Published in the 2002 Computer Measurement Group (CMG) Conference, Reno, NV, Dec. 2002.

LOAD TESTING, BENCHMARKING, AND APPLICATION PERFORMANCE MANAGEMENT FOR THE WEB

Daniel A. Menascé
Department of Computer Science and E-center of E-Business
George Mason University
Fairfax, VA 22030-4444
menasce@cs.gmu.edu

Web-based applications are becoming mission-critical for many organizations and their performance has to be closely watched. This paper discusses three important activities in this context: load testing, benchmarking, and application performance management. Best practices for each of these activities are identified. The paper also explains how basic performance results can be used to increase the efficiency of load testing procedures.

1. Introduction

Web-based applications are becoming mission-critical to most private and governmental organizations. The ever-increasing number of computers connected to the Internet and the fast growing number of Web-enabled wireless devices create incentives for companies to invest in Web-based infrastructures and the associated personnel to maintain them. By establishing a Web presence, a business has the potential of reaching out to a very large number of customers. Of course, they need to know that a site exists, the site has to earn their trust, and customers need to have a good experience when they visit the site. Customers should be able to easily and in a timely manner find what they want. A Web site should be always there when customers need it.

A critical question is what is the Return on Investment (ROI) of Web-based applications? There are several important factors to take into account in order to answer this question. First, offering services and products over the Web cuts down some costs in personnel and physical infrastructure that would be devoted to direct customer assistance, either in person or over the telephone. On the other hand, to provide online access to products and services, a company needs an adequately sized IT infrastructure that provides the services needed by its customers with the quality of service (QoS) they expect.

QoS for Web applications is usually measured in terms of response time and availability from its user's point of view and in terms of throughput from the site management standpoint. Poor QoS translates into frustrated customers, which translates into loss of business opportunity. The amount of money a business needs to spend in its IT infrastructure depends on the traffic it expects to see at its site. One needs to spend enough but no more than is required in the IT infrastructure. Besides, resources need to be spent where they will generate the most benefit. For example, one should not upgrade the Web servers if most of the delay is in the database server. So, in order to maximize the ROI, one needs to know when and how to upgrade the IT infrastructure. In other words, not spending at the right time and spending at the wrong place will reduce the cost-benefit of the investment. In this paper, we provide a framework, based on benchmarking, testing, and application performance management, which allows companies to maximize the ROI on their IT expenditures.

Section two defines in more detail the important QoS metrics to be considered. The next section presents a multi-layer reference model to analyze Web-based applications. Section four describes the main features of benchmarking, testing, and application performance testing. Section five presents some important requirements of tools used in these three activities. Section seven provides some basic performance concepts that should be taken into account when carrying out load testing. Finally, section seven presents some concluding remarks.

2. Web-Based Application QoS

Quality of Service (QoS) is a key concept in assessing how well Web-based applications deliver what customers expect in terms of response time and availability. *Availability* measures the percentage of time a Web-based application is available to customers. Typical availability goals may vary according to the type of application. Some critical applications, such as on line brokerage, may have more stringent availability requirements from the user perspective than other applications, such as

online travel sites. Availability requirements may also vary according to the time of day or according to special events. For example, during times of high market volatility, it is essential for an online brokerage site to exhibit an availability as close to 100% as possible. An online ticket sales service has to exhibit very high availability when tickets for a sporting event or for a concert go on sale. Unfortunately, these are the times, when Web sites are subject to "flash crowds" and their resources are stressed to their limits, which may cause user requests to be rejected due to lack of enough resources, thus decreasing the availability. Even in industries in which availability is not as critical from the user's perspective—e.g., e-tailers as opposed to online brokerage sites—availability is crucial to the bottom line of the business.

The availability of a Web or E-commerce site is not the same for all its customers. In fact, the geographical location of a customer may impact the availability of a Web application given that different customers may have their traffic served by different ISPs and different networks. For example, recent measurements by Keynote of Olympic.org, the site of the International Olympics Committee, showed 99.33% availability overall internationally, 98.9% for users from Asia, and 99.48% for European users (www.keynote.com).

Another key factor in determining the quality of service offered by Web and E-commerce sites is the *response time*. But, what is response time anyway? How should it be measured? What are its main components? These are important questions to consider since they affect the choice of measurement procedures and tools.

Let us start with the definition of response time. In Web-based environments, we need to measure end-to-end response time. In other words, we need to measure response time outside the firewall. This type of measurement indicates customers' perception of the time it takes a Web page to download or the time elapsed for a keyword search to return the results. When defining end-to-end response time, one must distinguish between the time it takes for the base HTML page to download and the time for the other components of the page (e.g., images, ad banners) to download.

