ARCHITECTURE AND IMPLEMENTATION OF A NEW USER INTERFACE FOR INTERNET SEARCH ENGINES

Fidel Cacheda, Alberto Pan, Lucía Ardao, Angel Viña
Department of Tecnoloxías da Información e as Comunicacións,
Facultad de Informática, University of A Coruña
Campus de Elviña s/n, C.P. 15.071 A CORUÑA, SPAIN
e-mail: {fidel, apan, lucia, avc}@gris.des.fi.udc.es

ABSTRACT

In this paper we expose the architecture and the implementation details of a new visualization system suitable to any Internet search engine or directory that tries to improve the search process done by users. Using this new system, users can get a more comfortable and efficient navigation through the search results without opening new windows and downloading simultaneously many Web pages in background. This multi-platform system, which uses a remote cache and many local caches to improve its performance, was designed in a modular way and developed using as base the Web directory called BIWE, developed by us.

1. INTRODUCTION

The World Wide Web dates from the end of the eighties and no one could have imagined its current impact. The boom in the use of the Web and its exponential growth are now well known. Just the amount of textual data available is estimated to be in the order of one terabyte and in addition other media are available. Thus, the Web can be seen as a very large, unstructured but ubiquitous database [1]. This leads to the need of efficient tools to manage, retrieve and filter information from this database: the Internet search engines and the Web directories.

Nowadays much research is made around search engines trying to improve their ranking algorithms, their indexing techniques, etc. But meanwhile the whole search process performed by the users doesn't improve and it is still the same as in the first nineties. Search engines try to get as many documents as possible hoping that their ranking algorithm will be able to place in the first places the most important Web pages. In other words, search engines try to obtain a high *recall*, but this implies reducing quite a lot the *precision* of their search [2].

This implies that users should perform a manual search among the results retrieved by a search engine, usually not checking all the different results, but the first ten or twenty ones. This manual search is quite time consuming because the user should select from the results obtained which one seems to be really suitable, then he has to go to that Web page and have a look to the information showed in the page. Finally, if this page is not what the user is looking for the same process must be repeated.

In this paper we will describe the architecture and the implementation details of a multi-platform system that improves the traditional search process in Internet. The system was developed using as base the Web directory called BIWE (Spanish Web Searcher in Internet) developed by us [3].

In the next section we describe the main types of search processes in Internet search engines and directories. In Section 3 we explain the objectives we want to achieve with this system, and then we are going to describe in detail the proposed system. First, we describe the new user interface used by the system and next we expose in detail the architecture and the implementation of our system. Finally we expose the conclusions obtained from this work.

2. STATE OF THE ART

During the last years research has been focused on search algorithms (trying to include as many factors as possible into the search) and on ranking algorithms (trying to show in first place the most important Web pages) in order to obtain the best precision and recall as possible [4]. But on the other hand, there is not much effort involved in the search process performed by users.

The search process performed by users, in an Internet search engine or directory, has the following steps [6]:

1. The user specifies some keywords related with his information need.
2. The Internet search engine performs the search and sorts the results according to the user's preferences.
3. The sorted results are showed to the user in his navigator.
4. For each "interesting" result, among the 10 or 20 first ones, the user:
   4.1. Clicks over the link.
   4.2. Waits until the main part of the page is downloaded.
   4.3. Checks the Web page downloaded.
   4.4. If it is not interesting then this page is discarded.
5. The user goes back to 1, trying to make a better search changing some keywords.

Most of this steps are automatic and don't require the user to do any effort, except step 4 which is a manual process completely dependent on the user interface used by search engines. This part is quite important because several factors depend directly on it:

- The ability to download, at the same time, many Web pages (parallelism).
- The efficiency to check several times a Web page previously downloaded.
- The better use of users' connection to Internet.

Basically, there are three different methods associated with the search process performed by users. The first one is the classical model used by many Internet search engines and directories. Its main characteristic is that when the user clicks over the link (step 4.1), the new page is downloaded in the same window (see figure 1) where the user had made the search.

