Chapter #8

SIMULATION IN WEB DATA MANAGEMENT

G. Papadimitriou, A. Vakali, G. Pallis, S. Petridou and A. Pomportsis
KAP

Abstract: The enormous growth in the number of documents circulated over the Web increases the need for improved Web data management systems. In order to evaluate the performance of these systems, various simulation approaches must be used. In this paper, we study the main simulation models that have been deployed for Web data management. More specifically, we survey the most recent simulation approaches, for Web data representation and storage (in terms of caching) as well as for Web data trace evaluation.

Key words: Web data accessing, Web caching, Simulation of Web information management

1. 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 information exchange and business transactions. Therefore, supportive approaches in data, information and knowledge exchange becomes 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 managing effectively the Web data. In that framework, it is essential to identify new concepts in an effective Web data management system. Simulation 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. According to this implementation, the Web documents and their links are simulated as graph nodes and graph arcs respectively. 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:** These simulation efforts include 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 behavior.
- **Web Data Storage:** Since Web data storage has a major effect on the performance of Web applications, new implementations (such as Web data caching and Web databases) have emerged. These implementations can be considered as one of the most beneficial approach to accommodate the (continuously growing) number of documents and services, providing also a remarkable improvement on the Quality of Service (QoS). The term of QoS has been introduced to describe certain technical characteristics, (such as performance, scalability, reliability and speed). In order to evaluate the performance of different cache management techniques, many Web caching simulations have been presented. So, one of the most important features of modern Web data management simulation efforts is based on the capability of storing effectively various Web documents.

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 both Web data workload and Web users' patterns are discussed in Section 3. In Section 4 an

overview of Web data caching simulation approaches is presented. Section 6 summarizes the conclusions.

2. WEB DATA REPRESENTATION

2.1 Web Document Structure

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[1] or XML[2]), 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 [18]) 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.

[1] W3C HTML Home Page: http://www.w3c.org/MarkUp
[2] Extensible Markup Language (XML): http://www.w3c.org/XML

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 [18]. 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.

2.2 The Web as a graph

The World Wide Web structure includes pages which have both the Web content and the hypertext links (that connect one page to another). An effective method to study the Web is to consider it as a directed graph, which simulates both its content and content's interconnection. In particular, in the *Web graph* each node corresponds to Web pages and arcs correspond to links between these pages. We can further separate these arcs in outgoing edges of a node (which simulate 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 [26], [30] and 15 billions edges, since the average node has roughly seven hypertext links (directed edges) to other pages [22], [25]. 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 whole graph we retrieve parts of it. This procedure is employed by and performing with software packages such as crawlers, robots, spiders, worms or wanderers. More specifically, we can think of this procedure as BFS (bread-first search) on a directed graph [10]: we begin with a random

initial list of URLs (Uniform Resource Locators) and build up the set of pages reachable from the first list through the outgoing hypertext links. The same process iterates from the new set of pages. In fact, this mechanism is more complicated for reasons, which deal with the frequency of the requests (to a given web server according to its load) and its rapid evolution (which implies that graph changes during the crawl).

Because of the above considerations, studies about the structure of the Web documents always deal with parts of the actual Web graph, usually from several millions to several hundreds of millions nodes. 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:

