Tunghai Science Vol. 5: 21−41
July, 2003

Performance Monitoring Web Applications via a Mobile-Agent Approach

Min-Hsiang Chuang Wen-Kui Chang[*]

Abstract

Performance testing can reveal the bottleneck and the peak performance of a website, and performance monitoring can make sure that the system is in its best condition. These two processes are becoming more and more important in order to ensure the quality of websites. However, a huge amount of bandwidth is needed to completely evaluate or even monitor a certain website as the process involves continuous exchange of data packages through the Internet. However, we are facing limited bandwidth as the World Wide Web (WWW) gets more and more popular and common, therefore we introduce the mobile agent technology as a solution.

In this research, we investigate the applicability of mobile agent technology to Internet applications, and employ the kernel properties of a mobile agent to propose a performance-monitoring framework in the web environment. The suggested approach holds several advantages such as reducing bandwidth and enhancing robustness of the executing platform. We present with two properties and validate their correctness using one demonstration example. This example is also used to verify the correctness of our framework.

Keywords: Software quality assurance, Software validation, Performance evaluation, World Wide Web (WWW), Mobile agent

1. Introduction

The World Wide Web (WWW) has become more and more popular and common around the world. It is even more common than television sets in some areas. Therefore, it has a very wide range and deep reach in human society. Furthermore, with so many web editors available with little cost or even free of charge, making websites is getting easier as time goes by. Not only do people have their own homepages describing their own interests, but more and more

[*] Department of Computer Sciences & Information Engineering, Tunghai University, Taichung 407, TAIWAN

companies are also building their web-based business. However, not all companies are successful in doing so. There are many factors contributing to this, and website quality is the major factor. A website without quality will have fewer people visiting it and will not survive under such a competitive environment.

There are many issues related to website quality, and the performance of the web server is one of the major issues. A server that has a low performance will not be able to survive the harsh condition of the Internet. Performance testing will be able to ensure that the website is properly tuned to suit its situation, and long term monitoring will allow us to identify the problems and fix them in time before the problems impact the websites. Therefore, in this research we first study the issues related to performance testing and performance monitoring of websites. After the detailed studies on performance issues, we propose a framework that not only carries out the performance testing and monitors the sites, but also takes care of the problems related to limited bandwidth we constantly face today.

2. Issues related to performance monitoring

2.1 Web environment characteristics

WWW stands for "World Wide Web". To access the web, you run a browser program. The browser reads documents, and can fetch documents from other sources. Information providers set up hypermedia servers from which browsers can get documents.

The browsers can, in addition, access files by FTP, NNTP (the Internet news protocol), gopher and an ever-increasing range of other methods. On top of these, if the server has search capabilities, the browsers will permit searches of documents and databases.

The documents that the browsers display are hypertext documents. Hypertext is text with pointers to other text. The browsers let you deal with the pointers in a transparent way by selecting the pointer, and you are presented with the text that is pointed to.

Hypermedia is a superset of hypertext. It is any medium with pointers to other media. This means that browsers might not display a text file, but might display images or sound or animations.

2.2 Performance monitoring

2.2.1 Objectives

For the Webmaster, it is hard to know how their websites are performing. Just because a site is fast and responsive from your desk, that doesn't mean it is like that from around the world.

2.2.2 Monitoring types

There are two major types of monitoring [1,3,8,10]. The first is what we call 'thin' monitoring. Thin monitoring monitors and captures metrics outside the firewall. Thin monitoring is primary used for the measuring the user experience. The second type of monitoring is what we call 'deep' monitoring. Deep monitoring monitors and captures metrics from inside the firewall. It is good for actually pinpointing the root of problems that effect thin monitoring metrics.

Thin monitoring

Thin monitoring monitors a website from the end-user perspective, including website URLs, hardware components, and sub-components. The tools imitate the user experience by checking the website at defined intervals, and alert the user when potential problems occur. The monitored items include [8] response time, the ISP lookup and connection times, the server hardware timing to get the website from the disk drive, the consumer 'shopping experience': how long it takes a customer to complete a standard set of purchases, round time, throughput, etc.

Thin monitoring can be further classified into the following monitoring areas.

1. Device monitoring monitors virtually any Internet device, including secure Web servers, mail servers, domain name servers, ports, Addresses and CGIs.
2. Website monitoring monitors any website to assure the site is up and running. Monitoring for site redirection is done by parsing for a specific key word or error message and checking of total bytes transferred.
3. Any TCP/IP port can be monitored to insure connections. This monitoring can be used for Email, FTP, Oracle, News Servers, Telnet, Gophers, DNS, IRC, etc. This is port monitoring.
4. Problem Alerts - Alerts can be set based on expected results. An alert can be sent via Email or messages to Pagers.
5. The customer can log on with a personal account and monitor desktop from a standard browser and see a complete history of all their active devices in real-time. This

information can be freely printed and copied into other applications for presentation. A Ping tracer can also execute from any of our locations. The service includes a choice of five Base report templates.

