Web Caching and Prefetching: What, Why, and How?

Sarina Sulaiman, Siti Mariyam Shamsuddin
[1] *Soft Computing Research Group, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Johor, Malaysia.*
sarina@utm.my, mariyam@utm.my

Ajith Abraham
Centre for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, Trondheim, Norway.
ajith.abraham@ieee.org

Shahida Sulaiman
School of Computer Sciences, Universiti Sains Malaysia,11800 USM, Penang, Malaysia
shahida@cs.usm.my

Abstract

The demand for Internet content rose dramatically in recent years. Servers became more and more powerful and the bandwidth of end user connections and backbones grew constantly during the last decade. Nevertheless users often experience poor performance when they access web sites or download files. Reasons for such problems are often performance problems which occur directly on the servers (e.g. poor performance of server-side applications or during flash crowds) and problems concerning the network infrastructure (e.g. long geographical distances, network overloads, etc.). Web caching and prefetching have been recognized as the effective schemes to alleviate the service bottleneck, minimize the user access latency and reduce the network traffic. In this paper, we express the discussion on what is the Web caching and prefetching, why we have to opt its and how to pertain of these two technologies.

1. Introduction

Due to the increase of processing capabilities of the single machines connected to the Internet, new and more demanding services have been developed. Multimedia, electronic mail, computer or video conferencing, and, last but not least, very easy to use graphical front ends to the wealth of information accessible via the World Wide Web (WWW) stressed the Internet to its limits. The WWW can be considered as a large distributed information system where users can access to shared data objects. Its usage is inexpensive, and accessing information is faster using the WWW than using any other means. Also, the WWW has documents that appeal to a wide range of interests, for example news, education, scientific research, sports, entertainment, stock market growth, travel, shopping, weather and maps.

However, the recent increase in popularity of the WWW has led to a considerable increase in the amount of traffic over the Internet. As a result, the Web has now become one of the primary bottlenecks to network performance. When objects are requested by a user who is connected to a server on a slow network link, there is generally considerable latency noticeable at the client end. Even if the Internet backbone capacity increases as 60% per year, the demand for bandwidth is likely to exceed supply for the foreseeable future as more and more information services are moved onto the Web [3]. In order to reduce access latencies, it is desirable to store copies of popular objects closer to the user. A loose definition of caching is the movement of Web content closer to the users [1]. Caching popular objects at locations close to the clients has been recognized as one of the effective solutions to alleviate Web service bottlenecks, reduce traffic over the Internet and improve the scalability of the WWW system.

The paper is structured as follows. What and why Web caching is presented in Section 2 that describes on reasons to use Web caching. In Section 3, we show the

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

Web caching works and Section 4 its architectures. Subsequently, we discuss on cache replacement algorithms in Section 5. In Section 6 and 7 we converse on Web prefetching and how to measure performance for Web optimization. Finally, Section 8 gives the concluding remark of our study.

2. What and Why Web caching?

Web caching is the temporary storage of Web objects (such as HTML documents) for later retrieval. There are three significant advantages to Web caching: reduced bandwidth consumption (fewer requests and responses that need to go over the network), reduced server load (fewer requests for a server to handle), and reduced latency (since responses for cached requests are available immediately, and closer to the client being served). Together, they make the Web less expensive and better performing.

Caching can be performed by the client application, and is built in to most Web browsers. There are a number of products that extend or replace the built-in caches with systems that contain larger storage, more features, or better performance. In any case, these systems cache net objects from many servers but all for a single user.

Caching can also be utilized in the middle, between the client and the server as part of a proxy. Proxy caches are often located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections. These systems serve many users (clients) with cached objects from many servers. In fact, much of the usefulness (reportedly up to 80% for some installations) is in caching objects requested by one client for later retrieval by another client. For even greater performance, many proxy caches are part of cache hierarchies, in which a cache can inquire of neighboring caches for a requested document to reduce the need to fetch the object directly.

