Web Data Accessing and the Web Searching Process

S. Petridou
Dept. Informatics,
Aristotle University
Thessaloniki, 54124,
Greece
spetrido@ccf.auth.gr

G. Pallis
Dept. Informatics,
Aristotle University
Thessaloniki, 54124,
Greece
gpallis@ccf.auth.gr

A. Vakali
Dept. Informatics,
Aristotle University
Thessaloniki, 54124,
Greece
avakali@ccf.auth.gr

G. Papadimitriou
Dept. Informatics,
Aristotle University
Thessaloniki, 54124,
Greece
gp@csd.auth.gr

A. Pomportsis
Dept. Informatics,
Aristotle University
Thessaloniki, 54124,
Greece
apompo@csd.auth.gr

Abstract

The enormous growth in the number of documents circulated over the Web increases the need for improved Web data management systems. Web data accessing and Web searching are the main processes in Web data management systems. In order to evaluate the performance of such systems, various simulation approaches must be used. In this paper, we survey the most recent approaches for Web data representation, Web data accessing as well as for Web search engines.

Keywords

Web technologies, Web data accessing, Information search and retrieval

INTRODUCTION

The World Wide Web is growing so fast that the need of effective Web data management systems has become obligatory. This rapid growth is expected to persist as the number of Web users continues to increase and as new Web applications (such as electronic commerce) become widely used. Currently, the Web circulates more than seven billions documents and this enormous size has transformed communications and business models so that the speed, accuracy, and availability of network-delivered content become absolutely critical factors for the overall performance on the Web.

The emergence of the Web has changed our daily practice, by providing rapid information exchange and business transactions. Therefore, supportive approaches in data, information and knowledge exchange become the key issue in new Web technologies. In order to evaluate the quality of these technologies many research efforts have used various simulation approaches.

During the last years, a great interest for developing simulation techniques on the Web Data Management Systems has been observed. By using simulation techniques, we can easily explore some models and produce tools for effectively managing Web data. In that framework, it is essential to identify the concepts in an effective Web data management system, which improve both the Web data accessing and the Web searching process. The efforts in this area have focused on:

- **Web Data Representation:** Due to the explosive growth of the Web, it is essential to represent it appropriately. One solution would be to simulate the Web as a directed graph. Graphs used for Web representation provide an adequate structure, considering both the Web pages and their links as elements of a graph. In addition, the emergence of XML, as the standard markup language on the Web (for organizing and exchanging data), has driven to new terms (such as ontologies, XML schemas) for simulating the Web data representation with a more effective way.
- **Web Data Accessing:** This process includes a collection of analytical techniques used to reveal new trends and patterns in Web data accessing records. The process of selecting, exploring and modeling large amounts of these records is essential for characterizing the Web data workload. In this context, workload characterization of Web data is clearly an important step for better understanding the Web data of behaviour.

- **Web Data Searching:** Due to the enormous size of Web documents, the search engines are the most widely used tools for retrieving Web data. Their goal is to crawl the Web, and retrieve the requested documents with low communication costs, at a reasonable interval of time.

The remainder of this paper is organized as follows. The next Section presents the main issues for Web data representation, with more emphasis on Web graph simulation models. The basic characteristics for Web data accessing patterns are discussed in Section 3. The Section 4 describes the most basic tools for searching and retrieving of the Web data. Section 5 summarizes the conclusions.

WEB DATA REPRESENTATION

Web Document Definition

Since, the amount of publicly available information on the Web is rapidly increasing (together with the number of users that request this information) various types of data (such as text, images, video, sound or animation) participate in Web documents. This information can be designed by using a markup language (such as HTML or XML), retrieved via protocols (such as HTTP or HTTPS) and presented using a browser (such as Internet Explorer or Netscape Communicator). We can further categorize Web documents into:

- **Static**: The content of a static document is created manually and does not depend on users' requests. As a result, this type of documents shows good retrieval time but it is not recommended in applications, which require frequent content changes. The hand-coded HTML Web pages processed by simple plain text editors (as well as the HTML documents created by more sophisticated authoring tools) are examples of static Web documents and (as noted in [14]) they define the first Web generation.
- **Dynamic**: Dynamic content includes Web pages built as a result of a specific user request (i.e. they could be different for different user accesses). However, once a dynamically created page sent to the client, it does not change. This approach enables authors to develop Web applications that access databases using programming languages (CGI, PHP, ASP etc.) in order to present the requested document. In this way, we can serve documents with same structure or up-to-date content. However, the dynamic content increases the server load as well as the response time.
- **Active**: Active documents can change their content and display in response to the user request (without referring back to the server). More specifically, active pages include code that is executed at the client side and usually implemented by using code such as Java and JavaScripts. Thus, active content does not require server's resources, but, it runs quite slowly since the browser has to interpret every line of its code.