Deep monitoring

Deep monitoring monitors the health of the on-line system components deep inside the firewall. Deep monitoring goes beyond thin monitoring which alerts you that a problem exists, pinpoints the cause and provides a complete development environment for scripts to take action. Its monitoring capabilities expand to include Applications, Transactions, Operating Systems (NT, UNIX), Databases, and SQL Scripts. Agents are also available for PeopleSoft and SAP. Its open architecture allows you to even build interfaces to custom applications. The metrics collected during deep monitoring include [8]: total response time, client time, network time, server time, browsing time, page loading time, connection time, downloading time, etc.

When performing deep monitoring, we traverse the website, making sure all aspects of the system are performing within the defined tolerance parameters. Additionally, the monitoring tools are designed to identify and respond to situations that could adversely affect the website. With deep monitoring scripts can be set up to perform preventive actions long before potential system failure. Scripts to take actions can be set based on a defined corporate escalation scale. For example one script can automatically generate a purchase requisition when a drive’s capacity reaches 60% while another can delete files unused for over 90 days.

2.2.3 Monitoring metrics

Throughput

In computer technology [7,13], throughput is the amount of work that a computer can do in a given time period. Historically, throughput has been a measure of the comparative effectiveness of large commercial computers that run many programs concurrently. An early throughput measure was the number of batch jobs completed in a day. More recent measures assume a more complicated mixture of work or focus on some particular aspect of computer operation. While "cost per million instructions per second (MIPS)" provides a basis for comparing the cost of raw computing over time or by manufacturer, throughput theoretically tells you how much useful work the MIPS are producing.

Another measure of computer productivity is performance, the speed with which one or a set of batch programs run with a certain workload or how many interactive user requests are being handled with what responsiveness. The amount of time between entering a single interactive user request and receiving the application's response is known as response time. A benchmark can be used to measure throughput.

Reliability

Reliability [7,13] has to do with the quality of measurement. In its everyday sense, reliability is the "consistency" or "repeatability" of your measures. Before we can define reliability precisely we have to lay the groundwork. First, we have to learn about the foundation of reliability, the true score theory of measurement. Along with that, we need to understand the different types of measurement error because errors in measures play a key role in degrading reliability. With this foundation, we can consider the basic theory of reliability, including a precise definition of reliability. We will find that we cannot calculate reliability - we can only estimate it. Because of this, there are a variety of different types of reliability, each of which has multiple ways to estimate reliability. In the end, it is important to integrate the idea of reliability with the other major criteria for the quality of measurement - validity - and develop an understanding of the relationships between reliability and validity in measurement.

2.3 Related tool

Several websites offer the monitoring of web performance, and without loss of generality, we take one website as the example.

The website is simply called Web Performance Monitoring [15]. From the name of the site and the organization, we can see clearly how professional they are in the area of performance monitoring. They not only offer performance monitoring, but also offer detailed description of the underlying components regarding performance monitoring. On their website we are able to perform simple performance monitoring of our own website. They also offer a tool, WebPerf, a system for measuring response time of specified URLs from multiple locations on the Internet.

3. Description of mobile agent technology

3.1 Mobile agent descriptions

In essence, a software agent is a computational entity that acts on behalf of others. It is also autonomous, proactive, and reactive, and exhibits capabilities to learn, cooperate, and move [2,6,9,11].

Furthermore, considering a mobile agent as a moving software agent, it generally consists of the following components as illustrated in Fig. 1.

Itinerary – This is the place where the route of the agent is recorded. Not only does it record the route, it also records the current position, so that we know exactly where the agent is. This is an essential part of a mobile agent; since the agent is moving, it needs to know where to start and where to go next and ultimately, where to end. We also have to know its current position so that we are able to trace it and make sure it will not get lost.

Code – This is the part where fragments of the program code are stored. By definition, an agent is a computational entity, which also means it is a program that can be executed. Without this code part, an agent will not be able to perform any tasks, not even traveling. Moreover, this part controls all other parts of the agent.

State – Status of the agent is recorded here. In our case it means the status of the tested hyperlinks will be recorded here. With this part, the agent then will be able to report to both the server and client about the status of the hyperlinks.