Finally, caches can be placed directly in front of a particular server, to reduce the number of requests that the server must handle. Most proxy caches can be used in this fashion, but this form has a different name (reverse cache, inverse cache, or sometimes httpd accelerator) to reflect the fact that it caches objects for many clients but from (usually) only one server [2].

3. How Web caching works?

All caches have a set of rules that they use to determine when to serve an object from the cache, if it is available. Some of these rules are set in the protocols (HTTP 1.0 and 1.1), and some are set by the administrator of the cache (either the user of the browser cache, or the proxy administrator).

Generally speaking, these are the most common rules that are followed for a particular request [2]:

1. If the object's headers tell the cache not to keep the object, it will not. Also, if no validator is present, most caches will mark the object as uncacheable.

2. If the object is authenticated or secure, it will not be cached.

3. A cached object is considered *fresh* (that is, able to be sent to a client without checking with the origin server) if:

   i. It has an expiry time or other age-controlling directive set, and is still within the fresh period.

   ii. a browser cache has already seen the object, and has been set to check once a session.

   iii. a proxy cache has seen the object recently, and it was modified relatively long ago.

Fresh documents are served directly from the cache, without checking with the origin server.

4. If the object is stale, the origin server will be asked to validate the object, or tell the cache whether the copy that it has is still good.

Together, **freshness** and **validation** are the most important ways that a cache works with content. A fresh object will be available instantly from the cache, while a validated object will avoid sending the entire object over again if it has not changed.

4. Web caching architectures / deployment schemes

Caching can happen at various levels for example the Web browser of a user, the user's hard disk, servers located in the institution in which the user is employed, the institution's Internet Service Provider (ISP), the regional Internet hub, the national Internet hub or at the global level. Caching can be accomplished by Web browsers; by specialized caches known as proxy caches and by Web servers (see Figure 1). Many popular Web browsers cache the Web pages browsed by the user. Very often such browsers enable the users to view the content downloaded earlier, by pressing a back button. In this case, the Web page is fetched from the browser's cache instead of fetching it again from the

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

original source on the Web, thereby avoiding unnecessary downloads.

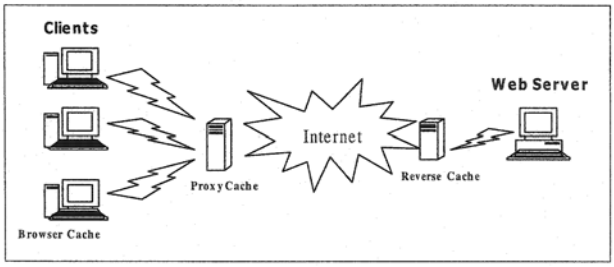

Figure 1. Web caching and prefetching can be implemented at three cache levels; on the client side, at the proxy server and also the website [3].

Features of these three kinds of web cache can be generalized as follows [4]:

In the client: Web caches can be built into most Web browsers. A first level cache for web users is generally implemented within a browser. Because this cache only uses some of the main memory, or a small disk space for storage, the size of a browser's cache is small. However, since the browser cache is the closest cache to the end user, a short response time is provided to the user if the requested objects are cached. Since this kind of cache is embedded into the browser, **the benefits of having cache objects cannot be shared by other users**.

Between the client and the server: A second level cache is usually provided within proxy cache servers using a local hard disk on the gateway server for storage. Proxy caches are often located near network gateways to reduce the bandwidth required over expensive dedicated Internet connections. This kind of cache can be used to store a significant number of files from many web servers and, as an additional benefit, **permits many users to share the resource from the nearest proxy servers** or their neighbor caches. As a result, this type of cache leads to wide area bandwidth savings, improved latency, and increases the availability of the static Web documents.

Near the servers: A reverse cache or inverse cache is placed directly in front of a particular web server. In contrast to a general proxy cache, the reverse cache only handles Web documents from one Web server. This is an attractive solution **to reduce the workload of a busy web server's by caching its static documents** so that the original server can be dedicated to providing service through generating dynamic pages.