Both dynamic and active Web documents introduce the second Web generation, where the content is machine-generated [14]. The common feature between these two Web generations is that they both design and present information with a human-oriented manner. This refers to the fact that Web pages are handled directly by humans who either read the static content or produce the dynamic and active content (executing server and client side code correspondingly). Finally, the third Web generation, also known as *Semantic Web*, focuses on machine-handled information management. The primary goal of the Semantic Web is to extend the current Web content to computer meaningful content. Current data representation and exchange standards (such as XML) could facilitate the introduction of semantic representation techniques and languages.

The Web Graph

The World Wide Web consists both of pages and hypertext links between them. An effective method to study the structure of the Web is to consider it as a directed graph. Particularly, in the *Web graph* each node corresponds to a page and arcs represent the hypertext links. We can further separate these arcs in outgoing edges of a node (which are the hypertext links contained in the corresponding page) and incoming edges (which represent the hypertext links through which the corresponding page is reached). Considering Web as a graph is proving to be valuable for applications such as Web indexing, detection of Web communities and Web searching.

The actual Web graph is huge and appears to grow exponentially over time. More specifically, in July 2000, it was estimated that it consists of about 2.1 billions nodes [24] and 15 billions edges, since the average node has roughly seven hypertext links (directed edges) to other pages [17]. Furthermore, approximately 7.3 millions pages are added every day and many others are modified or removed, so that the Web graph might currently (November 2002) contain more than seven billions nodes and about fifty billions edges in all.

In studying the Web graph, two important elements should be considered: its giant size and its rapid evolution. As it is impossible to work on the actual graph we retrieve parts of it, usually from several millions to several hundreds of millions nodes. This procedure is called crawl and is performed by software referred as crawlers, robots or spiders. Actually simulation efforts focus on subgraphs (which are supposed to be representative) in order to make

observations about the Web entirety and can be categorized in:

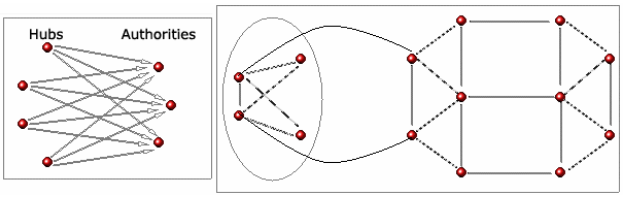

Figure 1. **Web communities on local structures of the Web graph**

- **Local approaches**: In this case, we can detect structures with an unusually high density of links among a small set of pages which is an indication that they may be topically related. Local structures are of great interest for "cyber-community" detection and thus for improving search engines techniques. A characteristic pattern in such communities contains a collection of hub pages (lists or guides) linking to a collection of authorities on a topic of common interest. More specifically, as shown in *Figure 1*, each page of the first set has a link to all pages of the second one, while there is no link between pages of the second set. The HITS (Hyperlink-Induced Topic Search) algorithm [17] is applied to modify subgraphs and computes lists of hubs and authorities for Web search topics. The Trawling algorithms [9] enumerate all such complete bipartite subgraphs of the Web graph. The results of the [19] experimentation suggest that the Web graph consists of several hundred thousand of such subgraphs, the majority of which correspond to communities with a definite topic of interest. An alternate approach detect communities based on the fact that some set of pages exhibit a link density that is greater among the members of the set than between members and the rest of the Internet, as shown in *Figure 1* (right) [18].