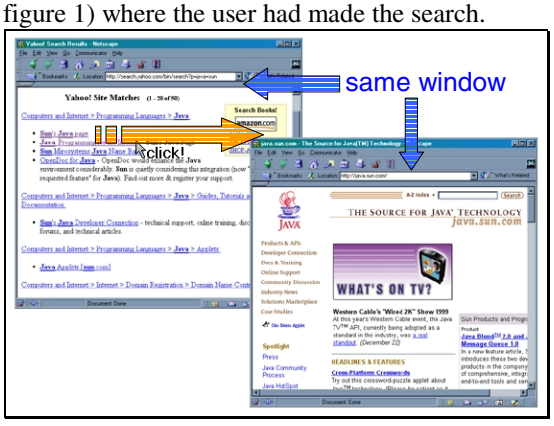

Figure 1: First method

Many important Internet search engines and directories use this method, although it presents many disadvantages:

- The user can just download one page at the same time (there is not parallelism).
- If the user wants to check several times the same page, he depends on his navigator cache and could even have to download again the same page.
- The Internet connection would be most of the time under-used, and therefore this means a waste of time and money for the user.

The second method is a small variation of this one. In this case, when the user clicks over a link a new window is opened and a Web page starts to be downloaded. This modification can be performed directly by users using their navigator options (commonly, most users will do it this way when they are using a search engine of the first method) or the own Internet search engine can implement it easily. In figure 2 we can see an example of this method.

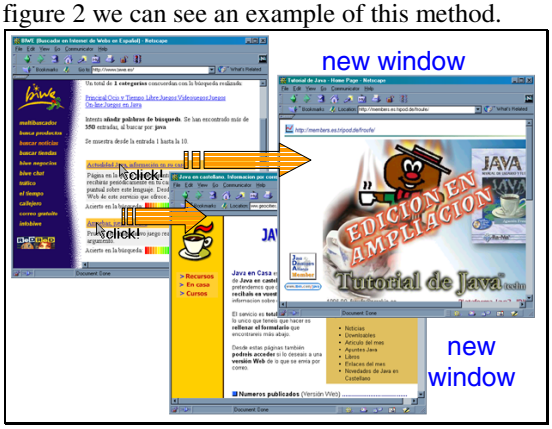

Figure 2: Second method

With this small variation we can achieve several advantages:

- The user can download many pages at the same time (parallelism is supported).
- The user's Internet connection is better used.

But there are still some disadvantages not repaired, and some newer ones:

- The user still depends on his navigator cache to re-check a Web page previously downloaded.
- The number of windows opened could be very high which can lead to failures with the navigator.
- Also, if the number of windows is high, the user can have problems to manage all the different windows (problems to locate a Web page among all windows, etc).

Finally, the third method tries to repair these problems changing the user interface (see figure 3). In this case, the window is divided in two frames: the left one is used to store the list of all the "interesting" pages and the right one is used to show the contents

of each Web page. The user must mark one page as interesting (just clicking over a button) and automatically the page is added to the left frame. Next the user just selects pages in the left side and observes them on the right side.

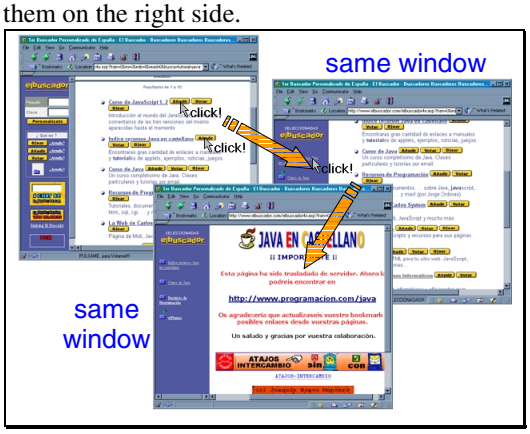

Figure 3: Third method

The main advantage of this method is that the user just works with one window, avoiding all the problems of the second method. But this method still has all the problems of the first one: no parallelism to download the pages, no facilities to re-check Web pages and the Internet connection is under-used.

In this brief state of the art we have described the most important and most used search processes for Internet search engines. Obviously there are other different methods which could not be described here due to space limitations.

3. OBJECTIVES

We want to obtain a system that makes easier the search process performed by users in an Internet search engine or directory. Firstly, the objectives we want to achieve from the server point of view are:

- The system must be multi-platform in order to allow a better installation in any search engine and, of course, to allow its utilization by any client, independently of his operating system.
- The system architecture must be modular to allow its distribution through several computers, to adapt the system to different requirements (number of users, minimum response times, etc.).

And, from the user point of view, we want to develop a new user interface that makes easier the search process performed by users. Specifically, the following objectives must be achieved:

- Improve the user interface of the search engine in order to get a more comfortable navigation through the search results. This interface should reduce as much as possible the number of windows used by the system.
- The user should be able to download simultaneously as many Web pages as he desires, without overload the interface. This way, the Internet connection can be better used.
- The system should provide a way to check several times pages previously downloaded, without depending only on the navigator cache (because users could have previously disabled this option of their browser).