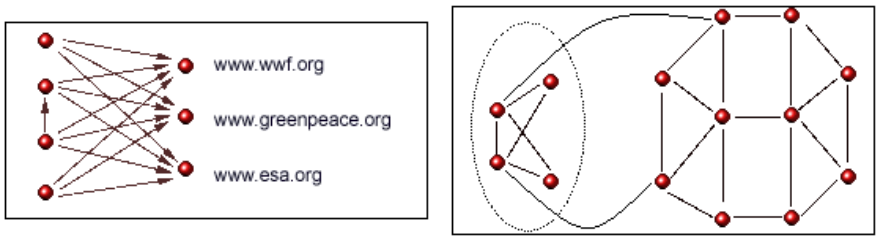

Figure 8-1. Web communities on local structures of the Web graph

– **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, 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, and hubs do not necessarily link to hubs [22], [25]. As an example, in *Figure 8-1* (left) we can consider hub-like pages on the left as the personal pages of ecologists, which co-cite the authoritative pages of ecological organizations on the right. The HITS (Hyperlink-Induced Topic Search) algorithm [25] is applied to modify subgraphs and computes lists of hubs and authorities for Web search topics. To construct such a subgraph, the algorithm first submits a search request to a traditional search engine and receives a root set of about 200 pages. The root set is further expanded into a base set of roughly 1000 – 5000 pages including all pages that are linked to by root-pages and all pages that link to a root-page. In a second step the algorithm performs an

iterative procedure determining the authority and hub weight of each page (before the start of the algorithm these values set to 1 and then are updated as follows: if a page is pointed to by many hubs, its authority weight is increased; correspondingly, the hub weight of a page is updated, if the page points to many authorities). As a result HITS returns as hubs and authorities for the search topic those pages with the highest weights. The Trawling algorithms [13], [25] enumerate all such complete bipartite subgraphs of the Web graph. The results of the [24] 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 8-1* (right) [23].

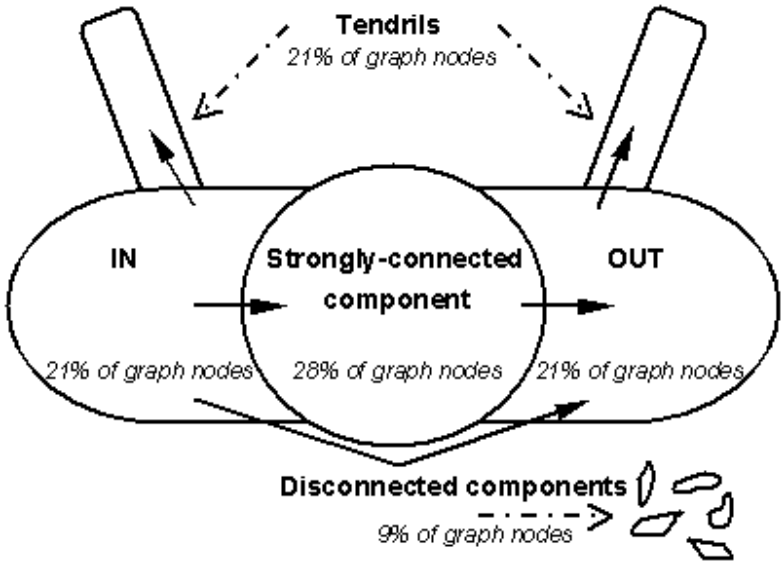

Figure 8-2. The bow-tie structure of the Web

– **Global approaches**: At a global level, a recent study [10] defines a *bow-tie* structure of the Web. Particularly, an experiment on a 200 millions nodes graph with 1.5 billions links, retrieved from a crawl of the Web, demonstrates that Web graph appears to consist of four components of equivalent sizes (as shown in *Figure 8-2*). The "heart" of this structure is the largest, strongly connected component (SCC) of the graph (28% of graph nodes) 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 (21% of graph nodes) can reach the core but cannot be reached from it whereas the right-stream or OUT component (21% of graph nodes) 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" (21% of graph nodes) 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 [25].

In order to evaluate such simulation efforts there is a need to consider appropriate measures and parameters. Therefore, statistical studies have considered several parameters to characterize the Web's graph structure. More specifically:

- the number of links to (in-degree) and from (out-degree) individual pages is distributed following a power low; as a sequence, the *average out-degree* of a node is about 7 [10]
- the sizes of the strongly-connected components in the Web graph are also distributed according to a power low [10]
- the probability that there is a path between a random start node u to a random final node v is 25%. Therefore, for around 75% of time there is no path between these nodes; during only the 25% of the time the *average connected distance* (diameter) can be defined and it was estimated to be about 16 [25]

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 search engines, focused search services or automatically refreshed directories. As an example, the Google's ranking algorithm (which called "RankPage") based on the link structure of the Web [9]. 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.

