Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 232 (2009) 89–99

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Statistically Customisable Web Benchmarking Tool

Katja Gilly[a,1], Carlos Quesada-Granja[a,2], Salvador Alcaraz[a,3], Carlos Juiz[b,4] and Ramon Puigjaner[b,5]

[a] *Física y Arquitectura de Computadores*
Miguel Hernández University
Elche, Spain

[b] *Ciències Matemàtiques i Informàtica*
University of Balearic Islands
Palma de Mallorca, Spain

Abstract

We describe a statistically customisable solution for Web benchmarking that includes three utilities: an Internet traffic generator for HTTP traffic, an static and dynamic Web pages generator in the Web server that is going to be tested and a performance monitor that reports some performance metrics during the test. The design of the tool is based on the implementation of the most important probability distributions that mainly characterise statistical properties of Internet traffic like the inter-arrival rate of the incoming Web traffic to the server, the number of objects contained in a Web page and the size of these Web objects. These properties can be customised for a specific capacity planning analysis. In this work, we describe the generation process of the Web pages and the customisable traffic characterization used by the benchmark. The monitoring process details and the validation of the probability distributions implemented are also included.

Keywords: Web Benchmarking, Dynamic and Static HTTP traffic

1 Introduction

This work is motivated by the need of generating a specific HTTP workload in a Web Server testing environment based on probability distributions to emulate a real user behaviour asking for Web content. It is well known that Internet traffic flows exhibit heavy-tailed probability distributions [2,7]; this is the main reason that

[1] Email:katya@umh.es
[2] Email:carlos.quesada@graduado.umh.es
[3] Email:salcaraz@umh.es
[4] Email:cjuiz@uib.es
[5] Email:putxi@uib.es

doi:10.1016/j.entcs.2009.02.052

lead us to implement a Web benchmarking tool based on these type of probability distributions.

Normally, a Web benchmark generates an HTTP request when the previous one has been satisfied. In fact, most of current available benchmarks usually generate the workload by creating sequential HTTP requests from each client involved in the test. This is the case of popular benchmarks as ApacheBench, Httperf, Webstone and SPECweb2005. ApacheBench is a command line program that comes with Apache HTTP Server; it basically tests how many requests per second a Web server (typically the Apache HTTP Server) can serve. Httperf is also a command line benchmark that is completely detailed in [8]. Webstone permits the generation of the file list that a set of concurrent Web clients are going to sequentially ask for during a period of time. SPECweb2005 generates a number of simultaneous sessions that corresponds to the number of load-generating processes that will continuously send requests to the HTTP server during the benchmark run. A new user session starts as soon as the previous user session ends, and this process continues until the benchmark run is complete. All these benchmarks generate requests individually or based on user sessions in a pure sequential scheduling, despite they can use concurrent client processes.

In 1998, Bradford *et al.* published in [3] the results obtained by SURGE, a tool for generating Web requests based on analytical models. In 2002, Kant *et al.* developed Geist, a benchmark that included the possibility of generating self-similar aggregated traffic [6]. They improved the traffic generation by handling the responses by separate processes to avoid the condition of receiving a response prior to generate the next request. The traffic generation is based on an HTTP log that represents the aggregate traffic as seen by the Web server. This reflects an improvement compared to previous benchmarks, but it still do not permit to customise the probability distribution that models a property that characterise the HTTP workload.

Our goal is to generate a specific and customisable workload based on probability distributions. These distributions will model some statistical properties of the Web test that the benchmark will perform. The properties that can be defined through these distributions are: the inter-arrival rate of the incoming Web traffic to the server, the number of objects contained in a Web page and the size of the Web objects. The fact of customising a benchmark by using probability distributions is very useful to reproduce some workloads that follow a determined characteristics that are described in literature. It can also be used to emulate real traffic in simulation scenarios and to validate the results that are obtained by some network simulators as OPNET Modeler or *ns*-2.

