is increased by maximum 4.3% on average compared with the non-prefetching case. When the reference count is increased, the request saving is decreased.

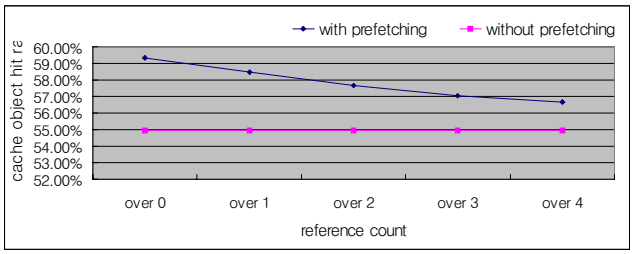

Figure 9. Cache Object Hit Ratio With and Without Prefetching Per Reference Count

6.2 Bandwidth Saving

Figure 10 reports the cache byte hit rate with and without the use of prefetching. The cache byte hit rate is raised by maximum 5% higher on average than the non-prefetching system When the reference count is increased, the bandwidth saving is decreased.

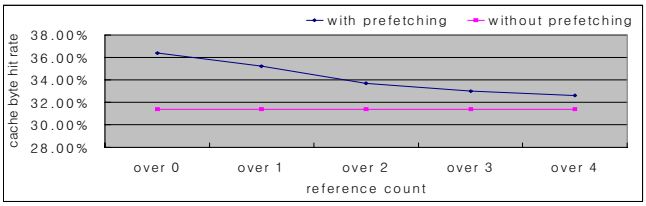

Figure 10. Cache Byte Hit Ratio With and Without Prefetching Per Reference Count

6.3 Accuracy

Figure 11 illustrate the accuracy of prefetching. In Table 4, when the reference count increases, the prefetched object hit rate and prefetched byte hit rate also increase.

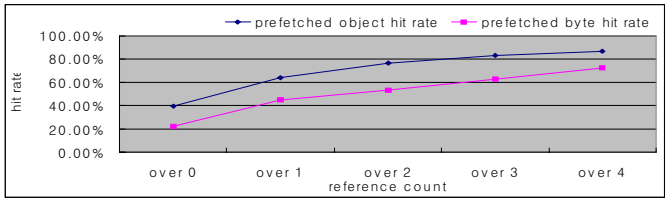

Figure 11. Accuracy Per Reference Count

6.4 Wasted Bandwidth

Table 2 shows the wasted bandwidth by prefetching and Figure 12 illustrates a rate of increase about bandwidth usage during off-peak periods. When the reference count increases, wasted bandwidth decreases.

Table 2. Wasted Bandwidth

Reference count	Daily average wasted bandwidth (%)	A rate of increase about bandwidth usage during Off-peak periods
Over 0	670.72 Mbyte (17.8%)	51.9 %
Over 1	172.65 Mbyte (4.6%)	28.3 %
Over 2	74.47 Mbyte (2.0%)	16.7 %
Over 3	35.91 Mbyte (1.0%)	10.8 %
Over 4	16.97 Mbyte (0.5%)	7.2 %

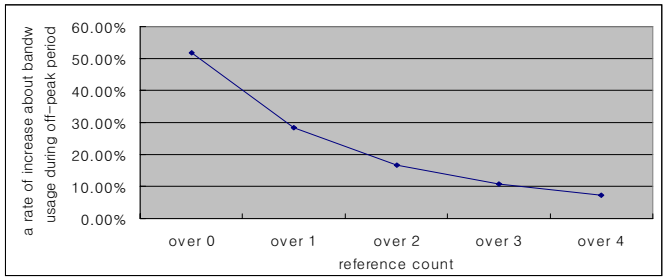

Figure 12. A Rate of Increase About Bandwidth Usage During Off-peak Periods Per Reference Count

6.5 Summary of Results

Table 3 summarizes performance metrics of prefetching per reference count. If the reference count is higher, then the request saving, the bandwidth saving, and overall accuracy are improved and the wasted bandwidth is decreased.

