INITIAL TOOL FOR MONITORING PERFORMANCE OF WEB SITES

Cristina Hava[*] & Stefan Holban
Faculty of Automation and Computer Engineering, "Politehnica" University Timisoara,
2 Vasile Parvan, Timisoara, Romania, chava@mail.dnttm.ro, havac@eeng.dcu.ie, stefan@utt.ro
John Murphy & Liam Murphy
School of Electronic Engineering, Dublin City University, Dublin 9, Ireland
murphyj@eeng.dcu.ie, murphyl@eeng.dcu.ie

Abstract

A very important application, which offers access to information on the Internet, is the World Wide Web (WWW or Web). The WWW is primarily responsible for the growth of e-commerce activity by the use of the Internet. To improve users' satisfaction with e-commerce WWW services, a lot of work has been done on characterizing the performance of Internet applications and servers. We discuss in this article about a monitoring application for performance measurement of the WWW Servers. By observing the behaviour of WWW servers and measuring their performance, we can estimate several performance indices. Our application simulates the access of a number of clients to the same Web page and its links and analyses the results. Three different cases are taken into account: the clients access the same Web page one after the other (i.e. serially), at the same time (i.e. in parallel), checking before a 'cache' structure.

1. Introduction

Many people who use the WWW want a good quality service, and the level of service expected does not diminish even when the service is "free". To improve users' satisfaction, the administrators of WWW servers have to manage their servers and to provide a good service. Typically this could involve to monitor the changes in performance indices (such as the minimum/average/maximum number of HTTP requests), the volume of total traffic accessing site's services, the average response time as it varies across hours, days, weeks, etc, the service availability, connection establishment and data transfer times. This type of performance assessment and evaluation is difficult because there are many factors, which can influence Web server performance.

There are some related work concerning WWW server and Internet performance tests developed. Some tests were run on the servers and others were done on the client machines.

A group of researchers from Japan have proposed a new method for measuring the performance of a WWW server. Their method is based on packet monitoring to reveal all the behaviour of the server, and then deriving several performance indices based on this monitoring [1]. Another project, testing the servers' performance, was PingER. It was run on different clients' machines. The project [2] involves monitoring end-to-end links between nearly 500 computers at over 300 sites throughout the world. This project uses ping to measure the response time, the packet loss percentages, the variability of the response time (both short-term and longer), and the lack of reachability. A similar project to PingER is Surveyor [3]. PingER sends a series of ICMP echo request messages grouped together on a fixed schedule. Surveyor constantly sends out small UDP packets on a Poisson schedule.

Our work's goal is to simulate the access of a number of clients to the same web page and its link and to observe how these influence the performance indices. The current work extends the previously

[*] Work carried out while under an ITRC scholarship at DCU Electronic Engineering.

done research described in [4], focussing on the data transfer time and the availability of the server at different moments of day.

In this paper we discuss about three different cases, which are taken into account by our simulator. Usually the clients can access the same Web page in a serial mode (one after the other) or in a parallel mode (all the clients access the Web page at the same time), with or without consulting a "cache" structure.

In the first case the simulator creates a thread for each client. Only one thread is created and run at a moment. Only when the thread finished his task, the following one is created. The advantage of this approach is that both the Web server and the network are not overloaded. Is expected that the average computed time would be smaller for a client. The main disadvantage is that there is only one client connected at a time.

In the second case of study all the clients' threads are created simultaneously. All the threads run in parallel. In this situation we can see the variation in responses to similar requests from the same source at the same instance in time. A possible advantage of this approach is that the client waiting times are expected to be shorter than for the first case. The disadvantage is possible overloading of both the network and the server.

The last case of study can be applied to both previous cases. Before accessing the server, the clients check a so-called 'cache' structure. It stores a copy of each web page immediately after a first access to an Internet address was made. If the access occurs after a previous one, the client is informed by consulting an indexed structure about the local path where the file is stored and they are much rapidly got. It is expected that the accessing time to be reduced.

