

Combining Queueing Networks and Web Usage Mining Techniques for Web Performance Analysis

Giuliano Casale^{*}
 Dip. di Elettronica e Informazione
 Politecnico di Milano
 via Ponzio 34/5
 20133 Milano, Italy
 casale@elet.polimi.it

ABSTRACT

The growing interest in Web applications that satisfy end-to-end Quality of Service (QoS) requirements is leading many organizations to build and analyze performance behavior models. In this direction, Web usage mining techniques may help in the automatic construction of user profiles from Web access logs. However, their use has been mainly limited to customer relationship management (CRM) issues and to market analyses. The aim of this paper is to explain how Web usage mining can be combined with queueing networks for effective Web capacity planning. After introducing a new general *relative cosine similarity* measure, we define a performance-oriented similarity for Web usage data. A methodology to devise the input parameters of a queueing network from the resulting clusters is also presented. Finally, the proposed approach is illustrated on a simple case study.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Modeling Techniques;
 H.2.8 [Database Management]: Database Applications—
Data Mining; I.5.3 [Pattern Recognition]: Clustering—
Algorithms

General Terms

Performance, Measurement, Algorithms

Keywords

Web usage mining, queueing networks, clustering, capacity planning, Web performance

^{*}This work has been partially supported by the Italian FIRB-Perf project.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA.
 Copyright 2005 ACM 1-58113-964-0/05/0003...\$5.00.

1. INTRODUCTION

The term Web Usage Mining (WUM) [13] denotes a collection of data mining techniques for the automatic discovery of valuable information regarding the interests and needs of Web site visitors. Such knowledge is usually mined from the access logs of a group of servers, including Web servers, application servers, and proxies. Usually, this activity raises the problem of handling a scalable analysis on huge amounts of data. Besides limiting the analysis to the generation of simple statistics, another possible solution is using the algorithmic techniques developed in the research area of WUM, among which association rules [1], OLAP-like strategies for Web datacubes [14], and cluster analysis of usage data [8]. Such methods have proven in past years to be effective in the automatic construction of Web user profiles that summarize past user behaviors, help in predicting future accesses, and can be used in a number of applications, including market segmentation, site personalization and topology improvement. However, WUM techniques have found only limited application in performance evaluation studies, where they have been mainly employed for traffic characterizations [2], and this is in contrast with the growing interest that many organizations are showing for performance-aware Web applications. The aim of this paper is to fill the gap between these areas of current research by showing how WUM techniques can fit in performance and capacity planning models. In particular, we show how the input workload of a multiclass queueing network may be inferred from usage profiles. The advantage of mapping user profiles to such models is that efficient solution techniques exist, and “what-if” capacity planning questions are easy to answer, e.g. “which response time will users experience after a CPU upgrade?”.

The remainder of this paper is organized as follows. Section 2 gives an overview of WUM techniques and queueing networks models. In Section 3 we introduce a new similarity measure for performance data. Section 4 describes how the resulting clusters can be used in the definition of a Web site performance model. In Section 5 our methodology is illustrated on a simple case study. Finally, Section 6 contemplates future work and concludes the paper.

2. PRELIMINARIES AND NOTATION

2.1 User Profiles Clustering

Let us consider a Web site composed of $|W|$ pages de-

scribed by the set $\mathbf{W} = (P_1, \dots, P_{|\mathbf{W}|})$. To simplify the exposition we shall refer to the terms Web page or Web object indifferently. Thus, for instance, a page P_k could be either an HTML document or a JPEG image. We assume that log files account site activities for a time period T . Let \mathbf{U} be the set of users that visit the site in T and $u \in \mathbf{U}$ one of such visitors. We call the i -th *session* of user u the set $S(u, i)$ described by the tuple:

$$S(u, i) = (p_1, \dots, p_j, \dots, p_{|\mathbf{W}|})$$

where p_j is a boolean variable that is one if and only if u has downloaded page $P_j \in \mathbf{W}$ during its visit; thus, multiple views of a same page are accounted only once. All variables $S(u, i)$ can be extracted directly from log files and collected in a *sessions dataset* \mathbf{S} , once that all logs have been filtered [6] and a proper sessionizing methodology has been chosen [4]. Using \mathbf{S} , several usage mining techniques can be applied to extract user behaviors. When cluster analysis is employed, the sessions of \mathbf{S} are partitioned in a number of classes, in such a way that observations that fall within the same class tend to be very similar, while members of different classes have a much lower degree of similarity. Hence, every cluster is roughly equivalent to a single usage profile, although further filtering and a qualitative analysis of the results may be required to ensure the semantic consistency of each cluster.

The key problem of defining a proper similarity measure θ between user sessions has been addressed in literature in several ways. A popular choice is the *cosine measure*. Let $S(u, i) = \{p_j\}$ and $S(v, k) = \{q_j\}$ be two sessions of \mathbf{S} , then the cosine measure is defined as:

$$\theta = \frac{\langle S(u, i), S(v, k) \rangle}{|S(u, i)|^{1/2} |S(v, k)|^{1/2}} \quad (1)$$

where $\langle S(u, i), S(v, k) \rangle$ is the scalar product $\sum_w p_w \cdot q_w$, and the terms at the denominator are $|S(u, i)| = \sum_w p_w$ and $|S(v, k)| = \sum_w q_w$. Thus, θ is the number of common pages of the two sessions, normalized over the geometric mean of their lengths. In many cases the *squared cosine* θ^2 is preferred, because it reduces the noise of terms with low similarity values. Another popular similarity measure is the *Jaccard coefficient*, that is the ratio of common pages to the cardinality of the union $S(u, i) \cup S(v, k)$, i.e.:

$$\theta = \frac{\langle S(u, i), S(v, k) \rangle}{|S(u, i)| + |S(v, k)| - \langle S(u, i), S(v, k) \rangle} \quad (2)$$

All these similarities allow a simple interpretation of the results, but their main drawback is that they do not capture any specific feature of the Web environment where users move. Hence, Joshi et al. [8] propose an additional factor in (1) that accounts the topological distance (i.e. minimum number of links to be followed) between each couple of pages. An integration of (1) with page content information, like words frequencies, is shown in [10]. In [3] both content and topological factors are linearly combined to define a new similarity measure for a clustering algorithm based on self organizing maps (SOMs).

A common problem of using Web usage clustering techniques is the quadratic time complexity required to compute the similarity matrix, that frequently becomes an important issue due to the dimensionality of the data set. A interesting solution, presented in [7], is that of calling the computation of θ only when required by the clustering algorithm, thus

minimizing the number of similarities to be computed. This indeed extends the applicability of the described techniques to large-scale datasets such those involved in real case studies.

2.2 Queueing Networks for Web Performance Analysis

A fundamental observation at the basis of many performance evaluation studies is that delays in computer systems are often caused by queueing for the access at different resources. For instance, the queue seen by a TCP packet arriving at a congested router is frequently the main determinant of its end-to-end latency. A product-form queueing network (QN) [12] is a stochastic model that allows to compute efficiently the impact of queueing delays on the overall response time of a request processed by one or more servers. Moreover, with *multiclass* QNs several statistical classes of services can be modeled, and this requires only the knowledge of the *average speed* L_{ir} (*loading*) at which component i provides service to a job of class r . For instance, a Web server i may be characterized by two loadings $L_{i,HTTP}$ and $L_{i,HTTPS}$ that describe the different average speeds (seconds/job) at which HTTP and HTTPS requests are served. When the number of requests arriving to the system is unbounded, it is usually described with an average arrival rate λ_r (class- r jobs/second) and the model is called an *open* QN; instead, constant-population models are referred to as *closed* QN and are specified by a set of per-class populations N_r (jobs). Figure 1 shows a simplified single-class open model for studying the performance of a small Web site organized on a three-tier architecture: a stream of λ HTTP requests/sec arrives at the Web server and is routed to the other servers (or back to the client) according to a first-order Markov chain (not represented in the diagram). Queued jobs can receive every instant an equal quantum of service (processor sharing policy) or can be served with a First-Come-First-Served policy, and the duration of service times can follow almost any distribution. These characteristics of queueing network have proven several times to be sufficient for a successful quantitative analysis of real-world computing infrastructures. Furthermore, the background load of each server (operating system, batch backups, ...) can also be studied with a (closed) QN [12]. The performance bottlenecks of both open and closed multiclass models can instead be identified efficiently using the geometric approach described in [5]. Also, using standard algorithms [12], it is easy to compute useful performance measures like the average response time of each job or the utilization of each server.

