Available online at www.elixirpublishers.com (Elixir International Journal)

Advanced Engineering Informatics

Elixir Adv. Engg. Info. 33 (2011) 2174-2178

ISSN: 2229-712X

An integration approach of intelligent predictive caching and static prefetching for internet web servers

[1]J B Patil and [2]B. V. Pawar

[1]Department of Computer Engineering, R. C. Patel Institute of Technology, Shirpur. (M.S.), India,

[2]Department of Computer Science, North Maharashtra University, Jalgaon. (M.S.), India.

ARTICLE INFO

Article history:

Received: 18 February 2011;

Received in revised form:

17 March 2011;

Accepted: 26 March 2011;

Keywords

Web caching,
Web prefetching,
Replacement policy,
Hit ratio,
Byte hit ratio,
Trace-driven simulation.

ABSTRACT

Web caching and Web prefetching are two important techniques used to reduce the noticeable response time perceived by users. By integrating Web caching and Web prefetching, these two techniques can complement each other since the Web caching technique exploits the temporal locality, whereas Web prefetching technique utilizes the spatial locality of Web objects [30]. In this paper, we develop algorithm Pre-IPGDSF# by integrating Web caching and Web prefetching in Web servers. For caching, we use innovative algorithm Intelligent Predictive Greedy Dual Size Frequency#, IPGDSF#, which is an enhanced version of algorithm GDSF# developed by us [13, 14, 15, 16, 17]. For prefetching, we use a static prefetching method. Using three different Web server logs, we use trace driven analysis to evaluate the effects of different replacement policies on the performance of a Web server. We specifically compare policies like GDSF, GDSF#, and IPGDSF# with the proposed integrated policy Pre-IPGDSF#. Using trace-driven simulations and for various standard cost criteria (hit rate and byte hit rate), we show that our proposed integrated policy outperforms all other algorithms under consideration.

Introduction

The enormous popularity of the World Wide Web has caused a tremendous increase in network traffic due to http requests. This has given rise to problems like user-perceived latency, Web server overload, and backbone link congestion. Web caching is one of the ways to alleviate these problems [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

One might argue that the ever decreasing prices of RAM and disks renders the optimization or fine tuning of cache replacement policies a "moot point". Such a conclusion is ill guided for several reasons. First, recent studies have shown that Web cache hit ratio (HR) and byte hit ratio (BHR) grow in a *log-like* fashion as a function of cache size [5, 26, 27, 28]. Thus, a better algorithm that increases hit ratios by several percentage points would be equivalent to a several-fold increase in cache size. Second, the growth rate of Web content is much higher than the rate with which memory sizes for Web caches are likely to grow. The only way to bridge this widening gap is through efficient cache management. Finally, the benefit of even a slight improvement in cache performance may have an appreciable effect on network traffic, especially when such gains are compounded through a hierarchy of caches [6].

Cao and Irani have surveyed ten different policies and proposed a new algorithm, Greedy-Dual-Size (GDS) in [5]. The GDS algorithm uses document size, cost, and age in the replacement decision, and shows better performance compared to previous caching algorithms. In [4] and [12], frequency was incorporated in GDS, resulting in Greedy-Dual-Frequency-Size (GDSF) and Greedy-Dual-Frequency (GDF). While GDSF is attributed to having best hit ratio (HR), it is having a modest byte hit ratio (BHR). Conversely, GDF yields a best HR at the cost of worst BHR [12].

We have proposed a new algorithm called Greedy-Dual-Frequency-Size#, (GDSF#), which allows augmenting or weakening the impact of size or frequency or both on HR and BHR [13, 14, 15, 16, 17]. We extended this policy further by incorporating the concept of future frequency which is used to assign weight (key value) to the document while storing it in the cache. This algorithm is an *intelligent* one as it can adapt to changes in usage patterns as reflected by future frequency. We call this innovative caching algorithm as *Intelligent Predictive GDSF#, (IPGDSF#).*

Compared with web caching, prefetching goes one step further by anticipating users' future requests and pre-loading the anticipated objects into a cache. When a user eventually requests the anticipated objects, they are available in the cache. In the past, several prefetching approaches have been proposed [31, 32, 33]. The general idea of prefetching is to request web objects that are highly likely to occur in the near future. Such systems often rely on a prediction model based on statistical correlation between web objects. These models are trained on previous web log data. Thus, prediction model construction is at the core of prefetching algorithms. In this paper, we propose a Static prefetching algorithm. We store the most popular documents from Web server logs in a prefetch buffer which is a part of main cache buffer.

