PERFORMANCE MEASUREMENT OF WORLD WIDE WEB SERVERS

Cristina Hava & Liam Murphy[1]

Abstract

The World Wide Web (WWW, or Web) is one of the most important Internet services, and has been largely responsible for the increasing popularity of the Internet in recent years. The WWW continues to expand in both size and content, and 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. In this paper we outline a monitoring application for performance measurement of WWW servers. By observing the behaviour of WWW servers and measuring their performance, we can estimate several performance indices. Our monitoring application simulates the access of a number of clients to the same Web page and its links. Our simulator allows the user to choose different scenarios based on different ways clients can access a Web page, and compare the resulting times.

1. Introduction

Up to the early 1980s most companies kept their data on powerful mainframe computers, and their employees accessed this data through terminals connected to the mainframe. Computer networks became popular only when networks of workstations or personal computers offered a big price/performance advantage over the mainframe-with-remote-terminals model. In the 1990s computer networks are dominant, delivering services to home users as well as workers in companies, universities and government institutions. Such networks allow access to information in many forms. For example, users can access financial institutions in order to pay their bill, manage their bank accounts, and handle their investments electronically. A very important application which offers access to information on the Internet is the World Wide Web (WWW, or Web). The WWW is largely responsible for the increasing popularity of the Internet among the general public in recent years, and continues to grow: last year, over 10 million Web sites were created, compared to only 200 Web sites in 1993 [1].

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”. Common performance measures include:

- how easy it is to find the desired information;
- how current the information is;
- quality of retrieved images, sound, video, and multimedia information;
- low connection establishment and download delays;
- high availability (i.e. the Web server being accessed doesn't "hang" or "crash").

Administrators of WWW sites have to manage their servers in order to meet these service requirements; or more accurately, to ensure that their servers are not the problem if users are not receiving acceptable service[2].

Typically this could involve tracking

- changes in performance indices such as the minimum/average/maximum number of HTTP requests;

[1] Performance Engineering Laboratory **http://www.eeng.dcu.ie/~pel**
School of Electronic Engineering, Dublin City University, Dublin 9, Ireland
havac@eeng.dcu.ie , **murphyl@eeng.dcu.ie**

[2] The service may be unacceptable due to limitations of the user's equipment or of the intervening network infrastructure, rather than the server being accessed (of course, the user's perception of the cause of the problem may not reflect this).

- the volume of total traffic accessing their site's services;
- the average response time as it varies across hours, days, weeks, etc;
- the service availability (e.g. in terms of blocked or refused access requests);
- 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. In particular, the volume and pattern of client accesses has a big impact on the server's performance. The goal of our work is to simulate the access of a number of clients to the same Web page and its links, and to observe how these influence various performance indices.

Related work

A number of related efforts concerning WWW server performance and Internet performance tests and measurements are being developed concurrently with our research area.

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. The method has been implemented as ENMA system [2]. ENMA can measure performance indices such as connection setup time, connection continuation time, and the number of concurrent connections.

The PingER project [3] 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 [4]. 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.

2. Types of Accesses

Our simulator allows the user to choose different scenarios based on different ways clients can access a Web page, and compare the resulting times. Two possibilities are supported:

1) The clients access the same Web page one after the other (i.e. *serially*). The simulator creates a thread for each client. A thread is killed after finishing its task, and then the following one is created. The advantage of this approach is that both the server and the network are not overloaded. The main disadvantage is that there is only one client connected at a time.

2) The clients access the same Web page at the same time (i.e. *in parallel*). The simulator creates all the client threads simultaneously. These threads run in parallel and are only killed when they finish their jobs. In this way it is possible to see the variation in responses to similar requests from the same source at the same instance in time. An advantage of this approach is that the client waiting times are expected to be shorter than for the serial case. The disadvantage is possible overloading of both the network and the server (and possibly the client as well).

A thread implements a client. A thread consists of modules that are in charge of tasks. The first task is to fetch a Web page. It has to establish a connection to the appropriate Web server through a socket. 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. Usually each protocol has a standard port number. For example, for HTTP protocol the standard port number is 80, and for FTP protocol it is 21.

3. Ping Implementation

"Ping" is one of the most useful IP network debugging tools available. To get information about the destination, ping sends a short data burst – a single IP packet – and listens for a single packet in reply. Ping is implemented using the required ICMP Echo function, documented in RFC792 [5].

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 function, which sends a packet on a round-trip between two hosts), and announcing timeouts (when IP packet's TTL field drops to zero). Each ICMP message is encapsulated in an IP packet. ICMP packet delivery is unreliable so the host cannot count on receiving ICMP packets for every network problem. There are about a dozen types of ICMP messages [5]. The ECHO REQUEST and ECHO REPLY messages are used to inform the sender that a given destination is reachable and alive. Their packet structures are documented in RFC792 and RFC791 [5].