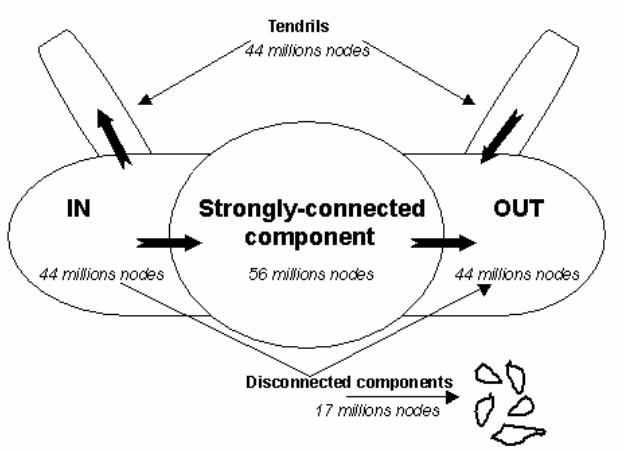

Figure 2. **The bow-tie structure of the Web**

- **Global approaches**: At a global level, a recent study [7] defines a bow-tie structure of the Web. Particularly, an experiment on a 200 millions nodes graph with 1.5 billions links demonstrates that Web graph appears to consist of four components of equivalent sizes (as shown in *Figure 2*). The "heart" of this structure is the largest, strongly connected component (SCC) of the graph and composes the core in which every page can reach every other or can be reachable by every other through a path of hypertext links. The remaining components can be defined by their relation to the core: left-stream nodes or IN component can reach the core but cannot be reached from it whereas the right-stream or OUT component can be reached from the core but cannot reach it. We can further explain the flow from the IN component to core as links from new Web pages to known interesting destinations and the lack of paths from OUT component to the core as set of pages whose links point only internally. Finally, the "tendrils" contain pages that do not link to the core and which are not reachable from it. The "tendrils" compose a set of pages that neither has been discovered yet from the rest of the Web community nor do they contain interesting links back to it. The remaining of about 9% of graph nodes consists of disconnected components.

As already mentioned, analysis of the Web's structure is leading to improved methods for understanding, indexing and, consequently, accessing the available information through the design of more sophisticated or focused search services. As an example, the Google's ranking algorithm (which called "RankPage") based on the link structure of the Web [6]. More specifically, Google ranks results pages uses information from the number of pages pointing to a given document. This information is related to the quality

of the page, as "high-quality" Web sites pointed by other "high-quality" Web sites.

WEB DATA ACCESSING

Capturing Web Users' Patterns

The incredible growth in the size and use of the Web has created difficulties in both the design of Web sites (to meet a great variety of users' requirements) and the browsing (through vast Web structures of pages and links) [4]. Most Web sites are set up with little knowledge on the navigational behaviour of the users (who access them). A relatively recent research discipline, called *Web Usage Mining*, applies data mining techniques to the Web data in order to capture interesting usage patterns. Simulating users' navigation patterns can be proved to be valuable both to the Web site designers and to the Web site visitors. For example, constructing dynamic interfaces based on visitors' behaviour, preferences or profile has already been very attractive to several applications (such as e-commerce, advertising, e-business etc). So far, there have been two main approaches to mining for user navigation patterns from log records:

- **Direct method**: In this case techniques have been developed which can be invoked directly on the raw Web server's log data. The most common approach to extract information about usage of a Web site is statistical analysis. Several open source packages that provide information about the most popular pages, the most frequently entry and exit points of navigations or the average view time of a page have been developed. This type of knowledge could be taken into consideration during system improvement or site modification tasks. For example, decisions about caching policies could be based on detecting traffic behaviour while identifying the pages where users usually terminate their sessions is important for site designers to improve their content.
- **Indirect method**: In this case the collected raw Web data are transformed into data abstractions (during a pre-processing phase) appropriate for the pattern discovery procedure. According to [27] the types of data that can be used for capturing interesting user's patterns are classified into the content, structure, usage and user profile data. Such data usually be collected from different sources (e.g. server log files, client level or proxy level log files) and can be processed in order to construct data abstractions such as *user* and *server session* [27]. A user session consists of page requests made by a single user across the entire Web while the server session is the part of user session that contains requests to a particular Web site. Once the data abstractions have been created standard data mining techniques, such as association rules, sequential patterns and clustering analysis, are used in patterns recognition [10].

In Web Usage Mining process, association rules discover set of pages accessed together (without these pages being necessarily connected directly through hyper-links). Detecting such rules could be helpful for improving the structure of a site or reducing latency due to page loadings based on pre-fetched documents. On the other hand, the action of detecting sequential patterns is that of observing patterns among server sessions such that the access to a set of pages is followed by another page in a time-ordered set of sessions. This type of information is extremely useful in e-business applications since placing advertisements aimed at certain user groups can be based on discovery of sequential patterns. Finally, clustering techniques can be used for categorizing both the users and the requested pages. User clusters involve users who exhibit similar browsing behaviour, whereas page clusters consist of pages with related content. The user clustering approach can improve the development of e-commerce strategies. Serving dynamic content focused on users' profile is a challenge in Web research. Moreover, information about page clusters can be useful for Web search engines.

