



MERCURY INTERACTIVE

# Common Web Site Problems Identified by ActiveTest

**White Paper**

**Mercury Interactive Corporation**  
1325 Borregas Avenue  
Sunnyvale, CA 94089  
408-822-5200  
[www.mercuryinteractive.com](http://www.mercuryinteractive.com)



## **Abstract**

Mercury Interactive's experts have conducted thousands of load tests on a variety of business-to-business and business-to-consumer Web applications. This paper examines the most common Web site performance problems encountered during load testing with ActiveTest™ and provides insight into their causes. In addition, it highlights some of the significant findings made by our load testing experts.

ActiveTest is a hosted, Web-based service powered by the industry-standard load testing tool LoadRunner®. It can generate a load equivalent to millions of customers on a single Web site. Moreover, ActiveTest uses real-time monitors to measure response times for transactions and gather metrics on all components within a Web infrastructure. With this data, ActiveTest experts can identify performance bottlenecks—inside and outside the firewall—and pinpoint their cause. Companies that implement recommendations made by ActiveTest experts have been able to increase Web site performance by an average of 400 percent.



## Table of Contents

Isolating Web Performance Problems with ActiveTest .....	6
ActiveTest Methodology .....	7
Common Web Performance Problems and Their Causes .....	9
Other Performance Problems .....	12
Summary .....	13
About Mercury Interactive .....	15

## Isolating Web Performance Problems with ActiveTest

When Web sites crash, deliver the wrong content or subject customers to delays, companies reap the repercussions. Both their reputations and revenue streams are impacted. Without automatic load and performance testing, however, it is nearly impossible to isolate the cause of Web site problems and quickly fix them. This is because most e-business applications rely on complex Web infrastructures that are an amalgam of servers, network devices and software.

After more than a decade of load testing and applying proper load testing methodology, Mercury Interactive's testing experts have uncovered the causes of common performance bottlenecks. To isolate these bottlenecks, they used ActiveTest, a hosted, Web-based service that enables e-businesses to load test their entire Web infrastructure.

### Testing Outside the Firewall

Powered by the industry-standard load testing tool LoadRunner, ActiveTest can generate loads equivalent to millions of customers on a single Web application. Moreover, ActiveTest uses real-time monitors to measure response times for transactions and gather metrics on all components throughout a Web infrastructure. With this data, ActiveTest experts can identify performance bottlenecks, pinpoint their causes and then fine-tune the performance of the sites.

More important, ActiveTest has the unique ability to test from outside the firewall. This is extremely valuable, since Mercury Interactive's testing experts now know that nearly 35 percent of bottlenecks are found only by generating the load over the Internet to the site. In this way, ActiveTest simulates the same end-to-end path that real online users travel and enables companies to isolate bottlenecks caused by routers, gateways, switches, bandwidth and ISP peering relationships.

ActiveTest uses multiple load testing farms and measurement points to triangulate problems that are outside the firewall but that cause performance problems for external users. ActiveTest therefore provides an ideal complement to in-house load testing tools and a means for companies to achieve a comprehensive load testing solution.

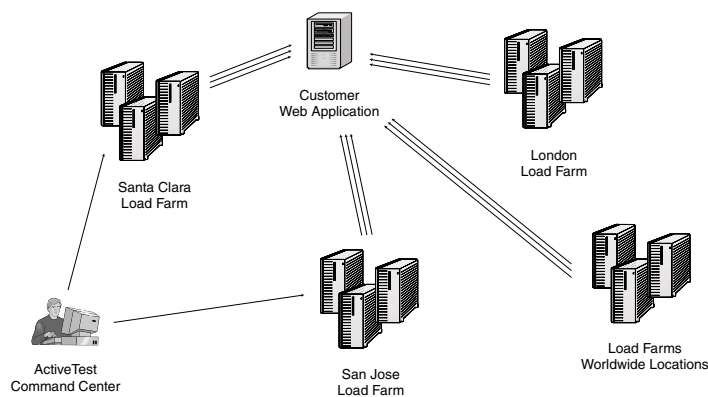


Fig. 1. This diagram shows a typical customer's Web application being tested by ActiveTest's multiple load farms via the Internet. The ActiveTest team coordinates the entire test from its Command Center.

## ActiveTest Methodology

In Web site load testing it is critical to create sufficient stress on the overall system in order to identify the system's bottlenecks. This is achieved by creating virtual users (Vusers) that emulate real-user interaction with the Web site—in particular, the most common business transactions for that site. Examples of common business transactions may include logging on or buying a book. By working with customers to understand and replicate their critical business processes, Mercury Interactive's ActiveTest team can determine which transactions are most common and most important for a given Web site.