4. THE PROPOSED SYSTEM

In the following sections we are going to describe the new visualization system proposed. This system has been designed and developed in a modular way, and adapted to BIWE, a Spanish Internet directory, also developed by us, which nowadays is one of the most used directories of our country.

4.1. The New User Interface

First of all, in order to achieve these objectives, we have designed a new user interface that will help users in their process of searching information in the Web. But, before we start describing this new interface, it is important to emphasize that this system can be installed in any kind of Internet search engine, without important changes on the normal user interface. Therefore, some users can use this new interface and, at the same time, the rest of them can carry on with their normal search.

In fact, in our implementation, the search process is the same, but the user can select to use the new interface and so, he will get the interface showed in figure 4.

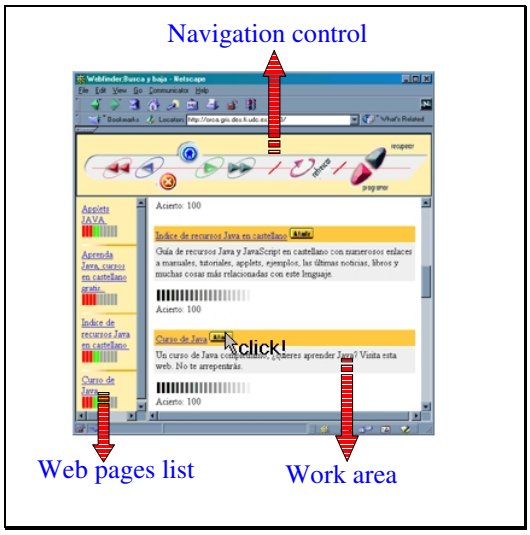

Figure 4: New user interface

Once the user has selected this new interface, his search process is modified. First, he just enters some keywords in the search engine, as he will do in a usual search. But then, things start changing: if he wants to

check a Web page, instead of clicking over the link and opening a new window; now the user must click over a button, which is beside the result he wants to check, and automatically this web page is added to the list of "interesting" pages and is started to getting downloaded by the system without disturbing the user interface. The user can select as many pages as he desires and no new windows will be opened to download these pages; instead all of them are downloaded in background.

To support this novel search process, the new interface is based on one main Web page divided into three independent zones: the work area, the Web pages list and the navigation control.

The **work area** is the biggest and most dynamic part of the interface. This is the place where the user performs his searches and where the search results will appear (like in a common search engine). Besides, the user can select as "interesting" any of the search results, and later, these selected Web pages will be showed in this area (remember that no new windows are opened).

The **Web pages list** is located on the left side of the interface and it shows a list with all the pages previously selected by the user as "interesting". For each page, the information showed is the following: the title of the Web page, where there is a link to its content, and an estimation of the percentage already downloaded for this page. At this point there is an important detail: this link is not to the original URL, but to the URL where this page is getting downloaded which could be local or remote to the user (see Section 4.2). Therefore the estimation pointed before is an important feature of each Web page in order to detect when a page could be visible or not.

Finally, the **navigation control** is the main part of the client side of this system. It is an agent that will interact between the client and the server sides, communicating both of them. On the one hand, it will notify the server of all the user actions and on the other, it will notify the user of any event in the server side. For example, the server is notified through the navigation control when a new user enters the system or if a Web page is removed from the list, and it also notifies the user of the Web pages downloaded by the server, among many other things.

Using this new interface, the algorithm explained in Section 2 has changed, basically in step 4 (remember that the rest are automatic). With this new interface step 4 is divided in the following tasks:

4.1. Add the page to the list of "interesting" Web pages by clicking over a button.
4.2. It is in background that the selected page is downloaded. The user waits until the main part of the page is downloaded (using the information showed in the Web pages list).
4.3. The user checks the Web page.
4.4. If it is not interesting then this page is removed from the list of "interesting" Web pages.

With this new method the advantages achieved are quite important:

- Users can download at the same time several pages (parallelism is achieved).
- The user interface is quite comfortable because just one window is needed to visualize all the pages used by the user: search results and selected Web pages.
- Users can easily access to Web pages previously downloaded independently of the navigator cache.

By the other side, the implementation of this new visualization system requires to the user an optional component to be installed in his computer (see the next section).

4.2. Architecture And Implementation

The architecture is divided into two zones: client and server. On the client side there are two main components: the user interface and optionally, a local cache server. And the server side is made up of other two components: the kernel of the system and a remote cache server.

