A Framework for Managing QoS and Improving Performance of Dynamic Web Content

Prasant Mohapatra and Huamin Chen
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824

Abstract— The proportion of dynamic objects has been growing at a fast rate in the world wide web. However, because of additional resource requirements and the changing nature of these objects, the performance of accessing dynamic web contents has been observed to be poor in the current generation web services. We propose a framework called *WebGraph* that helps in improving the response time for accessing dynamic objects. The WebGraph framework manages a graph for each of the web pages. Both the nodes and the edges have attributes that are used in managing the web pages. Instead of recomputing and recreating the entire page, the node and edge attributes are used to update a subset of the weblets are are then integrated to form the entire page. In addition to the performance benefits in terms of lower response time, the WebGraph framework facilitates web caching, QoS support, load balancing, overload control, personalized services, and security for both dynamic as well as static web pages.

Keywords—**Dynamic Web Requests, Internet, Quality of Service, World Wide Web, WebGraph, Weblet, Web Servers.**

I. Introduction

Performance and management of web servers has been a very active area of research in recent years. Several techniques have been proposed for improving the performance and management of web services. Some of the most common techniques that have been proposed include mirroring, caching web contents at proxy servers, distributed server farms with load balancers. These approaches are effective for web sites that have predominantly static web contents. However, all of these techniques are limited in terms of handling dynamic requests, scalability, overload, personalized services, and Quality of Service (QoS) assurances.

The current generation web service used in the e-commerce environment suffers from several serious problems. Because of the presence of a high proportion of dynamic contents, the response time for acessing these sites has been very poor [4]. Both network and server contribute to the response delay. Poor response delay lead to significant revenue losses in e-commerce environments. The revenue loss in 1998 was estimated to be 1.9 billion dollars owing to long response delays [22]. In addition, overload conditions have serious impositions on the performance of web servers. Furthermore, QoS support through service differentiation and personalized services are highly desirable features in most web sites, especially the ones used in commercial environments.

An examination of several dynamic pages on the web revealed that in most cases, only a part or a few parts of the pages are dynamic in nature. Other portions of these pages constitute static images or text. However, for every access of the dynamic pages, the entire page gets constructed and rendered to the clients or the proxies. Thus the static characteristics of these pages are not being exploited in the current model of web access. Our initial motivation was to exploit this nature of the dynamic pages. Thus we developed a framework, called *WebGraph* that uses a graphical representation of web pages to serve dynamic pages very efficiently. Parts of the web page that have the same attributes are called *weblets*. The weblets can be static or dynamic and are used to construct and reconstruct the web pages. In addition, we have also enriched the framework such that it can be used for other important attributes such as overload control, QoS assurances, caching efficiency, personalized services, and load balancing in web servers. This enrichment is achieved by attaching attributes to the graph elements (nodes and edges) in the framework. WebGraph interacts between the primary server(s) and the proxies or edge servers while facilitating the service of requests from the clients.

In addition to the performance benefits, overload control, and QoS support, WebGraph also facilitates personalized and value-added services, easy management, and publishing of web contents. The paper discusses and justifies these claims. We also outline the implementation and management details of the framework in addition to the discussions on its impact on the performance, caching behavior, overloads, and QoS.

The rest of the paper is organized as follows. The WebGraph framework is discussed in detail in Section II. The impact and benefits of the scheme is discussed in Section III. The experimental set-up and preliminary results are depicted in Section IV. The related work is discussed in Section V, followed by the concluding remarks in Section VI.

II. WebGraph Framework

The main idea of the WebGraph framework is to divide the dynamic web pages into multiple components known as weblets. Web pages can be formed from these weblets by creating templates in a markup language. Weblets typically represent parts of a dynamic page where changes do not happen (static parts) or happen concurrently (dynamic parts). Thus, a weblet can be a static object (representing an image or text) or a dynamic object (representing a part of the page where data changes dynamically). Upon access to a dynamic page, instead of recreation of the entire page, only the weblets (based on their attributes as discussed later) that have changed are recreated and integrated with the precomposed partial page.