3. WORKLOADS FOR SIMULATING WEB DATA ACCESSING

3.1 Web Data Workload Characterization

Table 8-1. A sample access log

986074304.817 81019 ccf.auth.gr TCP_MISS/503 1180 GET http://www.mymobile.com/ - DIRECT/www.mymobile.com –
986074304.828 51360 med.auth.gr TCP_MISS/000 0 GET http://www.battle.net/includes/ads.js - DIRECT/www.battle.net –
986074312.188 3140 med.auth.gr TCP_MISS/000 0 GET http://www.battle.net/includes/ads.js - DIRECT/www.battle.net –
986074312.302 53 med.auth.gr TCP_HIT/200 16590 GET http://www.battle.net/ - NONE/- text/html
986074320.238 7210 med.auth.gr TCP_MISS/000 0 GET http://www.battle.net/includes/ads.js - DIRECT/www.battle.net –
986074334.489 13742 med.auth.gr TCP_MISS/503 1202 GET http://www.battle.net/includes/ads.js - DIRECT/www.battle.net -
986074345.604 6 ccf.auth.gr TCP_MISS/503 1180 GET http://www.mymobile.com/ - DIRECT/www.mymobile.com –
986074359.079 50 med.auth.gr TCP_HIT/200 10673 GET http://www.auth.gr/index.el.php3 - NONE/- text/html
986074360.125 56 med.auth.gr TCP_IMS_HIT/304 252 GET http://www.auth.gr/auth.css - NONE/- text/css

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

Therefore, 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 requests which are issued by clients and be processed by servers. Then, these requests are recorded in files which called log files [17]. Entries in the log file are recorded when the request is completed, and the timestamp records the time at which the socket is closed. *Table 8-1* presents a sample of Squid logs. The first attempt to characterize Web user behavior was presented in [12]. The authors tried to synthesize the workload of Web data by analyzing the user behavior (captured at the browser). The task of workload characterization is not simple since Web workloads have many unusual features. Firstly, the Web requests have high variability (file sizes, time arrivals). According to [29], this is due to the variability in CPU loads of the Web servers and the number of their connections. 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 behavior. 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 Web workloads [6]:

– **Trace-based approach:** The most popular way to characterize the workload of Web data is by analyzing the past Web servers log files. In [3] a detailed workload characterization study, which uses past logs, is presented for World-Wide Web servers. Most of these tools are downloaded free from the Web. 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. In this context, the Webalizer[3] is a log file analysis tool, which produces highly detailed, easily configurable usage reports in HTML format. Calamaris[4], Squid-Log-Analyzer[5], Squidalyser[6] are tools which analyse only the logs of Squid proxy server. On the other hand, 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

[3] Webalizer site: http://www.mrunix.net/webalizer
[4] Calamaris site: http://calamaris.cord.de
[5] Squid-Log-Analyzer site: http://squidlog.sourceforge.net
[6] Squidalyser site: http://ababa.org

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. In [17] the author examines the disadvantages of using captured log files and investigates what can be learned from logs in order to infer more accurate results.

– **Analytical approach:** Another idea is for the Web data workload characterization to use traces 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 [6] the authors created a realistic Web workload generation tool, which mimics a set of real users accessing a server. In [11] 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.

Finally, another approach for synthesizing Web workloads is to process the current requests. Using a live set of requests produces experiments that cannot be reproducible. The disadvantage of using current requests is the high real load. So, the hardware and the software may have difficulties handling this load.

3.2 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) [7]. Most Web sites are set up with little knowledge on the navigational behaviour of the users (who access them). Therefore, 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).