Customers' perception of a Web application response time varies according to many different factors that are outside a Web site's environment, and that affect the customers' perception of response time. These factors include: the Web site's ISP, the customers' ISPs, the bandwidth of customers' connection to their ISPs, the networks involved in routing packets from customers to the site, and the delays imposed by third-party services used by the Web site. Examples of third-party services include Content Delivery Networks (CDNs) [Buzen2002] used to provide images and streaming media to customers or ad networks that provide ad banners to the pages served by a site. So, measuring response time from a single geographical location and at a specific time window does not provide the complete picture on a Web site's response time. End-to-end response time is time- and space-dependent. This means that one needs to be able to know how users from different locations, with different connectivity, and at different times of the day, perceive a Web site. For instance, according to recent measurements by Keynote, the site of the International Olympics Committee delivered variable response time for users in different parts of the globe, ranging from over 9.0 seconds for users in Asia to 3.97 for users in Europe, on average, from January 21 to February 11 (www.keynote.com).

Let us now zoom in into the end-to-end response time and break it down into its components [Zhi2001]. Suppose you are running a travel site and a customer submits a search for flights for a given itinerary. The result of the search is presented to the user in 4.03 seconds. What are the main components of the 4.03 seconds? Let us divide the end-to-end response time into the following components: i) DNS lookup time (i.e., time to convert your site human-readable name—say www.mytravelsite.com—into an IP address, ii) time required for the browser to establish an initial TCP connection with a server at your site, iii) redirection time, if your site, redirects requests to another site, iv) time to download the first byte of the base page requested, iv) time to download the entire base page, and v) time to download the content of the page with all its embedded objects, such as images and ad banners. Some of the components of the end-to-end response time are only network related (e.g., TCP connect time), while others involve both the network and several resources within your site (e.g., firewalls, load balancers, Web servers, application servers, database servers, and internal networks).

Table 1 provides an example of a possible breakdown of the 4.03 seconds of our example search request. Note that each file's download starts with a DNS request (which may need zero time, if that host name was previously requested), then a TCP Connection (assuming HTTP/1.0 or the use of a firewall), then the arrival of the first packet, then the arrival of the rest of the data, which may use multiple packets transmitted without waiting for an acknowledgement from the receiver.

Table 1 – End-to-End Response Time Breakdown

		Starting Time	Outside the Firewall			Inside the Firewall				Completion Time
			Network	External Server	Total Outside	Web Server	Application Server	Database Server	Total Inside	
First HTML File	DNS Lookup		0.01	0.02	0.03				0	0.03
	Initial TCP Connection		0.08		0.08	0			0	0.08
	First Packet Download		0.08		0.08	0.02	0.06	1.4	1.48	1.56
	Last Packet		0.4		0.4	0.1			0.1	0.5
	TOTAL		*0.57*	*0.02*	*0.59*	*0.12*	*0.06*	*1.4*	*1.58*	*2.17*
Image 1	DNS Lookup	2.2	0		0				0	2.2
	Initial TCP Connection	2.2	0.08		0.08	0			0	2.28
	First Packet Download	2.2	0.08		0.08	0.02			0.02	2.3
	Last Packet	2.2	0.15		0.15	0.06			0.06	2.41
	TOTAL	2.2	*0.31*	*0*	*0.31*	*0.08*	*0*	*0*	*0.08*	*2.59*
Image *n*	DNS Lookup	3.59	0		0				0	3.59
	Initial TCP Connection	3.59	0.08		0.08	0			0	3.67
	First Packet Download	3.59	0.08		0.08	0.02			0.02	3.69
	Last Packet	3.59	0.2		0.2	0.06			0.06	3.85
	TOTAL	3.59	*0.36*	*0*	*0.36*	*0.08*	*0*	*0*	*0.08*	*4.03*
External CDN Server	DNS Lookup	2.5	0.01	0.03	0.04				0	2.54
	Initial TCP Connection	2.5	0.03	0	0.03				0	2.53
	First Packet Download	2.5	0.04	0.03	0.07				0	2.57
	Last Packet	2.5	0.05	0.03	0.08				0	2.58
	TOTAL	2.5	*0.13*	*0.09*	*0.22*	*0*	*0*	*0*	*0*	*2.72*

The table shows a breakdown of the five components into outside the firewall components and inside the firewall components. The outside the firewall components include network (transmission, DNS servers, and network round-trip times), Content Delivery Network, and Ad Network. The inside the firewall components include Web server, application server, and database server. For example, the time to download the first packet, 1.56 seconds, includes 0.08 of network time, 0.02 of Web server time, 0.06 seconds of application server time, and 1.4 seconds of database server time. Note that when the first packet of the response page is sent, the site has already processed the search request, which includes accessing the database to locate the flights that satisfy the given itinerary. When the remaining elements of the page are being downloaded, a CDN is involved to supply images.

The inside the firewall components of the response time depend on the load, measured for example in requests/sec, submitted by customers to the site. If a site is experiencing a high volume of requests, the level of contention for site resources (e.g., processors, storage devices, and LANs) will be much higher than when the load is light [MA02].