Unlike other scripting languages like Active Server Pages [2] and JavaServer Pages [19] in which all the scripts within a dynamic page must be executed serially, in WebGraph these components are capable of being executed independently in parallel and apart from the template. Thus any change in a dynamic page is reflected by re-executing only the related weblet(s).

A web page can have multiple weblets, and conversely a weblet can be embedded in multiple web pages. Furthermore, a weblet can include one or more weblets. A web page can be represented as a directed graph where the nodes are weblets and the edges represent the inclusion relationship. A directed edge from weblet-A to weblet-B indicates that weblet-A is included in weblet-B. Several attributes are defined for each of the nodes as well as the edges. Node attributes are different from the edge attributes. Node attributes refer to the properties of the weblets, whereas the edge attributes define the control information regarding the inclusion of the node in rendering the web page. These attributes provide information to the primary and proxy servers on how to manage

This work was supported in part by the National Science Foundation under Grant No. CCR-9988179.

the data efficiently. The attributes could include information regarding the caching nature, security, quality of service, time to live, and any other related information.

To clarify the concept and the structure of the WebGraph, let us consider an example of the web page of a company xyz.com shown in Figure 1. The top and bottom of the pages include advertisements. The global advertisement (GA) targets a national audience, and the local advertisement (LA) targets the audience within a smaller region. Like the network television (i.e., ABC, NBC, CBS, etc.), which allows the local channels to include local advertisement, WebGraph facilitates the proxies to include localized advertisements or any other information (static or dynamic) that are of local interest. The site also contains news articles and weather information in addition to display of transactions on which the company thrives. Each of these components contain static as well as dynamic elements. For example, the news would contain textual links, static images, and stock indexes, that need updating at different time intervals.

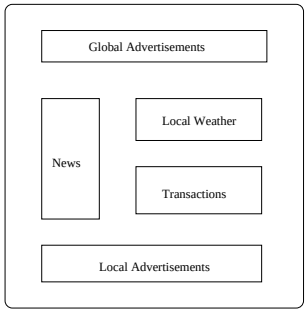

Fig. 1. An example web page for a company xyz.com.

In the WebGraph framework, the page shown in Figure 1 is represented as a graph as shown in Figure 2. The nodes represent different weblets and the edges denote their inclusiveness. The direction of the edges define the dependence relation of the weblets. The "News" weblet includes static images, textual links, stock indexes and the site logo. All of these weblets are static excepting the stock indexes. The weblet with site logo is included in multiple weblets. In the WebGraph of Figure 2, the dynamic weblets are stock indexes, radar images, current conditions and forecast, account profile, shopping cart, GA, and LA. The node attributes are marked as W1–W14, and the edge attributes are marked as L1–L15.

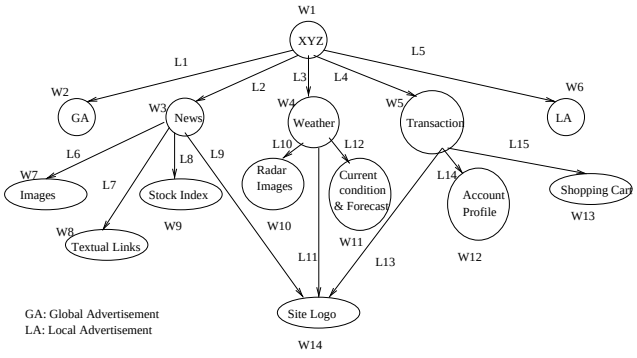

Fig. 2. The WebGraph for the web page of xyz.com.

The possible attributes for nodes (should be considered just as an example) are tabulated in Table I. We have shown four different attributes for the nodes: dependency, TTL, QoS, and security. There could be more or less number of attributes as desired or depending on the applications and the web site composition. The dependency attribute defines the validity of a weblet, which means that if all the weblets on which it depends on (is composed of) are valid then it is considered valid. A weblet is considered valid if its TTL has not expired and its other attributes are satisfied. The leaf nodes have no dependencies. A node is considered invalid if its TTL has expired, and will need to be retrieved from the primary server. A TTL of "0" means that the object is not cached and is retrieved for every access. The nodes that have dependent nodes do not have any TTL as they are not updated as a whole. The third attribute for the nodes is the QoS. As an example, we have considered three levels of QoS: A, B, and C, corresponding to delay-sensitiveness, throughput-sensitiveness, and loss-sensitiveness, respectively. For example, the stock index node would be delay sensitive, the radar image node would be throughput sensitive, and the account profile would be loss sensitive. The nodes with dependecies would have the QoS field as blank because their QoS would depend on the QoS requirements of the dependent weblets. The fourth parameter in Table I refers to the security requirements of a site. Some weblets may require the use of a secure protocol and authentication while others may be accessed by anyone. The security attribute makes that differentiation. A node that has dependable nodes with different security level is marked as "mixed".