### Identifying Performance Bottlenecks

To identify performance bottlenecks during a load test, companies must carefully monitor and evaluate the performance of the entire Web infrastructure. ActiveTest automates these processes by using real-time monitors to track thousands of parameters in the following areas:

- Transaction response times for each defined step in the business process
- Web server throughput and hits per second returned from the Web server
- Network behavior and errors
- Metrics for any component in the Web infrastructure (e.g., application server, Web server, database, firewall, load balancers)

### Isolating the Cause

Determining that you have a performance problem can be relatively simple. Isolating its cause can be—and often is—far more difficult and time consuming. During load testing, ActiveTest manages traffic and measures the load on the different tiers of the Web site architecture. In addition, ActiveTest can combine several business processes to accurately mimic real users and to stress particular components of the architecture. For example, if one of the tested business processes queries the database, ActiveTest can incrementally increase the number of Vusers to create more load on the database server. In this way, ActiveTest can gather data needed to pinpoint the cause of performance problems from anywhere within the infrastructure.

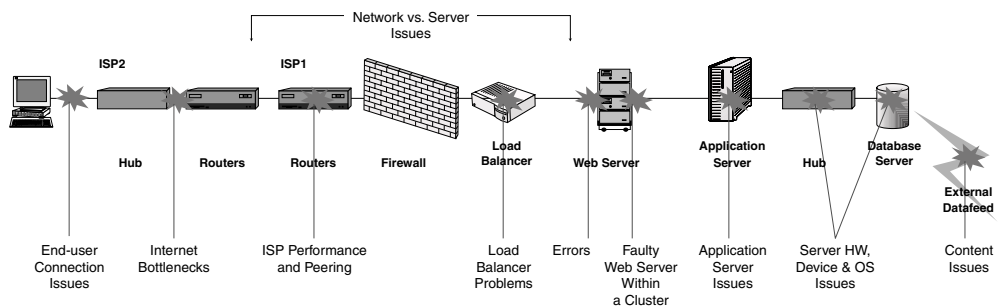


Fig. 2. ActiveTest pinpoints performance problems anywhere within the Web infrastructure.

### Increasing Site Performance

Most companies that test with ActiveTest are surprised to learn that their sites can handle only 15 percent of the traffic they expected before performance degrades. They also learn about ways to increase capacity and fine-tune their system for optimal performance. On average, companies that perform three to five runs with ActiveTest find that they can increase site performance by 400 percent.

The following example shows a two-run comparison in which ActiveTest was used to isolate the cause of a performance problem. After the problem was corrected, the site's scalability improved significantly.

In the first run, approximately 250 Users were on the site, causing the response time to increase to almost 110 seconds. During the test, the ActiveTest team evaluated the transaction performance of the Web server and network pipe. With this data, they pinpointed the site bottleneck to the database server. The ActiveTest team further determined that database indexes were not properly set up and that long database table scans were impacting performance.

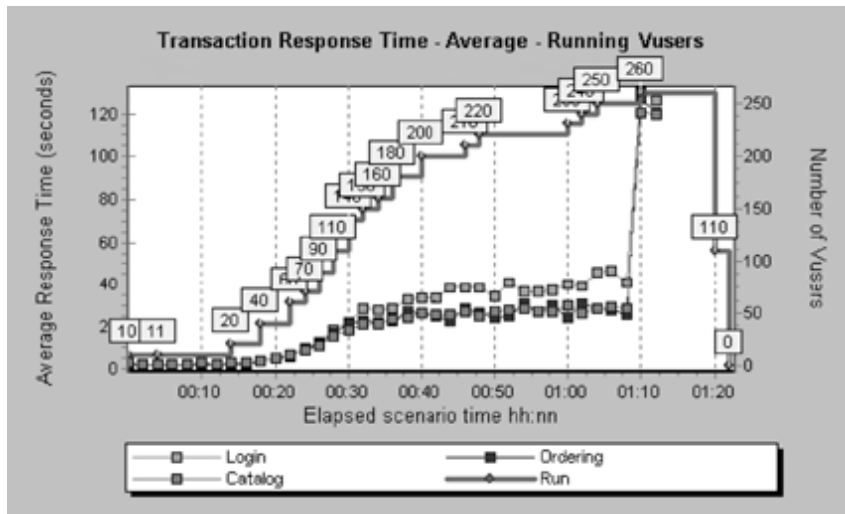


Fig. 3. Transaction response time for 250 virtual users indicates a performance bottleneck.

