SPEEDING-UP, FILTERING and MANIPULATING the WEB to MEET SPECIFIC USER NEEDS

Christos J. Bouras
Computer Engineering and informatics Department
Research Academic Computer Technology Institute
Patras, Greece
bouras@cti.gr

Agisilaos S. Konidaris
Computer Engineering and informatics Department
Research Academic Computer Technology Institute
Patras, Greece
konidari@cti.gr

Abstract

In our days the Web has become a large repository of information. This information is constantly updated and altered. In this work we describe a software architecture that has been used to implement a tool that may potentially aid individual Web users but also groups of users, to upgrade their browsing sessions and make these sessions more productive. The software that is presented here, uses documented and widely accepted techniques to improve users' browsing sessions. It also utilizes new experimental and emerging techniques. The software can be used in various client/server configurations. We present the software modules and their configuration, as well as the interaction that is carried out between them.

1 INTRODUCTION

The Web has become the largest repository of information known to man. The information is growing exponentially and the information acquiring process has become very complex. Not all information is useful to all users on the Web. Specific users have specific information needs, especially the users that use the Web as a working tool. This need can help users in acquiring the information they truly can use, as quickly and efficiently as possible.

A lot of research has been put in the field of browsing enhancements and Web speed-up techniques. In this work we have put together our previous work [7,8] with the work of many other researchers. We have applied our fragment based, Web speed-up approach, to other performance enhancement techniques and have developed a system that can help users and ISPs at improving Web performance.

This work was supported in part by the PABE-T 2000 project: "Development of an autonomous Intelligent Agent for the implementation of a cooperative browsing model on the Internet with the use of advanced technologies of machine learning". The project was sponsored by the Greek Ministry of Development (General Secretariat of Research & Technology)

The goals of the system described in this paper, are:

1. The provision of advanced browsing services to the Internet users

2. User and mediator driver information filtering

The provision of advanced browsing services mainly consists of browsing speed-up techniques. The information filtering service consists of information fragmentation techniques and the independent manipulation of these information fragments.

This paper is structured as follows. Initially we present a short review of related work and an overview of the software developed. We present the complete software architecture and describe each module in brief. After the architecture presentation we describe the possible usage configurations of the software and present the alternate scenarios. After that, we describe the techniques used to implement the various software modules. Finally we present our conclusions and future work.

2 RELATED WORK

There have been several attempts to deliver software that manipulates Web content and provides Web acceleration. All of these attempts specifically focus on certain parameters such as content filtering (for example Net Nanny - http://www.netnanny.com/, Cybersitter - http://www.cybersitter.com/, etc), acceleration techniques and servers (Volera Excelerator - http://www.volera.com, webROCKET 2002 - http://www.ascentive.com etc.) or simple guidance software (bots) that attempt to "guide" users through the vast amounts of information available, in order to help them find information they actually need.

WebWatcher, a project at CMU, is a "tour guide" agent for the world wide web. It is a actually a software tool that "runs" in parallel with the browser

and highlights information that it believes a user is really interested in, on every page the user visits.

WebMate, another CMU project, is a more sophisticated intelligent agent that provides searching enhancements, offline browsing, HTTP header filtering, HTML error handling and dynamic resource set-up. Of course it also provides browsing enhancements similar to WebWatcher.

The previously described software packages are only two out of many similar "intelligent" software packages that are available, and can filter and speed up browsing sessions. The main difference of the software described in this paper, compared to all these packages is the integration of many different Web manipulation techniques into one package. Our software package takes advantage of a series of proven and widely accepted techniques in order to manipulate the Web to the users' advantage.

3 OVERVIEW OF THE SOFTWARE

The software developed consists of several modules that constantly interact during a browsing session. The underlying Web architecture, that the software may enhance, is the well known client/server architecture. When fully exploited, the software utilizes a 3tier client-mediator server-Web server architecture and the software modules are installed on the clients and one or many mediator servers. In this section we describe all software modules implemented both on clients and the mediator server.