Host – This is where the server position is stored. It is quite vital since an agent is only running on the Agent Transfer Protocol (ATP) [9]. The agent has to remember where it came from so that it will be able to return to the server after the tasks assigned are completed. If not, it will be lost in the network and perhaps jam up the network.

Other necessary details – The agent needs to show who created it, and this is the place to put that. Other information related to this agent is also stored here so that people will know what this agent does and who the owner is.

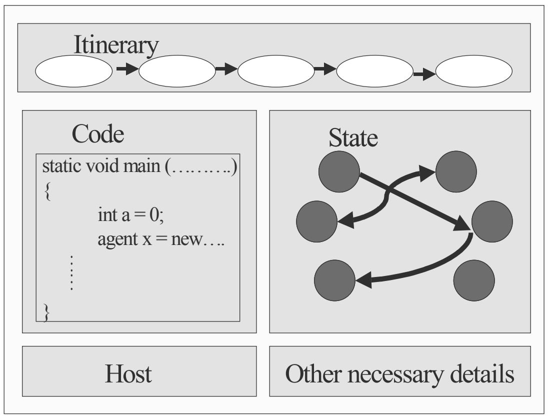

Fig. 1. The basic structure of a mobile agent

In addition, there are quite a number of different interfaces for implementing mobile agents today. The most profound ones are Concordia by Mitsubishi Electric Information Technology Center America (ITA) [6] and IBM Aglets [9]. In this research, we choose Aglets due to its availability.

3.2 Related research

Though the mobile agent is quite a new technology, it has been utilized in several areas in computer science. We will look at related research on mobile agents for network management [2,11].

The authors on mobile agents for network management [2] realized as early as 1998 that mobile agent framework is an emerging standard, and they tried to utilize it in the network management area. Basically they categorized basic network management into three different areas, namely fault management, configuration management, and performance management. They wanted to use mobile agents in these areas, so as to automate what we would normally do manually. The ultimate goal is to have a plug-and-play network that will automatically configure itself under different circumstances. This is also what recent new technologies are hoping to achieve.

Furthermore, in another paper [11], the authors realized that they could actually apply mobile agent technology in another way. They utilized the mobile agent framework in network management back in 1998. Several problems were faced during the implementation of their

JAMES project. The major problems were that the application has to be centered on the mobile agents, and a complicated interface between the agents and the application has to be written in order to make the application work. With these problems, the end users will have difficulty using the application because they not only need to configure security permissions, but also have to manage the troublesome agent platforms.

Hence, the authors developed another approach for utilizing mobile agent technology, which is called the application-centric mobile agent system (ACMAS). With this new framework, the mobile agents become the underlying mechanisms that perform messaging instead of all the necessary tasks. The end users now see only the application without worrying about the complicated configurations. Components are utilized to divide the tasks among the different agents. The advantage of the new framework is that the mobile agents now interact directly with the applications only from the inside, and thus no agent platforms are needed. In addition, users can self-configure the components they need to specialize the application to suit their needs.

4 Monitoring evaluation methodology

4.1 Monitoring framework

We first discuss the properties that the agent will possess so as to make the testing process easier. The agent should be able to do the following:

1. Simulate simultaneous access of a required number of users, e.g. 1000, 10000, or 1 million
2. Record the response time or other necessary data
3. Compile and tabulate the collected data and represent it in different forms, such as graphs, diagrams, or charts
4. Report the results to the client

The desired testing framework proceeds by the following steps as shown in Fig. 2:

1. Initialize

The target website initiates the test by accessing the web page of the agent server, with load/stress testing as the chosen option. The number of simulated simultaneous accesses can be set here, so the agent knows how many times it has to simulate to complete the test.

2. Dispatch

The server then creates an agent with the specification obtained earlier, and sends the agent out. After the agent has been sent out, it is on its own. The agent is responsible for generating the required number of simultaneous accesses automatically. During the test, the agent need not be connected to the server.

3. Test and collect data

The test is supposed to be incremental. The agent is required to automatically generate an increasing number of accesses to simulate the number of users during a short period of time. It may start at 100 simultaneous accesses and increase at an interval of 50 or even 500. It will only terminate the test when the required number of simultaneous accesses is reached. During each round of tests, the agent records necessary information for calculation at a later stage.