2. Four- Step HTTP Transaction

For all three different cases study, a thread implements each client. A thread consists of a number of modules, each of them in charge of a certain task. The tasks can be summarized in a four-step description [5]:

Step1: Establish a connection

Before a client and a server can exchange information, they must first establish a TCP/IP connection. The Internet uses the TCP/IP suite of protocols to let computers communicate. To distinguish protocols, applications use a unique number, called "port number" for each protocol. For example, for HTTP protocol the standard port number is 80. Also the user has to set some parameters for the Internet connection such as protocol, server name, the path for the web page, and optionally the port number of server.

Step2: Client issues a request

Each HTTP request a client issues to a Web server begins with a method (a command the client uses to specify the purpose of its server request), followed by an object's URL. The client also specifies the version of the HTTP protocol it is using and optionally information encoded in a particular header style.

Step3: Server issues a response

After a Web server receives and interprets a request message, the server responds with an HTTP "response message". The response message always begins with the HTTP protocol version, followed by a three-digit status code and reason phrase, and optional information encoded in a particular header style. Finally the server sends the data required by the client.

Step4: Server terminates the connection

It's the server's responsibility to terminate a TCP/IP connection with a client after it performs the client's request. However, both the client and the server must manage an unexpected closing of the connection.

3. "Ping" Mechanism

"Ping" is one of the most useful IP network debugging tools available. The idea comes from an equipment (SONAR), which was used on submarines. It sent a short sound burst and listened for an echo. Analyzing the echo the sender gets some information about the obstacle, which reflected the sound (e.g. distance, consistency, etc.). To get information about the destination, in IP networks ping sends a short data burst – a single packet- and listens for a single packet in replay. Ping is implemented using the required ICMP Echo function, documented in RFC792. [6]

Internet control Message Protocol (ICMP)

ICMP protocol is a required protocol tightly integrated with IP. ICMP functions include: announcing network errors (such as a host being unreachable), announcing network congestion, assisting troubleshooting (ICMP supports an Echo functions, which sends a packet on a round – trip between two hosts), announcing timeouts (when IP packet's TTL field drops to zero). Each ICMP message is encapsulated in an IP packet. Unfortunately the ICMP packet delivery is unreliable so the host cannot count on receiving ICMP packets for any network problem. The ECHO REQUEST and ECHO REPLAY

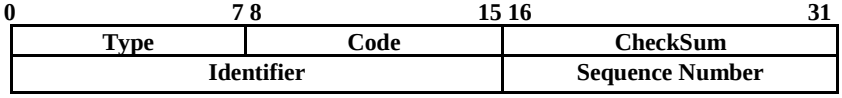

Figure 1: ECHO REQUEST packet's structure

messages are used to inform the sender if a given destination is reachable and alive. Their packet structures are documented in RFC792 and RFC791 [6].

Ping places a unique sequence number on each ECHO REQUEST packet it transmits and reports the sequence number it receives back. Unfortunately when multiple threads are used, and each thread issues a ping using his own raw socket, they don't receive only their own response from the server. The responses will be posted to all the threads. Each thread has to implement a mechanism to detect its own message among these replies. The ECHO REQUEST packet has a field Identifier (Figure 1) and each thread sets this field with its own ID. When a reply comes each thread compares the Identifier field with his ID and if it matches, that thread should process the response; otherwise it should ignore the response and wait for another one.

4. Tests' Result

We studied the servers' performance for all three-study cases. We choose six different servers to test them. These included two servers in Ireland, two in Europe and two in the US. Two of the servers were Universities and four were Portal sites and Figure 2 below summaries their main characteristics.

Server	Location	Type	Images	Size KB
1	Europe	Portal	78	73
2	USA	Portal	89	169
3	Ireland	University	25	85
4	Europe	University	12	90
5	Ireland	Portal	149	126
6	USA	Portal	28	33

Figure 2: Server Types and Locations

Serial access vs. parallel access

In the first case of study (serial access mode) only one client is connected at a moment. Our simulator creates a thread, which fetches the main page, and then computes the time elapsed. After parsing the HTML file, it sends a ping for all links found and computes the total time. Only after that, another test for a single connection can be run. For a set of serial connections the average time for five sequential accesses to the same Web is measured every hour.