When web users interact with a site, data recording their behaviour is stored in files (called Web server's log files), which can sum up to several megabytes per day (in case of a medium size site). A relatively recent research discipline, called *Web Usage Mining*, applies data mining techniques to the Web data in order to capture interesting usage patterns. So far, there have so far 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, the average view time of a page (or the hourly distribution of access) 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 [34] 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 can be collected from different sources (e.g. server log files, client level or proxy level log files). Server log files keep information about multiple users who access a single site. However, the collected data might not be reliable since the cached pages requests are not logged in the file. Another problem is the identifying of individual users since in most cases the web access is not authorized. On the other hand, client level collected data reflects the access to multiple web sites by a single user and overcomes difficulties related to page caching, user and session identification. Finally, proxies log files collect data about requests from multiple users to multiple sites. All of above data can be processed in order to construct data abstractions such as *user* and *server session* [34]. 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 [14].

In Web Usage Mining process, association rules discover set of pages accessed together (without these pages being necessarily connected directly through hyper-links). For example, at a cinema's chain Web site, it could be found that users who visited pages about comedies also accessed pages about thriller films. 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. As an example, at an ISP's Web site, it might be revealed that visitors accessed the Products page followed by the News page. This type of information is extremely useful in e-business applications since analyzing products bought (or advertisements views) can be based on discovery of sequential patterns.

Finally, clustering techniques can be used for categorizing both the users and the requested pages. More specifically, clusters are groups of items that have similar features, so we can recognize user and page clusters. 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. [21] proposes the *WebWatcher* a tour guide agent for the Web browsing. WebWatcher simulates a human guide making recommendations that help visitors during their navigation. It suggests the next page based on the knowledge of user's interests and of the content of the web pages as well as it improves its skills interacting with users. In [33] the authors present the *Web Utilization Miner (WUM)*, a mining system, which consists of an aggregation module and a mining module. The first module executes a pre-processing task on the web log data and infers a tree structure of detecting user sessions where as the second one is a mining language (MINT) which performs the mining task according to a human expert. [7] 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) [8] computes small sets of long rules using a dynamic threshold whose value is adapted to the length of the traversal path. Finally, in [15] 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.

4. SIMULATION OF WEB DATA CACHING

Web data caching techniques are used to store the Web data, in order to retrieve them with low communication costs.

4.1 Web Data Caching

The explosive growth of the World Wide Web in recent years has resulted in major network traffic and congestion. As a result, the Web has become a victim of its own success [1]. These demands for increased performance have driven the innovation of new approaches, such as the Web caching [2], [36], [37].

It is recognized that deploying Web caching can make the World Wide Web less expensive and better performing. In particular, it can reduce the bandwidth consumption (fewer requests and responses that need to go over the network), the network latency perceived by the client (cached responses are available immediately, and closer to the client being served) and the server load (fewer requests for a server to handle) [1], [37]. Furthermore, it can improve the network reliability perceived by the client.

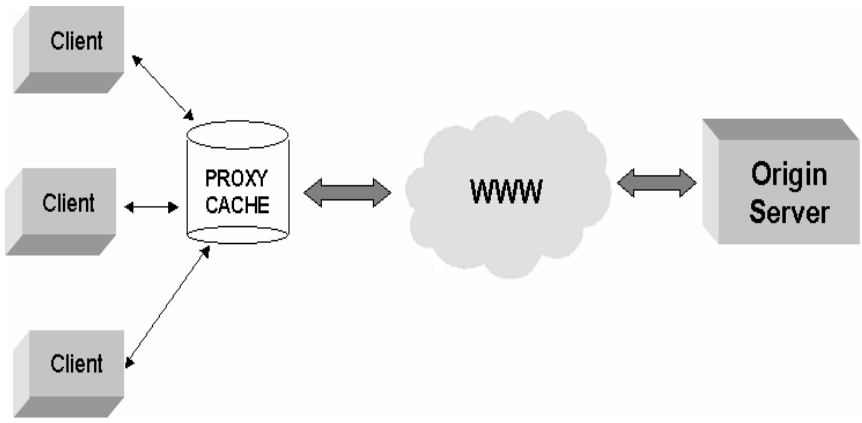

Figure 8-3. Web data caching

Web caching has many similarities with a memory system caching. A Web cache stores frequently used information in a suitable location so that it can be accessed quickly and easily for future use. Caching can be performed by the client application and is built into every Web browser. Caching can also be utilized between the client and the server as part of a proxy as illustrated in *Figure 8-3*. A proxy cache server intercepts requests from clients, and if it finds (called a cache hit) the requested object in the cache, it returns the object to the user without disturbing the upstream network connection or destination server. If the object is not found (a cache miss), the proxy attempts to fetch the object directly from the origin server. For greater performance proxy caches can be parts of cache hierarchies, in which a proxy requests objects from neighboring caches instead of fetching them directly from the origin server. *Table 8-2* presents the main metrics which assess the cache performance.

