Chapter 9

Performance Testing

LEARNING OBJECTIVES

- to define the kind of testing that measures the speed of software
- to analyze techniques that simplify the intrinsically complex performance testing of software transaction mixes
- to assess the potential business liabilities of ignoring performance testing

9.1 INTRODUCTION

We advance from testing techniques that validate the software behavior to testing techniques that validate the software "speed." Speed in this context means that a tester measures aspects of software response time while the software is laboring under a controlled amount of work, called a "workload." To make the software reveal its true production speed, the tester must execute the performance tests in a testing environment that approximates the intended production environment as closely as possible. These execution testing techniques are fundamentally different in objective and approach from functional testing where the objective is validating correct code behavior regardless of speed.

Performance testing occurs after the functional testing is mostly completed and the software has become quite stable (fewer and fewer changes or corrections). Functional defects in the software may be revealed during performance testing, but this is not the testing objective.

The objective of performance testing is to validate the software "speed" against the business need for "speed" as documented in the software requirements. Software "speed" is generally defined as some combination of response time and workload during peak load times. These peak load times may occur during lunch time, at the opening of the stock market day, or after midnight when all online customers are in bed (overnight batch workload). Performance testing is achieved by a series of tests that introduces increasingly more workload of increasingly more complex business transaction mixes.

9.2 WORKLOAD PLANNING TECHNIQUES

Your first thoughts about performance testing may revolve around timing measurement precision. If the workload is incorrectly planned or executed, the most precise timing measurements will not bear any resemblance to the timing exhibited by the software in production. So we need to consider workload planning as the first step in performance testing.

The performance tester's first workload challenge is to identify which business transactions and activities need to be measured for performance. In all but the simplest software applications, it is difficult to identify the most important transaction and activity candidates for performance testing. The difficulty lies partly with the software end user's typical reaction, "everything must respond in less than 3 seconds." Somehow the tester must convince the end-user that different groups of transactions and activities will have different performance requirements based on different business priorities.

For example, customer purchase transactions may need to be completed in less than 3 seconds in order to keep the customer interested in buying more merchandise. By contrast, the transactions like credit card validation or warehouse ship ordering that occur to complete the purchase can be done 3 hours later and the customer will still get his or her merchandise delivered on the same date next week. This slow response time grouping may not thrill the employees, but as long as they can get their job done in time, the business investment required to speed up the employee transaction grouping response time faster than 3 hours has a very low return on investment.

Another example would be bank customer online requests for fund transfers. Because it is the policy of most banks to make the fund transfer available no sooner than overnight, the online customer request confirmation for fund transfer may have a 5 seconds response time requirement whereas the transfer transaction itself has a 12 hours response time (batch, overnight).

Lurking in the background of all online transaction rate discussions is the "Rule of 8." The Rule of 8 is a human behavior discovered in the 1970s by measuring the productivity of computer users as their system response time slows from subseconds to 20 seconds. The conclusion back in the 1970s and reconfirmed in the 1990s is that when the system response slows down beyond 8 seconds per transaction, human productivity falls off dramatically. The popularly accepted explanation for this phenomenon is that when the system responds in 8 seconds or less, the computer user can retain his or her train of thought about intended next actions. Response times longer than 8 seconds cause the computer user to forget what he or she was doing or intended to do next. For this reason, there was much emphasis placed on response times of 8 seconds or less for mainframe and client/server applications during the 1980s and 1990s.

Here is an interesting side note to the Rule of 8. Computer center staff continually need to manage the expectations of their users. If something does not work correctly, the users get upset. If something does not work fast enough, the users get upset. By the late 1970s, large mainframe computers had the capacity for delivering subsecond response times over a wide range of software activities for a large number of users. Computer center managers opted to "tune" these mainframes to a 4–5 seconds (intentionally slowed down) response time. Users became accustomed to the 4–5 seconds response time as the norm and certainly satisfactory with respect to the Rule of 8. Then, as new software was added to the mainframe, there was reserve capacity that the computer center managers could "tune" back in so that the response time never slowed down beyond the 4–5 seconds range. The users came to expect that adding software to the mainframe's workload would not slow down the response time as they had experienced in the 1960s and early 1970s. The computer center managers found a way to manage performance expectations of their users without giving the users the absolute best performance that the technology could provide.

