# Web Page Response Time 101

Understanding and measuring performance test results

*by Alberto Savoia*

**Human beings don't like to wait. We don't like waiting in line at a store, we don't like waiting for our food at a restaurant, and we definitely don't like waiting for Web pages to load. Have you ever abandoned a Web site simply because it was too slow? I am sure you have. The consequences of this abandonment behavior can be very serious: In a recent study by Zona Research, it was estimated that each year Web sites lose several billions of dollars due to what they call "unacceptable download speed and resulting user bailout behavior." Since the impact of poor performance can be so devastating, Web site quality assurance efforts have to give performance testing the same level of priority and attention they give to functionality testing.**

**QUICK LOOK**

- 4 laws of Web site performance
- Examining the factors behind response times

But testing the performance of a Web site can be quite difficult, because the main unit of measurement—page response time—is affected by a lot of different and interacting factors. In this article we'll look at the basic concepts and tools necessary for measuring and analyzing Web page response time. As this article's title suggests, this is an introductory article on a very complex subject, and I've simplified some formulas and explanations; but I'm quite sure that after reading it you'll know more about Web page response time than most people in your organization, and you will never look at a Web page the same way again.

Typically the most limited bandwidth is between the end user and its ISP. Since today a large percentage of people are still connecting to the Internet using 56.6K (or even 28.8K) modems, a Web site design that doesn't take this into account is bound to lose a lot of visitors. Keep in mind that modem speeds are typically expressed in bauds, and that the user's modem baud rating will seldom match the *actual* bits per seconds (bps) that their modem will typically achieve (for a number of reasons). Your best bet, then, is to convert baud rates to the Kbytes per second that users are most likely to achieve in their environment. For practical purposes, assume that 56K modems will achieve an average transfer rate of 4Kbytes/sec and, knowing that many users will access your Web site at that speed (this is particularly true for consumer-oriented sites), use that as a minimum bandwidth number.

Once you have the page size and the minimum bandwidth, you can calculate the *raw download time* (see Figure 2).

Using this formula, we see that it will take approximately ten seconds for all the bytes in a 40Kbyte page to travel over a 56.6K-modem connection.

Unfortunately, the raw download time is not what the end user will experience—because, as we shall see, it takes several distinct cycles to send the entire content of a page, and each cycle is subject to unavoidable Internet delays. Nevertheless, the raw download time is a very useful metric because it sets a lower limit for page response time based on an end-user connection speed. If you have a 120Kbyte page, 56.6K modem users will seldom download it in less than thirty seconds, and most 28.8K modem users will have to wait at least a minute. Ouch!

$$DT_{RAW} = \frac{\text{Page Size in Kbytes}}{\text{Minimum Bandwidth in Kbytes/sec}}$$

$$DT_{RAW} = \frac{40\ \text{Kbytes}}{4\ \text{Kbytes/sec}} = 10\text{sec}$$

**FIGURE 2** Calculating raw download time

## Round-Trip Time

In the context of Web page response time, round-trip time (RTT) indicates the latency, or time lag, between the sending of a request from the user's browser to the Web server and the receipt of the first few bytes of data from the Web server to the user's computer. RTT is important because every request/response pair (even for a trivially small file) has to pay this minimum performance penalty. As we shall see in the next section, the typical Web page requires several request/response cycles.

The simplest way to measure RTT is by using "ping," a standard troubleshooting tool available on most network operating systems. On Microsoft Windows, for example, you can access ping by opening a DOS command window and simply typing **ping** followed by either the hostname or the IP address of the server. (Figure 3 shows what I would type to check my company's site.)