Several mining systems have been developed in order to extract interesting navigation patterns. In [26] the authors present the *Web Utilization Miner (WUM)* system which, firstly, executes a pre-processing task on the raw Web data in order to infer a tree structure of detecting user sessions and then performs the mining task. [4] presents the *Hypertext Probabilistic Grammar (HPG)* model which simulates the Web as a grammar, where the pages and hyperlinks of the Web may be viewed as grammar's states and rules. Data mining techniques are used to find the higher probability strings which correspond to the user's preferred navigation path. However, this model has the drawback that returns a very large set of rules for low values of threshold and a small set of very short rules for high values of threshold. As a sequence, the heuristic *Inverse Fisheye (IFE)* [5] computes small sets of long rules using a dynamic threshold whose value is adapted to the length of the traversal path. Finally, in [11] the *WebSIFT* system is presented which performs Web Usage Mining based on server logs. WebSIFT uses content, structure and usage information and composed of pre-process, pattern mining and pattern analysis modules.

Characterizing the Workload of Web Users' Patterns

Figure 3. **A sample access log**

Due to the enormous size of Web data accessing patterns, it is essential to devise workload characterization that will be representative of the underlying users' behaviour. Analysis derived from these patterns is reviewed in an effort to characterize the entire structure of the Web. In this context, one of the important steps before capturing Web users' patterns is to model the users' behaviour. The purpose of this approach is to understand the characteristics of the submitted workload and then to find a model for the Web data behaviour using a collection of analytic techniques (such as data mining).

Furthermore, workload characterization is the key issue for simulation approaches on Web data management. In fact, workload characterization is an essential source of information for all the simulation models, which define a compact description of the load (by means of quantitive and qualitive parameters). Visually, the workload has a hierarchical nature and measurements are collected at various levels of detail. However, the complex nature of the Web complicates measuring and gathering of the Web usage loads. Web data workloads usually consist of patterns which are issued by clients and be processed by servers. Then, these patterns are recorded in files which called log files [13]. Entries in the log file are recorded when the pattern is completed, and the timestamp records the time at which the socket is closed. *Figure 3* presents a sample of Squid logs. The task of workload characterization is not simple since they have many unusual features. In this context, the patterns of Web users can be represented conventionally as multidimensional vectors, where each dimension corresponds to a single feature. The Web patterns have high variability (file sizes, time arrivals). Another feature of Web workloads is that the traffic patterns have also high variability and therefore, it can be described statistically using the term of *self-similarity*. Studies have shown that self-similarity in traffic has negative results in the performance of Web data management systems.

Capturing a specific set of Web logs is essential in order to simulate an application's behaviour. So the majority of simulation efforts use Web workloads that are characterized by several approaches. These approaches deal with characterizing associations and sequences in individual data items (Web logs) when analyzing a large collection of data. In that framework, there are two common simulation approaches for characterizing the workload of Web users' patterns[3]:

- **Trace-based approach:** The most popular way to characterize the workload of Web users' patterns is by analyzing the past Web servers log files. In [2] a detailed workload characterization study, which uses past logs, is presented for World Wide Web servers. It is common to analyze the Web server logs for reporting traffic patterns. In addition, many tools have been developed for characterizing Web data workload (such as Webalizer, Calamaris etc.). But, characterizing the workload with captured logs has many disadvantages, since it is tied to a known system. Despite the fact that this approach is simple to implement, it has limited flexibility. Firstly, this workload analysis is based completely on past logs. But the logs may lose their value if some references within them are no longer valid. Secondly, the logs are inaccurate when they return objects that may not have the same characteristics with the current objects. Finally, the logs should be recorded and processed carefully because a false can lead to incorrect temporal sequences. For example, the requests for a main page can appear after the requests for images within the page itself. So, all the above can lead to incorrect results.
- **Analytical approach:** Another idea is for the Web data workload characterization to use patterns that do not currently exist. This kind of workload is called synthetic workload and it is defined by using mathematical models, which are usually based on statistical methods, for the workload characteristics. The main advantage of the analytical approach is that it offers great flexibility. There are several workload generation tools developed to study Web proxies. In [3] the authors created a realistic Web workload generation tool which mimics a set of real users' patterns accessing a server. In [8] another synthetic Web proxy workload generator is (called *ProWGen)* described. However, the task of generating representative log files is difficult because Web workloads have a number of unusual features. Sometimes, in attempting to generate artificial workloads, we make significant assumptions such as that all objects are cacheable, or that the requests follow a particular distribution. These assumptions

may be necessary for testing, but are not always absolutely true.