The **system kernel** is implemented over a Web server using Java Servlets [5]. This is the main part of the system because the kernel interacts with components of both sides; on the one hand it interacts with the client side of each user in order to detect all the user's actions, and on the other hand the kernel also gives orders to the remote cache server.

The system kernel uses the HTTP (HyperText Transfer Protocol) and TCP to interact with the client side and it is used to receive requests from the user. Basically the kernel is notified of the following events on the client side:

- Entry of a user in the system, in order to keep a trace of this user's session.
- Selection of a page as "interesting" by a user, in order to start to download this page.
- Navigation through the list of Web pages.

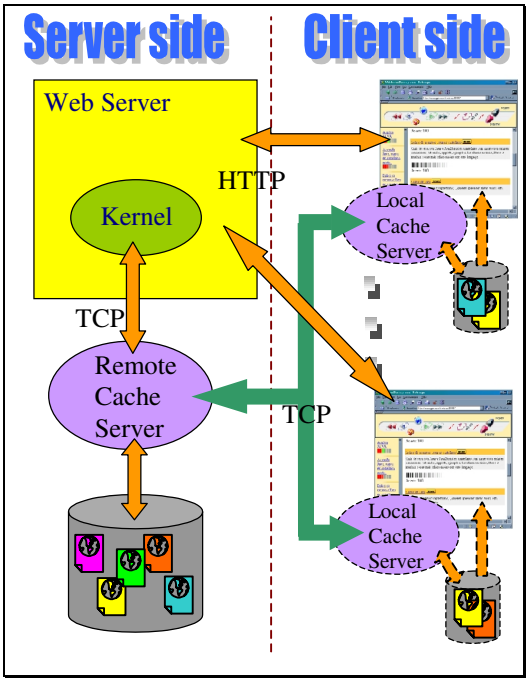

Figure 5: Architecture

The kernel communicates with the remote cache server using directly the TCP protocol through sockets. Basically the kernel, when a user has requested a Web page, notifies the remote cache server of the Web page to download and of the user who requested it. It is important to use sockets to communicate both parts, because the remote cache server can be placed in any other computer, which will increase the modularity of the system,

The **remote cache server** is a Java application that manages a cache in the server side that stores the Web pages requested by users. This application will interact with the kernel and with the local cache servers of each user. As mentioned before, this cache server will be notified by the kernel when a user wants to download a Web page. The cache server will receive which user it is and the URL he wants. Next, this server will connect to the Web site and download all the components of the Web page; this means download not only the HTML source, but also the images, frame sets, etc. Then, it is important to modify the relative links to make them absolute, avoiding problems with references to other Web pages, images, applets, etc. Once the download process has finished, the cache server will store in disk all the components of the Web page, and if it is necessary it will remove some Web pages from disk if the maximum size of the cache was reached.

And the final task of the remote cache server is the communication with the client side of the system, specifically with the local cache server of the user who requested the Web page. First, the kernel will have detected if the user has installed the local cache server, in this case the remote server will send to the local one the Web pages that the user has requested in a gradual way. Therefore, as soon as the remote server has downloaded a component of a Web page requested by a user, this component will be sent back to the local cache server. This communication is done using a TCP connection through sockets.

The **local cache server** is a very similar application (also developed in Java) as the former one. It is an application that must manage a small cache in each client side, in order to store the pages requested by users. Users don't need to install this application in order to use the entire system, but the functionality of the system increases because users can check off-line the Web pages downloaded using the local cache server. In fact, this application is quite small to allow a quick download and a simple installation.

The **user interface** is also an important component of the system because it is the part in charge of the connection between the user and the kernel of the system. The main part of the user interface is the navigation control, which is an applet that controls all user actions and notifies the system kernel of them using the TCP protocol. The Web pages list and the work area are dynamic Web pages, generated in the server side using Java Servlets. In this way, each user will have his own vision of the global cache of the server side.

Some actions of this part are transparent to the user, such as, the connection of a new user to the system, which is directly notified to the system kernel. But the main part of them are common user's actions, such as: select a Web page as interesting by clicking over a button, remove a Web page from the list, refresh the Web pages list and some others.

Once we have seen the architecture and implementation of the system, we can describe the new working operation. Initially, a user will use his Web browser to connect to our system and make a search in the work area of the new interface, where the search results will be showed. Then the user can select a Web page as "interesting" by clicking over a button that will call a Java Servlet of the system kernel. In this way, the kernel is notified that this user wants to download that Web page. In its turn, the system kernel will send a message to the remote cache server to notify it of the Web page to download and which user has requested it. Also, the kernel will refresh the Web pages list of the user's interface, adding the page requested.