Caching and prefetching have often been studied as separate tools for reducing the latency observed by the users in accessing the Web. Less work has been done on integration of caching and prefetching techniques. Kroeger et al [36] study the combined effect of caching and prefetching on end user latency. Yang and Zhang have proposed an Integrated Prefetching and Caching Algorithm using a Correlation-Based Prediction Model [37, 38]. Lan et al. [39] have proposed a Rule-Assisted Prefetching in

Tele:
E-mail addresses: jbpatil@hotmail.com,bvpawar@hotmail.com

Web-Server Caching. Yang et al. [40, 41] have proposed a method for Mining Web Logs to obtain a Prediction Model and using the model to extend the well-known GDSF caching policy. Curcio, Leonardi, and Vitaletti [42] have proposed an Integrated Prefetching and Caching for the World Wide Web via User Cooperation. Teng, Chang, and Chen [30] developed algorithm IWCP (Integration of Web Caching and Prefetching), by integrating Web *caching* and Web prefetching in client-side proxies.

In this paper, we propose a new technique to integrate caching and prefetching. We integrate our static prefetching algorithm with our caching algorithm IPGDSF#. We call this Integrated Web caching and prefetching algorithm as *Intelligent Predictive GDSF# with Prefetching (Pre-IPGDSF#).* We compare Pre-IPGDSF# with algorithms like GDSF, GDSF#, and IPGDSF#. Our simulation study shows that Pre-IPGDSF# outperforms all other algorithms under consideration in terms of hit rate (HR) as well as byte hit rate (BHR).

The remainder of this paper is organized as follows. Section 2 introduces Pre-IPGDSF#, a new algorithm for integrating Web caching and prefetching. Section 3 describes the simulation model for the experiment. Section 4 describes the experimental design of our simulation while Section 5 presents the simulation results. We present our conclusions in Section 6.

Pre-IPGDSF# Algorithm

We extract *future frequency* from the Web server logs. Then it is used to extend our GDSF# policy. Our idea is similar to the work of Bonchi et al. [18, 19] and Yang et al. [20]. While the Web caching algorithm in [18, 19] was designed to extend the LRU policy, Yang et al. [20] extended GDSF policy. We will be extending our policy GDSF# [13, 14, 15, 16, 17].

As pointed out early in caching research [21], the power of caching is in accurately predicting the usage of objects in the near future. In earlier works, estimates for future accesses were mostly built on measures such as access frequency, object size and cost. Such measures cannot be used to accurately predict for objects that are likely to be popular but have not yet been popular at any given instant in time. For example, as Web users traverse Web space, there are documents that will become popular soon due to Web document topology, although these documents are not yet accessed often in the current time instant [20]. Our approach is based on predictive Web caching model described by Yang et al. [20]. However, there are many noteworthy differences. Firstly, we use simple statistical techniques to find future frequency while Yang et al. use sequential association rules to predict the future Web access behavior. Secondly, for simplicity we do not try to identify user sessions. We assume that a popular document, which is used by one user, is likely to be used by many other users, which normally is the case for popular documents. We demonstrate the applicability of the method empirically through increased hit rates and byte hit rates.

Similar to the approach by Bonchi et al. [18, 19], our algorithm is an *intelligent* one as it can adapt to changes in usage patterns as reflected by future frequency. This is because the parameter *future frequency,* which is used in assigning weight (key value) to the document while storing in the cache, can be computed periodically in order to keep track of the recent past. This characteristic of adapting to the flow of requests in the historical data makes our policy intelligent. We call this innovative caching algorithm as *Intelligent Predictive GDSF#, (IPGDSF#).*

In GDSF#, the key value of document *i* is computed as follows [13, 14, 15, 16, 17]:

$$H_i = L + f_i^{\lambda} \times c_i / s_i^{\delta}.$$

where λ and δ are rational numbers, L is the inflation factor, c_i is the estimated cost of the document i, f_i is the access frequency of the document i, and s_i is the document size.