SEARCHING ON THE WEB

Searching for Web Documents

The World Wide Web is a huge, heterogeneous and distributed collection of documents. However, the key aspect of the Web that makes it a valuable recourse is that an important piece of this information can be searchable and, consequently, accessed by using search services such as Web search engines [21]. *Excite*, *Lycos*, *AltaVista* and *Google* are examples of most common search engines.

A Web search engine provides a front end to a database of indexed Web documents. Using a search engine can be a start point of Web activity. In this case, users apply a query to a local database of Web resource, using a list of terms that express their information need, and receive an ordered list of Web pages that correspond to their query as better as possible. Search engines consist of three components:

- **Spiders:** they are computer programs that traverse the Web in order to identify pages. More specifically, spiders browse the Web on the search engine's behalf, as a user follows links from one page to another, using a start set of URLs. According to the topics supported by a search engine, there exist *general purpose* or *specialized* search engines. Correspondingly, the spiders can either crawl generally the Web or focus on specialty sites. For example, *Excite News* indexes news sites whereas *Lycos Multimedia Search* restricts its index in audio, video and captures content.
- **Index:** it is a database that contains a copy of each page gathered by spiders. Once the search engine has completed the crawling cycle, it begins the indexing procedure. There are two basic methods of indexing: *full-text indexing*, where every word of gathered pages is inserted into the database (*AltaVista* has a full-text database) and *keyword indexing*, where only the important words and phrases are put into the database (*Excite* has a keyword index).
- **The search and retrieval mechanism**: it is the technology that allows users to query the database and resulted pages are returned by using a rank policy. The way a search engine ranks the results pages is a very crucial issue. There exist search engines that look *on the content* of pages in order to determine relevance. In this case, issues such as the presence of search terms in the title, first heading, URL, HTML meta tag or the frequency of appearances in the pages are taken into consideration during ranking procedure. On the other hand, there are search engines that look *off the content* trying to improve the rank of the results list. For example, *Google* rank a Web page based on the number of "important" pages that link to it.

Based on the above, the search engines can be classified into two major types:

- the **individual search engines**, that use spiders to construct their own searchable database and
- the **meta search engines**, also known as *parallel* or *multi-threaded search engines*, that do not have a local database and rely on other sources

Therefore, a meta engine searches the Web by making requests to multiple individual engines [21]. *MetaCrawler* is a typical example of a meta search engine. *Figure 4* shows the architecture of typical search and meta search engines. The obvious advantage of a meta engine is the combination of results of multiple search engines through a single user interface. However, this could also be proved to be main disadvantage if one thinks the lack of an efficiently rank policy in a single engine. As shown in *Figure 4*, the results returned from the meta engines ordered through a combined rank policy. Usually, meta search engines consider the titles, keywords, summaries and URLs provided by each one of the engines behind them. Some meta search engines, such as *MetaCrawler*, collate the results and, eventually, post on the screen a single list with the duplicate pages removed. Others, such as *Dogpile*, response with separate lists of results, one for each engine that was searched, where one can be find the same page more than once.

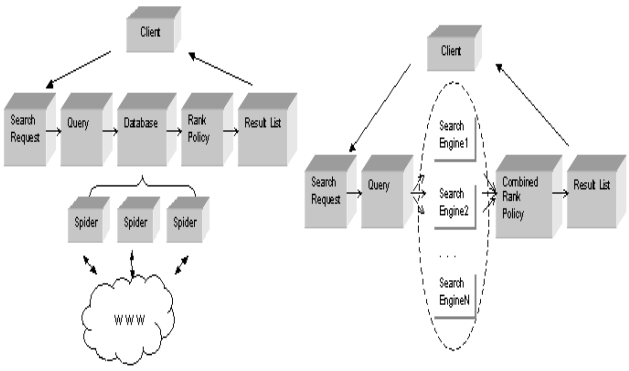

Figure 4. **The architecture of a typical search (left) and meta search (right) engine**

In spite of the great number of search engines, finding the accurate information on the Web is remain a complicated problem and several important questions bring up: can we index all global information that the Web contains and how can efficiently access it? Furthermore, how regularly can we update indices of this information? In a study at NEC Research Institute [20] the coverage of six Web search engines is analysed yielding interesting results:

- None of the search engines covers more than about one third of the indexable Web (for example they do not index pages with authentication requirements or pages generated after post action of a web form).
- The combination of the six engines covers about 3.5 times as much of the Web as one engine.
- The freshness of the several databases varies significantly. However, there is a proportional relationship between the comprehensiveness and freshness of a search engine (for example, the most comprehensive search engine had the largest percentage of broken links).

