Beyond Performance Testing

by:
R. Scott Barber

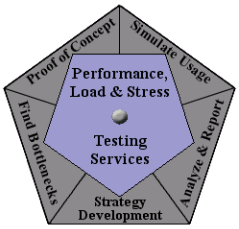
Part 12: Testing and Tuning Common Tiers

This is the second of four articles on our final theme in this series, which I call "the performance testing and tuning team." As we've discussed throughout this series, performance testers can provide exponentially more value to the primary tuners if they have a solid working knowledge of the systems and technologies they're testing. Accordingly, this article and the final two explore common areas of poor performance and describe things you can do to assist in the testing and tuning of these areas. While these three articles won't make you a tuning expert, I hope they'll make you more aware of some of the things that you can do to add value to the team. These final three articles will also introduce many topics in passing that you may want to research in more detail. Where appropriate, I'll recommend reference material for further research.

So far, this is what we've covered in this series:

This article focuses on Web, application, and database servers, which are common tiers related to testing and tuning. All Web-based systems have a Web or presentation tier, and most Web-based systems that are at all interactive, such as e-commerce sites, include some kind of application or business logic tier and a data storage tier. As we discussed in Part 9, tiers don't always correlate with physical machines but are logical distinctions that often correlate with hardware and/or software applications. Here we discuss specific issues related to these tiers.

This article is intended for mid- to senior-level performance testers and members of the development team who work closely with performance test engineers. You should have read at least Parts 9 and 11 before reading this article; it would be helpful to have read Parts 5, 6, 7, 8 and Part 10 as well.



The Web or Presentation Tier

For a Web-based system, the Web or presentation tier is generally the least complicated tier and the easiest to both test and tune. Of course, this doesn't imply that it should be ignored. Not fewer than half of the Web applications I've been involved in testing have required some testing and tuning specific to the Web server. Often some very simple Web server–specific testing can have profound effects on the production system. Some of this is included in the kinds of performance testing we've already discussed, but there are also other ways to test particular attributes of a Web server that can add significant value to the analysis and tuning processes.

Testing

During every performance test that you execute against a Web application measuring end-to-end response time, you exercise the Web server. With very rare exceptions, all of the requests sent by your user experience scripts are sent to the Web server. Analyzing those individual requests and response times to identify pages that are loading more slowly than expected is a great way to start testing this tier and is one way to determine if more Web server–specific testing would be worthwhile. Instead of looking at your timers (page load times), look at the response times for individual commands, each of which represents a particular interaction with the Web server. You'll usually find that most commands have very short response times. The few that have longer times may be worth looking at more closely.

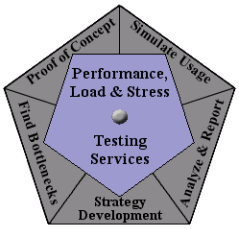
The first thing to look at is whether those commands trigger activities that take place exclusively on the Web server or whether they trigger activities on other tiers before a response is sent back to the client. I discussed how to determine which actual `GET` or `POST` is related to a command ID in Part 8 of this series (see the section entitled "Evaluate Commands with Slow Responses"). Loading graphics and static HTML pages generally exercises only the Web server, so those are the specific activities we're interested in. If one of these activities contributes significantly to the "too slow" total page load time, the Web server is a viable place to look for tuning opportunities.

Some specific indications that the Web server is a significant contributor to slow page load times are as follows:

- small graphics (under 50 KB) returning slowly
- all graphics returning slowly
- header files returning slowly
- a significant and sudden increase in response times at a specific and identifiable load below what the Web server was expected to handle
- significant resource utilization (that is, CPU or memory usage) at loads that the Web server was expected to handle

If none of these indications apply, it's unlikely that the Web server is the bottleneck, and therefore it probably doesn't deserve additional attention at this time.