The common kinds of caching include browser and proxy caching, active Web caching, adaptive Web caching and push caching. However, another current types of intelligent caching; intelligent Web caching [5], mobile environment for intelligent genetic search and proxy caching [6] and hybrid cache-index forwarding for mobile WWW [7]. A summary of Web caching architectures is depicted in Table 1.

Table 1. Summary of Web caching architectures

Architecture	Description	Advantage	Disadvantage
Proxy (known as forward proxy caching)	Deployed at the edge of the network	Easy to deploy	Single point of failure
- Reverse proxy caching	Deployed near origin	Server farm management	Single point of failure
- Transparent proxy caching	Intercepting HTTP request	Eliminate single point of failure	Violates end-to-end statement
Active Web caching	Applets; Caching for dynamic documents	Caching dynamic documents and personalized cache	Issues of privacy
Adaptive Web caching	Optimizing global data distribution. Consists of multiple, distributed caches which dynamically join and leave cache groups; CGMP, CRP	Tackling "hotspot" phenomenon	**Assumption:** Deployment of cache clusters across administrative boundaries is not an issue.
Push caching	To keep cached data close to those clients requesting that information. (concept of Mirror site)	Targeted providers	**Assumption:** Ability to launch caches which may cross administrative boundaries
Intelligent Web Caching	Applying neural network and network analysis.	Adaptable to environment	High computational Applied on 3-tier design
Mobile Environment for Intelligent Genetic	Implementation of mobile agents with	Efficient search for group of people share	Spend a lot of time for fetching documents

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

Search and Proxy Caching	genetic search and proxy caching algorithms	interests in some subject	from the Internet onto the local disk
Hybrid cache-index forwarding for mobile WWW	Hybrid MowgliWWW and CINDEX schemes to send the caching data information on mobile hosts per document to the base station and transfers all of the cache-index data from the old base station to the new one during the handover phase	Supports for high mobility of mobile hosts and provides a high cache hit ratio.	Does not suffer from wireless network delays, because the cache-index transmission is performed not by the mobile host but by the base station

5. Cache replacement algorithms

The efficiency of proxy caches is influenced to a significant extent by document placement/replacement algorithms. Cache placement algorithms try to place documents within a cache whereas cache replacement algorithms replace documents from a cache. Such algorithms are beneficial only if they can improve the hit ratio. In contrast to cache replacement, **the subject of cache placement has not been well researched [8]**.

Cache replacement algorithms often aim to minimize various parameters such as the hit ratio, the byte hit ratio, the cost of access and the latency. The most widely used cache replacement algorithms [8] include Least Recently Used (LRU), Least Frequently Used (LFU), LRU-Min, LRU-Threshold, SIZE, Lowest Latency First, Hyper-G, Greedy-Dual-Size (GDS), Hybrid, Lowest Relative Value (LRV), LNC-R-W3, Size-adjusted LRU (SLRU), Least Unified-Value (LUV), Hierarchical Greedy Dual (HGD) and Smart Web Caching. Several of these attempted to maximize the hit ratio. These happened depend on evaluation of the cache replacement algorithms performance. A summary of cache replacement algorithms is explained in Table 2.

There are several ways to categorize cache replacement algorithms [9]. Aggarwal *et al.* [10] suggested traditional algorithms, key based algorithms and cost based algorithms as approaches. The differences between these three categorizes are the ways the replacement algorithms retrieve or replace the objects from caches either the least number of times to cache or least frequently retrieve from the cache or based on primary key or use the cost function.