3. A SIMILARITY MEASURE FOR WEB PERFORMANCE DATA

3.1 The relative cosine measure

The clusters obtained from usage logs using Jaccard coefficient or cosine-like measures are generally well-suited for a qualitative description of user sessions. However, a certain degree of heterogeneity between members of a same cluster frequently remains, as we can show with the following example. Let us consider two sessions $S(u, i)$ and $S(v, k)$, such that $|S(u, i)| = 1$ and $|S(v, k)| = 9$, and assume that $\langle S(u, i), S(v, k) \rangle = 1$. Then, using (1), we obtain a value

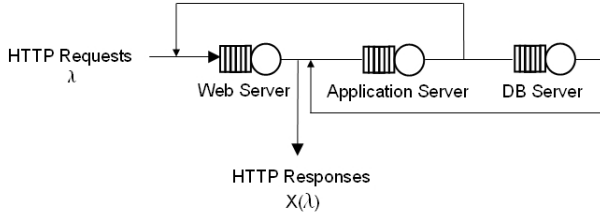


Figure 1: Simplified queuing network model of a Web site with dynamic pages.

$\theta = 0.33$ that is rather large and does not capture the remarkable difference between $|S(u, i)|$ and $|S(v, k)|$. Such uncontrolled values of θ can promote the aggregation of oversized clusters with high intra-cluster variance. While these inaccuracies may be accepted in a qualitative study, a quantitative performance model requires narrow margins of error for the input data. Therefore, we propose a new similarity measure that provides a higher degree of intra-cluster consistency, and without changes in the computational cost. Let $S(u, i)$ and $S(v, k)$ be two sessions of \mathbf{S} , we define $S(u, i) \rightarrow S(v, k)$ as the *relative similarity* of session $S(u, i)$ with respect to $S(v, k)$. Informally, we wish to capture that if user u visits a subset of the pages accessed by user v , then she maximizes her relative similarity with v ; conversely, user v should get a lower value of similarity if part of her session is not in common with u . Then, taking only the minimum of the two relative similarities we can promote the aggregation of sessions with similar lengths and similar number of common pages. A possible quantitative expression of $S(u, i) \rightarrow S(v, k)$ is given by the following formula:

$$S(u, i) \rightarrow S(v, k) = \frac{\langle S(u, i), S(v, k) \rangle}{|S(u, i)|}$$

that is the ratio of common pages to the length of $S(u, i)$. We can now define θ as the quantity

$$\begin{aligned} \theta &= \min(S(u, i) \rightarrow S(v, k), S(v, k) \rightarrow S(u, i)) \\ &= \frac{\langle S(u, i), S(v, k) \rangle}{\max(|S(u, i)|, |S(v, k)|)} \end{aligned}$$