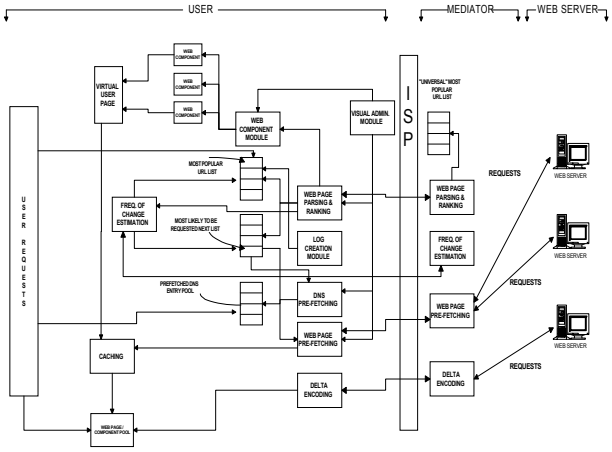

Figure 1. The overall software architecture

The client software is installed as a complete software package containing the software modules described below (shown in Figure 1) and Microsoft Internet Explorer 6.0. The add-on modules are:

1. **Web page parsing and ranking module.** This module is responsible for parsing the already visited Web pages and for ranking the links contained inside them, in order to determine the "most likely to be requested next" links. The module's algorithms take advantage of the historical data maintained by the request filtering and log creation module. The module keeps a "most likely to be requested next" list which contains all requested URLs and a number of links contained in these pages. These links are ranked appropriately.

2. **DNS pre-fetching module.** This module implements DNS pre-fetching utilizing the "most likely to be requested next" list created by the Web parsing and ranking module.

3. **Web page Pre-fetching module.** This module is responsible for pre-fetching Web pages contained in the "most likely to be requested next" list. This module also takes advantage of the overall "most likely to be requested next" list that is maintained by the mediator.

4. **Delta encoding and compression module.** This module is responsible for determining the deltas of changed resources and also compression and decompression. This module is available only in the fully deployed version of the software since the mediator server is a pre-requisite. The information is transferred in deltas only between the clients and the mediator servers.

5. **Web component module.** This module is responsible for fragmenting the Web pages requested and pre-fetched. The module takes advantage of the information inserted by the user in the Visual administration module. It is the most important module in relevance to information filtering, since it receives user preferences as an input and is responsible for creating what we call "Virtual user oriented pages" that consist of information that is required by specific users. For more analysis on this procedure please see [7,8].

6. **Frequency of change estimation module.** This module is responsible for estimating the frequency of change of resources. This estimation plays a very important role in pre-fetching and caching. The estimation relies on the log maintained at the client but mostly on the "universal log" data that is frequently communicated to clients by the mediator server.

7. **Caching module.** The caching module is responsible for maintaining the client cache. It utilizes data from the logging module and the frequency of change estimation module. This

module computes TTLs for all the objects kept in cache.

8. **Log creation module.** This module is responsible for the creation of the software's logs. The most important log is the request log, which keeps track of the users' request patterns.

9. **Visual administrator module.** This module is the user/client software interaction interface. The user "feeds" the system with certain preferences and configuration parameters, which are then used by various other software modules.

The software modules that are installed on the mediator server are the same as those installed on the client but the various parameters they receive are much broader than the ones at the client. This is normal, since, many clients are connected to the mediator server and consequently the mediator server has a much broader view of user preferences. We will not go into detail about the mediator software modules, due to space limitation.

4 USAGE SCENARIOS

4.1 Independent user

In this scenario the software package will have the form of a browser add-on, plug-in or a properly configured version of a commercial browser. Every user will be able to install the software on his/her PC and use its services without having to subscribe or use the services of any independent mediator server. This scenario is shown in Figure 2.