Table 2. Summary of cache replacement algorithms

Algorithm /Policy	Description	Advantage	Disadvantage
LRU	Least Recently Used documents are removed first	Efficient for uniform objects and simple to implement	Only consider time factor
LFU	Least Frequently Used documents are removed first	Simplicity	Only consider time factor. May keep obsolete documents indefinitely
SIZE	Big documents are removed first	High request hit rates	May keep small documents indefinitely; Low byte hit rate
GDS	Advancement of SIZE. Using an H value to remove obsolete small documents	Overcome drawbacks of SIZE	Does not take into account the delays included by the network and the frequency at which documents are accessed
LRV	Using relative value to estimate objects in cache repository	Includes access statistics for all objects. The replacement decision are made in constant time	Needs additional data to be kept in memory. The cost model does not include access latencies for the objects.
Smart Web Caching	Using neural network to estimate cache object priority. Applied with LRU	Cache space efficient; Considering multiple performance factor	High computational power

6. Web prefetching

In the context of Web caching, the term prefetching refers to the operation of fetching information from remote Web servers even before it is requested.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

Objects such as images and hyperlinked pages that are cited in a Web page (say a HTML page) may be fetched well before they are actually called for. It should be noted that a tradeoff occurs. Web objects are prefetched assuming they will be requested in the near future. An accurate selection of such objects would lead to a reduction in the access latency, whereas inaccurate picks would only result in wasted bandwidth.

Prefetching techniques gain significance due to the fact that there are limits on the performance betterment that may be obtained by applying just plain caching [11, 12, 13]. As noted by Douglis *et al.* [12], the rate of change of Web objects puts a barrier on the gains that may be obtainable by employing a proxy cache is limited to about 50% (under their workloads) irrespective of its design. This in effect means that one out of two documents will not be found in the cache. Prefetching is a means to overcome this restriction of plain Web caching strategies (that do not employ it) and often complements such strategies.

According to Lam and Ngan [14] instead of using caching, they study the effectiveness of using prefetching to resolve the problems in handling dynamic web pages. Prefetching is a proactive caching scheme since a page is cached before the receipt of any page request for the page. In addition to the problem of which pages to be prefetched, another equally important question is when to perform the prefetching. To resolve the prediction and timing problems, they explore the temporal properties of the dynamic web pages and the timing issues in accessing the pages to determine which pages to be prefetched and the best time to prefetch the pages to maximize the cache hit probability of the prefetched page. If the required pages can be found in the cache validly, the response times of the requests can be greatly reduced. The proposed scheme is called temporal prefetching (TPF) in which the researcher prioritizes prefetching requests based on the predicted usability of the to-be prefetched pages.

6.1 Web prefetching examples

It is easy to visualize the following three prefetching instances (see Figure 2): prefetching between Web clients and Web servers, prefetching between Web clients and proxy caches, and prefetching between proxy caches and Web servers.

6.2 Web prefetching schemes

Cho [15] provides an interesting approach by considering the speed and moving direction of the mobile user.

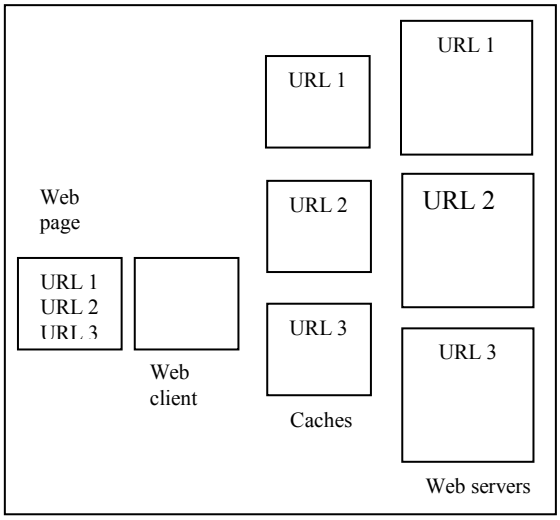

Figure 2. Prefetching possibilities

These two aspects are important elements of the movement pattern. The speed provides about the velocity with which a user changes locations. Moreover, the size of the user's area is largely dependent on the speed.

Whenever the user crosses the borders of the current zone, new prefetching zones is computed. Depending on the speed in the moment that the user leaves the scope of a zone, the new one considers more or less adjacent network cells. Frequency prefetching method [16] analyzes mobility pattern of user accumulated during fixed period. Informations that is worth being used to the future with this are prefetched. Frequency is based on the speed. If predict with data that is accumulated during given period, there is problem to itself. To solve this problem, some factors should be added, and Yoo *et al.* [17] propose Frequency, Interest and Popularity (FIP) scheme that mobile users have preference.