The rest of the paper is organised in the following way. Section 2 discusses the motivation of using probability distributions to characterise the workload the benchmark introduces in the Web server. The next three sections outline the proposed benchmark. The Web page generator and the Web traffic generator are presented in Section 3 and 4, respectively. Some details about the monitoring process are described in Section 5. The experimental results and the validation of the probabil-

ity distributions implemented are included in Section 6. Section 7 discusses some concluding remarks.

2 Motivation of using probability distributions

Internet literature describes the general characteristics of the traffic flows in Internet and the structure of the Web pages that are mainly demanded by the users [3,5,9,1]. Hence, based on the characteristics reported by these works, some performance experiments in Web servers need to reproduce this workload synthetically by using a Web benchmark. Normally, the features of the Web are detailed in the form of probability distributions. Some examples are the distribution of the objects per page request, the number of objects per page, the number of page requests per site, the user think time, the inter-arrival rate, etc. Therefore, our contribution to this field is the development of a statistically customisable benchmarking tool with probability distributions to reproduce the experiments reported in Web traffic literature.

We have selected three statistical properties to be configured in the benchmark. Previous works have investigated the probability distributions that model each of these properties. The benchmarking tool we present in this paper has implemented the distributions that can model these properties, although the benchmark can also be configured with other distributions. The statistical properties and their corresponding probability distributions according to literature are the following:

- **The number of objects per page.** A Web page can contain many kinds of information in addition to textual information. These elements are defined as objects and can be images, audio, video, scripts, ..., etc. The probability distribution that models the number of objects per page is Pareto [1].
- **The objects file size.** The distribution that models the file size of the objects that compound the Web pages is usually heavy-tailed and it is normally set as Pareto or Lognormal [9,1,5].
- **The inter-arrival time.** It is defined as the differences between consecutive HTTP requests arrivals to the Web server. The inter-arrival time is normally modeled as Pareto [1], but some authors consider it can be hyper-exponential distributed [10].

We have also considered the possibility of modeling all these properties with other distributions. The benchmarking tool can be customised with the five probability distributions that are: *uniform*, *exponential*, *hyper-exponential*, *pareto* and *lognormal*.

3 Web Page Generation

The Web pages the benchmark asks for have to be previously created in the Web server by following the characteristics introduced by the user. We have considered the creation of both static and dynamic Web pages in the Web server. The static

pages are basically composed by HTML code that include a certain number of objects. The HTTP requests that ask for dynamic content require the Web server to access to a database server to obtain the data requested. Dynamic Web pages can also include a certain number of objects. The number of objects each page contains is modeled according to the distribution specified by the user.

We organise the Web server site construction in two different steps. Firstly we create the Web objects that compose the Web pages by following the distribution assigned to model the size of the Web objects. Once all the Web objects are created, the second step consists of the inclusion of these objects in the Web pages by following the distribution that model the number of objects that are contained in the Web pages.

The benchmarking client sends the information introduced by the user to the Web server. This information determines the characteristics of the Web pages that are going to be generated. It includes the probability distributions that affect to the number of objects per page and the size of these objects. A process starts in the Web server to generate the file structure that afterwards is going to be demanded by the Web client. Once the file structure has been created, the paths and the names of the Web files are sent back to the client to inform the benchmark about where are the files the client is going to ask for.

The generation process varies depending on the type of Web pages generated. It is briefly described in the following subsections.

3.1 Static Web pages

Static Web pages consist of an HTML file that includes a number of objects that usually are images or multimedia files. We have generated the objects as text files, that are embedded in the HTML file with frames, because it is easier to control the size of text files than images or other file types. The frames are organised in a matrix structure whose number of rows (r) and columns (c) depend on the number of objects (n) that are included in the Web page by using the following expressions.

$$c = \lfloor \sqrt{n} - 1 \rfloor \qquad r = \left\lfloor \frac{n-1}{c} + 1 \right\rfloor . \tag{1}$$

When $c < 1$, we set c to 1. This frame structure permits to correctly visualise all the frames in the HTML Web page as it can be observed in Figure 1a.

The Web objects generation is done by an application that adds characters to a text file. The size of each Web object is predetermined by the probability distribution specified by the user.