Table 3. Summary of Performance Metrics Per Reference Count

Reference count	Request saving	Bandwidth saving	Wasted bandwidth	Accuracy	
				Prefetched object hit rate	Prefetched byte hit rate
Over 0	4.30 %	5.01 %	17.8 %	39.4 %	22 %
Over 1	3.53 %	3.78 %	4.6 %	64.3 %	45.1%
Over 2	2.72 %	2.27 %	2.0 %	76.5 %	53.5 %
Over 3	2.10 %	1.59 %	1.0 %	83.3 %	62.6 %
Over 4	1.62 %	1.19 %	0.5 %	87.1 %	72.6 %

When the amount of reduced bandwidth usage by prefetching is increased in volume and the amount of wasted bandwidth by prefetching is decreased, effectiveness of prefetching increases. Accordingly, we define effectiveness of prefetching in the following equation:

$E_p = B_s / B_w$

E_p: Effectiveness of prefetching

B_s: The amount of saved bytes by prefetching (Mbyte)

B_w: The amount of wasted bytes by prefetching (Mbyte)

Table 4 shows the effectiveness of prefetching and the total rate of decrease of bandwidth per reference count by prefetching.

Table 4. Effectiveness of Prefetching Per Reference Count

Reference count	B_s (Mbyte)	B_w (Mbyte)	E_p	Bandwidth Saving
Over 0	189.16	670.72	0.28	5.01 %
Over 1	141.82	172.65	0.82	3.78 %
Over 2	85.65	74.47	1.15	2.27 %
Over 3	60.08	35.91	1.67	1.59 %
Over 4	45.06	16.97	2.66	1.19 %

In the previous section, we do not select a particular reference count for prefetching because the appropriate reference count for prefetching must be determinated by the from results of simulation. To select an appropriate reference count for prefetching, we should consider the effectiveness of perefetching and the amount of bandwidth saving per reference count. Effectiveness of prefetching with a high reference count is increased, but the amount of bandwidth saving is decreased. Therefore, we should choose a reference count that guarantee E_p more than 0.5 with the bandwidth saving more than 3%. Consequently, the most appropriate reference count is more than 2.

When the reference count is more than 2, the average number of bytes of prefetched objects is 201,649,369 bytes. Peak period is 64,800 seconds (19 hours). Bandwidth Saving can be calculated as

Bandwidth saving = (141824895 * 8) / 54000= 20.52 Kbps

Thus, the amount of bandwidth saved by prefetching when the reference count is more than 2 is 20.52 Kbps.

7. Summary and Future Work

We have presented a prefetching scheme for Web traffic with the goal of reducing peak bandwidth usage. Our results indicate that statistical, batch, proxy-side prefetching can lead to significant reduction of peak bandwidth usage using extra network resources to prefetch Web objects during off-peak periods. The amount of bandwidth saving by the proposed prefetching scheme applied to POSTECH was 20.52 Kbps.

We can increase the cache object hit rate and byte hit rate using our scheme. Also, our prefetching scheme shows high accuracy through the increased hit rate of prefetched objects ranging from 22% to 73%. Through our simulation results, we confirmed the efficiency of our prefetching scheme. Consequently, a cache server with our prefetching scheme can use network bandwidth effectively.

The work presented in this paper investigated the use of the Web in an academic environment. We believe that we could achieve similar or better results in an industrial environment where the off-peak periods are typically larger (e.g., from 20:00 to 08:00).

REFERENCES

[1] Z. Wang and J. Crowcroft, “Prefetching in World Wide Web,” IEEE Globecom’96, http://www.cs.url.ac.uk/staff/zwang/papers/.

[2] V. Padmanabhan and J. Mogul, “Using Predictive Prefetching to Improve World Wide Web Latency,” Computer Communication Review, 26(3):22-36, July 1996.