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

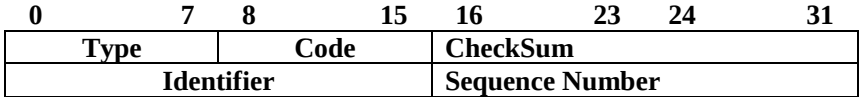

Figure 1. ECHO REQUEST packet's structure

4. Preliminary Results

To begin we choose six different servers to test. 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 and Parallel Transaction Processing

When the user chooses serial connections from the connection types offered by our application, a single thread is created. It fetches the main page and ensures that all the parts of the page are downloaded and then computes the time elapsed. It does some further processing before the user can start another test for a connection. For a set of serial connections the average time for five sequential accesses to the same Web is measured every hour.

When we choose parallel connections a number of threads are created simultaneously. These all try to fetch the same Web page from the server. 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 to the client may also vary. This results in different

times for the all the threads to fetch the same Web page. Furthermore the server could be overloaded or the network congested and this will be apparent in the computed response time.

Shown below are the results for serial connections, Figure 3, 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.

Observing Figures 3, 4 and 5 the connection time increases with the number of parallel clients trying to connect to the server. An example of this is shown in Figure 6, where a comparison of the response times of three of the servers for serial and parallel connections at 11am during the day. It can be seen that there is a noticeable increase in the response time for 20 clients in parallel for these servers.