3.2 Dynamic Web pages

The generation of dynamic Web pages implies that a database server is accessible from the Web client to create a new database with n columns. Each column i, $i \leq n$, will contain a string character whose size is $2^{(i-1)}$. The purpose of this is to build a string character of any possible size by concatenating the text contained in different database columns. The final string character is afterwards included in a

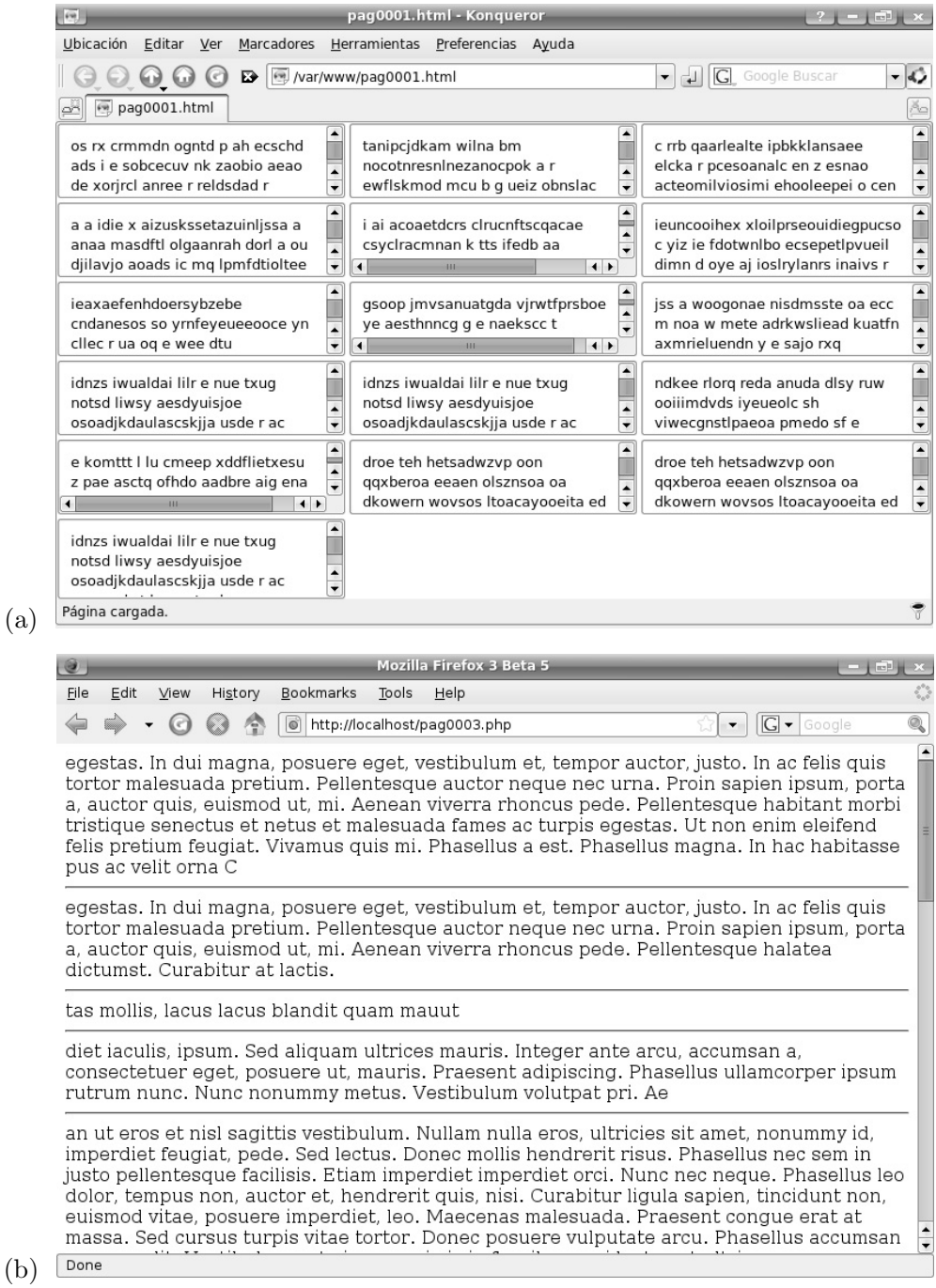

Fig. 1. An example of a (a) static and (b) dynamic Web page generated by the benchmark

text file that is going to be embedded as an object in the dynamic Web pages. The objects are included in the dynamic Web pages by using frames as it is shown in Figure 1b. Table 1 shows a database design and content example.