Dar *et al.* [18] propose to invalidate the set of data that is semantically furthest away from the current user context. This includes the current location, but also moving behaviors like speed, direction of the user. Ye *et al.* [19] make use of predefined routes to detect the regions of interest for which data is required. In such a way they have location information for the whole ongoing trip and do not have to compute the target areas while on the move.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

In [20], an adaptive network prefetching scheme is proposed. This scheme predicts the files' future access probabilities based on the access history and the network condition. The scheme allows the prefetching of a file only if the access probability of the file is greater than a function of the system bandwidth, delay and retrieval time.

Prefetching method is a well established technique to improve performance in tradition distributed systems based on fixed nodes, and several papers exist about this topic [21,22]. Some papers have also considered the utility of this technique in the framework of mobile computing, in general from the viewpoint of improving the access to remote file systems; the use of mobility prediction has been also considered for this purpose [23,24,25,26].

7. Measuring performance for web optimization

Measurement of the Web caching performance is needed to establish the efficiency of a Web caching solution [8,5,27]. Consequently, some benchmarks or standards are necessitated which the performance of a particular Web caching solution may be evaluated. Such benchmarks may assist in choosing the most suitable Web caching solution for the problem we countenance. In this situation, a possibility for a particular structure may beneficial for certain applications while other applications may require some other substitutes.

Simultaneously, some organizations may choose for proxy based caching solutions. They may try to overcome the problem of configuration Web browsers by forcing the use of browsers that provide auto-configuration. In the case of massive organizations, they will may use of network components such as routers and switches [8, 28]. Another alternative, they can employ transparent caching. Some organizations may prefer highly scalable solutions anticipating future needs. Besides, organizations which Web sites contain highly dynamic content might occupy Active Cache [29] or possibly will utilize Web server accelerators. Obviously, the subject of measurement of performance is controlled not just to find the competence of a given Web caching solution but also to cover evaluation of the performance of cache consistency protocols, cache replacement algorithms, the role of fundamental protocols such as HTTP and TCP and etc.

7.1 Parameters for measuring Web performance

Several metrics are commonly used when evaluating Web caching policies [30]. These include the following [31]:

i. Hit rate – the hit rate is generally a percentage ratio of documents obtained through using the caching mechanism versus the total documents requested. In addition, if measurement focuses on byte transfer efficiency, weighted hit rate is a better performance measurement [32].
ii. Bandwidth utilization – an efficiency metric. A reduction in the amount of bandwidth consumed shows the cache is better.
iii. Response time/access time – the response time is the time takes for a user to get a document.

The are various parameters such as user access patterns, cache removal policy, cache size and document size that can significantly affect cache performance. Other common metrics that are used to quantify the performance of Web caching solutions proposed by Mohamed [5] including hit ratio, byte hit ratio, response time, bandwidth saved, script size and current CPU usage.

Performance of Web caching solutions may be quantified by measuring parameters such as those listed as follow [8]:

i. price
ii. throughput (e.g. the number of HTTP requests per second generated by users, the rate at which a product delivers cache hits etc.)
iii. cache hit ratio (the ratio of the number of requests met in the cache to the total number of requests)
iv. byte hit ratio (the fraction of the number of bytes served by the cache divided by the total number of bytes sent to its clients)
v. the number of minutes until the first cache hit/miss after a breakdown
vi. the cache age (the time after which the cache become full)
vii. hit ratio/price (e.g. hits/second per thousand dollars)
viii. downtime (e.g. time to recover from power outrages or cache failures).

Techniques for measuring the efficiency and usefulness of Web caching solutions have been evolving slowly. This is because Web caching is a relatively new discipline. In Web caching, theory has advanced much faster than practice [8].