TABLE I
NODE (WEBLET) ATTRIBUTES OF THE WEBGRAPH.

Weblet	Dependency	TTL	QoS	Security
W1	W2,W3,W4,W5,W6	-	-	mixed
W2	-	10	B	no
W3	W7,W8,W9,W14	-	-	no
W4	W10,W11,W14	-	-	no
W5	W12,W13,W14	-	-	mixed
W6	-	5	B	no
W7	-	1440	B	no
W8	-	60	C	no
W9	-	5	A	no
W10	-	15	B	no
W11	-	30	C	no
W12	-	0	C	yes
W13	-	0	C	yes
W14	-	10,000	B	no

Table II shows the possible attributes of the edges. Although these attributes are shown in a binary mode in our example, it need not have to be that way. The attributes in all the three columns indicate if an edge would be removed on not based on the increase of load, inability to satisfy the required QoS, or due to security reasons. For example, under high server load, L6 and/or L10 could be removed. Similarly if the required QoS cannot be satisfied then some objects may provide incorrect or useless information. A delayed stock index value may not be useful or a very low quality radar image may be incomprehensible and thus be useless. The third attribute controls the security aspects. In case of any breach in security or while servicing in an insecure mode, some links should be removed. These four attributes may be changed and other attributes can be added based on the web application environment; our intent is to emphasize on the concept of the proposed framework.

WebGraph interacts between the primary server (the source server) and the proxies and/or the edge servers (e.g. Akamai servers [3]). For the ease of explanation, we do not distinguish between the proxies and the edge servers. The implementation and the support mechanism of WebGraph is the same for both of these types of servers. WebGraph is usually created and updated at the primary server. The proxies maintain and can modify parts of the WebGraph as mentioned later. When a request from a client comes to the primary server through a proxy, the primary server sends the requested page and the corresponding We-

TABLE II
EDGE ATTRIBUTES OF THE WEBGRAPH.

Edges	Load	QoS	Security
L1	yes/no	no	no
L2	yes	no	no
L3	yes	no	no
L4	no	no	yes
L5	yes/no	no	no
L6	yes	yes	no
L7	yes	no	no
L8	yes	no	no
L9	no	yes	no
L10	yes	no	no
L11	no	yes	no
L12	yes	no	no
L13	no	yes	no
L14	no	no	yes
L15	no	no	yes

bGraph (the graph with all its attributes). The proxy stores the WebGraph and the weblets constituting the page, and while rendering the page in response to future client requests, uses the WebGraph framework. The topology of the graph is expected to remain static for an elongated period of time. The overhead for updating changes in the WebGraph is insignificant since the graph topology is unlikely to be changed frequently.

The proxies serve requests using the WebGraph, which reduces the bandwidth demand on the Internet, expedites the client response time, while reducing the load on the primary server. In the WebGraph framework, a considerable amount of load is transferred from the primary server on to the proxy servers. Thus server bottlenecks can be avoided using this framework.

III. IMPACT OF WEBGRAPH

In this section we describe the usage of WebGraph framework and how we can derive benefits from it in terms of performance, quality of service support, load balancing, overload control, and personalized services.