The first thing you notice is that before it could ping the Web site, the hostname www.keynote.com had to be resolved into its IP address (206.79.179.108) by the Domain Name Service (DNS). After resolving the IP address, ping sends it four small packets of thirty-two bytes each and measures the time it takes for the server to respond. Finally, it provides you

```
c> ping www.keynote.com
Pinging www.keynote.com [206.79.179.108] with 32 bytes of data

Reply from 206.79.179.108: bytes=32 time=103ms TTL=247
Reply from 206.79.179.108: bytes=32 time=110ms TTL=247
Reply from 206.79.179.108: bytes=32 time=105ms TTL=247
Reply from 206.79.179.108: bytes=32 time=104ms TTL=247

Ping statistics for 206.79.179.108:
        Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round-trip time in milliseconds:
        Minimum = 103ms, Maximum = 110ms, Average = 105ms
```

**FIGURE 3** Measuring round-trip time using "ping"

| URL | TOTAL SIZE OF THE PAGE (BYTES) | SIZE OF THE DISPLAYED TEXT (BYTES) | SIZE OF THE HTML TAGS (BYTES) | NUMBER OF DIFFERENT IMAGES CONTAINED | SIZE OF THE BIGGEST IMAGE (BYTES) | SIZE OF ALL IMAGES (BYTES) |
|---|---|---|---|---|---|---|
| http://www. newyorktimes.com | 57768 | 10461 | 47307 | 39 | 17978 | 115656 |
| http://www. morocco-today.com | 53748 | 17628 | 36120 | 12 | 43787 | 141898 |
| http://www. kansascitystar.com | 33645 | 7271 | 26374 | 30 | 17302 | 136180 |
| http://www. milwaukeetimes.com | 26546 | 3790 | 22756 | 13 | 40787 | 88458 |
| http://www. bahraintribune.com/home.asp | 20996 | 7130 | 13866 | 17 | 21678 | 92599 |

**TABLE 1** Web page sizes from five newspaper sites

with some packet loss data and statistics on the round-trip time. In this case, the RTT delay seems pretty consistent for all measurements (with an average of 105ms, a minimum of 103ms and a maximum of 110ms), but keep in mind that this number can change dramatically at different times of day, and even within the same ping. A user's geographical location, their choice of ISP, and the current traffic levels on the Internet can greatly affect the delay. (A great free tool for getting a feeling of Internet latency between different backbones is the "Internet Health Report," which can be found at **www.internetpulse.com.** I strongly encourage you to check it out to get a real-time view of delays across the Net.)

## Turns

A typical Web page consists of a *base page* and several additional objects such as graphics or applets. As suggested in the previous section, these objects are not transmitted along with the base page; instead, the base page HTML contains instructions for locating and fetching them. Unfortunately for end-user performance, fetching each of these objects requires a fair number of additional communication cycles between the user's system and the Web site server—each of which is subject to the RTT delay I just mentioned.

The term *turns* defines the number of TCP connections required to completely download a page. For HTTP 1.0, the minimum number of turns required to download a typical base page is four. One turn is needed for DNS look-up (to translate a URL into an IP address) and three turns to request and receive the base page. In addition to these four turns, each distinct object in a Web page requires a minimum of three turns (more if the object is handled by another server and requires an additional DNS look-up, as is the case with most banner ad servers).

Since each turn is subject to the round-trip time delay, the very verbose HTTP 1.0 protocol is not terribly efficient for the typical modern Web page, which may have thirty, forty, or more distinct objects in it. Fortunately, most browsers allow four simultaneous connections so four turns can take place concurrently, which significantly improves things. The newest version of the HTTP protocol, 1.1, is designed to significantly reduce the number of TCP connections required for a complete page download; but until 1.1 becomes more widespread, most users and Web sites will have to pay a relatively high performance price for having a lot of objects in each page. (For resources on understanding the underlying mechanisms behind the HTTP protocol, see the StickyNotes feature at the end of this article.)

How do you calculate the number of objects in a Web page? The simplest way is to use the online page size tool I previously introduced. Alternatively, you can study its HTML source and look for IMG, APPLET, or OBJECT tags, which instruct the browser to fetch additional objects. Once you know the number of objects, you can calculate the number of turns using the formula in Figure 4 where NumObjects is the number of objects in the page. The constant **4** takes into account that it takes one turn to do DNS lookup and three turns to download the base page. NumObjects is multiplied by **3** because it takes three turns to download each object, and divided by **4** because, as you might recall, most browsers are set up to download up to four objects concurrently.