Table 8-2. Caching Metrics

Caching Metrics	Definitions
Hit rate	It is defined as the ratio of documents obtained through using the caching mechanism versus the total documents requested. A high hit rate reflects an effective cache policy.
Byte hit rate	It is defined as the ratio of the number of bytes loaded from the cache to the total number of bytes accessed.
Saved bandwidth	This metric tries to quantify the decrease in the number of bytes retrieved from the origin servers. It is directly related with byte hit rate.
User response time	The time a user waits for the system to retrieve a requested document.
System utilization	It is defined as the fraction of time that the system is busy.
Latency	Latency is defined as the interval between the time the user requests for a certain content and the time at which it appears in the user browser.

However, if at some point the space required to store all the objects being cached exceeds the available space, the proxy will need to replace an object from the cache. Cache Replacement Algorithms play a main role in the design of any caching component and some of them are discussed in [5]. In general, cache replacement policies attempt to maximize the percentage of requests which successfully are served by the cache (called hit ratio) [4]. In order to evaluate these algorithms in various caching systems, some simulation approaches are usually used. Simulation is a very flexible method to evaluate the caching policies because it does not require full implementation. Otherwise, we should have developed an integrated caching scheme. The simulation results have shown that the maximum cache hit rate

that can be achieved by any caching algorithm is usually no more than 50% [28].

4.2 Simulating Caching Approaches

It is useful to evaluate the performance of proxy caches both for Web data managers (selecting the essential system for a particular situation) and also for developers (working on alternative caching mechanisms). Simulating the Web data will help also to an effective data management on the Web [28].

In this context, new simulation approaches are needed for describing the Web. In [16] an encouraging development for simulating the Web is presented. In this paper, the authors use a class of Parallel Discrete Event Simulation (PDES) techniques for constructing appropriate models for the World Wide Web. More specifically, they use the Scalable Simulation Framework (SSF), which is being developed by Cooperating Systems Coorporation. SSF provides an interface for constructing process-oriented, event-oriented and hybrid simulations. SSF provides also some mechanisms for constructing PDES that can scale to millions of Web objects. Therefore, this framework, in conjunction with scalable parallel simulations, makes it possible to analyze the behaviour of the complicated Web models.

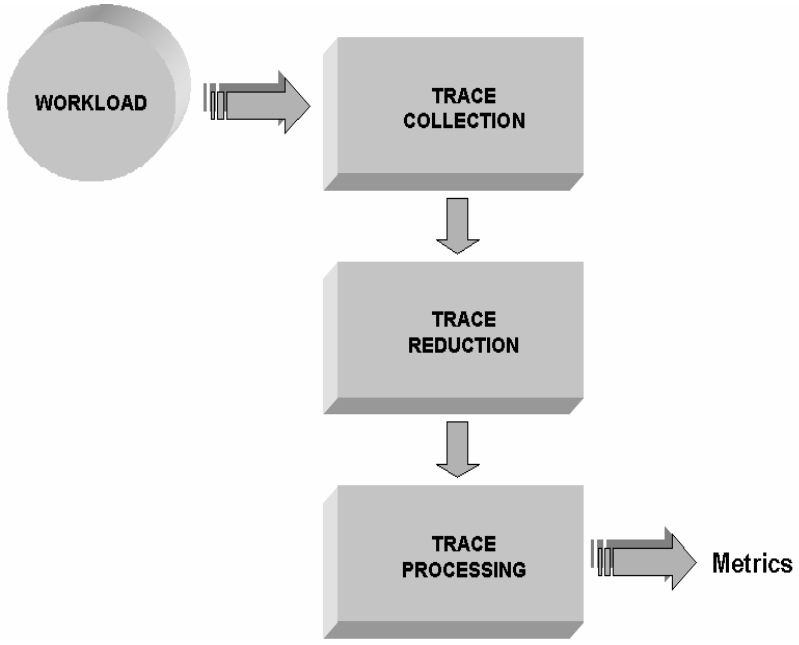

Figure 8-4. The Three Stages of Trace-driven Simulation

In the literature, several alternative approaches are available. They can be summarized as follows:

– **Simulations using captured logs:** This kind of simulation is the most popular and is directly related with Web performance. Many research efforts have used trace-driven simulation to evaluate the effects of various replacement, threshold, and partitioning policies on the performance of a Web server. The workload traces for the simulations come from Web servers' access logs. They include access information, configuration errors and resource consumption. In this approach the logs are the basic component and they should be recorded and processed carefully. In general, a trace driven simulation can be considered of having three main stages: trace collection, trace reduction and trace processing [19]. As illustrated in *Figure 8-4*, trace collection is the process of determining the sequence of Web data that made by some workload. Because these traces can be very large, trace reduction techniques are often used to remove the full trace of data that are needless or redundant. In the final stage, trace processing is used to simulate the behaviour of a system, producing some useful metrics, such as hit rate, byte hit rate etc. More specifically, authors in [32] trace-driven simulation is used to evaluate their proposed algorithm (LNC-R-W3-U) with different cache replacement algorithms. In this work, the authors gathered a seven-day snapshot of requests generated by clients in a lab at Northwestern University. The simulation results show that the LNC-R-W3-U improves the delay saving ratio by 38% when compared to LRU (the most popular algorithm). Another work [28] uses trace-driven simulation based on access logs from various servers to evaluate the most popular documents with client access profiles. The basic idea of this proposal is (for servers) to publish their most accessed objects, called "Top 10" (although there may be more than ten popular objects). In particular, the authors captured traces from several Web servers from a variety of environments, such as universities, research institutions, and Internet Service Providers (ISPs) both from Europe and the United States. All these traces exceed the four million requests. Then, the authors used these captured logs to investigate the costs and benefits of their approach. Performance results have shown that this approach can prefetch more than 60% of future requests, with less than 20% corresponding increase in traffic. Finally, in [31] the authors use trace-driven simulation to evaluate a new caching policy, taking into account some criteria such as hit rate, byte hit rate and latency. In particular, the authors developed a simulator in C++ which models the behaviour of a proxy cache server. According to this simulation model, the authors captured logs from proxy caches of various institutes such as Digital

Equipment Corporation, Boston University, NLAR and INRIA. The experimentation results show that the new caching policy improves the performance.

– **Simulations using synthetic workloads:** In this approach synthetic traces are usually used to generate workloads that do not currently exist. Authors in [35] propose a new cache algorithm (RBC) which uses synthetic traces for the simulation of caching continuous media traffic. The selected workload has a predefined distribution of requests among different object types and a predefined object size distribution. In particular, the objects are ranging either from 3 to 64 KB (for objects of image/text) or from 100 KB to 15MB (for objects of audio/video). The simulation results show that RBC achieves higher hit ratio as compared to several existing algorithms under the above workload. In [20] an adaptive prefetch scheme using a synthetic trace set is presented. According to this scheme, the authors presented a prediction algorithm and studied its performance through simulations. Although the trace set is very limited, this algorithm achieves a high hit rate. Furthermore, authors in [11] use synthetic workloads to evaluate the performance of different cache replacement algorithms for multi-level proxy caching hierarchies. The workload follows distinct distributions, such as Zipf-like popularity, heavy-tailed file size distribution etc. According to this simulation model, the client's requests are forwarded to the lower level proxies. All the requests that failed from the upper level proxies are forwarded to the Web servers. At different levels of the hierarchy, the proxies support different replacement policies. Results have shown that this approach improves the performance, combining different policies at different levels of the proxy cache hierarchy. Finally, in [27] a new Web benchmark that generates a server benchmark load, which is focused on actual server loads, is presented. This tool would be used to compare the traffic generated by the benchmark and the desired traffic patterns. The results have shown that these predictions are sufficiently realistic.
– **Simulations using current requests:** This kind of simulation utilizes current requests of a live network. The advantage is that the cache is tested on a real traffic. The drawback is that the experiments are not reproducible (especially when connected with live networks or systems).