At this point, if the user has the local cache server, when the remote cache server has downloaded all the components of the Web page, it will send them to the local cache server of the user who made the request.

Otherwise, the remote cache server doesn't need to perform any action.
Finally, the user can check, in the working area, a Web page previously select just by clicking over the link in the Web pages list, but the place where this link points to is very important. If the user has installed the local cache server then he will access through the link to the page stored in his own computer. In other case, the user will access to the Web page stored on the server side, in the remote cache server through the Web server.
At this point it is important to point up that each user is handled individually using sessions. This implies that each Web page requested by each user is managed separately, although this doesn't mean that each page is stored separately. This is necessary because each Web page is sent back to the user who requested it.

One of the main advantages of this implementation is the high degree of independence of the system, achieved due to Java and the communication structure among all the components. On the one hand, Java allow the system to be platform independent which is quite important in order to obtain a system suitable to be installed over any search engine or directory. On the other, the use of sockets to communicate allows a more flexible architecture because the different layers can be located in different computers. This is quite important in the server side because the system kernel could be placed in a computer (for example, the host where the search engine is running) and the remote cache server in another one.
Another special characteristic of this system is the presence of a cache server in the client side. With this local cache the user can check instantaneously the Web pages previously selected without delays, and these Web pages will remain in the local cache for future enquiries. But on the other side, the user will need to download and install the local cache server on his computer; this is the reason why the local cache server is a small application and with an easy installation process.
Related with the former, is the double cache system used (one remote and many local). This double cache means that users will need two hops to download the Web page required to their computers, which could be a problem because this will slow down the user's navigation. But we have to take into account that the remote cache is global to all users and that in Internet search engines and directories many searches are repeated, which implies that an important percentage of the Web pages required will already be stored in the remote cache. Anyway, Internet search engines and directories usually have a high throughput in order to avoid congestion problems accessing their Web sites, which makes the last hop (remote to local) quite short.

5. CONCLUSIONS

In this paper we have described the architecture and the implementation details of a multi-platform system that improves the traditional search process in Internet search engines and directories.
This system improves the user interface of search engines in order to get a more comfortable navigation through the search results, without opening new windows and downloading simultaneously several Web pages. Additionally, the system allow users to check off-line Web pages previously downloaded using their local cache servers.
In order to get an easier installation and adaptation to any kind of search engines or directories, the system architecture was developed in a modular way and the system implementation was done using the features of a multi-platform language as Java.
This new visualization system uses a double cache system (one remote and many locals) to improve the efficiency of the system, reducing the download times through the remote cache and allowing users to check off-line Web pages through their local cache. On the other side, the local cache server is optional, because users must download and install it. Therefore, users can use the whole visualization system without installing anything, although the performance obtained is reduced.
In future researches we will use this system to schedule the download of Web pages off-line. In this way, users could select the Web pages to download off-line and later, connect to the server and download automatically to the client side the pages scheduled. Finally, users could check off-line in their computers the Web pages selected without additional costs.

ACKNOWLEDGMENTS

This work has been partially supported by PGIDT 99PX11050/B.

REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto, "*Modern Information Retrieval*", Ed. Addison Wesley, ISBN: 0-201-39829-X

[2] G. Salton and M. McGill, "*Introduccion to Modern Information Retrieval*", Ed. McGraw-Hill, ISBN: 0-07-066526-5

[3] F. Cacheda, A. Pan, L. Ardao and A. Viña "*A Layered Architecture based on Java for Internet and Intranet Information Systems*". International Journal of e-Business strategy Management. November/December 1999, volume One. Pages: 123-129.

[4] E. Lagergren and P. Over, "*Comparing interactive information retrieval systems across sites: The TREC-6 interactive track matrix experiment*". Proceedings. of 21st Annual Int. ACM SIGIR Conference, pages 164-172, Melbourne, Australia, 1998.

[5] O. Lubling and L. Malave, "*Developing Scalable, Reliable, Business Applications with Servlets*". Java Developer Connection White Paper.

[6] B. Shneiderman, D. Byrd and B. Croft, "*Sorting out searching: A user-interface framework for text searches*". Communications of the ACM, 41(4): 95-98, 1998.

[7] M. Chen, M. Hearst, J. Hong and J. Lin "*Cha-Cha: A System for Organizing Intranet Search Results*". Proceedings of the 2nd USENIX Symposium on Internet Technologies and SYSTEMS (USITS), Boulder, CO, October 11-14, 1999.