A. Performance Impact

The primary performance parameter in web service is the response time. WebGraph is likely to reduce the response time of web requests significantly because of two reasons. First, the weblet-based approach allows for partial processing of web pages and reduces the unnecessary recomputations of the entire page comprising of one or more dynamic weblets. Thus the processing of complex page structures would be faster. The second reason for the faster response time would be due to the transfer of load (to a certain degree) from the primary server to the proxies. This load transfer reduces the load of the primary server which are more likely to be the bottlenecks compared to the proxies. The load transfer also reduces the bandwidth demand on the Internet as the requests and file transfers from the primary server is reduced. Only the specific weblets are recomputed and sent across (from the primary server to the proxies) rather than transferring the entire page all the time. Since the proxies are geographically closer to the clients, it makes more sense to transfer additional load on to the proxies, thereby reducing the bandwidth demand on the longer access path to the primary servers.

B. Impact on Web Caching

WebGraph provides an efficient solution to the problems in caching dynamic web pages. Since the proxy has the image of the graph that forms the page, all the weblets are cached at the proxies. Upon access to a page, the proxy servers checks the validity of the weblets by examining the node and edge attributes. Only the invalid weblets are requested from the primary server or back-end servers and are used in reconstructing the page, which is rendered to the clients. Since the weblet attributes define their validity, the capacity of the web cache can be utilized very efficiently. Thus, by using WebGraph, dynamic requests can be cached, and only parts of it are retrieved from the primary servers as and when necessary.

C. Quality of Service Support

The QoS support can be handled in a much efficient manner in the WebGraph framework. One of the attributes of the weblet could define the QoS parameter as described in our example in the previous section. The QoS could be related to delay, throughput, loss rate, or any other service parameters. Thus for each of the weblets, the QoS constraint can be imposed for better delivery of data. A weblet that has strict timing requirements is expected to be retrieved in a timely manner. A time-sensitive weblet (where the delayed values are of no use) may get dropped if delayed. Similarly for dynamic images (e.g., directions map from Mapquest.com) the QoS support can be provided through resource allocation using the WebGraph framework.

The edge attribute on QoS-sensitivity defines the links that can be removed if the required QoS requirements cannot be satisfied. For example, a poor quality radar image may be incomprehensible. Thus a significant amount of resources may be saved rather than getting used in data delivery that is of no use. In other words, if the quality of an object falls below certain level of QoS because of resource unavailability, the edges from the WebGraph are removed and those objects are not fetched. This approach results in saving of resources, which could be used by other objects.

Another approach for QoS support can be also provided by WebGraph. For example, different versions of the weblets could be used for representing different quality of the data or image. Based on the QoS requirement, the appropriate weblet can be used for creating the web page. Thus depending on the QoS requirement, different sets of edges will be added or removed. In addition, attributes can also be used for service differentiation; preferred customers are given additional or different weblets representing value-enhanced services, etc.

On another issue, it is being predicted that there will be an influx of handheld wireless devices surfing the Internet. This scenario will create a different problem. The handheld devices may be limited in resources (bandwidth, buffering capacity, power consumption, processing ability) and thus may not be able to handle the rich contents of a web page. Using the WebGraph framework, we can have the attributes of some of the weblets configured to match the needs and resource availability of the clients. For example, dense and high-quality images can be replaced by lightweight and lower quality graphics if they do not hinder the purpose. Thus for such applications, by using the edge attributes, we can reconfigure the topology (by removing weblets corresponding to dense contents and adding weblets corresponding to the lightweight contents) of the graph in the WebGraph framework and render the pages effectively and efficiently.

D. Load Balancing

For traditional web applications, the whole process must reside on one server irrespective of their capacity or load. In the WebGraph framework, the weblets within a dynamic page can be distributed on different servers according to their capacity and workload. Some CPU intensive work can be run on servers with high frequency CPU's. Likewise, I/O intensive work can be run on those with high I/O bus capacity.

In addition, the affinity of the weblets with respect to the nodes can be exploited to better utilize the caching effect at the nodes as proposed earlier in [21]. The scheduling is more flexible and robust than other load balancing schemes. The load balancing approaches using the weblets can be used for clusters both at the primary server as well as at the proxies. At the proxy server clusters the distribution of weblets may ease the manageability and distribute the network access load evenly. Weblets with very short update duration should be spread on different nodes, otherwise a single node may get choked with network accesses to reach the server for updates. Load balancing on the basis of other constraints, such as content-awareness [7], can be acieved through the proposed framework.