We now consider how to find future frequency, ff_i for document i from the Web logs. We mine the preprocessed Web log files. We extract the unique documents from the logs. Then we arrange these documents in the temporal order. Now for each unique document, we extract the number of future occurrences of that document. We call this parameter as *future frequency*, *ff*. With this parameter, we can now extend GDSF# by calculating H_i, the key value of document i as follows:

$$H_i = L + (f_i + ff_i)^{\lambda} \times c_i / s_i^{\delta}.$$

Here we add f_i and ff_i together, which implies that the key value of a document i is determined not only by its past occurrence frequency f_i, but also by its future frequency ff_i. By considering both the past occurrence frequency and future frequency, we can enhance the priority i.e. the key value of those objects that may not have been accessed frequently enough in the past, but will be in the near future according to the future frequency. The more likely it occurs in the future, the greater the key value will be. This will promote objects that are potentially popular objects in the near future even though they are not yet popular in the past. Thus, we look ahead in time in the request stream and adjust the replacement policy.

Finally, we make the policy intelligent by periodically updating future frequency when some condition becomes false, e.g. at fixed time intervals or when there is a degradation in the cache performance.

The idea of static prefetching is based on the rather obvious fact that for any given Web server document access patterns change very slowly [34, 35]. The documents that were accessed frequently today will probably be accessed frequently tomorrow. In other words, popular documents will remain popular.

Keeping the above fact in mind, we try to store the most popular documents from Web server logs. We allocate 20% of the total cache size for the prefetch buffer; the caching algorithm uses the remaining 80%. The prefetching stops when the prefetch buffer is full. This set of popular documents is updated periodically to maximize the performance of the integrated algorithm. For example, we can re-calculate the set of prefetched documents daily based on document frequencies for the previous day. The Web server would use the request log file for the day to determine the documents to be cached during the following day. In Figure 1, we give a formal description of the algorithm.

Procedure Prefetch **{**
while (space taken by prefetched documents < 20% of cache)**{**
find popular document *p*
insert *p* in cache
}
}
Procedure IPGDSF# **{**
Initialize *L = 0*
Find future frequency ff_i
loop forever {
do {
Process each request document in turn:
let current requested document be *i*
if *i* is already in cache

$H_i = L + (f_i + ff_i)^{\lambda} \times c_i / s_i^{\delta}$
else
while there is not enough room in cache for *i* **{**
let *L = min(*H_i*),* for all *i* in cache
evict *i* such that $H_i = L$
}
load *i* into cache

$H_i = L + (f_i + ff_i)^{\lambda} \times c_i / s_i^{\delta}$
} ***while*** (condition)
update (future frequency)
}
}

Figure 1: Pre-IPGDFS# algorithm.

Simulation Model for the Experiment

In case of Web Servers, a very simple Web server is assumed with a single-level file cache. When a request is received by the Web server, it looks for the requested file in its file cache. A *cache hit* occurs if the copy of the requested document is found in the file cache. If the document is not found in the file cache (a *cache miss*), the document must be retrieved from the local disk or from the secondary storage. On getting the file, it stores the copy in its file cache so that further requests to the same document can be serviced from the cache. If the cache is already full when a file needs to be stored, it triggers a replacement policy.

Our model also assumes file-level caching. Only complete documents are cached; when a file is added to the cache, the whole file is added, and when a file is removed from the cache, the entire file is removed.

For simplicity, our simulation model completely ignores the issues of *cache consistency* (i.e., making sure that the cache has the most up-to-date version of the document, compared to the master copy version at the original Web server, which may change at any time).

Lastly, caching can only work with static files, dynamic files that have become more and more popular within the past few years, cannot be cached.

Workload Traces

In this study, logs from three different Web servers are used: a Web server from an academic institute, Symbiosis Institute of Management Studies, Pune, India; a Web server from a manufacturing company, Thermax, Pune, India, and a Web server for an E-Shopping site in UK, www.wonderfulbuys.co.uk.

Experimental Design

This section describes the design of the performance study of cache replacement policies. The discussion begins with the factors and levels used for the simulation. Next, we present the performance metrics used to evaluate the performance of each replacement policy used in the study.

Factors and Levels

There are two main factors used in the in the trace-driven simulation experiments: cache size and cache replacement policy. This section describes each of these factors and the associated levels.