The software installed on the client in this scenario would be activated by a URL request by the user. The following simple algorithm would be executed on the client for every user request:

For every URL requested by user
If in cache then
Use Frequency of change module to determine if it has changed
If has NOT changed then
Show user cached copy
Show user related components
Get most likely to be visited next links from ranking module
Pre-fetch most likely links
DNS-prefetch the rest of the links
Else
GET page from original location
Rank page by using the ranking module
Slice page components with component module
Replace old copy in cache
Get most likely to be visited next links from ranking module
Pre-fetch most likely links
DNS-prefetch the rest of the links
Else
GET page from original location
Rank page by using the ranking module
Slice page components with component module
Save page to cache
Get most likely to be visited next links from ranking module
Pre-fetch most likely links
DNS-prefetch the rest of the links
End if
Loop

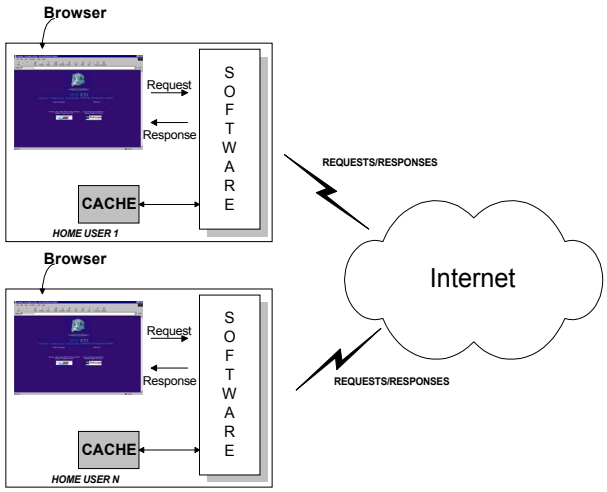

Figure 2.The software installed at the client

4.2 Independent user with the use of a mediator server

This scenario is actually the enhancement of the previous one. The home user will install the software package and will be able to configure a mediator server, in order to use it during their browsing sessions. This procedure is similar to the configuration and usage of contemporary proxy servers with browsers.

It is obvious though, that under this scenario, the installation and configuration of the mediator server software is required. This scenario is presented in Figures 3 and 4.