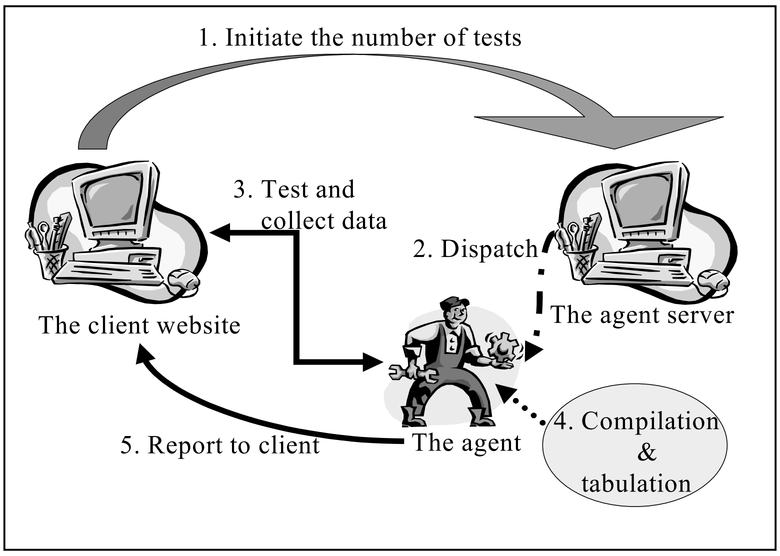

Fig. 2. The desired testing framework

4. Compile and tabulate

Once the agent has finished simulating the required number of simultaneous accesses, it gathers the information obtained earlier and compiles the raw data, which is mainly the response time during each round of the tests. Then the agent tabulates the results according to the number of simulated accesses so as to show the client the load and stress that the server was facing during each round of the tests.

5. Report

The agent reports the results and the tables in various forms to the client, so that the client can have choices of different types of displays. The agent can report to the server as well if needed. In addition, the agent can determine the performance of the server, such as the bottleneck and the peak performance of the server, based on the data obtained. It will be disposed of after a certain period of time so that it will not take up CPU resources.

In addition, the mobile agent can carry out the designed tasks for a pre-set period of time. In other words, the mobile agent can monitor the situations that the website encounters and warn the system administrator once problems occur.

4.2 Monitoring the process in detail

While the agent is named a performance monitoring agent, it not only monitors the performance of the client website, but also performs performance testing. First, we will focus on the processes for performance monitoring.

Monitoring, by definition, is a program that observes, supervises, or controls the activities of other programs. Therefore, when we monitor the performance of a website, we are observing the performance of the website. The mobile agent will obtain the settings before hand. From the settings the mobile agent can determine how long it will need to monitor the web server and how frequently it should carry out the performance testing. Once the mobile agent starts the monitoring process, it will be on its own and it will be disposed of once the monitoring is done. During the monitoring process, once the mobile agent encounters with any unusual error, it will generate a warning message and notify the system administrator of the client website. The monitoring framework can be represented as in Fig. 3.

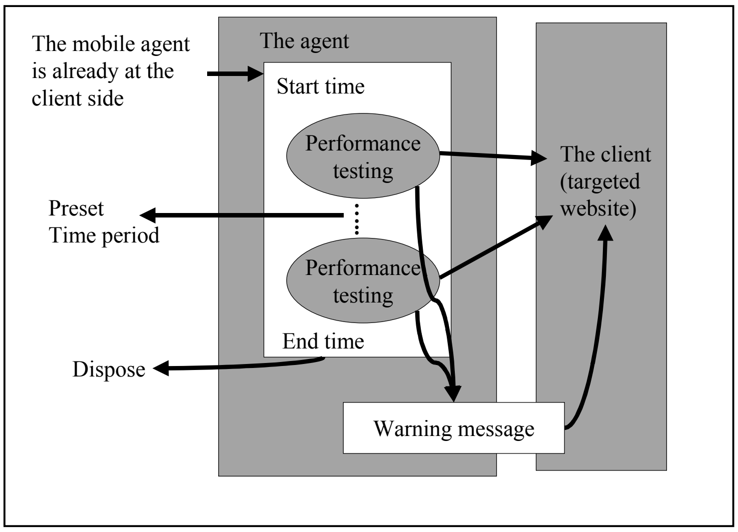

Fig. 3. The monitoring process in detail

Now we will look at the performance testing process in detail. The main objective of the agent is to simulate the number of simultaneous accesses to the website, so the agent will be required to repeat the testing process and collect the necessary data.

From this, we can deduce that the agent will have a mechanism to generate numerous simultaneous accesses. The detailed performance testing procedures are illustrated in Fig. 4. This is where we place the information that we gathered while the client accessed the website. The client will determine beforehand the number of accesses that he wants to simulate, and input that number when he accesses the website. The agent will use that number as the upper limit and start the testing process. The testing process will be incremental. It will start from the minimum and increase the number of simultaneous accesses by 50, 100, or even 500.