that can be computed in a single step and does not require the computation of both relative similarities. Henceforth we shall refer to the newly defined measure as the *relative cosine similarity*. In the previous example, using the relative cosine formula, θ drops from $\theta = 0.33$ to a reasonable $\theta = 0.11$. Note that a similar value can be obtained using Jaccard coefficient or the squared cosine measure instead of the cosine measure (1). Therefore, in order to prove the increased consistency offered by the relative cosine, we show the results of the following random experiment: we generated a sample of 1000 random paths, chosen from a set of $|\mathbf{W}| = 100$ pages, for different choices of the sessions lengths (namely 2, 4, 8 or 12 pages). The pages were drawn from \mathbf{W} with a Zipf distribution to mimic the average access frequencies of a real Web page [11]. The resulting similarities and standard deviations are listed in Table 1. Compared to Jaccard coefficient, the relative cosine gives higher similarities for sessions with identical lengths, and reduces the variance of measures. Yet,

Table 1: Mean similarity μ and std. deviation σ of different similarities applied to synthetic Web data.

$ S_A $	$ S_B $	cos μ, σ	jacc μ, σ	rel. cos μ, σ	sq. cos μ, σ
2	2	0.18, 0.26	0.20, 0.35	0.18, 0.26	0.10, 0.18
4	4	0.25, 0.19	0.22, 0.26	0.25, 0.19	0.10, 0.11
8	8	0.34, 0.13	0.30, 0.19	0.34, 0.13	0.13, 0.09
12	12	0.35, 0.10	0.29, 0.13	0.35, 0.10	0.13, 0.07
2	4	0.19, 0.21	0.16, 0.23	0.13, 0.15	0.08, 0.11
2	8	0.23, 0.17	0.13, 0.11	0.11, 0.08	0.08, 0.09
2	12	0.23, 0.14	0.11, 0.07	0.10, 0.06	0.07, 0.06
4	8	0.29, 0.15	0.22, 0.17	0.20, 0.11	0.11, 0.10
4	12	0.29, 0.12	0.18, 0.10	0.17, 0.07	0.10, 0.07
8	12	0.35, 0.11	0.28, 0.14	0.29, 0.09	0.14, 0.08

the average similarity is not penalized as in the squared cosine. Also, we remark that for sessions with similar lengths our measure performs identically to the original cosine, but without the drawbacks of having high similarities between paths with very different lengths. These results clearly motivate the use of the relative cosine measure.

3.2 Performance-oriented similarities

An important issue involved with the use of cosine-like measures is that pages contribute equally to the value of θ . Being interested in the performance evaluation of the system, we argue that the contribute of a single page to θ must be proportional to the load that it imposes to the system. This workload can be characterized using, if available, the disk retrieval latency T_{HD} (for non cached-pages), the download time T_{Dow} and the CPU processing time T_{CPU} ; for dynamic pages we can also consider the implied load at the application server T_{Dyn} and at the DB server T_{DB} . Then, we call *time demand* of a page p_j the quantity $\tau(p_j \in S(u, i))$ defined as

$$\tau(p_j) = T_{HD} + T_{Dow} + T_{CPU} + T_{Dyn} + T_{DB}, \quad (3)$$

that is the overall time spent by the system to process the requests for page p_j in session $S(u, i)$. Clearly, $p_j = 0$ implies $\tau(p_j) = 0$. In general, if P_j has been downloaded multiple times, $\tau(p_j)$ has to be computed as the cumulative service demand of the requests. Note that $\tau(p_j)$ is not a response time, since it does not include the time spent by the queued requests waiting for service. Now let $\tau(S(u, i))$ be the tuple defined as

$$\tau(S(u, i)) = (\tau(p_1), \dots, \tau(p_j), \dots, \tau(p_W)),$$

we can integrate $\tau(\cdot)$ with the relative cosine similarity as follows:

$$\theta = \frac{\langle \tau(S(u, i))^{1/2}, \tau(S(v, k))^{1/2} \rangle}{\max(|\tau(S(u, i))|, |\tau(S(v, k))|)} \quad (4)$$

where the term

$$|\tau(S(u, i))| = \sum_w \tau(p_w), \quad p_w \in S(u, i)$$

is the total amount of time required by the system to serve $S(u, i)$. The above formula is immediately obtained substituting $\tau(S(u, i))$ to $S(u, i)$ in the formula of the relative cosine, and adding a square root to the terms at the numerator to normalize θ in $[0, 1]$. In this way, the numera-