Cache Size

The first factor in this study is the size of the cache. For the Web server logs, we have used seven levels from 1 MB to 128 MB. Similar cache sizes are used by many researchers [3, 18, 19, 22, 23, 24, 25]. The upper bounds represent the *Total Unique Mbytes* in the trace, which is essentially equivalent to having an infinite size cache [29]. An infinite cache is one that is so large that no file in the given trace, once brought into the cache, need ever be evicted [25, 28].

It allows us to determine the maximum achievable cache hit ratio and byte hit ratio, and to determine the performance of a smaller cache size to be compared to that of an infinite cache.

Replacement Policy

We show the simulation results of GDSF, GDSF#, IPGDSF#, and Pre-IPGDSF# for the Web server traces for hit rate, and byte hit rate. For these algorithms, we consider the cost function as one. In GDSF#, IPGDSF#, and Pre-IPGDSF#, we use the best combination of $\lambda = 2$ and $\delta = 0.9$ in the equation for H_i.

Performance Metrics

The performance metrics used to evaluate the various replacement policies used in this simulation are *Hit Rate* and *Byte Hit Rate*.

Hit Rate (HR) Hit rate (HR) is the ratio of the number of requests met in the cache to the total number of requests.

Byte Hit Rate (BHR) Byte hit rate (BHR) is concerned with how many bytes are saved. This is the ratio of the number of bytes satisfied from the cache to the total bytes requested.

Simulation Results

In this section, we present and discuss simulation results for Thermax, Wonderfulbuys, and Symbiosis Web servers.

Simulation Results for Thermax

Figure 2 gives the comparison of Pre-IPGDSF# with other algorithms.

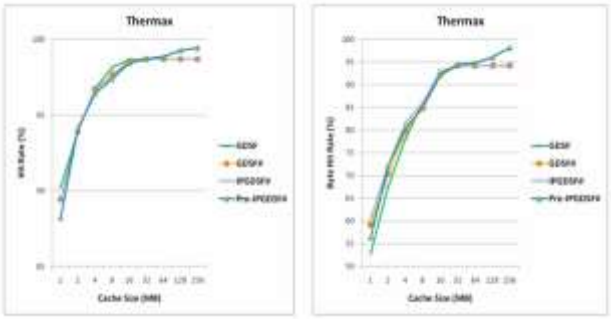

Figure 2: Comparison of Pre-IPGDSF# with other algorithms using Thermax trace

From Figure 2, it can be seen that when a cache size is small, in fact, there is a minor loss in hit rate and byte hit rate compared to other algorithms as can be expected because 20% of the original cache area is now allocated to prefetch buffer. In case of byte hit rate, Pre-IPGDSF# outperforms all other algorithms under consideration.

Thus, for a cache size of 256 MB, there is an increase of 0.75% over GDSF (from 98.71% to 99.46%) in case of hit rate and an increase of 4% over GDSF (from 94.19% to 98.19%) in case of byte hit rate for the Thermax Web server log.

Simulation Results for Wonderfulbuys

Figure 3 gives the comparison of Pre-IPGDSF# with other algorithms.

For the Wonderfulbuys data, at the lower end of cache size, only in case of hit rate there is a minor loss. In case of byte hit rate, Pre-IPGDSF# is superior to all the algorithms under study. At the higher end of cache size, Pre-IPGDSF# proves its superiority over the remaining three algorithms. Thus, for a cache size of 128 MB, there is an increase of 0.21% over GDSF (from 99.66% to 99.87%) in case of hit rate and an increase of 0.43% over GDSF (from 99.27% to 99.70%) in case of byte hit rate.

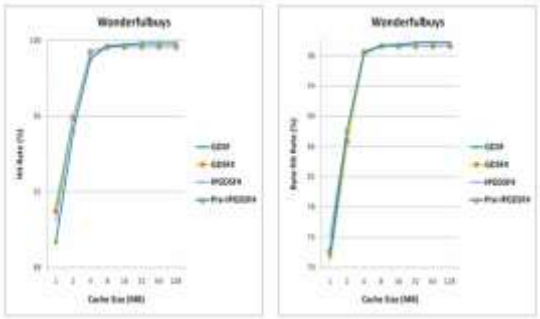

Figure 3: Comparison of Pre-IPGDSF# with other algorithms using Wonderfulbuys trace

Simulation Results for Symbiosis

Figure 4 gives the comparison of Pre-IPGDSF# with other algorithms.