While the test is going on, we will need the agent to collect some information for us so that we are able to determine the load and stress of the server under different number of simulated simultaneous users, the threshold of the server, and finally the performance of the server. The kind of information we will be collecting is the response time, which is one of the monitoring metrics as discussed earlier in section 2.3 “Performance monitoring”. From the response time we will be able to verify whether the server is still able to serve more users or the server is already at the peak of its performance.

After we have collected all the response times for different numbers of simultaneous accesses, the agent will be responsible for compiling the data and tabulating them into different forms, such as bar graphs and pie charts. Then it will send the tabulated results back to the client and display the results to let the client know the load and stress that the server will face if given a certain number of simultaneous accesses. The agent will also be able to conclude from the data the performance of the server. In other words, the agent will be able to tell when the server will reach its peak or when the server will face its bottleneck. From the results the client can also learn the efficiency of the server and decide whether to control the number of users that can simultaneously connect to the site.

4.3 Advantages of using mobile agent technology

Mobile agent technology is used to solve several problems faced today. The advantages of using mobile agent technology in our research can be categorized into three categories, namely bandwidth, CPU utilization, and robustness.

Bandwidth

This is the most important advantage of mobile agent technology. Even though we have bigger and bigger bandwidth with new technologies, due to the applications we use and the requirements of the clients, we constantly face the problem of not having enough bandwidth to satisfy both the applications and the client requirements. Especially in the case of performance monitoring, we need to perform the test procedures continuously for a long period of time, and thus the consumed bandwidth is even bigger than for other applications.

By using mobile agent technology for performance monitoring, we are able to decrease the bandwidth to a minimum. The reason is that we only require a small amount of bandwidth while the agent is being compiled and sent out to the client to perform the tasks. Once the agent is on the client side, there will be no connection between the agent server and the agent, and the server will be able to accept other requests without having to sacrifice the previous requests.

Robustness

A common goal that designers of complex systems strive for is robustness. Robustness is the ability of a system to continue to operate correctly across a wide range of operational conditions, and to fail gracefully outside of that range. One of the major characteristics of a mobile agent is its robustness. Agents can work on their own without having to connect to the agent server permanently. Therefore, after we apply mobile agent technology to performance monitoring, we are able to do a better job than with the conventional approach. The reason is that, with the conventional approach, there needs to be a constant connection between the test server and the client, since testing the performance of the client involves the sending and receiving of data packets. Once the connection is lost or unstable, the test server will not be able to continue the test from where it has stopped, or even it can continue, the result may be not as accurate as desired.

With the mobile agent approach, the test can be carried out as usual without having a permanent connection, and therefore robustness will be achieved with little difficulty.

5 Demonstration example

Now we describe a demonstration example to show exactly how our framework will function. The targeted website is the homepage of our faculty – Computer science and information engineering [14]. The reason for this choice is that mobile agents need to run on a specific protocol, the Agent Transfer Protocol (ATP) [9]. In order to set up this protocol, we will need to install necessary software, namely the IBM Aglets Software Development Kit (ASDK) [9]. Therefore we need a site that will authorize us to perform such an action.

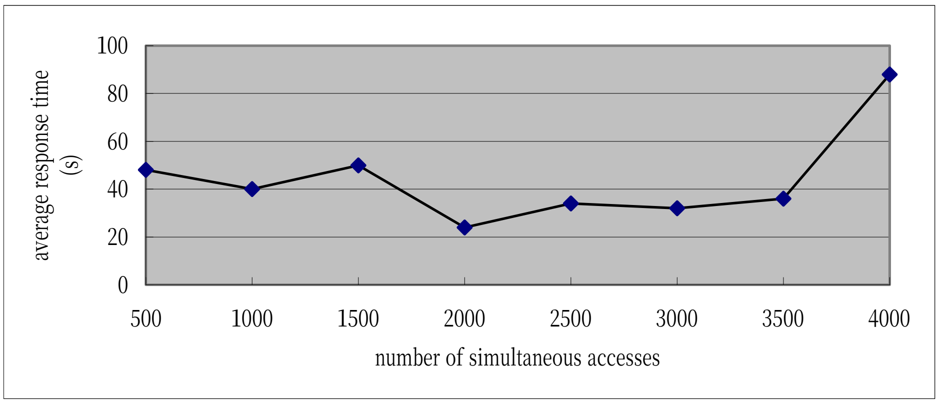

Fig. 4. The profile of the performance testing results

After ASDK has been set up and running, we send an agent out to carry out the performance monitoring. Since one advantage and also the obvious purpose of using a mobile agent for testing is the network bandwidth, we purposefully record the network status of the agent server. We notice that there is only a short period of connection while the agent is being compiled and sent out to the targeted website. Once the agent is at the client side, we notice no network connection at all. In order to ensure that the mobile agent is working as we expected, we altered our design so that the agent will return to us after one single round of performance testing. The result of the single round of testing is summarized in Fig. 4.