id	...	c16	c8	c4	c2	c1
1	...	Lorem ipsum dolo	r sit am	et,	co	n
2	...	sectetuer adipis	cing eli	t. D	on	e
3	...	c rutrum, lectus	ullamco	rper	ul	l

Table 1
An example of the database design and content used to generate dynamic Web pages.

4 Workload Generation

The generated workload is composed by HTTP requests that ask for the static and dynamic Web pages that have been previously created. The inter-arrival time among HTTP requests is modeled by the distribution specified by the user. As the workload generator has to follow the inter-arrival time modeled by the distribution, the workload generator has to create a different process for each request sent to the Web server to avoid a sequential treatment of responses. Each processes created represents an individual client that requests a Web page.

There are some differences between the implementation of the downloading of the static and dynamic pages. Let us describe both methods separately.

4.0.1 Static Web pages.
Although HTTP 1.1 enables pipelining of some requests in one TCP connection, we have chosen to download the objects that are embedded in the HTML file in parallel connections to reduce the total Web page download time [4]. Once the HTML page is retrieved by the client, a parallel TCP connection is open for each object included in the HTML file. This allows us to better control the object download time and its repercussion in the Web server performance. The Web server workload implied in the generation of a response that contain a static Web page basically consists of some read operations on the disk drive to retrieve the information contained in the HTML file. This includes a read operation for each of the objects that are embedded in the static Web page.

4.0.2 Dynamic Web pages.
In this case, the complete dynamic Web page is sent through the same TCP connection. Once the request from the client arrives to the Web system, the database server needs to be accessed. It is necessary to retrieve the text information of each object embedded in the Web page requested from the database server as it is described in previous Section. A read operation in the database is essential to obtain the content of each object. Hence, the database server has to be accessed as many times as the number of objects the dynamic Web page requested has. Once all the information contained in the objects is obtained, the complete dynamic Web page is sent through the TCP connection. Therefore the workload in the Web system increases with the number of objects embedded in the Web page requested.

5 Performance Monitoring Process

Before starting the workload generator, the benchmarking tool executes the performance monitoring process in the Web server to get some information about the CPU utilization while the test is being performed. The processes involved in the Web server system are the Web server process and the database server process. In the case of static pages the database server is not monitored. A log is created with the information obtained by the monitor during the test.

Other performance metrics are also got during the workload generator execution, these include the downloaded objects size for each Web page, the response time to download each object and time needed to get a complete Web page.

The information obtained by the monitor permits to estimate the error introduced in the implementation of the probability distribution depending on whether it were used to model the inter-arrival time, the number of objects per page or the objects size.

6 Results and validation

The benchmarking tool presented in this paper is completely implemented in C. The Web server used is Apache HTTP Server and the database server used is MySQL. Dynamic Web pages are designed with PHP, while static pages are HTML files.

The test-bed scenario consists of two computers that are connected through a FastEthernet switch. The processor of the computer that acts as the Web server is an Intel Pentium D at 3000 MHz and it has 2 GB of RAM Memory. The client runs in an AMD Turion 64 at 1600 MHz and with 512 MB of RAM.

Let firstly analyse the tests performed with the benchmarking tool and the experimental results obtained to then proceed with the validation of the probability distributions implemented.

6.1 Experimental results

The set-up of the experiments consist of nine different tests that include all the probability distributions implemented by the benchmarking tool and that are fully described in Table 2. The first column of the Table represents the number of the test. Each of the tests has been executed during 10 minutes with 10 different seeds.