This example clearly shows that one must be able to measure the components of response time inside and outside the firewall, at different load levels, from different geographical locations, and at different times.

3. A Reference Model for Web and E-commerce Applications

We use here the *reference model* for Web and E-commerce applications presented by Menascé and Almeida in [MA00]. This model is used in the subsequent sections to discuss a framework for managing the QoS of Web-based applications. We will use the example of a simplified online travel site to illustrate the concepts in the remaining sections of this paper.

3.1 Business Model

The top level of the reference model is composed of elements related to the nature of the business, its revenue goals, its partnerships with other businesses, its position within the industry. In our example, the business is an online site used to book seats on flights operated by a large number of airlines and to sell cruises to islands in the Caribbean. Our online travel site has established partnerships with other businesses in order to get access to their databases of flight schedules.

The business model specifies that revenue will come from sales commissions on booked flights and cruise packages and from the placement of advertisement from travel-related companies on the site pages. The cruise section of the site is multimedia-rich and includes streaming video, delivered by a CDN. Ad banners are delivered by an ad network.

The revenue goals for the site are a function of the number of bookings/week, number of cruise packages sold per week, and the number of ad banner impressions per week. Based on the amount of money brought to the business by each of these activities, one can compute the revenue per week, defined as *revenue throughput* [MA00]. Note that the revenue throughput depends on the number of completed transactions per unit time—the site throughput—and on the number of displayed ads per unit time. When a site has poor performance, sessions tend to be abandoned by users, and the site loses revenue. Another relevant metric at the business level is the *potential lost revenue throughput* [MA00], measured in dollars per unit time that are in your customers shopping carts but are not converted into sales.

We need to be able to determine, at the business level, the QoS goals for the site, in terms of performance and availability, and how the site should rank in terms of its competitors. There are industry benchmarks, such as the Keynote indices (e.g., Broker Trading, Business 40, Consumer 40, Government 40, Streaming, and Wireless SMS), that you can use to establish comparisons with other sites in your industry. For example, the travel site may decide that its performance and availability goals are to be relatively close to Keynote's Consumer 40 Index, or they may choose to measure competitors directly, possibly doing transactions that are similar in function to the ones they measure on their own site.

There is a direct relationship between the QoS requirements establish by an e-business and the amount of money spent in the IT infrastructure and IT personnel to deliver that kind of service. The goal should be to get the best possible performance by spending as little money as possible to provide the service. At the business level, one has to evaluate decisions regarding what to do in-house and what to outsource.

At this level level, one has to deal with issues such as marketing campaigns and changes in the business model. Our travel site may decide to launch a marketing campaign to promote the site close to Thanksgiving when many people travel. If successful, the campaign will attract a large number of visitors to the site and increase the number of transactions that have to be processed by the site per unit time. Will the site be able to handle this surge in traffic? Changes in the business model may also have an impact on the site's IT infrastructure. If the site decides to offer hotel and car reservations in addition to the previous services, the processing load on the site will increase. Again, one has to assess if this new type of load can be supported with acceptable QoS levels for both the existing and new load.

3.2 Functional Model

At this level, we have to look at the various functions offered by a site as well as the kind of support technologies (e.g., Flash, JavaScript, Active X, cookies, SSL) used to implement them. In our travel site example, the functions offered could be: Display the Home Page, Search for Flights, View Flight Reservations, Reserve a Flight, Book a Flight, Browse through Cruises, Show Cruise Video, Reserve a Cruise Package, and Book a Cruise Package.

For example, SSL is used for authentication in the functions Reserve a Flight, Reserve a Cruise Package, Book a Flight, and Book a Cruise Package. Streaming media is used in the Show Cruise Video function.

3.3 Customer Behavior Model

The next layer down in the reference model describes how customers navigate through the site and what their characteristics are. Customers of a Web or E-commerce site interact with it through *sessions*, which are sequences of requests coming from the same user during the same visit to the site. The navigational pattern of a user can be captured by a graph, called a Customer Behavior Model Graph (CBMG), as shown in Fig. 1 [MARPFM00, MA00, MAFM99]. The picture shows the possible states in which customers of our travel site can be found, along with the possible transitions between states and the probability that such transitions occur. We have only shown in Figure 1 the flight booking aspect of the travel site.

The states a customer can be found in Fig. 1 are: viewing the home page, searching for flights, retrieving previously completed reservations, making a reservation, and booking a flight. The figure also shows that a customer that completes a search has a 30% probability of submitting a new search, 6% probability of reserving a flight, 4% probability of booking a flight, and 60% probability of leaving the site after performing the search. Note that the arrows in the graph that do not go to any other state are actually going to the Exit state not shown explicitly.