Finally, many research efforts have used a combination of these approaches, which are often called as hybrids. According to these approaches, research efforts are trying to evaluate the Web data management systems using both captured logs and synthetic workloads.

5. CONCLUSIONS

This paper presents a study of simulation in the Web data management 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).

REFERENCES

[1] M. Abrams et al. *Caching Proxies: Limitations and Potentials*. Proc. of the 4th International WWW Conference, pp. 119-133, 1995.

[2] C. Aggarwal, J. Wolf, P. S. Yu. *Caching on the World Wide Web.* In IEEE Transactions on Knowledge and Data Engineering Vol.11, No.1, pp.94-107,January-February, 1999.

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

[4] M. Arlitt, R. Friedrich, T. Jin. *Performance Evaluation of Web Proxy Cache Replacement Policies.* Hewlett-Packard Technical Report HPL 98-97, Performance Evaluation Journal, May 1998.

[5] G. Barish and K. Obraczka. *World Wide Web Caching: Trends and Techniques.* IEEE Communications Magazine, Vol. 38, No. 5, pp. 178-185, 2000.

[6] P. Barford and M. Crovella. *Generating representative Web workloads for network and server performance evaluation.* Proc. of the SIGMETRICS '98 Conference, June 1998.

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

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

[9] 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.

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

[11] 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, June 2002.

[12] L. D. Catledge and J. E. Pitkow. *Characterizing Browsing Strategies in the World-Wide Web*. Computers Networks and ISDN Systems Vol. 26, No. 6, pp. 1065-1073, 1995.

[13] 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.

[14] 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.

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

[16] 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.

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

[18] 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.

[19] 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.

[20] Z. Jiang and L. Kleinrock. *An Adaptive Network Prefetch Scheme.* IEEE Journal on Selected Areas in Communications, Vol. 16, No. 3, pp. 358-368, April 1998.

[21] T. Joachims, D. Freitag, T. Mitchell. *WebWatcher: A Tour Guide for the World Wide Web*. Proc. of 15th International Joint Conference on Artificial Intelligence, pp. 770-775, Aug 1997.

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

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

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

[25] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, E. Upfal. *The Web as a Graph*. Proc. of 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2000.

[26] M. Levene and R. Wheeldon. *Web Dynamics.* Software Focus, Vol. 2, pp. 31-38, 2001.

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

[28] E. P. Markatos, C. E. Chronaki. *A Top-10 Approach to Prefetching the Web.* Proc. of INET'98, Geneva, Switzerland, July 1998.

[29] 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.

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

[31] N. Niclausse, Z. Liu, P. Nain. *A New Efficient Caching Policy for the World Wide Web.* Proc. of the Workshop on Internet Server Performance (WISP'98), 1998.

[32] J. Shim, P. Scheuermann, R. Vingralek. *Proxy Cache Algorithms: Design, Implementation, and Performance*. IEEE Transactions on Knowledge and Data Engineering, 1999.

[33] M. Spiliopoulou and L. Faulstich. *WUM: A Web Utilization Miner*. Proc. of International Workshop on the Web and Databases, pp. 184-203, Valencia, 1998.

[34] 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.

[35] R. Tewari, H. M. Vin, A. Dan, D. Sitaram. *Resource-Based Caching for Web Servers.* Proc. of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, CA, January 1998.

[36] A.Vakali. *Evolutionary Techniques for Web Caching.* Distributed and Parallel Databases, Journal, Kluwer Academic Publishers, 2002.

[37] A. Vakali and G. Pallis. A Study on Web Caching Architectures and Performance". 5th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2001), July 2001.