Although quantifying the performance of caching clarifications, aspects for example client side latencies, server side latencies, aborted requests, DNS lookup

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

latencies, cookies, different popularity characteristics among servers, the type of content, network packet losses etc must not be ignored. Many of these parameters are often interrelated. For example, hit ratio is affected by inadequate disk space in a cache server, by insufficiencies in the object placement/replacement policies, by network overload and so on. Maximizing a single parameter alone may not be adequate [8].

8. Conclusion

The Web caching and prefetching technologies are the most popular software based solutions [33, 34]. Caching and prefetching can work individually or combined. The blending of caching and prefetching enables doubling the performance compared to single caching [13]. These two techniques are very useful tools to reduce congestion, delays and latency problems. Consequently, basic knowledge on how these both techniques work; architectures/deployment scheme, placement and replacement algorithms and lastly how to measure its performance are essential to realize an accomplishment of the Web caching and prefetching.

9. Acknowledgements

This work is supported by MOSTI and RMC, Universiti Teknologi Malaysia, MALAYSIA. Authors would like to thank *Soft Computing Research Group* (SCRG) for their continuous support and fondness in making this study a triumph.

10. References

[1] Krishnamurthy,B., and Jennifer.,R., *Web Protocols and Practice*. First edition. Upper Saddle River: Addison Wesley, 2003, pp. 407-441.

[2] Web Caching, *Caching Tutorial for Web Authors*, 2008. http://www.web-caching.com/mnot_tutorial/intro.html.

[3] Wang, J., *A Survey of Web Caching Schemes for the Internet.* ACM Computer Communication Review, 25(9): 1999, pp.36-46.

[4] Wang, Y., *A Hybrid Markov Prediction Model for Web Prefetching*, Master thesis, Department of Electrical and Computer Engineering, University of Calgary, Alberta, 2003.

[5] Mohamed, F., *Intelligent Web Caching Architecture*. Master thesis, Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, Malaysia, 2007.

[6] Cvetkovi´c, D., Peši´c, M., Petkovi´c, D., Milutinovi´c, V., Horvat, D., Ko˘Covi´c, P., and Kova˘ cevi´c, V., *Architecture of the Mobile Environment for Intelligent Genetic Search and Proxy Caching*. Telecommunication Systems 18:1–3, 2001, pp. 255–270.

[7] Ahn, K. H. and Han, K. J., *A Hybrid Cache-Index Forwarding Scheme for Mobile WWW*, CIC 2002, LNCS 2524, 2003, pp. 461–469.

[8] Nagaraj, S. V., *Web Caching and Its Applications*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2004.

[9] Podlipnig, S. and Boszormenyi, L., *A Survey of Web Cache Replacement Strategies,* ACM Computing Surveys, 35(4): 2003, pp.374-398.

[10] Aggarwal, C., Wolf, J. L. and Yu, P. S., *Caching on the World Wide Web*, IEEE Trans. on Knowledge and Data Engg., 11(1):95-107, January 1999.

[11] Abrams, M., Standridge, C. R., Abdulla, G., Williams, S., and Fox, E. A., Caching proxies: Limitations and Potentials, *In Proceedings of the 4th International WWW Conference*, Boston, MA, December 1995.

[12] Douglis, F., Feldmann, A., Krishnamurthy, B., and Mogul, J., *Rate of Change and Other Metrics: A Live Study of the World-Wide Web,* In Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems (USITS-97), 1997, pp. 147-158.

[13] Kroeger, T. M., Long, D.D.E., and Mogul, J.C., *Exploring The Bounds of Web Latency Reduction from Caching and Prefetching,* in Proceedings of the USENIX Symposium on Internet Technology and Systems, 1997, pp. 13-22.

[14] Lam K. Y., and Ngan, C.H., *Temporal Pre-Fetching of Dynamic Web Pages*, Information Systems Journal, Elsevier, Volume 31, 2006, pp. 149-169.

[15] Cho, G., *Using Predictive Prefetching to Improve Location Awareness of Mobile Information Service*, Lecture Notes in Computer Science Vol. 2331, 2002, pp. 1128-1136.