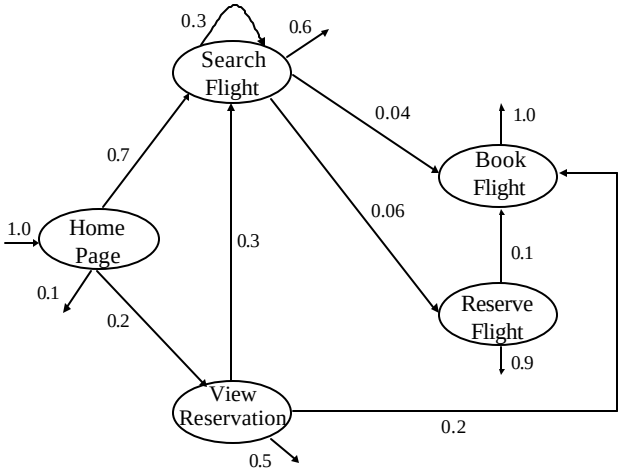

Figure 1 – Simplified Customer Behavior Model Graph (CBMG) for the Travel Site.

Not all customers behave the same way when they visit a site. So, one must consider groups or clusters of sessions that represent customers with "similar" behavior. Table 2 shows some examples of types of sessions. Each type of session can be described by a CBMG that represents a common behavior of a group of users.

Table 2 – Examples of Types of Sessions

Home ⇒ Search ⇒ Exit
Home ⇒ Search ⇒ Reserve Flight ⇒ Exit
Home ⇒ Search ⇒ Reserve Flight ⇒ Book Flight ⇒ Exit
Home ⇒ View Reservation ⇒ Book Flight ⇒ Exit

If we analyze a CBMG using the methods in [MA00], we can determine the average number of times per session that each state is visited. For example, for the CBMG of Fig. 1, we would have the following numbers: Home Page (1.00), Search Flight (1.0857), View Reservation (0.20), and Book Flight (0.0899). So, 8.99% of the visitors to our travel site end up booking a flight.

It is also important to know how long a customer takes between transitions. For example, some customers may take 15 seconds on average to go from Search Flight into Reserve Flight while others may take 60 seconds to make the same transition. This is called the *think time*. It is important to incorporate this feature into the customer behavior model because of the impact it has on the site performance. As customers submit their requests faster, i.e., with smaller think time, a site has a higher arrival rate of requests to process.

What if a customer submits a search request for a flight and the travel site takes 45 seconds to reply to the customer? Well, it is quite likely that the customer will no longer be there to see the result. The customer would have abandoned the session! How long are customers willing to wait for a page to download? The answer depends on the type of customer and on the type of page. An industry de facto standard of 8 seconds has been used as a threshold after which customers become frustrated and abandon a Web site. While customers may be less willing to wait for the result of a search request, they will tolerate longer waits when they are waiting for their selected flight to be booked.

3.4 IT Resource Level

At this level one must consider the several IT resources, including hardware (e.g., server boxes, storage boxes, LAN segments, routers, and firewalls), software (e.g., operating systems, web servers, database management systems, ERP systems, transaction processing monitors, performance monitoring tools, capacity planning tools), and personnel (e.g., programmers, system analysts, system and network administrators, DBAs, performance analysts, capacity planners, Web content developers, and graphic designers).

4. Three Key Activities in QoS Management

Managing the QoS of Web-based applications requires a combination of three types of activities: benchmarking, testing, and application performance management (APM). These three activities involve performance measurement. What distinguish them is how these measurements are taken, when they are taken, where they are taken, and how and to whom they are reported.

4.1 Benchmarking

Benchmarking is a process used to compare the performance of a hardware and/or software system, called the system under test (SUT). An SUT could be a Web-based application. The following are important components of any benchmark:

- *Workload specification:* determines the type of requests and the frequency by which these requests are submitted to the SUT. Some benchmarks, such as TPC-W (www.tpc.org) specify a controlled environment in which all the requests submitted to the SUT come from a specified workload generation process [Menascé2001, Menascé2002b]. Others, such as Keynote Performance Benchmarks (e.g., Broker Trading, Business 40, Consumer 40, Government 40, Streaming, and Wireless SMS) generate requests in a realistic environment, in which other

sources—real customers—also generate requests to the SUT. Consider for example Keynote's Business 40 (KB40) Internet Performance Benchmark. The workload of this benchmark is composed of requests to download the home page of a Web site. These requests are generated by automated agents every 15 minutes, 24 hours a day, seven days a week, and are reported at Keynote's Web site for the period of Monday to Friday between 9 am and 3 pm Eastern time.

- *Specification of metrics:* determines what to measure. Consider again the example of Keynote's KB40 benchmark. What is measured by the benchmark is the average download time of the home pages of 40 important US-based business Web Sites, spread through categories such as publishing, search engines, business services, financial services, high-technology, and communications. This benchmark is used to measure download speed for well-connected major business users.

- *Specification of the measurement procedure:* determines how the values of the metrics should be obtained. Again, in the case of the KB40 benchmark, measurements are obtained by averaging the results obtained by agents attached to key points in the Internet backbone in the 25 largest metropolitan areas of the United States. The exact location, i.e., backbone provider and geographical location, of each agent has to be specified as part of the measurement procedure.