The average response time of these experiments is detailed in table 6.1. The object response time of dynamic pages is not obtained because the complete Web page is sent in just one step to the client. It can observed that the static Web pages response time is always lower than the one monitored when requesting dynamic pages.

6.2 Validation

Validation consists of verifying that the benchmarking tool output matches with the distributional models that characterise the file sizes, number of objects per page and

T	Inter-arrival Time (sec)	Objects Size (bytes)	Objects per page
1	Constant ($c = 0.03$)	Constant ($c = 200$)	Constant ($c = 30$)
2	Uniform ($v_{min} = 0.01, v_{max} = 0.05$)	Uniform ($v_{min} = 0, v_{max} = 400$)	Uniform ($v_{min} = 0, v_{max} = 60$)
3	Exponential ($\mu = 0.03$)	Exponential ($\mu = 200$)	Exponential ($\mu = 30$)
4	Exponential ($\mu = 0.03$)	Pareto ($x_m = 57, k = 1.4$)	Constant ($c = 1$)
5	Pareto ($x_m = 0.018, k = 2.5$)	Pareto ($x_m = 57, k = 1.4$)	Constant ($c = 1$)
6	Hyper-exponential ($\mu_1 = 0.025, \mu_2 = 0.1, p_1 = 0.933$)	Pareto ($x_m = 57, k = 1.4$)	Exponential ($\mu = 30$)
7	Exponential ($\mu = 0.03$)	Lognormal ($\mu = 200, \sigma^2 = 40000$)	Pareto ($x_m = 18, k = 2.5$)
8	Pareto ($x_m = 0.018, k = 2.5$)	Lognormal ($\mu = 200, \sigma^2 = 40000$)	Pareto ($x_m = 18, k = 2.5$)
9	Hyper-exponential ($\mu_1 = 0.025, \mu_2 = 0.1, p_1 = 0.933$)	Lognormal ($\mu = 200, \sigma^2 = 40000$)	Pareto ($x_m = 18, k = 2.5$)

Table 2
Set-up details of the experiments performed

inter-arrival times. We have chosen some of the performance results obtained by the performance monitor to compare visually how well an empirical data set fits analytic distribution models.

Figure 2 shows the CDFs of the observed inter-arrival rate data versus its probability distribution and details the number of their corresponding tests. In the case of the Figure 2a the distribution is uniform. It indicates a good fit between the observed inter-arrival rate CDF and the uniform distribution despite a small peak before 10 ms. Figures 2b and 2c show the results for an inter-arrival rate modeled with exponential and hyper-exponential distributions, respectively. Pareto distribution results are shown in Figure 2d, and in this case there is also a small peak

	Static Requests		Dynamic Requests
T	Page Resp. Time (s)	Object Resp. Time (s)	Page Resp. Time (s)
1	0.0178	0.000606	0.0212
2	0.0192	0.000495	0.0277
3	0.0266	0.000693	0.0409
4	0.0092	0.001228	0.0139
5	0.0037	0.000476	0.0055
6	0.0102	0.001834	0.0149
7	0.0305	0.000614	0.0379
8	0.0259	0.000649	0.0274
9	0.0339	0.000610	0.0437