After adjusting table indexes, the ActiveTest team used the same scripts to retest the system. The results of the second run show an astounding improvement in Web site performance. For 250 users, the response time was reduced to less than 10 seconds. The transaction response time does not reach 110 seconds until the number of Users reaches 1,000. This represents a four-fold increase in performance.

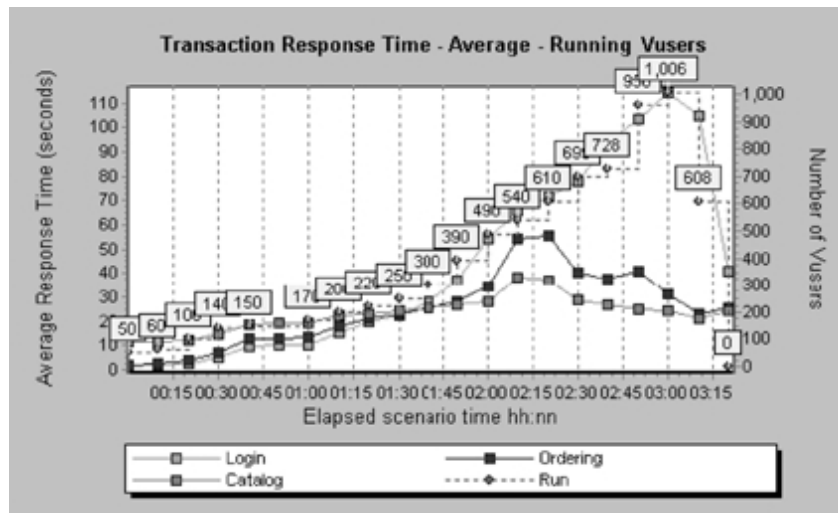


Fig. 4. Response time decreased to less than 10 seconds for 250 virtual users after isolating and fixing the performance bottleneck.

## Common Web Performance Problems and Their Causes

Web application performance problems can manifest themselves in several ways. Typical performance problems include:

- Long response time from the end-user's point of view
- Long response time as measured by the servers
- Memory leaks
- High CPU usage
- Too many open connections between the application and end users
- Lengthy queues for end-user requests
- Too many table scans of the database
- Database deadlocks
- Erroneous data returned
- HTTP errors
- Pages not available

Without automatic load and performance testing, however, it can be difficult—sometimes impossible—to isolate Web site problems and quickly fix them. The following sections describe the most common performance problems and their causes—namely, databases, Web servers, application servers and the network itself.

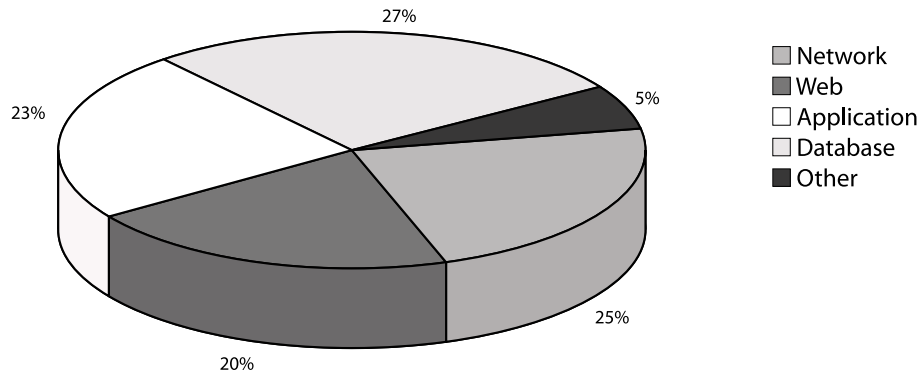


Fig. 5. Common Web site problems identified by ActiveTest. Sample is based on 1,000 load test runs.

### Databases

Dynamic Web sites that rely heavily on database operations are particularly at risk for performance problems. Many e-businesses do not realize that operations such as table scans or database connection setups can quickly exhaust system resources under load. As a result, 27 percent of Web site performance problems can be traced to malfunctioning database operations. Following are the most common database problems that the ActiveTest team has encountered:

- **Insufficient indexing.** Efficient database index design is key to good performance. Insufficient index design can easily drain system resources on the database server. Optimizing indexes through index tuning can increase performance dramatically.
- **Fragmented databases.** A fragmented database also can deplete system resources. The fragmentation causes long sequential scans of the database and a large number of database pages to be read. Compacting fragmented databases can improve performance, help accelerate queries and free up resources. Compacting simply places a table's records into adjacent database pages. As a result, only the minimum number of database pages needs to be read in order to retrieve all the records.
- **Out-of-date statistics.** As the data in a column changes, index and column statistics become out of date. This causes the query optimizer to make less-than-optimal decisions on how to process a query and thereby degrades database performance. This problem can be avoided by running automatic statistic updates, a feature found in most database servers.
- **Faulty application design.** When faulty application design is the problem, the client or application is the controlling entity, rather than the database server. The client then governs the types of queries, when they are submitted and how the results are processed. This in turn impacts the type and duration of locks, amount of I/O and the processing load on the server.