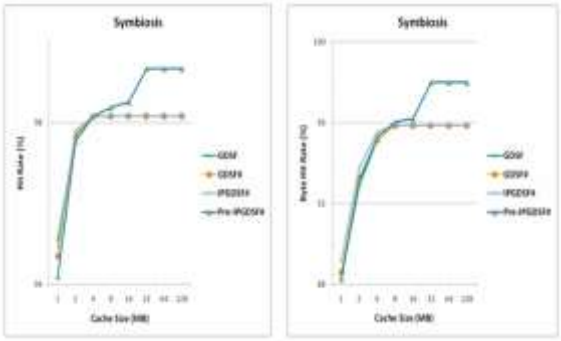

Figure 4: Comparison of Pre-IPGDSF# with other algorithms using Symbiosis trace

For the Symbiosis data, similar trend is noticed. Here, in case of both hit rate and byte hit rate, there is a minor loss for smaller cache sizes. However, at the upper end of cache size, Pre-IPGDSF# easily outperforms the remaining three algorithms under scrutiny. In case of cache size of 128 MB, there is an increase of 0.18% over GDSF (from 98.16% to 99.34%) in case of hit rate and an increase of 2.13% over GDSF (from 95.87% to 98.00%) in case of byte hit rate.

In summary, in case of Web servers, Pre-IPGDSF# outperforms all the other algorithms under study when the cache size is large. Minor loss in smaller sized cache can be explained by the fact that in already small cache, some portion of original cache buffer is utilized for the prefetch buffer.

6. Conclusions

In this paper, we have proposed an integrated algorithm called Intelligent Predictive Web caching algorithm with Prefetching, Pre-IPGDSF#, capable of adapting its behavior based on access statistics. For this we have also proposed a simple but effective static prefetching scheme.

We have integrated this static prefetching algorithm with our caching algorithm IPGDSF#. This algorithm is in turn based on the algorithm, GDSF#, which we proposed in [13, 14, 15, 16, 17]. IPGDSF# considers future frequency in calculating the key value of the document, i.e. we look ahead in time in the request stream and adjust the replacement policy. The future frequency is mined from Web server logs using the simple statistical techniques. We make the policy intelligent by periodically updating future frequency when some condition becomes false.

We have compared Pre-IPGDSF# with cache replacement policies like GDSF, GDSF#, and IPGDSF# for three Web servers using trace-driven simulation method. We have used metrics like hit ratio and byte hit ratio to measure and compare the performance of these algorithms.

We have found that for small cache sizes there is minor loss in case of the metric hit rate. However, for larger cache sizes, Pre-IPGDSF# outperforms all other algorithms in terms of both hit rate and byte hit rate.

References

[1] M. Arlitt, R. Friedrich, & T. Jin, "Workload Characterization of Web Proxy Cache Replacement Policies", In *ACM SIGMETRICS Performance Evaluation Review*, August 1999.

[2] M. Abrams, C. R. Standridge, G. Abdulla, S. Williams, & E. A. Fox, "Caching Proxies: Limitations and Potentials", In *Proceedings of the Fourth International World Wide Web Conference*, Pages 119-133, Boston, MA, December 1995.

[3] M. Arlitt & C. Williamson, "Trace Driven Simulation of Document Caching Strategies for Internet Web Servers", *Simulation Journal*, Volume 68, Number 1, Pages 23-33, January 1977.

[4] L. Cherkasova, "Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy", In *HP Technical Report HPL-98-69(R.1)*, November 1998.

[5] P. Cao & S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", In *Proceedings of the USENIX Symposium on Internet Technology and Systems*, Pages 193-206, December 1997.

[6] S. Jin & A. Bestavros, "GreedyDual*: Web Caching Algorithms Exploiting the Two Sources of Temporal Locality in Web Request Streams", In *Proceedings of the Fifth International Web Caching and Content Delivery Workshop*, 2000.

[7] S. Podlipnig & L. Boszormenyi, "A Survey of Web Cache Replacement Strategies", *ACM Computing Surveys*, Volume 35, Number 4, Pages 374-398, December 2003.