Table 3
Average Page and Object Response Time for dynamic and static requests

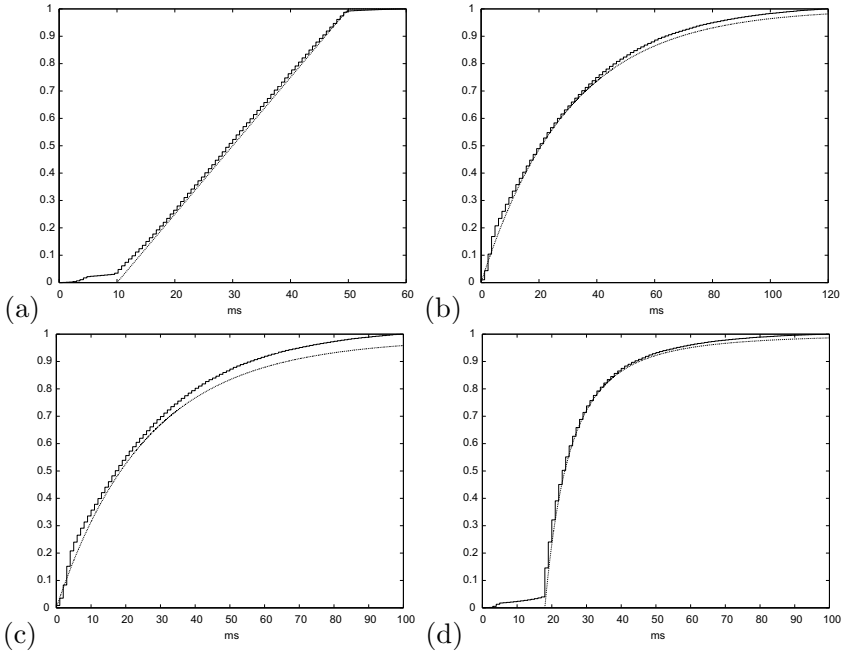

Fig. 2. CDF of (a) Inter-arrival time as Uniform (Test 2); (b) Inter-arrival time as Exponential (Tests 3-4,7); (c) Inter-arrival time as Hyper-exponential (Tests 6,9); (d) Inter-arrival time as Pareto (Tests 5,8)

below 18 ms.

The reason of the peaks observed in Figures 2a and 2d is due to the use of signals in the C code to schedule processes. There can be some delays among the signal and its associated callback function execution. As every request is scheduled by

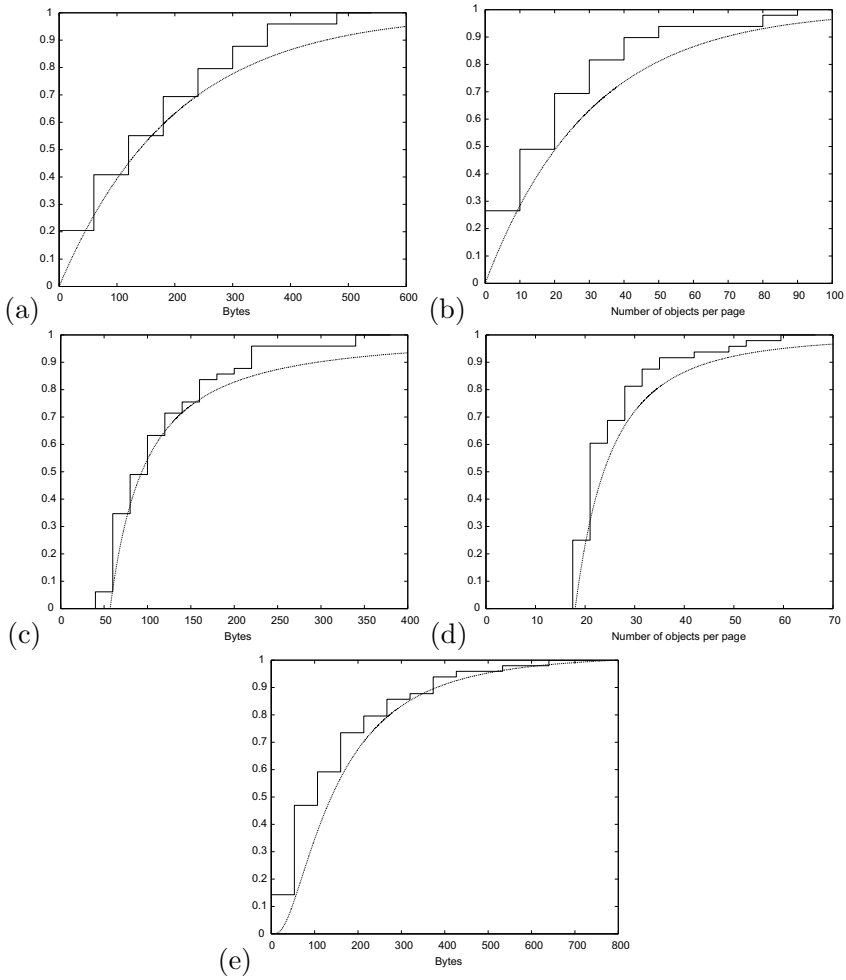

Fig. 3. CDF of (a) Objects Size as Exponential (Test 3); (b) Number of objects per page as Exponential (Test 3); (c) Objects Size as Pareto (Tests 4-6); (d) Number of objects per page as Pareto (Tests 7-9); (e) Objects Size as Lognormal (Tests 7-9)