Benchmarking is an important activity at the business level of the reference model since one needs to know how well a Web-based application is doing with respect to the competitors. One also needs to know how different types of customers perceive the response time and availability of a site. It could very well be that business users (i.e., those with high-bandwidth connectivity to the Internet) perceive a response time that is acceptable while home users (i.e., those with dial-up connections) have a very bad experience at the site. This may lead to a significant loss in revenue if home users account for a substantial share of the buying customers.

For transactional Web sites, such as ecommerce sites, one needs to benchmark complete transactions and not just Web page downloads. Since not all e-businesses are the same, one may need to create scripts that reflect critical sessions containing the site's critical transactions. This set of scripts along with a set of metrics and measurement procedure constitutes a benchmark for the site. Benchmarks need to be run on a regular basis from different Internet locations in order to know what the customer experience is.

If a Web site provides customer access through wireless devices such as Palm OS devices, RIM pagers, SMS and WAP phones, it is important to benchmark the site under the conditions experienced by customers using these wireless devices.

4.2 Performance Testing

Performance testing, also called *load testing*, is the process of running a specified set of *scripts* that emulate customer behavior at different load levels and taking measurements of a site's performance for each of the load levels.

There are several circumstances that call for load testing. Suppose a business is expecting a significant increase in the number of visitors due to a marketing campaign. So, instead of the current 3,000 session starts per hour at the peak hour, the business expects to see twice as many sessions per hour. Dial-up customers are currently experiencing an average response time of 6.5 seconds on search requests, your most critical e-business function. What will be their response time when the load on the site increases to 6,000 sessions per hour? Suppose that the site is adding new functionality or that the Web pages have been redesigned. What will be the customers' experience in terms of response time after the changes? One must be able to answer these questions before the customers so that potential performance problems are detected and fixed prior to making the Web application available to customers. You may need to do load testing if you plan to implement changes in your IT infrastructure.

Performance testing is usually carried out with controlled and parameterized workloads. A key concept in load testing is the notion of a *virtual user (VU)*. A virtual user emulates what a real user does. A load test is only valid if the behavior of virtual users has characteristics similar to those of actual users. Remember the Customer Behavior Model of section 3.3? Well, one needs to make sure that virtual users navigate through a Web site according to patterns similar to real users, with realistic think times, and that they will react as frustrated users and abandon the session in case of excessive response times.

If a load test does not allow one to mimic what happens when real users visit a site, one may get results that are totally inconsistent. For example, customers who abandon a session end up using

less site resources than those who complete a session. So, if the capacity of the infrastructure is planned assuming that all started sessions will complete, over provisioning may occur. Also, if one does not consider session abandonment, one will not be able to accurately quantify revenue throughput nor potential lost revenue throughput. These are important metrics at the business model level.

The main important parameters to vary during a load test are:

- Workload intensity, typically measured in started sessions/hour.
- Workload mix, described by the combination of scripts that determine what the typical sessions are and what customers do in each session type.
- Customer behavior parameters, such as abandonment threshold and think time.

Typical results of a load test include:

- Number of completed and abandoned sessions per hour as a function of the number of started sessions/hour.
- Revenue throughput and potential lost revenue throughput as a function of the number of started sessions/hour.
- Individual page download times and transaction completion times versus the number of started sessions/hour.

4.3 Application Performance Management (APM)

Application Performance Management (APM) is a collection of management processes used by organizations to ensure that the QoS of their e-business applications meets the business goals. An e-business application is implemented by a collection of Web pages delivered by Web sites with the support of all servers, programs, and databases that make up the site. As discussed previously, QoS goals include response time and availability. Both goals are very important and have to be considered simultaneously. Poor response time in a highly available Web site is as undesirable as a site that can respond fast but is not available close to 100% of the time.

There are two basic approaches to APM: reactive and proactive. In the former case, companies monitor their site's QoS and react when problems arise. This is called the "fire-fighting" approach. The danger of this approach is that while performance and/or availability problems are not identified, diagnosed, and corrected, a Web site will operate in a less than optimal manner. As discussed before, these conditions may pose threats to a business. The other, preferred approach, is one in which companies proactively seek to maintain the QoS of their applications through management processes that reduce occurrences of poorly performing applications.

APM involves real-time monitoring of the QoS of a Web site. As we saw in the previous section, response time and availability are influenced by factors that occur both inside and outside the firewall. This means that end-to-end measurements from multiple locations as well as detailed monitoring inside the firewall should occur.

When QoS goals are not met, APM tools should trigger alarms by sending detailed e-mail and or pagers to the proper members of the technical staff, and provide fast diagnostic capabilities. Performance and/or availability problems that take long to be addressed are bad news. The tools used for APM should allow for a clear identification of the type of problem (e.g., DNS server vs. the load balancer at the site) so that the right team can be put to work on the problem as fast as possible.