E. Overload Control

The response time from overloaded servers is usually unacceptable. As a result, a number of requests are aborted causing severe financial losses in the e-commerce environment. Not much has been reported on the overload control issues is web servers. In fact, most of the current generation e-commerce servers do not employ any overload control support. In the absence of this support, these servers are prone to cause denial of services. In the WebGraph framework, an attribute can be added at the link level, which gets used during the overload situations. Another approach is to have a second version of lightweight weblet for some of the objects that can be comprehensible at a coarser granularity. These attributes are labeled on the basis of the relative priority of different weblets during overloaded configurations. For efficient overload control, the web server must take action before the occurrence of the overload situation. Thus a load indicator can be used to indicate the onset (likelihood of overload in near future) of overload. With the onset of an overload, the primary server or the proxy server can use the attributes of the edges to decide what edges can be removed entirely or be replaced with another edge with a lower service demand. Based on the load index, a prioritized approach of eliminating edges can be deployed. Many other perceived load management approaches can also adopted using the WebGraph farmework.

F. Personalization of Web Pages

The WebGraph framework allows and facilitates personalized services that are of high value and significance in the e-commerce environment. Based on the history of accesses and the cookie information, personalized graphs for different classes of users or individuals can be created. These graphs will be stored at the proxies and will be used upon access by the clients to disseminate data in a very effective manner. The attributes of the weblets and the edges would depend on the personal requirements of the individuals or the class. Several value-added services can be also considered that can be supported by the proposed framework.

Since the proxies also have limited access to some changes in the WebGraph, localized information such as local advertisements, schedule of local events, etc. can be integrated with the original web page. Similar approach is used in the contemporary television services, as indicated earlier. Thus the personalization support can be adopted at different levels; for a single user, a group os users, or for a service locality.

IV. Experimental Implementation and Performance Study

To demonstrate the feasibility and validate the expected performance improvement, we implemented a small prototype set-up of a web server and a proxy server using the WebGraph framework. The server was built on the Apache server 1.3.12 on Microsoft Windows 2000. WebGraph exists as a dynamic loadable module and weblets are Windows DLL. URL's with suffix *wgr* are interpreted by Apache as WebGraph requests and handled by the WebGraph module. The WebGraph configuration is set up in Apache initialization process to read all the templates and load the associated weblet files. In the proxy side, WebGraph is directly integrated into the Apache proxy module.

TABLE III
Weblets used in the experiment.

Name	*Nature*	*Mempry*	*CPU*	*Cache*	*TTL*
A	CPU intensive	90%	80%	yes	10s
B	Reference counter	90%	90%	no	0s
C	Database query	90%	90%	yes	20s

Currently only system load, proxy side cache-ability and TTL attributes are realized in the prototype. The proxy side cache-ability is a directive particular for proxy servers that tells whether a certain weblet can be loaded and executed on the proxy side. For example, the configuration *(memory: 80%, CPU: 90%, cache-ability: yes, TTL: 10s)* of the weblet A tells that the weblet can be loaded for execution only when the current system memory and CPU usage are below 80% and 90%, respectively, and the weblet can be run on both the proxy server and web server and the output is valid for 10 seconds. If the system resource status cannot satisfy the weblet's requirement, an adapted output (probably with a lower resolution weblet) is rendered the client. The content adaptation consumes less system resources than the normal output. The CPU and memory usage are obtained from the Windows NT performance counter and their values are read every 50ms.

We setup a web page with three weblets in it to simulate three different types of tasks that are commonly experienced in dynamic web pages. Table III lists the nature and link attributes of the three weblets in the example page. The performance measurement was done for the proxy and web server configuration with and without WebGraph. The web server has a Pentium III 733MHZ CPU with 128MB RAM, and the proxy is with a Pentium II 350MHZ CPU with 128MB RAM, both run a modified Apache web server on MS Windows 2000. The client runs Netscape browser for Linux. The WebGraph output includes an embeded Javascript code to reload the same URL immediately. It was observed that the memory usage during the experiment kept relatively constant around 80%, so only request processing time and CPU usage are used for performance measurement.