Searching for XML-based Data

The flexibility of the XML standard has improved the performance of Web searching process. In that framework, the following subsections present the main research efforts (Xyleme and Niagara) that have been done the last years for developing a robust search engine.

Xyleme

Xyleme is a research project whose goal is to implement a smart and user friendly search engine, able to answer queries relying on the structure of XML data. In this context, it is targeted on discovering XML pages on the Web that are of interest for people, synthesizing information from distinct documents and finally maintaining them up to date. Xyleme started as an "opened" group of researchers in September 2000. At the end of 2001, 25 researchers were employed for this project. In general, Xyleme is a dynamic warehouse for XML data of the Web [28]. The Xyleme system runs on a local network of Linux PCs. As illustrated in *Figure 5*, Xyleme is implemented between three autonomous machines:

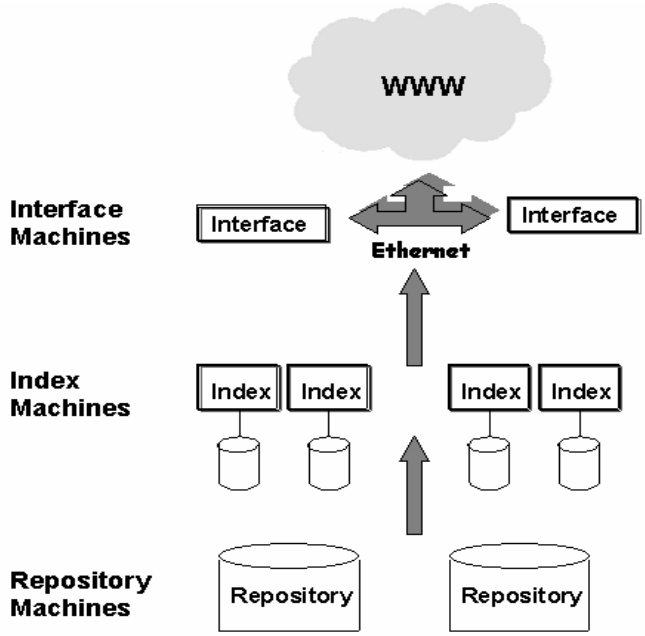

Figure 5. Xyleme Architecture

- **Repository Machines:** They are responsible for storing the XML documents. Documents are clustered according to a semantic classification. Each repository machine stores the XML data as trees in a conventional DBMS (DataBase Management System) until a certain depth and then stores the XML pages as byte streams. More details about these approaches can be found in [16].
- **Index Machines:** These machines have enough memory to fit the indexes that they are maintaining. For example, there will be several index machines for only one repository machine.
- **Interface Machines:** These kinds of machines are connected to the Web. They run applications and send either tasks or processes to the other machines.

The main feature of Xyleme, which distinguishes from other systems, is that it is based on warehousing. In particular, the performance of Xyleme is heavily depended on the efficiency of the repositories, replicating XML documents. So, the main functionalities of Xyleme can be summarized into the following modules:

- **Data Acquisition Module:** It decides when to update a document in Xyleme, evaluating the importance of the document, its size etc.
- **Query Processor Module:** The task of query processor module is to extract data from the repositories. Xyleme uses for this purpose a query language, which is a mix of OQL and XQL. More details about this module can be found in [1].

- **Change Control Module:** This module manages query subscriptions, processes temporal queries and decides which documents to adapt.
- **Semantic Module:** This module is responsible for analyzing documents and defining connections between summaries and real documents which are called views. Using data mining techniques and some human interaction, Xyleme provides views that can easily be queried by users.

Niagara

Niagara is another very promising tool which effectively manages the XML documents. The Niagara Internet Query System is designed to enable users to pose XML queries over the Web, retrieve XML documents and monitor them. It was initiated by a group of researchers in the University of Wisconsin-Madison. Instead of Xyleme, Niagara is open-source software and it runs either on UNIX PCs or Windows PCs. The Niagara has two main components:

- **Search Engine**
- **Query Engine**

The system has been implemented in a Java platform. Both the Query Engine and Search Engine are also written in Java and structured as multi-threaded servers. The next paragraphs describe the architecture of the Niagara Query System.