Many of the problems previously described cause the perpetual use of temporary tables. This often results in long requests and responses. In addition, the use of ODBC instead of a native database can further contribute to poor performance.

## Web Servers

Web server bottlenecks are the cause of poor performance in 20 percent of all test cases. These bottlenecks usually result from poor configuration and insufficient resources. For example, combining Web and application servers or using Active Server Pages technology can quickly exhaust server resources.

Load test results often show high transaction response times coupled with normal network latency measurements. Such low throughput and hit rates usually are the result of a faulty Web server. Using real-time monitors and specific script combinations, the ActiveTest team can exercise different Web site resources, such as Web servers, in order to isolate the actual cause of the problem. Following are common problems that ActiveTest has linked to misconfigured Web servers.

- **Poor Web server design.** Often poor design leads to inefficient communication between the end client and the application server. This can lead to requests not being evenly distributed to the application server and thereby cause abnormal use of memory and the CPU. Other times, poor design algorithms can contribute to inefficient data/page caching. A single software configuration setting often can rectify this problem.
- **Memory problems.** Most Web servers handle file caching quite well. However, physical memory constraints and mismanagement of server memory can easily cause a system bottleneck. Too often, memory issues are mistaken for other performance problems, such as poor disk storage and high CPU usage. For this reason, it is important to first eliminate Web server memory problems before exploring other possible causes of poor performance.
- **High CPU usage.** Processor bottlenecks occur when one or more processes use nearly all of the computer's processor time. Poorly written code can easily exhaust this resource and cause the system to hang. In some cases, poorly written DLLs have exhausted CPU time and created a bottleneck on the Web server. When CPU usage exceeds 70 percent for normal traffic, the system typically will be unable to accommodate traffic surges and fail.

## Application Servers

Application servers account for more than 20 percent of system bottlenecks. Moreover, since the application server is the hub of many system architectures, its performance problems can impact the entire architecture. Identifying these problems, however, can be challenging because infrastructures are so complex. For example, application servers communicate directly with databases, so bottlenecks often are confused with database server transactions. Typical causes of application server performance problems include:

- **Poor database tuning.** That is, the application server is not optimizing queries and is sending too many of them to the database, taxing the CPU and generating errors.
- **High CPU usage due to poor cache management.** When cache is not utilized properly, the available cache is completely consumed. As a result, the information must be constantly swapped to and from the hard drives. This takes far more time than simply accessing the cache memory.

Several factors can contribute to poor cache management. One is lack of memory, which occurs when the process needs more memory than what is available at the moment. This situation can be identified by monitoring memory usage and/or bursts in CPU performance. By enhancing memory on the machine in use, companies can easily remedy this problem.

- **High CPU usage due to poor session management.** When sessions are not managed efficiently by the application, the application server cannot manage the load. A common cause of poor session management is incorrect software configuration, which can compact waiting queues and cause timeouts. Poor concurrent handling of client page requests also can slow the rendering of the pages, as discussed in the Web server section.

Unusual behavior by the application server hardware, including wildly fluctuating database calls, queuing requests, application load times and throughput to the Web server, often are indications of a problem. The source of these problems can be found by generating a consistent load and holding it steady during a test. This methodology enables companies to determine whether such fluctuations are a result of rapidly changing load and requests, or of bottlenecks within the application server.

## Networks

Network bottlenecks are revealed often only under heavy load conditions, such as those experienced by informational sites that use many static Web pages. Pinpointing the bottleneck requires specialized testing that creates the load without significant server or database involvement. The ActiveTest team has found that in almost 25 percent of all cases, the pipe to the Internet could not sufficiently handle the desired load and caused delays in incoming and outgoing requests. Moreover, the ActiveTest team has frequently uncovered bottlenecks between the customer's Web site and the ISP.

The ActiveTest team also has found that the firewall, load balancers, gateways and routers can contribute significantly to performance problems. Hardware incompatibility or improper software configuration are often the culprits. Pinpointing these problems can be difficult, since they are often evident only when traffic levels are high. For example, to determine whether a lull in performance is due to a software or a hardware problem, the ActiveTest team must first try to optimize the software settings. If that fails, then the hardware is most likely incompatible.