Fig. 4 is the kind of diagram that we expect the agent to display to the client if the client only requested a simple performance test. It shows the load of the server during different periods of the test. Also, we can see that the bottleneck of the web server is around 3500 simultaneous accesses as the average response time increases to around 90 at 4000 simultaneous accesses. We are able to determine from the figure that the peak performance of the web server is probably at 2000 simultaneous accesses as the average response time is even less than that for 500 accesses.

After one day, the agent returned with the monitoring report of our faculty website. The default number of users used to simulate the situation is 3500, and hence, according to Fig. 4, the upper limit of response time will be around 35s.

Fig. 5 shows the summary of the monitoring process by the mobile agent approach. From the results the system administrator can determine when the web server is performing at its peak and when it is performing at its poorest.

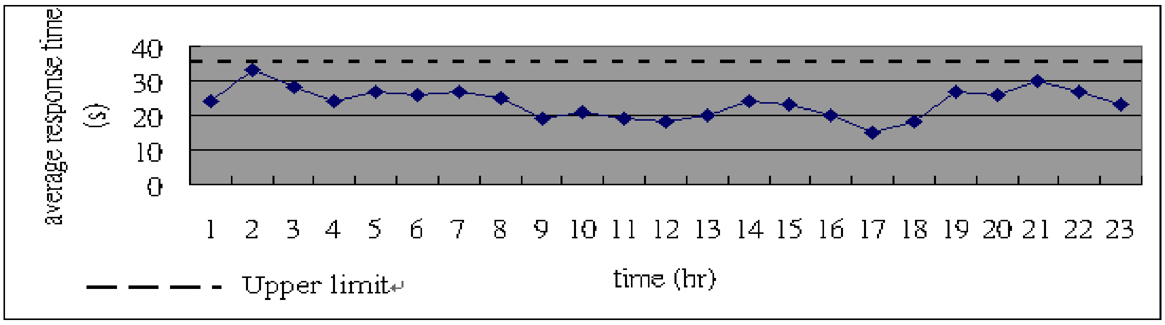

Fig. 5. The performance monitoring profile

Comparing tools

We now compare the mobile agent approach for performance monitoring and the monitoring tool provided by Web Performance Monitoring [15].

First, we compare the bandwidth that the mobile agent approach and the conventional web-based approach utilize. The mobile agent is well known for its efficiency under limited bandwidth, and therefore we derive an equation to calculate the agent efficiency by comparing different bandwidth utilizations.

First, we make an assumption:

The complexity of the application is directly proportional to the execution time and the size of code. In other words, a more complicated application will have a longer execution time and a larger code.

To simplify the derivation, we use the following notation. Let

ε = the efficiency of the mobile agent, which is the degree of enhancement of bandwidth utilization by the mobile agent,

BWconv = the bandwidth utilization of the conventional approach

BWagent = the bandwidth utilization of the mobile agent approach

Strans = the size of the conventional transaction, which mainly depends on the size of the web page, therefore we take it to be the size of the web page

Sagent = the size of the mobile agent

Tex = the execution time for the conventional approach

TtransC = the transmission time for the conventional approach

TtransA = the transmission time for the mobile agent approach

Generally speaking, bandwidth is directly proportional to the amount of data transmitted or received per unit time. In a qualitative sense, bandwidth is proportional to the complexity of the data for a given level of system performance. For example, it takes more bandwidth to download a photograph in one second than it takes to download a page of text in one second. Large sound files, computer programs, and animated videos require still more bandwidth for acceptable system performance

By the general definition of bandwidth, we can say that the bandwidth utilized by an approach equals the size of the content multiplied by the time duration. Therefore, we can derive two equations representing the bandwidth utilized by the conventional web-based and the mobile agent approach.

The bandwidth utilized by conventional web-based testing (BWconv) can be defined as follows:

$$BWconv = C1 \times Strans \times (Tex + TtransC)$$

where C1 denotes an arbitrary constant.

The bandwidth utilized by the mobile agent approach for testing (BWagent) can be defined as follows:

$$BWagent = C2 \times Sagent \times TtransA$$

where C2 denotes an arbitrary constant.