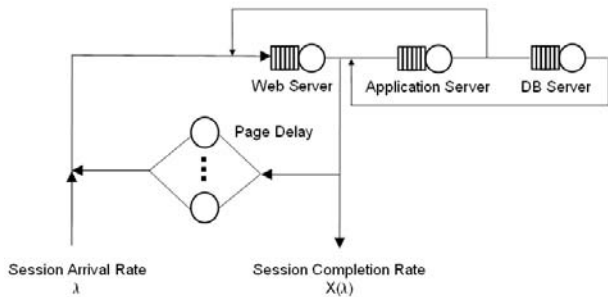


Figure 2: Web site model that includes the inter-arrival times of requests within a session.

tor becomes the sum of the geometric averages of the times spent on pages common to both $S(u, i)$ and $S(v, k)$, with the nice side-effect of making θ both normalized and robust to the presence of noise in $\tau(S(u, i))$ or $\tau(S(v, k))$. An open issue we leave for future work is if the maximum at the denominator could become a problem in presence of unreliable data.

4. INFERENCE OF QUEUEING NETWORK PARAMETERS

Once that the similarity matrix has been computed with (4), any non-metric clustering algorithm can build the desired performance classes. These may include several outliers and uncorrelated sessions, whose removal would lead to an underestimate of the workload. A better idea is to group the clusters with low intra-similarities in a single *non-typical workload* cluster.

Before considering the QN parameter estimation problem, we propose in Figure 2 a modified version of the QN discussed in Section 2.2. The main difference is that the arrival rate λ is now expressed in sessions/sec, rather than in requests/sec, and this allows a more intuitive mapping of sessions $S(u, i) \in \mathbf{S}$ to the QN model: in fact each session of a cluster can now be represented with a single circulating job of the model, and the requests associated with a session are modeled as a series of passages at the Web server, each delayed by an average time of Z_r (seconds/job), for all sessions belonging to cluster r . The presence of the delays Z_r is expressed graphically by the *Page Delay* object. For the QN of Figure 2, the required input parameters are: i) the loading L_{ir} of cluster- r sessions at the three servers; ii) the page delays Z_r , and the rates λ_r at which sessions of class- r joins the system.

We propose the following approach to compute L_{ir} , Z_r and λ_r . Let $c(r)$ be the cardinality of cluster r , $t(h_1)$ the arrival time of request h_1 , and $\tau^i(S(u, k))$ the vector derived by $\tau(S(u, k))$ by setting to zero all times demands that have not been served by server i . Then the QN parameters can be computed as:

- $L_{ir} = \frac{1}{c(r)} \sum_{S(u, k) \in r} |\tau^i(S(u, k))|$
- $Z_r = \frac{1}{c(r)} \sum_{S(u, k) \in r} \frac{1}{|\tau(S(u, k))|} \sum_{\substack{(h_1, h_2) \in S(u, k) \\ h_2 \text{ follows } h_1}} t(h_2) - t(h_1)$

- $\lambda_r = \frac{c(r)}{T}$

where T denotes the period in which the system was observed. The three terms require little comments, since they are obtained directly from their definitions: L_{ir} is computed as the cumulative demand of cluster- r sessions at server i , normalized over the number of sessions $c(r)$; Z_r is the mean of the average inter-arrival times of sessions $S(u, k) \in r$; finally, λ_r is the average arrival rate of new sessions in T . A refinement of the above formulas may be easily obtained introducing additional weights that account for the popularity of each page within the cluster to model the different frequencies at which the pages are visited.