If one or more of these indications apply, it may make sense to develop some specific tests targeted against the Web server. My approach to this is as follows. I first ask a developer to put these files on the Web server and provide me with a valid URL to access them from the machine I'm using to record



scripts:

- an HTML document containing only unformatted text (approximately one screen's worth)
- graphics files of various sizes (1 KB, 10 KB, 50 KB, 100 KB are common, but use what you have easily available as long as you note the actual file sizes)
- one HTML document containing each of the graphics exclusively
- one HTML document containing all of the graphics

Once this is done (which should take someone with full access to the Web server less than an hour), simply record a script that requests each of these objects sequentially (you can put each request in a timer for easy reporting) and make appropriate modifications for delays. Because there won't be any links to follow, you'll have to record this script by physically typing the URL from the developer into the browser's address bar.

Before running this script, you'll want to disable caching on the Web server and on the client/script. This will force the script to get the actual file each time. Then execute the script, looped 10 or more (I prefer 100 when time permits) times with a single user. This will give you various single-user response times to use as a comparison, both to one another and to future tests. While this test is executing, the Web server administrator will likely want to monitor resource usage on the Web server. If the test doesn't provide enough insight, you can re-execute it with increasing numbers of users.

While this test has little to no realism, it isolates the Web server in a way that helps determine if it's a bottleneck and provides information that may be critically helpful in tuning. Remember that it may be helpful to turn on, or increase the level of, logging on the Web server.

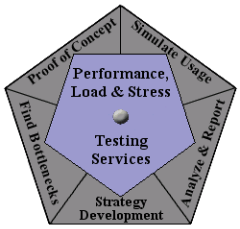
Note that if all Web server transactions seem to be slow and there doesn't seem to be anything tuned poorly on the Web server, you may want to ping the Web server from your client. The latency may actually be in the network, not the Web server. We'll discuss this in detail in Part 13 of this series.

Tuning

Tuning the Web server is generally the responsibility of the system administrator, but it may help for you to know some common things that may need to be tuned. There are several good books and resources about tuning parameters for each of the countless Web servers on the market, such as Apache, IIS, iPlanet, and Netscape Enterprise Server. One example is the Web site dedicated to configuring or tuning Apache.

In my experience, these are the things that commonly need to be tuned on a Web server if it turns out to be the cause of the bottleneck:

- number of processes
- number of server threads
- allowable open/persistent connections
- session duration
- caching



- logging
- operating system parameters (that is, virtual memory settings and such)
- hardware resources (that is, disk I/O, RAM, and the like)
- partition configurations
- other programs/processes/software on the same machine

This isn't an inclusive list, just some common places to look as you get started in your tuning process. For further information, I recommend two books: *Web Performance Tuning: Speeding Up the Web* by Patrick Killelea and *Speed Up Your Site: Web Site Optimization* by Andrew B. King. These books are great introductions to tuning various components of Web-based applications. *Web Performance Tuning* contains specific advice regarding hardware, operating systems, and software common in 1998. While some of the specific information may be outdated, the concepts presented in this book are still valid. *Speed Up Your Site* is about optimizing the code for individual Web pages. This is great information when you find that your download times are slow, or your bandwidth utilization is high though your server response times are fast.

The Application or Business Logic Tier

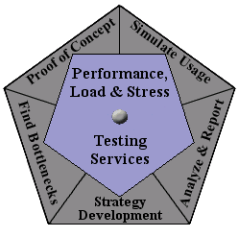
The application tier of a Web-based system is generally the tier that contains the business logic. This can be thought of as the brains of the application, where input taken from the user (via the Web server) is processed in some way. For instance, this is where search parameters are converted into valid SQL statements to be passed to the database, or shipping costs and applicable taxes are calculated for an e-commerce site. Testing and tuning this tier is significantly more complex than testing and tuning the Web or presentation tier.