In this test, a relative low end computer acted as a HTTP proxy for the client to simulate the proxy server. In this architecture, all weblets that are not cache-able or cannot meet the load conditions in the proxy server are executed by the primary web server. From the web server's perspective, the incoming traffic is initiated by the proxy and is confined mostly to individual weblets resulting in reduced server load.

Table IV presents the average values under different configurations. The *Location* column indicates where the data were collected. The *Configuration* column identifies the configuration of the server-proxy set-up. *CPU Usage* is measured in percentage, and *Processing Time* is the request processing time in Apache log file measured in seconds. The logging code was modified to provide high resolution of timing. The proxy side worked as a request relay so the processing time includes both request forwarding time and waiting for server response time. The proxy was monitored via Windows 2000 Performance Monitor. *Throughput* is measured in the number of request completion per second. Some of the cells in Table IV are empty. When used behind a proxy with WebGraph functionality, the server sees the requests as individual weblet initiated from the proxy, so the throughput does not make sense here, so does processing time. The CPU Usage for the proxy without WebGraph functionality was not recorded.

From Table IV, we observe that the WebGraph framework provides throughput improvement of about 60% for the small prototype imple-

TABLE IV
PERFORMANCE IMPROVEMENT DUE TO WEBGRAPH.

Node	*Configuration*	*CPU Usage*	*Proc. Time*	*Throughput*
Server	Server and proxy	87%	12.92	1.03
Proxy	(proxy without WebGraph)	-	12.80	1.03
Server	Server and proxy	49%	-	-
Proxy	(proxy with WebGraph)	90%	2.12	1.67

mentation with the simple example. Thus, we believe, there is a very high potential of this framework and needs further validations and enhancements. In addition, as we predicted earlier, the load on the primary server is drastically reduced (by about 40% for the example) even if it is serving much high throughput. We are in the process of evaluating these potentials of the proposed WebGraph framework through a large-scale implementation and considering examples of portal web sites.

V. RELATED WORK

There has been a lot of approaches proposed earlier for improving performance of web servers. Several work has been reported on the web caching issues [6], [8], [9]. Frequently accessed static web objects can be retrieved from the cache saving a significant amount of network bandwidth and delay. Web caching, although effective for most static web sites, is not applicable for dynamic pages, which are usually not cached.

Load balancing has been a very effective technique for managing load at server clusters. Mirror and replication of servers are been extensive used in the commercial environment. Several effective load balancing techniques have been proposed by researchers [5], [10], [12]. Overload control issues in web servers have not been adequately addressed yet. A content-based overload control technique was proposed in [1]. The service differentiation approaches proposed for web servers [13] can be also used for load management. These work have targeted primarily for providing QoS support at the web servers.

There has been very limited work on addressing performance issues of dynamic requests. Cao et al. [11] have proposed the concept of active cache for caching dynamic contents on the web, which may not be applicable for dynamic requests that need to access a back-end server. Further the active cache framework cannot provide all the additional features that WebGraph can provide as discussed in Section III. Caching of dynamic data has been studied in [15] and [18], where the work has focussed on rewriting the server applications to handle the insertion and deletion of cache items. The main goal in these approaches is the improvement of cache hit rates for dynamic requests. A publishing framework for web sites was proposed in [14] that uses a graphical representation of the web pages. However, the graph does not includes any attributes, and thus is used only for managing the publishing aspect. The works in [16] and [17] addressed dynamic page caching by separating static portions from dynamic portions and used special tags in the markup language to instruct the client which part can be cached and how to fetch dynamic content.

Our work is different from all of the previous work on the performance of dynamic requests. Almost all of the earlier works have focussed on the caching behavior of the dynamic requests, which is just one aspect of the WebGraph. Unlike most of the earlier work, no client-side support is needed for WebGraph. Since we are targeting WebGraph implementation for proxies, the performance benefits would be for large groups of clients rather than programming individual clients and deriving performance benefits for them. In addition, as mentioned earlier, WebGraph can support and facilitate several other functionalities, which are equally important in several web applications environments.