The efficiency of the mobile agent (ε) is the ratio of the bandwidth utilized by the conventional approach to that of the mobile agent approach. Hence, the relationship can be represented as:

$$\begin{aligned} \varepsilon &= BWconv / BWagent \\ &= C1 \times Strans \times (Tex + TtransC) / C2 \times Sagent \times TtransA \\ &\quad (Strans / Sagent) \times 〔 (Tex + TtransC) / TtransA 〕 \end{aligned}$$

assuming $C1 / C2 \quad 1$

Because transaction time and execution time are directly proportional to size under our assumption, therefore

$$\begin{aligned} \varepsilon &= BWconv / BWagent \\ &\quad (Strans / Sagent) \times (2Strans / Sagent) \end{aligned}$$

Therefore, after we simplify the above equation, we have:

$$\varepsilon = 2(Strans / Sagent)2 \qquad (1)$$

From equation 1, we can say that the agent efficiency equals two times the square of the ratio of the size of the web page to the size of the agent. Under most circumstances, the conventional web page is about 10 times larger than a mobile agent. Therefore,

$$\varepsilon = 2x102 = 200$$

From this result, we can say that the mobile agent is 200 times more efficient than the conventional web-based approach to testing. In other words, the mobile agent approach will only utilize 1/200 of the bandwidth required by the conventional approach to achieve the same results.

Now we look at an example to prove our theory. When we accessed the website of Web Performance Monitoring and requested monitoring our faculty website for 1 day, with a period of 10 minutes, we recorded a total of 14,061,845 packets. On the other hand, when we switched to the mobile agent approach, we only recorded a total of 68,382 packets. From these data, we can see that the tested agent efficiency ε is indeed about 200 as 14,061,845 divided by 68,382 equals 205.6 to the nearest tenth.

Next, we investigate the robustness of the two different methods. A robust system can be taken to be the same as a system having a long mean period of time between each failure. In most cases, under the web environment, we will focus on the mean period of time between connection failures. A system with a longer mean period between connection failures has greater robustness.

To simplify the derivation of the equation, we use the some of the same variables used in mobile agent efficiency ε. In addition, we define the following:

Let γ = agent robustness, which is the degree of reduction of connection failures by the mobile agent

$$\gamma = (Tex + TtransC) / TtransA \quad (2)$$

The reason that we define γ as in equation 2 is that the mean period of time between connection failures of the conventional approach greatly depends on the execution time (Tex) and the transmission time (TtransC), whereas that of the mobile agent approach mainly depends on the transmission time of the mobile agent (TtransA). The logic is that under normal circumstances, if a web page fails to load and execute within the total of execution time and transmission time, it will be considered as a failure. The same applies to the mobile agent approach. If the mobile agent fails to be transmitted within the transmission time, it is considered a failure.

After we simplify equation 2 with the same assumptions as those in equation 1, we have

$$\gamma = (Strans + Strans) / Sagent$$

$$= 2(\text{Strans}/\text{Sagent}) \qquad (3)$$

From equation 3 we can evaluate the robustness of the mobile agent approach. Since as we reported earlier, a normal web page is about 10 times the size of a mobile agent, we can say that the robustness of the application utilizing the mobile agent approach is about 20 times that of the conventional web-based application.

Now we again use an example to prove our theory. On the website of Web Performance Monitoring, we saw a number of example outputs. We selected the sample outputs of 7 different websites, namely Amazon, CNET, Geocities, Lycos, Microsoft, Sun, and Yahoo. With the 7 websites, we recorded a total number of 167 failures. Hence, the average number of failures in 1 day equals to 23.857. The mean period of time between each connection failure of the conventional approach is 1.006 hours (24 / 23.857). This result indicates that when we want to monitor any website with Web Performance Monitoring, we will have to expect an average failure rate of 1 hour in 1 day.

As stated earlier, a conventional web page is generally 10 times larger than a mobile agent. Therefore,

$\gamma = 2 \times 10 = 20$

With this, we can calculate the mean period of time between each connection failure of the mobile agent, which is 20.12 hours (1.006 x 20). With this result, we can say that the mobile agent approach can survive continuous performance monitoring for 20.12 hours before it encounters a failure. Table 1 summarizes the above comparisons.

Table 1. Comparisons of the two performance monitoring methods

Methods / Metrics	Web Performance Monitoring	Proposed mobile agent approach
Total number of packets received in 1 day	14,061,845	68,382
Mobile agent efficiency (ε)	1	205.6
Robustness	1	20
Mean period of time between each	1.006	20.12

connection failure (hr)		

6. Conclusion

Nowadays people are mostly interested in making interactive and efficient websites. Most of them have neglected the quality of the websites. Quality websites attract people, whereas websites of low quality deter potential visitors. There are many methods for improving the quality of websites, and testing the performance of the web server is one of the more efficient ways. Once we ensure that the server is performing at its peak, the chances of being low quality will be less, especially when we constantly monitor the website.