Testing

Determining whether tests specific to the application tier are required or useful follows basically the same process as for the Web tier. Let the user experience tests you've already created exercise the application tier while you observe resource utilization. The system administrator can assist with that. If you (or the administrator) see that resources seem to be stretched too thin, you can start limiting or modifying your existing user experience tests to narrow your search for the offending pages and/or requests.

If there are no suspect resource utilization metrics, it's time to look more closely at your page load times. If a particular subset of pages is exhibiting unacceptable slowness, you'll need to determine what's unique about them. You can identify the underlying `GET` or `POST` of each command that has a slow response in order to evaluate whether these pages have requests for certain objects (like graphics) in common. If this doesn't point conclusively to something known to be handled by a different tier (like an excessively large graphic or file download), your next step is to find out if the offending request is accessing and causing activity on the application tier. This is often a collaborative process with the development team.

Once you determine that slow pages do access the application tier, you should start the bottleneck detection process discussed in detail in Parts 7–10 of this series. Unlike for the Web tier where you'll



be able to recommend tests to the development team that might add value, you'll likely have to depend on them to turn on logging during your tests of the application tier and to tell you which tests will be most helpful. Often these tests will require the techniques for collecting data by tier discussed in Part 9.

Tuning

The task of tuning the application tier is often shared by the tester, the system administrator, and the development team. It's often a balancing act involving modification and configuration of the hardware, the operating system, the application server software, and custom code. As the performance tester, you can expect to be asked to create complex tests and to execute them many, many times as you try out different combinations of configurations and modifications. While you most likely won't know enough to recommend many different configuration options, one important way you can contribute is to take notes and document the results of making changes. This is actually extremely useful to the administrators and developers, and keeps you actively involved with the team.

The following is a generic list of things to think about when the application tier has been identified as the bottleneck:

- hardware resources (that is, disk I/O, RAM, and so on)
- operating system parameters (that is, virtual memory settings and such)
- logging
- management of processes and/or threads
- business logic algorithms
- custom code
- partition configurations
- other programs/processes/software on the same machine

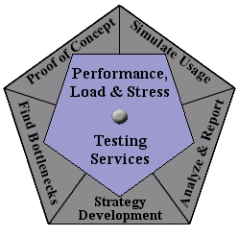
Once again, there are countless books and resources dedicated to configuration and tuning of specific application servers, such as IBM WebSphere, JBoss, and BEA WebLogic. One example of a book dedicated to configuring or tuning the IBM WebSphere application server on a specific hardware platform is *Java and WebSphere Performance on IBM Iseries Servers* by IBM Redbooks.

The Data Storage Tier

Data storage tiers very often hide significant bottlenecks that don't present themselves until performance testing. DBAs typically test and tune databases for expected single-user scenarios, whereas it's the multi-user scenarios and/or data variance involved in performance testing that commonly bring database bottlenecks to light.

Testing

Determining whether the database is a bottleneck starts in exactly the same way as for the application tier. First look at the slow page loads and find out whether those pages involve database access. If they



do, ask the DBA to monitor the database during tests. Make certain that the DBA understands which data you're using and the sequencing of the activities in the test. Often you'll be asked to vary your testing to help the DBA monitor and determine the cause of the slowness. A common request will be to vary the load or the test configuration to determine if the bottleneck is related to multiple users accessing the same row or table.

You may find yourself using some of the techniques from Part 9 to isolate the data storage tier. This will help you determine if the bottleneck is actually on that tier or just appears to be because of a problem like inefficient SQL generation on another tier.

Tuning

Database tuning can be extremely complex and varies dramatically from one database to another. Most enterprise databases have so many interrelated configuration options that there are literally millions of configurations possible, so it's virtually impossible to follow the scientific method of making one change at a time. Even an experienced DBA often has to follow an educated trial-and-error process. You can help most by designing tests that are easily re-executed, monitoring end-to-end response times, and taking notes — just like for the application tier.