5. CASE STUDY

We now illustrate our methodology on a simple case study. We remark that the following example should not be considered as a validation of our methodology, but just a practical example of the steps required to devise the QN model. In this example we use the log data of the university Web site <http://www.polimi.it>. At the time logs were collected, the site was composed of over 3600 *static* Web objects. Access data were collected using the extended common logfile format, and without session information (no cookies). Note that the absence of cookies makes harder to reconstruct user sessions, and thus represents an obstacle, rather than a simplification, for our analysis. Also, note that the presence of proxies and browser caches does not affect the present analysis, since we are interested only in the processed workload, and not in requests that have not been served. During the data preparation phase, we filtered out all HTTP requests that did not imply the download of some data (head methods, not-modified status codes, ...). Then, we used a standard session timeout value of 30 minutes to sessionize the requests, and we extracted a random sample \mathbf{S} of 1000 sessions, whose statistical properties are summarized in Table 2. After computing the similarity matrix with (4), the clustering was performed using Chameleon [9]. To limit the complexity of the resulting queueing network, we fixed the number of clusters to 15. The results, expressed in terms of average intra-cluster and extra-cluster similarities, are shown in Table 3: the first eight clusters offer good intra-cluster similarities and almost all clusters have very low values of the extra-cluster similarities, thus proving again the quality of the relative cosine similarity. Clusters from 9 to 14 are merged in the non-typical workload cluster. The results of the application of the inference formulas are then shown in Table 4. Since the extended common logfile format does not give information on file retrieval times or on CPU times, we computed the values τ_{pj} estimating the download times T_{Dow} , assuming an hypothetical bandwidth of 100Mbit/s. Moreover, as the site contained only static pages, we considered only two service centers: the Web server and the page delay, parametrized with the terms $L_{WS,r}$ and Z_r respectively. Using the inferred parameters, we solved the model and determined the average utilizations of each class at the Web server. Qualitatively, we found that the Web server was lightly loaded, being at an average utilization level below 1%, and this suggests that queues are rather infrequent. This result could be explained by the fact that a consistent part of the student services are offered by other servers of the campus (e.g. departmental servers). As an example of the power in this context of QN models, we can easily devise

Table 2: Statistical properties of the session sample (μ =mean, σ =std. deviation, me =median).

Requests	16889	Sessions	1000
Requests per session	$\mu=16.889$ $\sigma = 4.8853$ $me = 19$	KBytes per session	$\mu=35.424$ $\sigma = 128.19$ $me = 14.136$

Table 3: Mean values μ and standard deviations σ of the intra-cluster similarities θ_I and of the extra-cluster similarities θ_E .

cluster	size	$\mu(\theta_I)$	$\sigma(\theta_I)$	$\mu(\theta_E)$	$\sigma(\theta_E)$
0	212	0.957	0.040	0.154	0.004
1	79	0.856	0.100	0.214	0.025
2	72	0.681	0.099	0.062	0.010
3	136	0.620	0.150	0.214	0.050
4	32	0.465	0.123	0.041	0.015
5	21	0.385	0.148	0.005	0.005
6	22	0.316	0.237	0.001	0.000
7	17	0.249	0.199	0.004	0.006
8	29	0.224	0.106	0.029	0.015
9	29	0.159	0.094	0.002	0.003
10	28	0.089	0.079	0.003	0.003
11	53	0.103	0.074	0.005	0.004
12	31	0.087	0.045	0.001	0.002
13	27	0.059	0.049	0.001	0.001
14	212	0.000	0.003	0.000	0.000

the variation of the average number of class- r requests waiting in queue under a change of the arrival rate of requests λ_r as

$$\frac{\partial Q_{WS,r}}{\partial \lambda_r} = \frac{L_{WS,r}}{(1 - U_{WS})^2}$$

where U_{WS} is the utilization of the Web server. Formulas of this type can be easily derived and are fundamental to evaluate the scalability of the architecture, since the growth of response times is in general non-linear and cannot be forecasted easily without resorting to queueing networks.