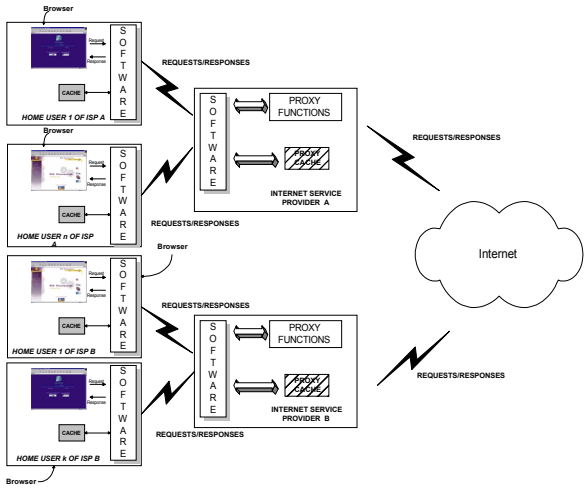

Figure 3. The software installed at the clients and the ISPs

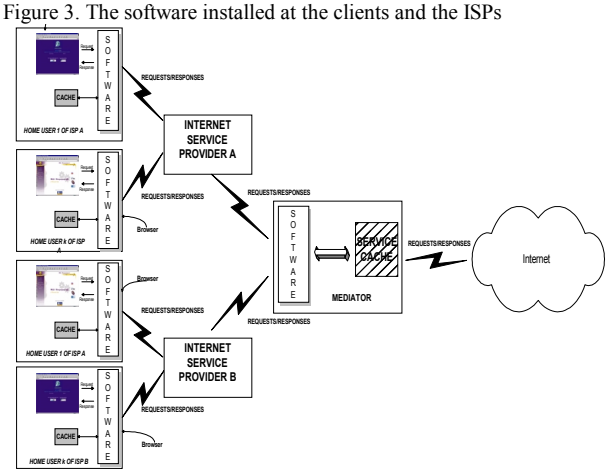

Figure 4. The software installed at the clients and a mediator server

4.3 Intra-business use

In this scenario, a company will purchase the software, and will install it on its server, making it the gateway to the Internet. The company will also purchase client software that will be installed on every company PC connected to the Internet via the server. The advantages of using the software under a intra-company model, is that the specialization of the company and the common interests of the users, enable the proper tuning of the software in order for it to meet specific needs. For example, a stock broker company, may tune the server software to keep track of specific Web sites or even domains in order to acquire specific information online very quickly.

5 BROWSING ENHANCEMENTS IN CLIENTS AND MEDIATORS

5.1 Web page ranking algorithms

The Web ranking algorithms have been used mostly by Web search engines [1,2,3,4]. The ranking of Web pages is a very important procedure to search engines in order for them to provide useful information to users. Web page ranking is one of the most important procedures in the software that has been developed, and is presented as an independent module.

5.2 DNS pre-fetching and caching

The basic problem that users come across on the Internet is known as User Perceived Latency (UPL). UPL is actually the delay that occurs between the time a page request is issued by a user, and the time that the page begins loading in the browser window. One of the most important contributing factors, to UPL is the request setup period that precedes the actual page transfer.

The connection setup involves the translation of the web address to the equivalent IP address. This translation is made possible after the client contacts its configured DNS server that sends back to the client the corresponding IP address of the requested server. The time spent during the client-DNS server interaction significantly contributes to UPL.

The DNS pre-fetching technique attempts to minimize or completely abolish the DNS translation period that is carried out before a request [5]. This is made possible by querying a DNS server before the actual request is issued by the user. This means that a browser software module is responsible for analyzing Web pages, identifying possible "next requests", ranking these possible requests in order of "most likely to be requested" and then querying the DNS server for the corresponding IP addresses. When returned to the browser, these DNS results are kept in a local cache for a period of time in case they are used by the user. It has been shown that this procedure significantly reduces UPL.

5.3 Caching

Caching is one of the key features of the software that has been developed. The usage quality and the quantity of resources kept in cache as well as the Time To Live (TTL) parameter for every resource, are the most important parameters that play a role in the efficiency of caching [6]. The basic issues that must be answered and configured properly are the following:

1. The co-operation of the server sided caches with the client-sided caches of the system. Since server sided caches are much larger and attempt to facilitate large numbers of users, it is obvious that

proper configuration must be carried out at the clients in order for them to keep cached copies of resources that are actually needed by individual clients.

2. The size of server and client sided caches.

3. The TTL configuration of the caches as well as the replacement algorithms that will be utilized.

5.4 Frequency of change estimation

The frequency of change estimation of Web resources problem is a common and very important issue for caching and pre-fetching. It is crucial to be able to estimate the changing frequency of a Web resource in order to know if it should be cached or pre-fetched.

The issue has been analyzed in several research papers. Contemporary Proxy servers tend to use the if-modified-since headers in order to decide whether a resource has changed or not. The basic problem with this procedure is that not all servers on the Internet are able to provide this header for their resources. Thus, proxy servers are usually misinformed, something that leads to unnecessary Web transfers.

Many methods of change frequency estimation have been proposed in the bibliography. Most of them are based in the probabilistic analysis of a selected setup period. This analysis helps to estimate changes after the setup period. The software developed has also used a probabilistic analysis approach based on logs.

5.5 Delta encoding and compression

It is clear that when a Web resource can positively be identified as changed, it usually has not changed completely, but only certain portions of the page have changed. The new version of a Web resource shows many similarities to the old version of the same resource. If only the differences between the new and the old resource could be sent to a client instead of the whole resource, during consecutive requests, it is obvious that Web transfers would be minimized. Delta encoding permits the transfers of deltas (=differences) between a server and the clients, instead of the whole version of a resource. The clients can construct the new version of a resource by using a stored copy of the resource (the older version) and the deltas sent by the server. This method is further optimized with the compression of deltas before they are sent by the server and their de-compression at the client upon receipt.