An important element in facilitating fast problem resolution is the capability to have a seamless integration between the measurements and reporting provided by a company's Enterprise Management System (EMS) and the measurements obtained outside the firewall.

The performance of a Web application depends on the performance of services provided to the company by service providers such as ISPs, Content Delivery Networks, ad networks, payment gateways, and others. A company may also be a provider of services to others. In fact, this model will become more prevalent as the notion of Web Services (see www.w3.org/TR/wsdl) becomes widely adopted. Whenever companies provide or use services from others, Service Level Agreements (SLAs) must be put in place. SLAs are documents that specify realistic performance guarantees as well as penalties for non-compliance [Overton2002].

SLA management involves three steps:

- *SLA design*: realistic SLAs are developed, agreed upon by the parties, and specified in a document that lays out thresholds, reporting and enforcement procedures, as well as penalties for non-compliance.
- *SLA measurement and reporting*: frequent (e.g., hourly, daily, monthly) measurement

and reporting on SLA compliance and computation of penalties for non-compliance.

- *SLA analysis*: longer-term re-evaluation of SLAs, which may be necessary due to changes in market conditions, changes in technology, and user behavior.

SLA management is therefore a fundamental aspect of APM.

5. Tool Requirements

We examine in this section important requirements that should be satisfied by tools that support the three activities discussed in the previous section.

5.1 Benchmarking

Business-level requirements:

- Need to be able to measure and compare your response time and availability with that of your competitors using metrics and measurement procedures that are well accepted and recognized in the industry.
- Need to be able to evaluate third-party service providers (e.g., CDNs) prior to making a contracting decision and determine whether or not these service providers meet your SLAs on an ongoing basis.

Customer behavior model level:

- Need to account for different user platforms, geographical location, connection speed, and type of client device and browser, including wired and wireless platforms.
- Need to be able to benchmark complete sessions and not just specific pages.

5.2 Testing

Business-level requirements:

- Need to be able to track revenue throughput and potential lost revenue throughput.
- Need to be able carry out load tests under the most realistic and thorough conditions possible to help avoid over- and under-provisioning.
- Need to know the impact of business decisions (e.g., marketing campaigns or new business models) on the IT infrastructure.

Functional level requirements:

- Need to be able to test functions supported by many different technologies including Flash, JavaScript, Active X, cookies, and SSL.

Customer behavior level requirements:

- Realistic modeling of customer behavior including user abandonment, tolerance to high response times for different types of pages and e-business functions, customer tenacity, customer experience with the site, and different think times.
- Need flexible and easy-to-use ways of recording scripts that represent different types of customer interactions.
- Need to be able to easily change load testing scripts to adapt to changes in customer behavior that occur over time.

IT resource level requirements:

- Need to be able to test Web applications on a regular basis in an actual production environment and not just in a scaled-down testing environment.
- Need to assess the impact of changes in system architecture, server types and their capacity, storages boxes, software, and networking bandwidth.
- Need to be able to carry out load tests on-demand and at scheduled times.

5.3 Application Performance Management

Business-level requirements:

- Need a fast way to detect the cause of performance and availability problems so that the right team can be put to work on the problem as fast as possible and the company's revenue stream is not reduced or even interrupted.
- Need a methodology for clearly designing and negotiating SLAs between your company and your providers. These SLAs must be realistic to avoid stiff penalties for non-compliance. Thus, before committing to SLAs, it is important to validate them in live environments.
- Need to review and renegotiate SLAs to keep them consistent with new conditions.

- Need reporting mechanisms on financial penalties for non-compliance of SLAs.
- Need to put in place a process to enforce and report SLA compliance that is acceptable by all parties involved. A credible third-party service is usually preferred.

Customer behavior level requirements:

- Need to identify which aspects of the customers' characteristic (e.g., their geographic location) are responsible for detected performance problems.

IT Resource level requirements:

- Need to integrate measurements obtained inside the firewall with measurements taken outside the firewall.
- Need to be able to integrate outside the firewall measurement with several EMS frameworks and provide for easy data correlation.
- Need a single and effective visualization interface with drill-down capabilities that display the components of delays inside and outside the firewall.
- Need detailed network level and page element details (e.g., DNS resolution, TCP connection, redirection, SSL handshake times, first packet response time, content download time, and content errors).
- Need measurement and reporting procedures regarding SLA compliance.

6. Fundamental Concepts in Load Testing

Load testing can be used to predict the performance of a Web site at any load level by simply increasing the number of VUs until the desired load is achieved. Running load tests for a large number of values of the number of VUs and workload mixes may be very time consuming and expensive. Faster, albeit less accurate results, can be obtained by combining load testing with performance models—analytic or simulation [Menascé2002a]. We provide in what follows some basic performance relationships that can be used to speed up scalability analysis with load testing.