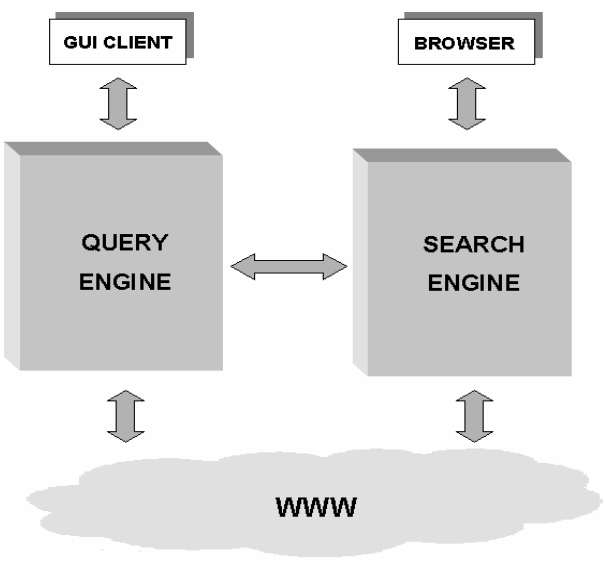

Figure 6. **Niagara Architecture**

As illustrated in *Figure 6*, the users use a Graphical User Interface (GUI) and send their XML-QL queries to the Query Engine for execution. Each query is then parsed and optimized. Afterwards, the Query Engine extracts a query to the Search one, asking for the URLs that arise from the XML files. Two of the main components of Search Engine are the *Index Manager* and the *Crawler*. The Crawler crawls the Web for new and updated XML documents and then it returns a list of them either to the Query Engine or to the users. On the other hand, Index Manager indexes the list of these documents. Afterwards, Query Engine receives the list and *Data Manager* (which is its main component) fetches asynchronously all the appropriate documents from the Web. Finally the results are given to the users via a GUI that can be run as an applet in a browser and connects to the Niagara server. More details about Niagara can be found in [25].

However, the Niagara Query System is not so flexible as Xyleme because it does not provide a semantic module in its architecture. So, the users should have prior knowledge for the structure of the XML files that they want to find. In other words, they should select by themselves the DTDs (Document Type Definitions) that define the structure of XML documents which are relevant to their queries.

CONCLUSIONS

This paper presents a study in Web data accessing and Web searching process. The extremely large volume of the Web documents has increased the need for advanced management software implementations that offer an improvement on the quality of Web services.

Selection of an appropriate evaluation methodology for Web data management systems depends on various concerns. In this context, several simulation approaches for Web data management have been developed during the last years. Firstly, these approaches are focused on simulating the structure of Web. Web graphs are the most common implementations for Web data representation. Secondly, it is essential to simulate the Web data workloads. This can be implemented using data mining techniques. These techniques study carefully the structure of Web data and find new trends and patterns that fit well with a statistical model. Finally, various systems have been developed for simulating Web caching approaches. These approaches are used for an effective storage.

All the previous simulation approaches, in conjunction with the emergence of search engines, try to improve both the management of Web data (on the server side) and the overall Web performance (on the user side). The emergence of XML standard simplifies the task of managing the Web data and it has many possibilities to dominate in the near future on the Web data management systems. So, the next generation of Web data management systems will be distinguished from their flexibility and customizability, providing also a significantly improvement on the QoS.

REFERENCES

[1] V. Aguilera, S. Cluet, P. Veltri, D. Vodislav, F. Wattez. *Querying XML Documents in Xyleme.* Proceedings of the ACM-SIGIR 2000 Workshop on XML and Information Retrieval, Athens, Greece, Jul. 2000.

[2] M. Arlitt, C. Williamson. *Internet Web servers: Workload Characterization and Performance Implications.* IEEE/ACM Transactions on Networking, Vol. 5, No. 5, pp. 631-645, Oct. 1997.

[3] P. Barford and M. Crovella. *Generating representative Web workloads for network and server performance evaluation.* Proceedings of the SIGMETRICS '98 conference, Jun. 1998.

[4] J. Borges and M. Levene. *Data Mining of User Navigation Patterns*. Proceedings of the Web Usage Analysis and User Profiling Workshop (WEBKDD99), pp. 31-36, San Diego, Aug 1999.