[16] Choi, I.S., *Applying Mobility Pattern to Location-aware Mobile Information Services*, Master Thesis, Chonbuk National University, Korea, 2003.

[17] Yoo, J. A., Choi, I. S., and Lee, D. C., *Prefetching Scheme Considering Mobile User's Preference in Mobile Networks,* Springer-Verlag, Berlin Heidelberg, ICCSA, 2005, pp. 889-895.

[18] Dar, S., Franklin, M. J., Jónsson, B. T., Srivastava, D., and M. Tan, *Semantic Data Caching and Replacement*, Proc. 22nd VLDB Conf. Mumbai, Bombay, India, 1996, pp. 330-341.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE

[19] Ye, T., Jacobsen, H.-A., and Katz, R., *Mobile Awareness in a Wide Area Wireless Network of Info-Stations,* Proc. of MobiCom'98, 1998, pp.109-120.

[20] Jiang, Z., and Kleinrock, L., *An Adaptive Network Prefetch Scheme*, IEEE Journal on Selected Areas in Communications, 16(3):1–11, April 1998.

[21] Korner, K., *Intelligent Caching for Remote File Service*, Proc.of the ICDC1990, 1990, pp.220-226.

[22] Patterson, R. Hugo, G., Garth, A., and Satyanarayanan, M., *A Status Report on Research in Transparent Informed Prefetching*, ACM SIGOPS Operating Systems Review, v.27 n.2, pp.21-34, April 1993.

[23] Liu, G., *Exploitation of Location-Dependent Caching and Prefetching Techniques for Supporting Mobile Computing and Communications*, Proc. of WIRELESS-94, 1994, pp.1-6.

[24] Liu, G.Y., and Maguire, G.Q., *A Predictive Mobility Management Scheme for Supporting Wireless Mobile Computing*, Proc. Int. Conf. ICUP-95, Tokyo, Japan, 1995, pp.268-272.

[25] Liu, G. Y., and Maguire, G.Q., *A Class of Mobile Motion Prediction Algorithms for Wireless Mobile Computing and Communication*, Mobile Networks and Applications, v.1 n.2, pp.113-121, Oct. 1996.

[26] Kistler, J. J., and Satyanarayanan, M., *Disconnected operation in the Coda File System*, ACM Transactions on Computer Systems v.10 n.1, pp.3-25, Feb. 1992.

[27] Kobayashi H., Yu S-Z, *Performance Models of Web Caching and Prefetching for Wireless Internet Access*, International Conference on Performance Evaluation: Theory, Techniques and Applications, PerETTA 2000, 2000, pp.1-8.

[28] Krishnamurthy B., and Rexford J, *Web Protocols and Practice: HTTP 1.1, Networking Protocols, Caching and Traffic Measurement,* Addison Wesley, 2001.

[29] Cao P., Zhang J., and Beach K., *Active Cache:Caching Dynamic Contents on The Web,* Distributed Systems Engineering, 6(1):1999, pp.43-50.

[30] Shi Y., Watson E., and Chen Y-S, *Model-Driven Simulation of World-Wide-Web Cache Policies,* Proceedings of the 1997 Winter Simulation Conference: 1997, pp.1045-1052.

[31] Abrams, M., *WWW:Beyond the Basics,*1997. http://ei.cs.vt.edu/~wwwbtb/fall.96/bppk/chap25/index.html.

[32] Abrams M., Standridge C.R., Abdulla G, Williams S, and Fox EA, *Caching proxies: Limitations and Potentials*, Virginia Polytechnic Institute & State University, Blacksburg, VA, 1995.

[33] Acharjee, U., *Personalized and Intelligence Web Caching and Prefetching*, Master thesis, Faculty of Graduate and Postdoctoral Studies, University of Ottawa, Canada, 2006.

[34] Garg, A., *Reduction of Latency in the Web Using Prefetching and Caching*. Doctor of Philosophy thesis, University of California, Los Angeles, United State., 2003.

978-1-4244-2328-6/08/$25.00 © 2008 IEEE