5.6 Log analysis

Log analysis is considered as a facilitating method for almost all of the software modules that have been implemented.

5.7 Web page pre-fetching

Web page pre-fetching is one of the basic enhancements of the implemented software. The pre-fetching engine is implemented as an independent software module that has two basic characteristics:

1. It can be configured by administrators

2. It constantly updates its functions based on client interactions

In a Web browser-Mediator server-Web server configuration, Web pre-fetching can be carried out between clients and mediators but also between mediators and servers. It is obvious that the pre-fetching engine implemented as a part of the client software must cooperate with the pre-fetching engine implemented as part of the mediator server software.

The client pre-fetching engine utilizes the specific client's request history, current request patterns and mediator server provided most popular lists in order to identify its own most likely to be requested next tables and their TTLs.

The mediator server pre-fetching engine utilizes all the clients' pre-fetching history, current request patterns and constructs the mediator server's most likely to be requested tables and their TTLs.

5.8 Web components

Another basic software module that has been implemented, is the Web component module. This module works together with all the other software modules and permits the fragmentation of Web resource information in order to enable its efficient manipulation. Web fragments are a methodology that has been proposed in the bibliography in many different forms. The basic denominator of all fragmentation techniques is their attempt to cut-down Web pages into independent components and then utilize, manipulate and connect these components as independent entities. This method enables the personalization of information and the efficient reduction of Web transfers. These are two of the basic goals of the implemented software.

6 CONCLUSIONS AND FUTURE WORK

In this paper we have described a software tool that utilizes the client/server architecture, an can efficiently improve user browsing sessions, both in information quality but also in interaction speed. We are currently in the experimental usage phase of the software. It is obvious that many parameters must be tuned properly in order for the software to be as useful as possible to the users. We intend to add more browsing enhancement modules in the future, but also improve our Web component technique. We are confident that our tool will aid individual users but also businesses in acquiring the information that they need efficiently and quickly.

7 REFERENCES

[1] V. N. Padmanabhan and J. C. Mogul, “Using predictive prefetching to improve world wide web latency”, ACM SIGCOMM Computer Communication Review, vol. 26, no. 3, pp. 22-36, 1996.

[2] K. M. Curewitz, P. Krishnan and J. S. Vitter, “Practical Prefetching via Data Compression”, Proc. ACM-SIGMOD Conference on Management of Data, pp. 257-266, 1993.

[3] P. Cao, E. W. Felten, A. Karlin and K. Li, “A Study of Integrated Prefetching and Caching Strategies”, Proc. ACM SIGMETRICS, pp. 171-182, 1995.

[4] M. Crovella and P. Barford, “The network effects of prefetching”, Proc. of the IEEE Conf. on Computer and Communications, (INFOCOM '98), pp. 1232-1240, 1998..

[5] E. Cohen and H. Kaplan, “Prefetching the means for document transfer: A new approach for reducing Web latency”, Proc. IEEE INFOCOM, 2000.

[6] R. Alonso, D. Barbara and Hector Garcia Molina, “Data Caching Issues in an Information Retrieval System” ACM Transactions on Database Systems, 15(3), pp. 359-384, 1990.

[7] C. Bouras and A. Konidaris, “Estimating and Eliminating Redundant Data Transfers Over the Web: A Fragment Based Approach" Proc. 3rd International Conference on Internet Computing (IC 2002), USA, 2002.

[8] C. Bouras and A. Konidaris, "Web Components: A Concept for Improving Personalization and Reducing User Perceived Latency on the World Wide Web”, Proc The 2nd International Conference on Internet Computing (IC2001), USA, Vol 2, pp. 238-244, 2001.