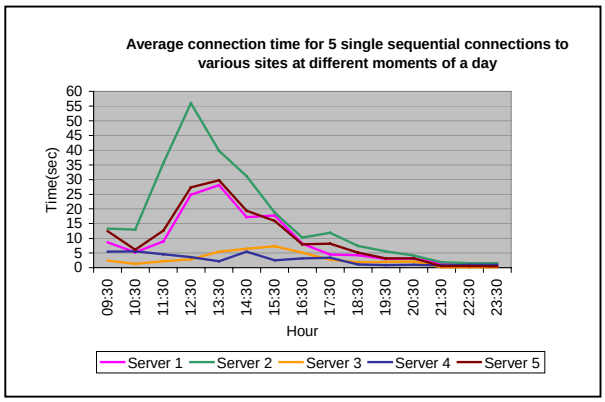

Figure 3: Serial Connections

In the second case of study we simulate multiple clients trying to fetch the same web page from the server. A number of threads equal with the clients' number are created simultaneously. All the threads perform similar tasks to the thread that was created for a single connection. The parallel threads have to share the processor, disks, memory and all server resources. The routing of the packets sent by the server

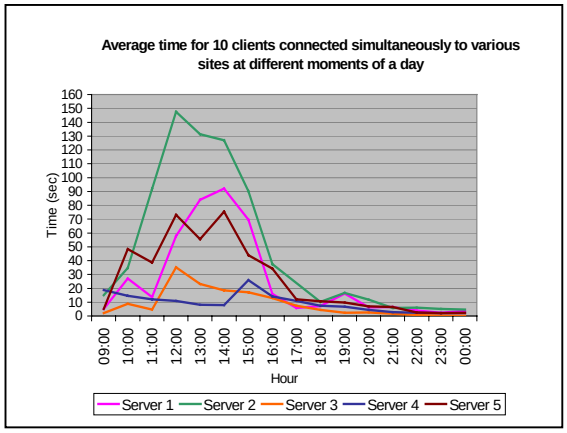

Figure 4: Ten Parallel Connections

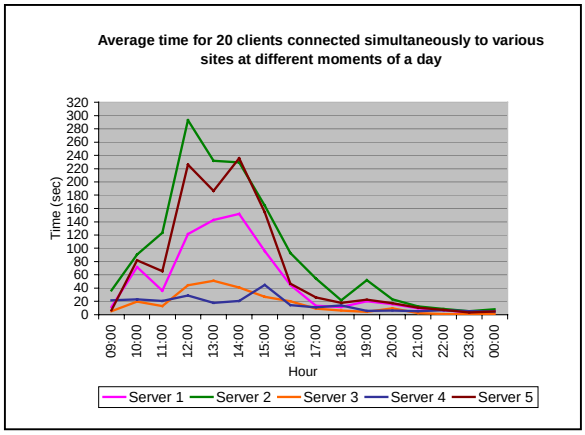

Figure 5: Twenty Parallel Connections

to the client may also vary. Furthermore the server could be overloaded or the network congested. All these facts can influence the computed time.

In Figure 3 are the results for serial connections to five different servers over different times of the day. The plotted number is the average of the download times. For the same servers at the same times there are parallel connections for 10 and 20 clients shown in Figures 4 and 5.

All three graphs show similar response time profile throughout the day with a peak in the morning that continues throughout most of the day and otherwise fairly low response times. From the graphs it can be seen that starting at 10am until 4pm the average time for getting the Web page is large in comparison with the afternoon and evening period. During the busy period the HTTP connection is established, but we have to wait longer time to obtain all the data from the WWW server. Following the previous pictures,

we can observe the connection time increases with the number of parallel clients trying to connect to the server.