[5] J. Borges and M. Levene. *A Heuristic to Capture Longer User Web Navigation Patterns*. Proceedings of the 1st International Conference on Electronic Commerce and Web Technologies, Greenwish, U.K., Sep. 2000.

[6] S. Brin and L. Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. Proc. of 7th International World Wide Web Conference, Brisbane, Australia, 1998.

[7] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. L. Wiener. *Graph Structure in the Web*. Proceedings of 9th International Conference (WWW9)/Computer Networks, Vol. 33, No. 1-6, pp. 309-320, 2000.

[8] M. Busari and C. Williamson. *ProWGen: A Synthetic Workload Generation Tool for Simulation Evaluation of Web Proxy Caches.* Computer Networks, Vol. 38, No. 6, pp. 779-794, Jun. 2002.

[9] S. Chakrabarti, B. E. Dom, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, J. M. Kleinberg. *Mining the Web's Link Structure*. IEEE Computer, Vol. 32, No. 8, pp. 60-67, 1999.

[10] R. Cooley, B. Mobasher, J. Stivastava. *Data Preparation for Mining World Wide Web Browsing Patterns*. Journal of Knowledge and Information systems, Vol. 1, No. 1, 1999.

[11] R. Cooley, P. Tan, J. Stivastava. *WebSIFT: The Web Site Information Filter System*. Proceedings of the Workshop on Web Usage Analysis and User Profiling (WEBKDD99), San Diego, Aug. 1999.

[12] J. Cowie, D. M. Nicol, A. T. Ogielski. Modeling the Global Internet. In Computing in Science and Engineering, Vol. 1, No. 1, pp. 42-50, January-February 1999.

[13] B. D. Davison. *Web Traffic Logs: An Imperfect Resource for Evaluation.* Proceedings of the 9th Annual Conference of the Internet Society (INET'99), Jun. 1999.

[14] S. Decker, F. Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, S. Melnik. *The Semantic Web - on the Respective Roles of XML and RDF*. IEEE Internet Computing, 2000.

[15] M. Holiday. *Techniques for Cache and Memory Simulation Using Address Reference Traces*. International Journal in Computer Simulation, Vol. 1, No. 1, pp. 129-151, 1991.

[16] C. C. Kanne and G. Moerkotte. *Efficient Storage of XML data*. Technical Report 8/99, University of Mannheim, 1999.

[17] J. M. Kleinberg, R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins. *The Web as a Graph: Measurements, Models, and Methods.* Proceedings of the International Conference on Combinatorics and Computing, pp. 1-18, 1999.

[18] J. M. Kleinberg and St. Lawrence. *The Structure of the Web*. Science Magazine, Vol. 294, pp. 1849-1850, Nov. 2001.

[19] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins. *Trawling emerging cyber-communities automatically.* Proceedings of 8th International World Wide Web Conference (WWW8), Toronto, Canada, 1999.

[20] S. Lawrence and C. L. Giles. *Searching the World Wide Web*. Science Magazine, Vol. 280, pp. 98-100, 1998.

[21] S. Lawrence and C. L. Giles. *Searching the Web: General and Scientific Information Access*. IEEE Internet Computing, Vol. 37, No. 1, pp. 116-122, 1999.

[22] S. Manley, M. Seltzer, M. Courage. *A Self-scaling and Self-configuring Benchmark for Web Servers.* Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '98/PERFORMANCE '98), pp. 270-271, Madison, WI, Jun. 1998.

[23] J.C. Mogul. Network *Behaviour of a Busy Web Server and its Clients.* Technical Report WRL 95/5, DEC Western Research Laboratory, Palo Alto, CA, 1995.

[24] B. Murray and A. Moore. *Sizing the Internet.* White paper, Cyveillance, Jul. 2002.

[25] J. Naughton, D. DeWitt, D. Maier et al. *The Niagara Internet Query System*. IEEE Data Engineering Bulletin, Vol. 24, No. 2, pp. 27-33, 2001.

[26] M. Spiliopoulou and L. Faulstich. *WUM: A Web Utilization Miner*. Proceedings of International

Workshop on the Web and Databases, pp. 184-203, Valencia, 1998.

[27] J. Srivastava, R. Cooley, M. Deshpande, P. Tan. *Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data*. SIGKDD Exploratios, Vol.1, No. 2, Jan. 2000.

[28] Lucie Xyleme. A Dynamic Warehouse for XML Data of the Web. IEEE Data Engineering Bulletin, Report number 198, Mar. 2001.