Now that we have a basic understanding of round-trip time and turns, and a way to measure them, we can come up with a rough estimate of the portion of the overall Web page download time that is attributable to them, as in Figure 5 (in which RTT represents the average round-trip time).

Figure 6 shows what the approximate impact of turns and delay would be on a page with forty objects and a ping time of 100ms.

This number is nothing to sneeze at; it means that even if you have a *big pipe* between the user and the server and are able to achieve sub-second raw download times, the page will not complete downloading in less than 3.4 seconds due to the number of turns and the delays associated with them.

$$\text{Turns} = 4 + \frac{3 \times \text{NumObjects}}{4}$$

**FIGURE 4** Calculating the number of turns

$$DT_{TURNS} = RTT \times \left(4 + \frac{3 \times \text{NumObjects}}{4}\right)$$

**FIGURE 5** Calculating delay due to turns

$$DT_{TURNS} = 100\text{ms} \times \left(4 + \frac{3 \times 40}{4}\right) = 3{,}400\text{ms} = 3.4\text{sec}$$

**FIGURE 6** Sample calculation with 40 objects and 100 ms ping time

## Processing Time

The last factor in the response time formula is the processing time required by the server and the client to *put together* the required page so it can be viewed by the requester. This can vary dramatically for different types of Web pages.

On the server side, pages with static content require minimal processing time and will cause negligible additional delay. Dynamically created pages (e.g., personalized home pages like my.yahoo.com) require a bit more server effort and computing time, and will introduce some delay. Finally, pages that involve complex transactions (e.g., credit card verification) may require very significant processing time and might introduce delays of several seconds.

On the client side, the processing time may be trivial (for a basic text-only page) to moderate (for a page with complex forms and tables) to extreme. If the page contains a Java applet, for example, the client's browser will have to load and run the Java interpreter, which can take several seconds.

Most Web sites consist of many hardware and software components from dif-

| | NO LOAD | 1,000 SESSIONS/HR | 2,000 SESSIONS/HR | 3,000 SESSIONS/HR |
|---|---|---|---|---|
| **Home Page** | 0.1 | 0.3 | 0.9 | 2.4 |
| **Product Info** | 0.4 | 0.5 | 1.6 | 5.5 |
| **Account Status** | 3.4 | 5.6 | 8.4 | 16.1 |
| **Credit Card Verification** | 15.5 | 18.5 | 22.4 | 37.7 |

**TABLE 2** Average page completion time (in seconds) at various load levels (measured on intranet at 100Mbits per second)

ferent vendors, each with a unique performance profile. Because of this varied anatomy, there isn't a simple formula that you can use to calculate processing time. The most practical way to understand and quantify this component is to actually measure it; and the simplest way to measure it is to bypass the Internet and to establish a direct connection to the site under test (using a fast 100Mbits/sec intranet, so bandwidth is maximized and delays are kept to a minimum). Then, with an old-fashioned stopwatch, measure the time it takes to *completely* load a page. (On Microsoft Explorer, you know a page has been completely loaded when the word "Done" appears on the bottom left corner of the window.)

Some pages will load so fast that you will barely have time to press the start/stop button on the stopwatch. (At 100Mbits/sec, it only takes about 1/100 of a second to transfer 100Kbytes.) Others will take a definite and measurable amount of time to complete loading. Since you are on a very fast connection with minimum delays and huge bandwidth, you can safely attribute any significant delays to server and client processing time. If you want to get really sophisticated, subtract 0.1 to 0.2 seconds from the measured time to account for delays in your reflexes and make sure you take at least five to ten measurements to get a statistically significant sample.

Keep in mind, however, that since every time you load a Web page the browser stores the various page components in its local cache, you must clear the cache between measurements. Otherwise, instead of fetching these objects from the server, you will be getting them from your local cache and bypassing the very process you are interested in timing. (Clearing your local cache is easy. On Microsoft Explorer, for example, follow the menu: **Tools→Internet Options→General,** and on the section "Temporary Internet Files" choose "Delete Files.")