The minimum, average and maximum response time for getting a Web page from a single site is plotted in Figures 6 and 7 at different moments. What can be seen is that the difference between the minimum and the maximum can be nearly an order of magnitude. For example there are periods during the day when the minimum response time is about 10 seconds while the maximum response time

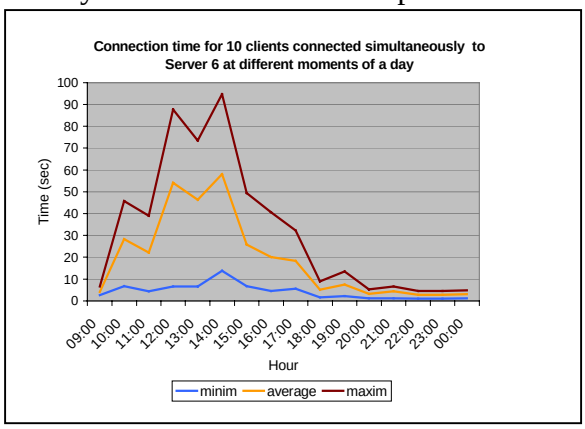

Figure 6: Minimum, Average & Maximum Response Times for 10 Parallel Clients

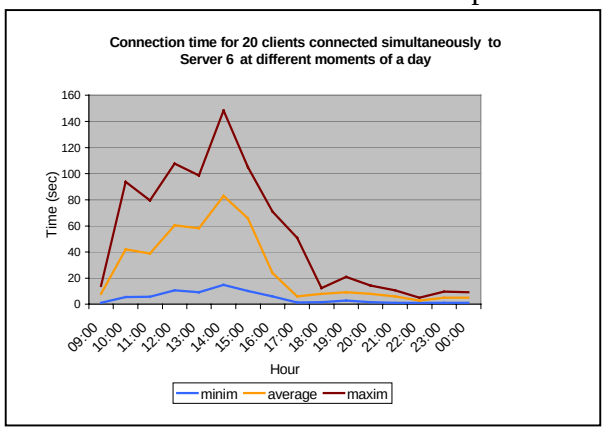

Figure 7: Minimum, Average & Maximum Response Times for 20 Parallel Clients

is about 90 seconds. All these tests were conducted during a normal working day.

Cache vs. No Cache

As we have seen in previous graphs, the WWW servers and network are overloaded during the working hours. In this case is very useful to use a cache to a proxy server from a company or university.

When there are a lot of requests for the same web pages from same location, only the first request downloads all the data from server and put them in the cache. Then, all the other requests read the information from the cache. In this case the time for fetching the web pages is smaller than the fetched time of the fist request. The worst situation is when all the requests are simultaneously. In this case only one request gets the information from the WWW server and put it in the cache, and all the other requests have to wait till they can read the information from cache. The fetched time for all the other requests is equal with fetching time of the fist request plus fetched time from the cache. We simulated this situation using a "cache structure" created on our computer.

The average fetched time for a site, with cache and without cache, when there are 10 and 20 requests in same time can be seen in Figures 8 and 9.

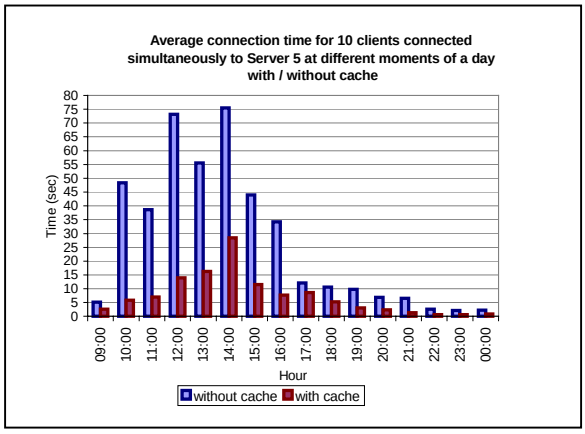

Figure 8: Multiple connection with and without "cache structure" (10 clients)

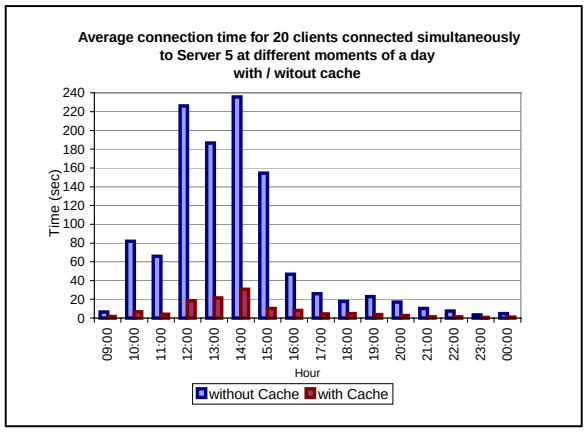

Figure 9: Multiple connection with and without "cache structure" (20 clients)

Availability of Servers

We send a short packet using "ping" mechanism to check the availability of the server. Because the ICMP packet delivery is unreliable, we have to send multiple times a packet for computing the ping time. A server is unavailable if all the trials were unsuccessful. Our decision was to send up to four such packets.