[3] Ken-ichi Chinen and Suguru Yanaguchi, “An Interactive Prefetching Proxy Server for Improvement of WWW Latency,” INET’97, 1997, http://www.isco.org/INET97/procceding/A1/A1_3.HTM.

[4] Arthur Goldberg, Ilya Pevzner and Robert Buff, “Caching Characteristic of Internet and Intranet Web proxy Traces,” In Computer Measurement Group Conference (CMG’98), Anaheim, CA, December 1998, http://www.cs.nyu.edu/artg.

[5] Carlos Maltzahn and Kathy J. Richardson, “On Bandwidth Smoothing,” In Web Caching Workshop WCW’99, 1999, http://www. ircache.net/Cache/ Workshop99/program.html.

[6] Themistoklis Palpanas and Alberto Mendelzon, “ Web Prefetching Using Partial Match Prediction,” In Web Caching Workshop WCW’99, 1999, http://www.ircache.net/Cache/Workshop99/program.html.

[7] Squid Internet Object Cache, available from http://squid.nlanr.net/ Squid/.

[8] Gihan V.Dias, Graham Cope and Ravi Wijayaratne, “A Smart Internet Caching System, ” INET’96 Conference, 1996, http://www.isoc.org/ isoc/ whatis/conferences/inet/96/proceedings/a4/a4_3.htm.

[9] Katsuo Doi, “WWW Access by Proactively Controlled Caching Proxy, ” Sharp Technical Journal, No. 66, December 1996.

[10] Brad Duska, David Marwood, and Michael J.Feeley, “The Measured Access Characteristics of World-Wide Web Client Proxy Caches,” In Usenix Symposium on Internet Technologies and Systems (USITS), Monterey, CA, USA, December 8-11 1997, Usenix, http://www. cs.ubc.ca/spider/marwood/Projects/SPA/Report/Report.html.

[11] Marc 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, http://ei.cs.vt.edu/ ~succeed/WWW4/ WWW4.html.

[12] Anawat Chankhunthod et al, “A Hierarchical Internet Object Cache, ” Technical Conference, Usenix 1996, http://excalibur.usc.edu/ cache-html/cache.html.

[13] James Griffioen and Randy Appleton. “Reducing File System Latency using a Predictive Approach,” Proceddings of the 1994 Summer USENIX Technical Conference, Boston, Massachusetts, USA, 1994, http://usenix.org/publications/library/proceedings/bos94/griffioen.html.

[14] Azer Bestavros, “Speculative Data Dissemination and Service to Reduce Server Load,” Network Traffic and Service Tome in Distributed Information System, In International Conference on Data Engineering, pages 180-189, New Orleans, LO, February 1996.

[15] Tomas M. Kroeger, Darrell D. E. Long, and Jeffrey C. Mogul. “Exploring the bounds of web latency reduction from caching and prefetching,” In Proceedings of USENIX Symposium on Internet Technology and Systems, December 1997, http://www.usenix.org/ publications/library/proceedings/ usits97/kroeger.html.

[16] Evangelos P. Margatos and Catherine E. Chronaki, “A top-10 Approach to Prefetching on the Web,” Technical report, In Proceedings of INET' 98 (The Internet Summit), Geneva, Switzerland, July 1998, http://www.ics.forth.gr/ proj/arch-vlsi/OS/www.html.

[17] Wcol Group, “WWW Collector – the prefetching proxy server for WWW,” 1997, http://shika.aist-nara.ac.jp/products/wcol/wcol.html.

[18] 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, http://www.cs.wisc.edu/~cao/.

[19] T. Berners-Lee, R. Fielding, and H. Frystyk, “Hypertext Transfer Protocol – HTTP/1.0,” RFC 1945, May, 1996.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1,” RFC 2616, June 1999.

[21] GNU wget Homepage, http://www.gnu.org/software/wget/wget.html.