6. CONCLUSIONS AND FUTURE WORKS

In this paper we presented a methodology to map the results of a Web usage mining process to a queueing networks performance model. We defined a new similarity measure suitable for a performance evaluation of a Web system. Our methodology was illustrated on a real case study. However, its validation has been limited to its ability of creating homogenous clusters with low intra-cluster variance. A detailed comparison with existing methodologies for inferring queueing networks parameters is left as a future work. These also include the integration of the proposed techniques in a single software tool for the generation of performance models automatically, a study of the behavior of our measures in presence of unreliable data and a validation on sites employing cookies and dynamic pages.

7. REFERENCES

[1] R. Agrawal, R. Srikant. Fast Algorithms for Mining Association Rules. *Proc. 20th Int. Conf. Very Large D.Bases (VLDB)*, 1994.

Table 4: Queueing networks parameters inferred from the log file.

class	clusters	λ_r (sess/sec)	$L_{WS,r}$ (sec)	Z_r (sec)
0	0	0.0053	0.00078	210.15
1	1	0.0020	0.00150	45.47
2	2	0.0018	0.00006	1.94
3	3	0.0034	0.00095	196.71
4	4	0.0007	0.00014	3.297
5	5	0.0005	0.00007	38.98
6	6	0.0005	0.00001	0.82
7	7	0.0004	0.00007	0.93
8	8	0.0007	0.00005	0.63
9	9-14	0.0102	0.00013	36.94

[2] M. Arlitt, T. Jin. A Workload Characterization Study of the 1998 World Cup Web Site. *HP Tech. Rep. HPL-1999-35(R.1)*, 1999.

[3] K. Benabdeslem, Y. Bennani, and E. Janvier. Connectionist Approach for Website Visitors Behaviors Mining. *Proc. of the ACS/IEEE Int. Conf. on Comp. Sys. and App. (AICCSA'01)*, 2001.

[4] B. Berendt, B. Mobasher, M. Spiliopoulou, and J. Wiltshire. Measuring the Accuracy of Sessionizers for Web Usage Analysis. *Proc. of the 1st SIAM Int. Conf. on Data Mining*, 2001.

[5] G. Casale, G. Serazzi. Bottlenecks Identification in Multiclass Queueing Networks using Convex Polytopes. *In Proc. 12th IEEE / ACM Int. Symp. on Model., Anal., and Simul. of Comp. and Tlc. Sys. (MASCOTS 2004)*, pp:223-230, IEEE Press.

[6] R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *J. of Knowledge and Inf. Sys.*, 1(1), 5-32.

[7] V. Estivill-Castro, J. Yang Categorizing Visitors Dynamically by Fast and Robust Clustering of Access Logs. *LCNS 2198*, pp. 498+, 2001.

[8] A. Joshi, R. Krishnapuram. Robust Fuzzy Clustering Methods to Support Web Mining. *In Proc. W.shop in Data Mining and Know. Discovery*, 15-1 - 15-8, 1998.

[9] G. Karypis, E.S. Han, and V. Kumar CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. *IEEE Computer*, 1999.

[10] B. Mobasher, H. Dai, T. Luo, Y. Sun, and J. Zhu. Integrating Web usage and content mining for more effective personalization. *In Proc. 1st Int. Conf. on E. Commerce and Web Tech.*, 165 - 176, 2000.

[11] M. Rabinovich, O. Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[12] E.D., Lazowska, J., Zahorjan, G.S., Graham, K.C., Sevcik, Quantitative System Performance: Computer System Analysis Using Queueing Network Models. Prentice-Hall, 1984.

[13] J. Srivastava, R. Cooley, M. Desdhpande, and P. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2), 2000.

[14] O.R. Zaiane, M. Xin, J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Techniques on Web Logs *Proc. Advances in Digital Libraries Conf. (ADL98)*, pp.19-29, 1998.