Consider Figure 2, which depicts several virtual users submitting requests to a Web site, and let:

- N_{VU} : number of virtual users.
- N_C : number of concurrent requests being processed by a Web site.
- Z : average think time, in seconds.
- R : average response time of a request, in seconds.
- X_0 : average throughput, in requests/sec.

Using the Response Time Law [DB78, MA02], we get the following relationship:

$$R = \frac{N_{VU}}{X_0} - Z \,. \tag{1}$$

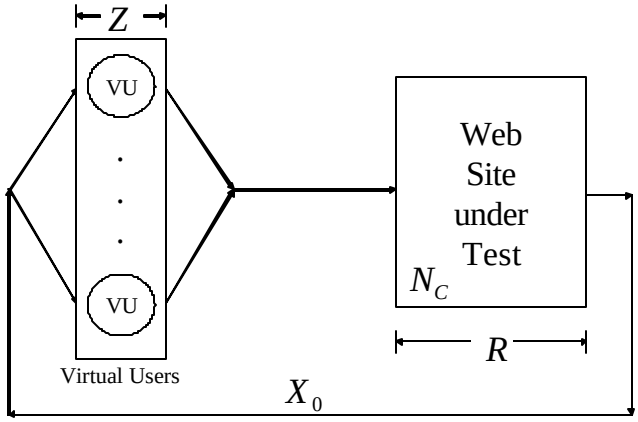

Figure 2 - A Web site receives requests from a number of Virtual Users (VU).

The throughput of a Web site is a function of the number N_C of concurrent requests in execution—the load level—and of the *service demands* of these requests at each of the resources (e.g., processors, storage devices, networks) of the site. The service demand, D_i, of a request at a resource *i* is defined as the average total time spent by the request receiving service from the resource [MA02]. This time does not include any queuing time and is therefore independent of the load level. So, we can write that

$$X_0(N_C) = f(D_1,...,D_K,N_C) \tag{2}$$

to indicate that the throughput is a function of the service demands at the *K* resources of a Web site and of the load level. The response time is also a function of the service demands and of the load level. Thus, we can write that

$$R(N_C) = g(D_1,...,D_K,N_C) \,. \tag{3}$$

So, combining equations 1-3, we get that

$$N_{VU} = [R(N_C) + Z] \times X_0(N_C) \,. \tag{4}$$

We can now use either an analytic or simulation model to predict the values of the response time and

throughput for different values of the load level N_C and use equation 4 to estimate the number of VUs needed to generate a given value of N_C.

Load testing tools can be quite useful in that regard since they can be used to generate scripts that are submitted for a low number of VUs in order to measure the service demands—remember that the service demands are load-independent. The service demands are then used as input parameters to performance models as explained in [MA02]

We illustrate these concepts through an example. Consider a Web site in which the service demands for processing and for I/O are 8 and 9 milliseconds, respectively. Using the Mean Value Analysis method [MA02] one can compute $X_0(N_C)$ and $R(N_C)$ for various values of N_C. Figure 3 shows the resulting curves of $R(N_C)$ and $X_0(N_C)$ versus N_C. The figure also shows that for N_C = 19, the number of VUs, computed using equation 4 and assuming an average think time of 8 seconds, would be equal to 897.

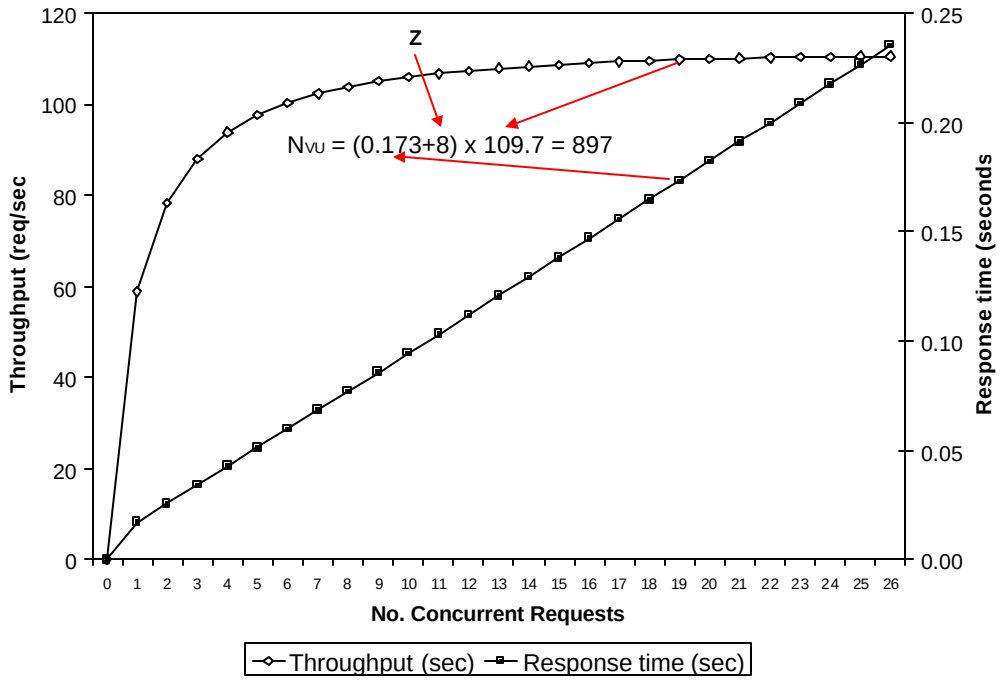

Figure 3 – Throughput and Response Time vs. No. of Concurrent Users.