We simulate multiple clients who try to send simultaneously a "ping" to a WWW server, at different moments of the day. Shown below is the result for successful rate of the server during one day. (Figure 10). We also monitored the variation of average ping time during one day when we have 20 clients sending simultaneously a ping to the server (Figure 11).

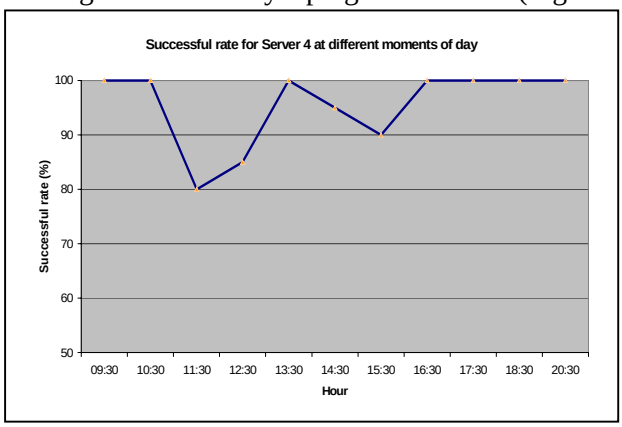

Figure: 10 Successful rate for 20 clients sending simultaneously a "ping"

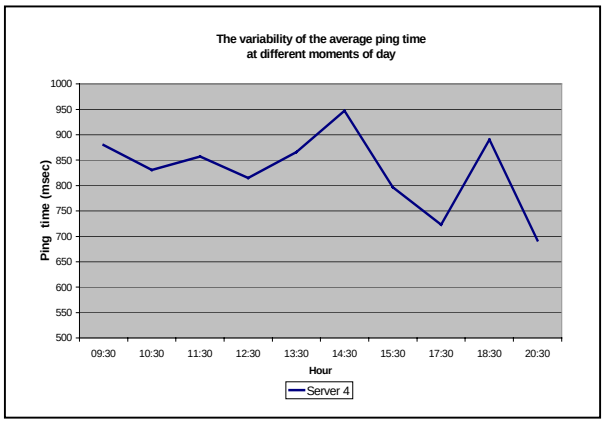

Figure 11: The variation of the average "ping" time for 20 clients

5. Conclusions and Future Work

Our work studies the variation of some of the performance indices measured on WWW pages. We mainly noticed large variations in the response times for download of the same WWW pages at similar times although some servers are more affected than others. The second result is that when requests are made in parallel with no caching, the average response time increases quite a lot. In the future we are going to investigate what is responsible for this increase. We also want to monitor the server and network performance for a longer period of time. Some extensions are to try and separate out the network, client and server delays in the total response time that is currently being measured. This might be achieved by modeling and simulation studies. We are thinking of implementing a tool to look at the links on a WWW page and to give an indication as to what the potential delays to them might be.

References

[1] Y. Nakamura K. Chinen, H. Sunahara, S.Yamaguchi and Y. Oie, "ENMA: The WWW Server Performance Measurement System via Packet Monitoring", INET'99
[2] L. Cottrell, W. Matthews and C. Logg "Tutorial on Internet Monitoring & PingER at SLAC" http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html
[3] S. Kalidindi and M. J. Zekauskas, "Surveyor: An Infrastructure for Internet Performance Measurements", INET'99
[4] Cristina Hava, Liam Murphy, "Performance measurement of the WWW Servers", 16th IEE UK Teletraffic Symposium, Harlow 2000
[5] Kris A. Jamsa, Suleiman Lalani, Steve Weakley "Web Programming", Jamsa Press, 1996
[6] Internet RFC/STD/FYI/BCP Archives http://www.faqs.org/rfcs/