Mobile agent technology is rarely utilized for web testing. However, as we study the characteristics of mobile agents, we realize that mobile agent technology is a potential candidate in this area. A mobile agent possesses the abilities to reduce the network bandwidth greatly and achieve a higher level of robustness as compared to conventional methods. We made an assumption and derived two reasonable equations that produce satisfactory results on validating the correctness of web applications.

We wish to further enrich our knowledge on the applicability of mobile agent technology in other aspects of performance monitoring, and contribute our efforts in using mobile agents for advanced features of web testing.

References

[1] http://www.sun.com/950901/columns/adrian/column1.html, Cockcroft, Adrian, *System Performance Monitoring*.

[2] Bieszczad, Andrzej, Pagurek, Bernard and White, Tony (1998), Carleton University, *Mobile Agent for Network Management.*

[3] http://www.bmc.com/technews/004/SiteAngel.html, Whichard, Brandon and Dean, Alisa C., *Performance Monitoring of Web Applications*

[4] Wen-Kui, Chang and Min-Hsiang, Chuang (November 3, 2001), Investigation on Performance Testing for Website Quality, the 37th Annual Conference of Chin*ese* Society for Quality and the 7th National Quality Management Symposium, Taipei, Taiwan, ROC, pp477-485

[5] Wen-Kui, Chang and Min-Hsiang, Chuang (November 23-24, 2001) Evaluating website reliability by the mobile-agent approach, The Fourth ROC Symposium on Reliability and Maintainability, Taipei, Taiwan, ROC, pp193-202

[6] http://www.concordiaagents.com/, Concordia-Java Mobile Agent Technology.

[7] Daniel, M. J. (2000) *Client-Server Software Testing on the Desktop and the Web*, Prentice Hall, 2000.

[8] http://www.dotcomfidence.com/, Dotcomfidence.

[9] http://www.trl.ibm.com/aglets/index.html, IBM Aglets Software Development Kit.

[10] http://www-flash.stanford.edu/architecture/papers/SIG96/, Martonosi, Margaret, Ofelt, David, and Heinrich, Mark, *Integrating Performance Monitoring and Communication in Parallel Computers*.

[11] Paulo Marques, Paulo Simões, Luís Silva, Fernando Boavida, and João Silva, Providing Applications with Mobile Agent Technology, *IEEE Openarch 2001.*

[12] Griss, M.L. and Pour, G.(2001) Accelerating Development with Agent Components, *IEEE Computer*, **34**(5): 37-43.

[13] Splaine, S. and Jaskiel, S.P.(2001) the Web Testing Handbook, *Software Quality Engineering*, STQE Publishing.

[14] http://www.csie.thu.edu.tw, The Department of Computer Science and Information Engineering of Tunghai University.

[15] http://www.webperf.org/, Web Performance Monitoring.

[16] Chuang, Min-Hsiang, “Utilizing Mobil Agent for Web Testing to Ensure Website Quality,” (M.S. Thesis), Dept. of Computer Science & Information Engineering, Tunghai University, Taiwan, May 2002.

Acknowledgement

The authors gratefully acknowledge financial support from the National Science Council, Taiwan, Republic of China, under project NSC 90-2213-E-029-005, entitled “A mobile agent for testing distributed applications.”

利用行動代理人機制評審電腦網站之性能

莊閔翔　　張文貴*

摘　　要

基本上，網站性能測試可顯露網站的性能極限及其瓶頸所在，而性能監控則可確信網站系統是否處於最佳狀況。這兩種方法在確保網站系統的品質目的上已是越來越重要；然而，欲透過網際網路完全地評審或控制電腦網站之性能，卻需要巨量的頻寬，尤其全球資訊網越來越流行時，網路頻寬的問題變得更為棘手，因此本文將導入行動代理人的技術作為解決方案之一。

在本研究中，我們將探討行動代理人的技術在網際網路方面的適用性，並利用行動代理人的核心特性，研究提出網路環境中的監控架構；所建議的方法具有若干優點，包括：減少網路的頻寬，及提升執行平台的堅固性。本文最後並以一個實際案例的演繹說明所建議監控架構的正確性。

關鍵詞：軟體品保、軟體驗證、性能評估、全球資訊網、行動代理人。

* 東海大學資訊工程與科學研究系