VI. CONCLUDING REMARKS

A new framework called *WebGraph* has been proposed in this paper for supporting dynamic as well as static contents in the web environment. The framework operates between the primary servers (the front-end as well as any back-end servers) and the proxies or edge servers. A graphical representation of the web with both node and edge attributes are stored for each of the web pages, which helps in rendering the web pages from the proxies. In addition to the performance in terms of lower response delay, WebGraph also facilitates caching, QoS support, overload control, load balancing, personalized services, and security. It provides a very flexible environment to managing and serving objects in the web environment.

REFERENCES

[1] T. Abdelzaher and N. Bhatti, "Adaptive Content Delivery for Web Server QoS," In *International Workshop on Quality of Service*, London, UK, June 1999.

[2] Active Server Pages, http://msdn.microsoft.com/workshop/server/.

[3] Akamai, Inc., http://www.akamai.com.

[4] V. A. Almeida and M. A. Mendes, "Analyzing the Impact of Dynamic Pages on the Performance of Web Servers," In *Computer Measurement Group Conference*, Anaheim, California, USA , December 1998.

[5] D. Andresen, T. Yang, V. Holmedahl, and O.H. Ibarra, "SWEB: Toward a Scalable World Wide Web Server on Multicomputers," In *Proceedings of the 10th International Parallel Processing Symposium*, Honolulu, Hawaii, USA , April 1996.

[6] M. Arlitt, R. Friedrich, and T.Jin, "Performance Evaluation of Web Proxy Cache Replacement Policies," Performance Evaluation, 39(1-4):149–164, February 2000.

[7] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel, "Scalable Content-aware Request Distribution in Cluster-Based Network Servers," Proc. of the 2000 Annual Usenix Technical Conference, San Diego, June 2000.

[8] O. Aubert and A. Beugnard, "Towards a Fine-Grained Adaptivity in Web Caches," In *Proceedings of the 4th International Web Caching Workshop*, San Diego, California , March/April 1999.

[9] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," IEEE Communications, May 2000.

[10] H. Bryhni, E. Klovning, and O. Kure, "A Comparison of Load Balancing Techniques for Scalable Web Servers," IEEE Network, July 2000.

[11] P. Cao, J. Zhang, and K. Beach, "Active Cache: Caching Dynamic Contents (Objects) on the Web," In *Proceedings of the IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98)*, The Lake District, England , September 1998.

[12] V. Cardellini, M.Colajanni, and P. S. Yu, "Dynamic Load Balancing on Web-server Systems," IEEE Internet Computing, Vol. 3, No. 3, May/June 1999.

[13] X. Chen and P. Mohapatra, "Providing Differentiated Services from an Internet Server," Int. Conf. on Computer Communications and Networks, pp. 214-217, 1999.

[14] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed, "A Publishing System for Efficiently Creating Dynamic Web Content," *IEEE INFOCOMM 2000*, March 2000.

[15] J. Challenger, A. Iyengar, and P. Dantzig, "A Scalable System for Consistently Caching Dynamic Web Data," *IEEE INFOCOMM 99*, March 1999.

[16] F. Douglis, A. Haro, and M. Rabinovich, "HPP: HTML Macro-Preprocessing to Support Dynamic Document Caching," In *USENIX Symposium on Internet Technologies and Systems*, Monterey, California, USA , December 1997.

[17] F. Douglis and J. C. Mogul, "Potential benefits of delta-encoding and data compression for HTTP," In *Proceedings of the NLANR Web Cache Workshop*, Boulder, Colorado, USA , June 1997.

[18] A. Iyengar, J. Challenger, "Improving Web Server Performance by Caching Dynamic Data," In *Proceedings of the 1997 USENIX Symposium on Internet Technologies and Systems*, December 1997.

[19] JavaServer Pages, `http://java.sun.com/products/jsp/`.

[20] J. C. Mogul, "Operating Systems Support for Busy Internet Services," In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, WA , May 1995.

[21] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum, "Locality-Aware Content Distribution in Cluster-Based Network Servers," Proc. of the Conf. on Architectural Support for Programming Languages and Operating Systems, San Jose, CA Oct. 1998.

[22] T. Wilson, "E-biz bucks lost under SSL strain," Internet Week Online, May 20 1999. `htto://www.internetwk.com/lead/lead052099.htm`.