[8] L. Rizzo, & L. Vicisano, "Replacement Policies for a Proxy Cache", *IEEE/ACM Transactions on Networking*, Volume 8, Number 2, Pages 158-170, April 2000.

[9] A. Vakali, "LRU-based Algorithms for Web Cache Replacement", In *International Conference on Electronic Commerce and Web Technologies, Lecture Notes in Computer Science*, Volume 1875, Pages 409-418, Springer-Verlag, Berlin, Germany, 2000.

[10] R. P. Wooster & M. Abrams., "Proxy Caching that Estimates Page Load Delays", In *Proceedings of the Sixth International World Wide Web Conference*, Pages 325-334, Santa Clara, CA, April 1997.

[11] S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, & E. A. Fox, "Removal Policies in Network Caches for World-Wide-Web Documents", In *Proceedings of ACM SIGCOMM*, Pages 293-305, Stanford, CA, 1996, Revised March 1997.

[12] M. F., Arlitt, L. Cherkasova, J. Dilley, R. J. Friedrich, & T. Y Jin, "Evaluating Content Management Techniques for Web Proxy Caches", *ACM SIGMETRICS Performance Evaluation Review*, Volume 27, Number 4, Pages 3-11, March 2000.

[13] J. B. Patil and B. V. Pawar, "GDSF#, A Better Algorithm that Optimizes Both Hit Rate and Byte Hit Rate in Internet Web Servers", *International Journal of Computer Science and Applications*, ISSN: 0972-9038, Volume 5, Number 4, Pages 1-10, 2008.

[14] J. B. Patil and B. V. Pawar, "Trace Driven Simulation of GDSF# and Existing Caching Algorithms for Internet Web Servers", *Journal of Computer Science*, Volume 2, Issue 3, Page 573, March-April 2008.

[15] J. B. Patil and B. V. Pawar, "GDSF#, A Better Algorithm that Optimizes Both Hit Rate and Byte Hit Rate in Internet Web