In addition to page type, another factor that has a significant impact on processing time is *server load*. The response time of a server is inversely proportional to its load. As the number of users increases, the speed with which the server can respond to each page request diminishes; first gradually, then dramatically until the server crashes. Understanding how user load impacts server performance is the domain of load testing, an activity that should be part of any Web site performance effort.

(Web site load testing is a broad and complex topic, but I've written articles on the subject for the last two issues of *STQE* magazine, which I believe serve as a good introduction to the subject. I encourage you to read them and put them into practice, because any page response time analysis that does not factor in server load levels will be of limited usefulness in predicting end-user response time.)

If you are expecting your Web site to experience an average traffic level of, say, 1,000 sessions/hr, and a peak level of 3,000 sessions/hr, you should design a test that will drive your system at several different loads: the average load, the maximum anticipated load, and one or two load levels in between. Most load testing tools report the page response time for different pages, but in order to compare apples to apples, I still recommend taking manual measurements with your stopwatch to complement (and double check) the data reported by the load testing tool. At the end, you should have a table that looks something like Table 2.

As you can see in this example, the processing time can vary dramatically from page type to page type, and the degradation in performance varies as the load is increased.

## Putting It All Together

As we have seen, the page response times experienced by end users are affected by six principal variables: page size, minimum bandwidth, number of turns, round-trip time, server processing time, and client processing time. Let's look again at the basic formula correlating all these variables, shown in Figure 1.

Now that we've talked about load issues, note that this formula's "Page Response Time" is really a function of server load. Because the impact of server load on response time can be so severe, we want to make that connection very clear.

By studying this formula you can easily spot several ways to minimize your page response times. Keeping the *total page size* small, for example, can tilt the whole equation in your favor. Since transfer rates between low- and high-bandwidth clients can be dramatically different, small total page sizes will prevent the loss of low-bandwidth users. Remember that the performance difference

between a 40K and an 80K page is only a fraction of a second for a DSL/T1 user, but it's twenty seconds for a 28K modem user.

Even if you manage to keep the size of the page small (say, 40Kbytes), you can undo the benefit of all that byte trimming by populating the page with too many small *objects*. That kind of clutter means that even end users with very high bandwidth connections will have to wait several seconds for all the objects to arrive.

Since the value of the *round-trip time* delay variable tends to increase as the distance between the server and the client increases, you might want to consider taking advantage of content distribution networks (CDNs) offered by companies such as Akamai and Speedera. CDNs keep frequently downloaded pages and page objects (such as company logos) on multiple servers located on different network backbones. When a client requests one of those objects, the request is served by the closest server to minimize delay.

Even if all your pages are very small and contain few objects, your response time will decrease dramatically if the server can't handle the load. Your highest load times are your site's best opportunities for visibility and business activity, so don't forget to make sure that your system can handle it without crashing and turning that opportunity into a disaster—make sure that you do a load test before a site launch, site redesign, or new hardware installation.

A final suggestion: minimize the use of applets and other technologies that require a fair amount of client-side processing, since this could have a severe performance impact on slower systems. Just because the delay is on the client side you shouldn't feel exonerated, because the Law of Responsibility will come back to bite you: poor performance is always going to be perceived as your fault.

Happily, the reverse is often true as well: a positive experience on your Web site is something that users will give *you* full credit for—and armed with these common-sense principles, you're halfway there already. STQE

---

*Alberto Savoia* (alberto.savoia@keynote.com) *is Chief Technologist of Keynote's load testing division and has also served as founder and CTO of Velogic, General Manager of SunTest, and Director of Software Research at Sun Microsystems Laboratories. His sixteen-year career has been focused on applying scientific methodology and rigor to software testing, and he holds several U.S. patents—including two on software test automation.*

*STQE* magazine is produced by STQE Publishing, a division of Software Quality Engineering.