The maximum value of the site throughput can also be estimated from the service demands by using the upper bound equation [MA02]:

$$X_0(N_C) \leq 1/\max\{D_i\}. \qquad (5)$$

In our example, the maximum throughput would then be 1 / max (0.008, 0.009) = 1 / 0.009 = 111.1 requests/sec, which is the horizontal asymptote of the throughput curve in Figure 3.

7. Concluding Remarks

The QoS of Web-based applications is measured in terms of end-to-end response time and availability, as perceived from many different geographical regions. The availability requirements may also vary according to the time of day or according to special events. One must be able to measure the components of response time inside and outside the firewall, at different load levels, from different geographical locations, and at different times.

When analyzing Web and E-commerce applications, it is useful to consider a four-level reference model composed of a business level, functional level, customer behavior model level, and IT resource level. The business level is concerned with the nature of the business, its revenue goals, its partnerships with other businesses, and its position within the industry. It is important to be able to assess the revenue throughput and potential lost revenue throughput. Decisions at the business level are aimed at obtaining the best possible QoS with the least possible amount of IT capital and personnel expenditures.

There are three key activities required to manage the QoS of your Web-based applications:

benchmarking, performance testing, and application performance management. In this paper we defined these activities and specified requirements for tools used to support them. Best practices in carrying out these three activities dictate that i) benchmarking has to be carried out on a regular basis from different Internet locations in order to know what your customer experience is, ii) complete customer sessions and not just page downloads must be benchmarked, iii) performance tests are only valid if the behavior of virtual users has characteristics similar to those of actual users, otherwise the results of the test may be totally inconsistent, iv) load testing must be carried out on a regular basis during the lifetime of Web application, v) application performance measurement must provide a seamless integration of measurements outside the firewall with Enterprise Management System measurements.

Running load tests for a large number of values of the number of VUs and workload mixes may be very time consuming and expensive. Faster, albeit less accurate results, can be obtained by combining load testing with performance models—analytic or simulation—as illustrated in this paper.

Acknowledgements

The author would like to thank Eric Siegel of Keynote Systems for providing the data used in Table 1.

References

[Buzen2002] J. P. Buzen, "Factors Shapping the Performance of Content Delivery Networks," *J. of Computer Resource Management*, CMG, Spring 2002, pp. 24—31.

[DB78] Denning, P. J. and J. P. Buzen, "The Operational Analysis of Queuing Network Models," Computing Surveys, vol. 10, no. 3, Sept. 1978, pp. 225-261.

[Menascé2002a] D. A. Menascé, "Load Testing of Web Sites," *IEEE Internet Computing*, July/August 2002.

[Menascé2002b] D. A. Menascé, "TPC-W: a Benchmark for E-commerce," *IEEE Internet Computing*, May/June 2002, pp. 83—87.

[MA02] D. A. Menascé and V. A. F. Almeida, *Capacity Planning for Web Services: metrics, models, and methods*, Prentice Hall, Upper Saddle River, NJ, 2002.

[Menascé2001] D. A. Menascé, "Testing E-commerce Scalability with TPC-W," Proc. 2001 Computer Measurement Group Conference, Anaheim, CA, Dec. 2-7, 2001.

[MA00] D. A. Menascé and V. A. F. Almeida, *Scaling for E-Business: technologies, models, performance, and capacity planning*, Prentice Hall, Upper Saddle River, NJ, 2000.

[MARPFM00] D. A. Menascé, V. Almeida, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. Meira Jr, "In Search of Invariants for E-Business Workloads," *Proc. Second ACM Conference on Electronic Commerce,* Minneapolis, MN, October 17-20, 2000

[MAFM99] D. A. Menascé, V. Almeida, R. Fonseca, and M. A. Mendes, "A Methodology for Workload Characterization of E-commerce Sites," *Proc. First ACM Conference on Electronic Commerce*, Denver, CO, November 3-5, 1999.

[Overton2002] C. Overton, "On the Theory and Practice of Internet SLAs," *J. of Computer Resource Management*, CMG, Spring 2002, pp. 32—45.

[Zhi2001] J. Zhi, "Web Page Design and Download Time," *J. of Computer Resource Management*, CMG, Spring 2001, pp. 40--55.