The ActiveTest real-time network monitor traces transactions through each network segment to determine which hops are causing the bottleneck. ActiveTest also records the route from the ActiveTest farm to the destination Web site, analyzes this data to see where response times were exceeded and identifies problems in different network segments.

## Other Performance Problems

Security is a necessary part of any Web infrastructure, but it also can lead to bottlenecks. Excessive use of HTTPS and other security measures can quickly deplete server resources and contribute to system bottlenecks. For example, when HTTPS is implemented on the Web server during a load test, system resources are exhausted by a relatively small load. This is caused by secured socket layer (SSL) resource-intensive operations.

Continuously open connections also can drain server resources. Unlike browsers, servers providing SSL services typically create numerous sessions with large numbers of clients. Caching the session identifiers from each transaction can quickly consume the server's resources. In addition, the "keep-alive" enhancement features of most Web browsers keep connections open until they are explicitly terminated by the client or server. As a result, server resources may be wasted as large numbers of idle browsers remain connected to the server.

During load testing, such problems can be easily identified. Once the Web site reaches its limit of open connections, the ActiveTest team correlates the number of users to hit-per-second values and to error messages of unsuccessful connections in order to help fine-tune the site.

The performance of secured Web sites can be improved by:

- Fine-tuning the SSL and HTTPS services according to the type of application
- Using SSL hardware accelerators, such as SSL accelerator appliances and cards
- Changing the level of security according to the level of sensitivity of the data (i.e., change the key length used for public key encryption from 1,024 to 512 bits)
- Avoiding the excessive use of SSL and redesigning those pages that have low levels of data sensitivity to use regular HTTPS

In addition to SSL services, a multitude of other problems can cause system inefficiencies. Examples frequently include objects that are not maintained, batch jobs that are improperly scheduled and performance boosting features, such as connection boosting, that are not used. As with SSL problems, all of these can be easily corrected.

It is important to remember that any component within the Web infrastructure—including routers, firewalls and load balancers—can potentially contribute to performance bottlenecks. Because these devices perform demanding operations, such as packet cracking, IP source address analysis, protocol analysis, URL parsing, cookie parsing and others, they can easily cause bottlenecks that are difficult to detect. For this reason, it is critical to monitor these components when load testing a system.

## Summary

Poor Web site performance can be caused by several factors—from a faulty database server to misconfigured hardware and software to resource-intensive security requirements. With Mercury Interactive's ActiveTest, companies can load test systems to pinpoint the exact cause of performance slowdowns anywhere within the Web infrastructure. This is critical since 35 percent of bottlenecks are found only by testing outside the firewall. ActiveTest delivers this capability, making it a complementary and vital part of any in-house load testing tool box.

By using ActiveTest, e-businesses can solve performance problems before they impact customers and fine-tune their Web infrastructure to ensure peak performance around the clock.

In summary, common problems pinpointed with ActiveTest include:

**General**

- Insufficient memory
- Incompatible service packs and application extensions (e.g., DLLs)
- Excessive queuing requests
- Too many secure HTTPS connections in use

**Database**

- Inefficient indexing
- Fragmented databases
- Out-of-date statistics
- Faulty application design

**Web server**

- Insufficient memory
- Poor Web server design
- High CPU usage

**Application servers**

- Poor cache management and high CPU usage
- Lack of memory
- Poor session management
- Poor database tuning

**Network**

- Inadequate Internet pipe
- Hidden bottlenecks between the customer's Web site and the ISP (ISP peering)
- Faulty hops (misdirected traffic and lost packets)
- Misconfigured software and incompatible hardware

For more information about the ActiveTest service, visit:  
[www.mercuryinteractive.com/products/activetest](http://www.mercuryinteractive.com/products/activetest)

## About Mercury Interactive

Mercury Interactive Corporation is the worldwide leader in Web performance management solutions. Mercury Interactive solutions turn Web application performance, scalability and user experience into competitive advantage. The company's performance management products and hosted services are open and integrated to best test and monitor business-critical Web applications.

Together with our world-class partners and award-winning service and support, Mercury Interactive provides the industry's best Web application performance solutions capable of supporting both traditional and wireless e-business applications.

More than 10,000 e-commerce customers, Internet service providers, applications service providers, systems integrators and consultants use Mercury Interactive solutions. Mercury Interactive is headquartered in Sunnyvale, California, and has offices worldwide. For more information on Mercury Interactive, visit our Web site at [www.mercuryinteractive.com](http://www.mercuryinteractive.com).