the probability distribution, the delay caused by the signals do not alter the next request time. Hence the inter-arrival time for the next request is reduced to fulfill the probability model used for inter-arrival times.

Figure 3 shows the CDFs of the observed number of objects per page and objects size data generated versus the analytic models. Due to the smaller number of values in these cases compared to the inter-arrival rate data, the plots in Figure 3 are more scaled. Figure 3a, 3c and 3d show the objects size CDF when using an exponential, pareto and lognormal distribution, respectively. The number of objects per page monitored are represented in Figures 3b and 3e.

In general, the distributional plots of Figure 2 and Figure 3 fit quite well for the data obtained, according to the number of values of each data set.

7 Conclusions and Future Work

A flexible Web benchmarking tool that can be statistically customised is presented in this paper. The tool permits generating a representative workload based on the probability distributions that mainly characterise some useful properties as the number of objects per Web page, the objects size and the inter-arrival rate. This tool also includes a Web page generator and a monitoring process that permits to obtain some important performance metrics in the Web client and server. This benchmarking tool fills a gap in the currently available set of benchmarks. The validation of the customised properties is also presented at the end of the paper.

Future work will include the implementation of user sessions that currently are not included and the possibility of mixing dynamic and static traffic in one test of the benchmark.

Acknowledgement

This work is partially supported by the Spanish Ministry of Education and Science through the TIN02006-022265 Research project.

The authors would like to thank Jose Antonio García Orza for his help in various parts of this work.

References

[1] Andreolini, M., M. Colajanni and M. Nuccio, *Scalability of contentaware server switches for cluster based web information systems*, in: *proceedings of WWW*, 2003.

[2] Arlitt, M. and T. Jin, *A workload characterization of the 1998 World Cup Web site*, Technical report hpl-1999-35r1, HP Labs (1999).
URL citeseer.nj.nec.com/article/arlitt99workload.html

[3] Barford, P. and M. Crovella, *Generating representative web workloads for network and server performance evaluation*, in: *ACM SIGMETRICS Performance Evaluation Review* (1998), pp. 151 – 160.

[4] Bent, L. and G. M. Voelker, *Whole page performance*, Technical report, University of California at San Diego (2002).

[5] Crovella, M. and A. Bestavros, *Self-similarity in world wide web traffic: Evidence and possible causes*, in: *Proceedings of SIGMETRICS: The ACM International Conference on Measurement and Modeling of Computer Systems.* (1996).
URL http://www.cs.bu.edu/fac/best/res/papers/sigmetrics96.ps

[6] Kant, K., V. Tewari and R. Iyer, *Geist: A web traffic generation tool*, Computer Performance Evaluation: Modelling Techniques and Tools **2324** (2002), pp. 325–352.

[7] Meiss, M., F. Menczer and A. Vespignani, *On the lack of typical behavior in the global web traffic network*, in: *proc. of World Wide Web*, 2005, pp. 510–518.

[8] Mosberger, D. and T. Jin, *httperf: A tool for measuring web server performance*, in: *First Workshop on Internet Server Performance* (1998), pp. 59–67.
URL citeseer.nj.nec.com/mosberger98httperf.html

[9] Pitkow, J. E., *Summary of www characterizations*, International WWW Journal **2** (1999), pp. 3–13.
URL citeseer.nj.nec.com/pitkow99summary.html

[10] Pothuri, S., D. W. Petr and S. Khan, *Characterizing and modeling network traffic variability*, IEEE International Conference on Communications **4** (2002), pp. 2405–2409.