Servers", *BRI'S Journal of Advances in Science and Technology,* ISSN: 0971-9563, Volume 10, No. (I&II), Pages 66-77, June, December 2007.
[16] J. B. Patil and B. V. Pawar, "GDSF#, A Better Web Caching Algorithm", In *Proceedings of International Conference on Advances in Computer Vision and Information Technology (*ACVIT-*2007),* Co-sponsored by IEEE Bombay Section, Pages 1593-1600, Aurangabad, India, November 28-30, 2007.
[17] J. B. Patil and B. V. Pawar, "Trace Driven Simulation of GDSF# and Existing Caching Algorithms for Web Proxy Servers",
In Proceedings of *The 6th WSEAS International Conference on DATA NETWORKS, COMMUNICATIONS and COMPUTERS (DNCOCO 2007),*
Trinidad and Tobago, November 5-7, 2007, Pages 378-384, ISBN: 978-960-6766-11-4, ISSN: 1790-5117.
[18] F. Bonchi, F. Giannotti, C. Gozzi, G. Manco, M. Nanni, D. Pedreschi, C. Renso, and S. Ruggieri, "Web Log Data Warehousing and Mining for Intelligent Web Caching," *Data and Knowledge Engineering*, Volume 39, Number 2, Pages 165-189, 2001.
[19] F. Bonchi, F. Giannotti, G. Manco, M. Nanni, D. Pedreschi, C. Renso, and S. Ruggieri, "Web Log Data Warehousing and Mining for Intelligent Web Caching,"
In *Proceedings of International Conference on Information Technology: Coding and Computing (ITCC'01* Pages 0599- , 2001.
[20] Q. Yang, and H.H. Zhang, "Web-Log Mining for Predictive Web Caching", *IEEE Transactions on Knowledge and Data Engineering*, Volume 15, Number 4, Pages 1050-1053, July/August 2003.
[21] L.A. Belady, "A Study of Replacement Algorithms for Virtual Storage Computers," *IBM Systems Journal*, Volume 5, Number 2, Pages 78-101, 1966.
[22] H. Braun and K. Claffy, "Web Traffic Characterization: An Assessment of the Impact of Caching Documents from NCSA's Web Server", In *Proceedings of Second International World Wide Web Conference*, Chicago, 1994.
[23] A. Bestavros, R. Carter, M. Crovella, A. Heddaya, and S. Mirdad, "Application-Level Document Caching in the Internet", In *Proceedings of Second International Workshop Services Distributed Networked Environments (SDNE'95)*, Whistler, BC, Canada, Pages 166-173, June 1995.
[24] E. Markatos, "Main Memory Caching of Web Documents", *Computer Networks and ISDN Systems*, Volume 28, Pages 893-905, 1996.
[25] M. Busari, "Simulation Evaluation of Web Caching Hierarchies", *MS Thesis*, Dept of Computer Science, Uni of Saskatchewan, Canada, 2000.
[26] C. R. Cunha, A. Bestavros, & M. E. Crovella, "Characteristics of WWW Client-based Traces", *Technical Report, BU-CS-95-010*, Computer Science Department, Boston University, 1995.
[27] V. Almeida, A. Bestavros, M. Crovella, & A., de Oliveria, "Characterizing Reference Locality in the WWW", In *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, Miami, 1996.
[28] M. Busari & C. Williamson, "On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics", In *Proceedings of IEEE Infocom*, Anchorage, Alaska, April 2001, 1225-1234.
[29] H. Bahn, S. H. Noh, S. L. Min, & K Koh, "Using Full Reference History for Efficient Document Replacement in Web Caches", In *Proceedings of Second USENIX Symposium on Internet Technologies and Systems*, Boulder, Colorado, USA, October 1999.
[30] W. G. Teng, C. Y. Chang, and M. S. Chen, "Integrating Web Caching and Web Prefetching in Client-Side Proxies", *IEEE Transactions on Parallel and Distributed Systems*, Volume 16, Number 5, Pages 444-455, May 2005.
[31] D. Duchamp, D., "Prefetching Hyperlinks", In *Proceedings of the Second USENIX Symposium on Internet Technologies and Systems*, Boulder, CO, 1999.
[32] E. Markatos and C. Chironaki, "A Top 10 Approach for Prefetching the Web", In *Proceedings of INET'98 Conference*, Geneva, Switzerland, 1998.
[33] T. Palpanas and A. Mendelzon, "Web Prefetching Using Partial Match Prediction", In *Web Caching Workshop*, San Diego, CA, 1999.
[34] J. Pitkow, and M. Recker, "A Simple Yet Robust Caching Algorithm Based on Document Access Patterns", In *Proceedings of Second International World Wide Web Conference*, Chicago, 1994.
[35] I. Tatarinov, A. Rousskov, and V. Soloviev, "Static Caching in Web Servers", In *Proceedings of Sixth IEEE International Conference on Computer Communications and Networks (IC3N'97)*, Las Vegas, Nevada, USA, September 1997.
[36] T. M. Kroeger, D. D. E. Long, and J. C. Mogul, "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching", In *Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, December 1997.
[37] Z. Zhang, "An Integrated Prefetching and Caching Algorithm for Web Proxies using a Correlation-based Prediction Model", *M. S. Thesis*, Department of Computing Science, Simon Fraser University, December 2000.
[38] Q. Yang, and Z. Zhang, "Model based Predictive Prefetching", In *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, Pages 291-295, September 03-07, 2001.
[39] B. Lan, S. Bressan, B. C. Ooi, and K. L. Tan, "Rule-Assisted Prefetching in Web-Server Caching", In *Proceedings of the 9th International Conference on Information and Knowledge Management*, Pages 504-511, Washington DC, USA, November, 2000.
[40] Q. Yang, H.H. Zhang, and I.T.Y. Li, "Mining Web Logs for Prediction Models in WWW Caching and Prefetching", In *Proceedings of Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Pages 473-478, August 2001.
[41] O. Yang, H. H. Zhang, and H. Zhang, "Taylor Series Prediction: A Cache Replacement Policy Based on Second-Order Trend Analysis, In *Proceedings of the 34th Hawaii International Conference on Systems Sciences*, IEEE Computer Society, Piscataway, NJ, 2001.
[42] M. Curcio, S. Leonardi, A. Vitaletti, "Integrated Prefetching and Caching for the World Wide Web", *Alcom-FT Technical Report Series, ALCOMFT-TR-01-41*, 2001.