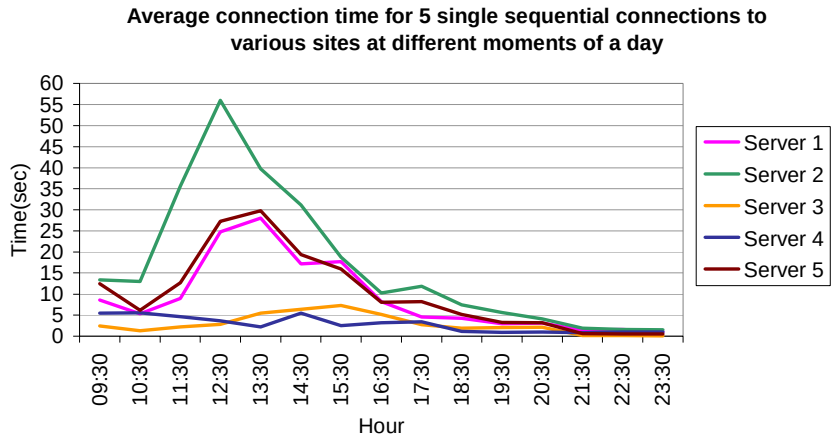

Figure 3. Serial Connections

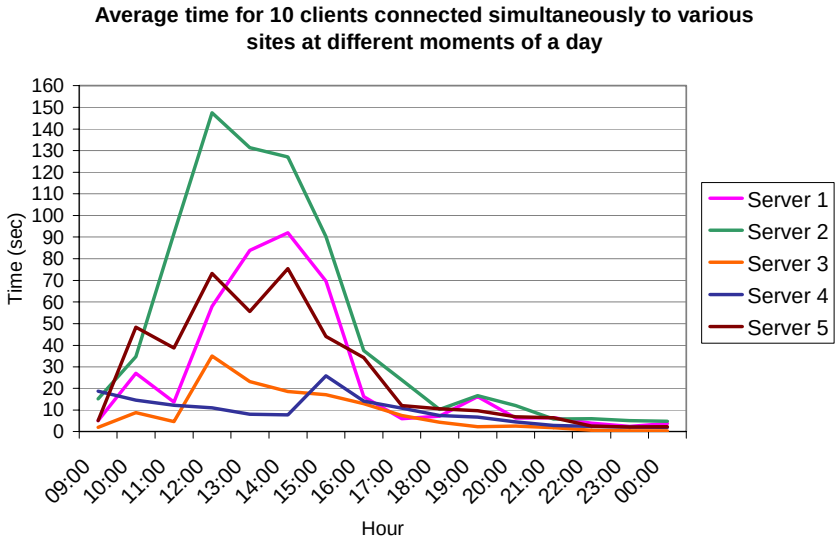

Figure 4. Ten Parallel Connections

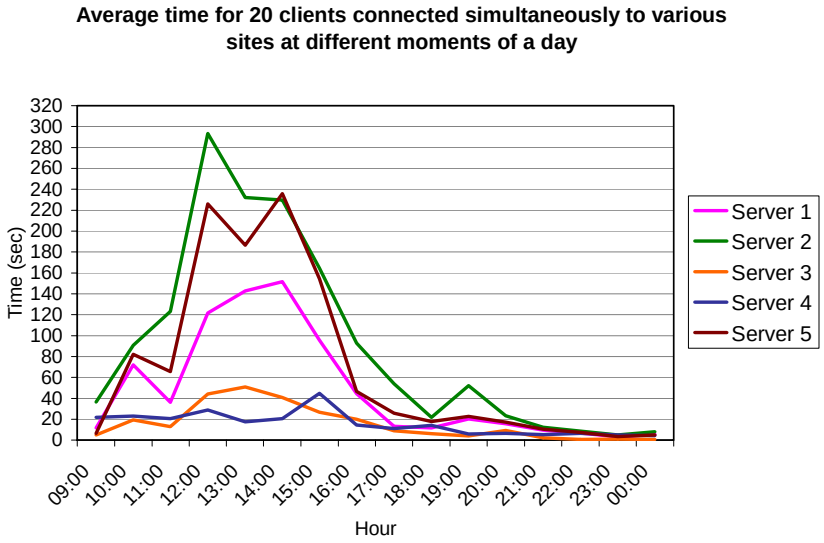

Figure 5. Twenty Parallel Connections

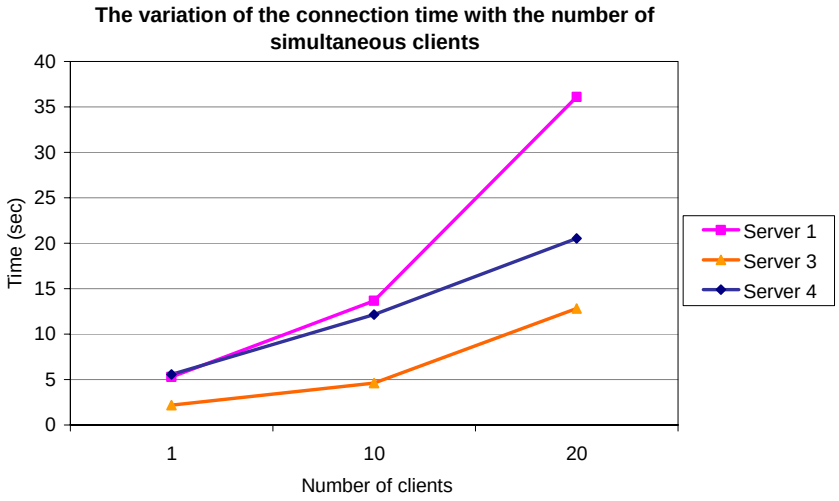

Figure 6. Response Time Variation - Serial & Parallel

Usually a WWW server can process several HTTP connections in parallel, but sometimes the number of concurrent connections is limited and the server could be saturated. In this case, the server rejects several accesses randomly and the data is slow to be delivered to the clients. Apart from the average values that we are observing in the above diagrams it is also possible to see that there is a large variation in the access times to the same page at the same time. In Figure 7 the minimum, average and maximum response time for getting a Web page from a single site is plotted for different times. 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 is about 90 seconds. All these tests were conducted during a normal working day.

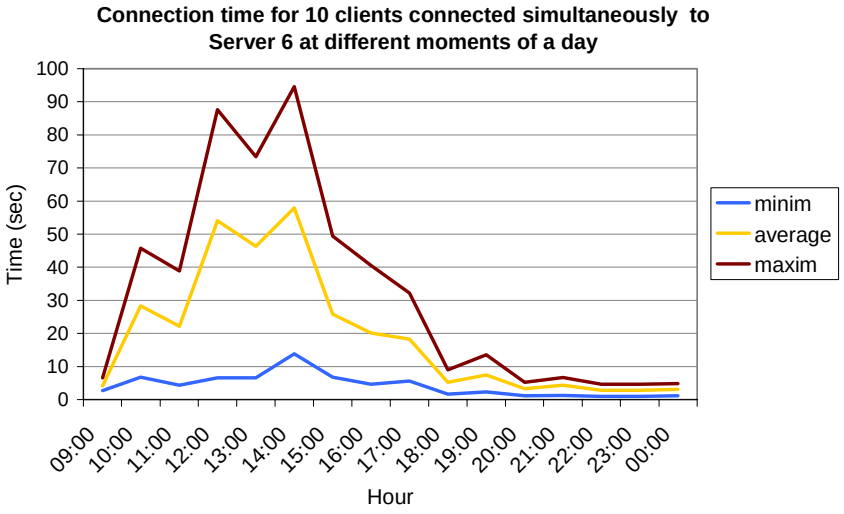

Figure 7. Minimum, Average & Maximum Response Times for 10 Parallel Clients

5. Conclusions and Future Work

This paper explores a first attempt to perform performance tests and measurements on WWW pages. The methodology used is simple and the results fairly straightforward. What has been achieved is to automatically download WWW pages and measure the response time with no caching taking place on the client machine or within the University networks. The results are mainly that there are large variations in the response times for downloads 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. What is driving this increased response time has yet to be investigated.

The future work for this project is to make an application that can run in the background and that can monitor the server and network performance for a longer time period. 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 modelling and simulation studies. It is also planned that a tool to look at the links on a WWW page, to give an indication as to what the potential delays to them might be, will be developed.

References

1. Hobbies' Internet Timeline v.5.0. http://www.isoc.org/guest/zakon/Internet/History/HIT.html
2. Y. Nakamura K. Chinen, H. Sunahara, S.Yamaguchi and Y. Oie, “ENMA: The WWW Server Performance Measurement System via Packet Monitoring”, INET’99
3. 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
4. S. Kalidindi and M. J. Zekauskas, “Surveyor: An Infrastructure for Internet Performance Measurements”, INET’99
5. Internet RFC/STD/FYI/BCP Archives http://www.faqs.org/rfcs/