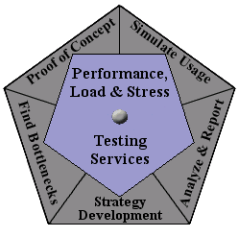
Here are some common areas to think about when the database turns out to be the bottleneck:

- hardware resources (that is, disk I/O, RAM, and so on)
- operating system parameters (that is, virtual memory settings and such)
- logging
- query optimization
- indexes
- stored procedures
- row and/or table locking
- complex joins versus temporary tables
- partition configurations
- other programs/processes/software on the same machine

Some commonly used databases are DB2, Oracle, Sybase SQL Server, and Microsoft SQL Server. Doing a Yahoo search on "DB2 tuning" yielded more than 121,000 entries. I scrolled through the first 200 and found every one to be potentially useful to someone wanting to tune that particular database. "Oracle tuning" yielded more than 250,000 hits. As you can see, a lot has been written about database tuning, so don't expect to become an expert in this area overnight.

Summing It Up

Testing and tuning common tiers is generally an iterative, collaborative process. There are countless possible areas to be configured and/or tuned. Through experience, you'll come to recognize what some symptoms indicate. But no matter how experienced you become, you'll find that you can help the most



by developing and executing tests with the idea of providing value to the developers and administrators and by keeping a notebook of changes and associated results. You can increase your ability to contribute to test creation and tuning by researching published tuning tips specific to the hardware and software being used by the application you're testing.

References

- *Web Performance Tuning: Speeding Up the Web* by Patrick Killelea (O'Reilly & Associates, 1998)
- *Speed Up Your Site: Web Site Optimization* by Andrew B. King (New Riders, 2003)
- *Java and WebSphere Performance on IBM Iseries Servers* by IBM Redbooks (Vervante, 2002)

Acknowledgments

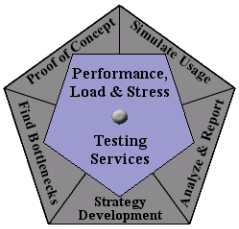
- The original version of this article was written on commission for IBM Rational and can be found on the IBM DeveloperWorks web site

About the Author

Scott Barber is the CTO of PerfTestPlus (www.PerfTestPlus.com) and Co-Founder of the Workshop on Performance and Reliability (WOPR – www.performance-workshop.org). Scott's particular specialties are testing and analyzing performance for complex systems, developing customized testing methodologies, testing embedded systems, testing biometric identification and security systems, group facilitation and authoring instructional or educational materials. In recognition of his standing as a thought leading performance tester, Scott was invited to be a monthly columnist for Software Test and Performance Magazine in addition to his regular contributions to this and other top software testing print and on-line publications, is regularly invited to participate in industry advancing professional workshops and to present at a wide variety of software development and testing venues. His presentations are well received by industry and academic conferences, college classes, local user groups and individual corporations. Scott is active in his personal mission of improving the state of performance testing across the industry by collaborating with other industry authors, thought leaders and expert practitioners as well as volunteering his time to establish and grow industry organizations. His tireless dedication to the advancement of software testing in general and specifically performance testing is often referred to as a hobby in addition to a job due to the enjoyment he gains from his efforts.

About PerfTestPlus

PerfTestPlus was founded on the concept of making software testing industry expertise and thought-leadership available to organizations, large and small, who want to push their testing beyond "state-of-the-practice" to "state-of-the-art." Our founders are dedicated to delivering expert level software-testing-related services in a manner that is both ethical and cost-effective. PerfTestPlus enables individual experts to deliver expert-level services to clients who value true expertise. Rather than trying to find individuals to fit some pre-determined expertise or service offering, PerfTestPlus builds its services around the expertise of its employees. What this means to you is that when you hire an



analyst, trainer, mentor or consultant through PerfTestPlus, what you get is someone who is passionate about what you have hired them to do, someone who considers that task to be their specialty, someone who is willing to stake their personal reputation on the quality of their work - not just the reputation of a distant and "faceless" company.