The emergence of Internet applications has caused great consternation among end users. Internet application response times often range in the 10 seconds–3 min range with little hope for improvement using the current technology. Knowing that the Rule of 8 is still alive and well, the tester must team with the developer in setting realistic end-user expectations about performance capabilities and limitations of a new Internet system.

9.2.1 Documenting the Performance Workload Requirements

The outcome of this workload analysis for a purchasing application might be the following groups of transactions and their performance requirements. Notice that the performance requirement is stated as a maximum response time (no slower than) rather than an average response time.

Draft 1 Performance workload plan

Transaction group	Response time requirement
Menu navigation	Max 3 seconds
Log on/Log off	Max 3 seconds
Product detail display	Max 4 seconds
Purchase steps	Max 7 seconds
Catalog search	Max 10 seconds
Credit card payment	Max 30 seconds
Product ship	Max 24 hours

9.2.2 Documenting the Performance Workload Peaks

The performance tester's second workload challenge is to determine peak usage of each group of transactions and the timeframes in which the peak usage occurs.

Peak usage is normally measured in terms of active users. For example, if 500 users normally log on to the purchasing system over the lunch hour and 2,000 users normally log on around 7P.M. in the evening, then the peak expected workload for all log on/log off is 2,000 users. Notice also that the peak expected workload is 2,000 users whether the user base is 2,000 users or 2,000,000 users.

When researching the peak workload of users for a particular group of transactions, specify that you are interested in numbers of *active* users at peak workload time rather than the number of *concurrent* users. Concurrent user workloads are of interest for specific tests such as database record lock contention, deadly embraces, or network packet collisions. The actual occurrence of truly concurrent activity is rare. For example, if the peak workload is 2,000 active users, then it will be most unusual to observe more than five truly concurrent users at any time. Truly, concurrent users press the Enter key on the same transaction at precisely the same time causing the same transaction to launch in the application and follow exactly the same execution path.

Active users access the system during the same timeframe, have been allocated system resources for a work session, and are launching a variety of activities or transactions based on their job role or customer needs. It is this allocation/reallocation of system resources during the active work sessions that will affect the vast majority of transactions being launched. Each launched transaction will compete with every other launched transaction for these finite system resources. It will be the objective of the performance instrumentation to measure how well the transactions compete for these resources under different workload situations.

The peak usage of each transaction group is now added to the workload plan:

Draft 2 Performance workload plan

Transaction group	Response time requirement	Peak active users/ customers	Day/time of peak activity
Menu navigation	Max 3 seconds	2,000	Mon–Fri 12–1 P.M.
Log on/Log off	Max 3 seconds	2,000	Mon–Fri 12–1 P.M.
Product detail display	Max 4 seconds	2,000	Mon–Fri 12–1 P.M.
Purchase steps	Max 7 seconds	500	Sat 9–11 A.M.
Catalog search	Max 10 seconds	2,000	Mon–Fri 12–1 P.M.
Credit card payment	Max 30 seconds	500	Sat 9–11 A.M.

The Mon–Fri 12–1 P.M. peak activity entry means that a peak of 2,000 customers use the system during lunchtime at least one workday of the week. From a performance testing perspective, it does not matter which specific day it occurs or if it occurs during several weekdays. The peak is the peak from a performance testing perspective. Later in the development process, the production staff will want very much to know which day(s) of the week the peak occurs for production schedule perspective.

The third performance tester's challenge is to clarify just how many different workload peaks need to be tested. Looking down the Draft 2 plan, you will notice that not all of the transaction groups peak Mon–Fri noon. Some of the groups peak during Sat A.M. Looking a little closer, you will realize that the weekday noon transaction groups are predominantly browsing activities in contrast to the Saturday transaction groups that are predominantly purchasing activities. This weekly workload has more than one peak. To complete the workload plan, you will need to revisit each of the two peaks and develop separate workload plans, now designated weekday browsing and Saturday purchasing.

Look for the peak active users for each transaction group in all workload plans. Failure to revisit each transaction group for all plans may cause performance test underreporting. For example, if you test the weekday browse peak workload without any purchases because purchases peak on Saturday, you may miss the fact that a few purchase transactions during the weekday could noticeably slow down a browse response. Although the number of customers making purchases during the week does not reach the Saturday 500 peak, you should strongly suspect that there may be at least 50 purchases (10% of Saturday peak) during the weekday browse workload peak. Similarly, the number of customers doing catalog searches on Saturday does not reach the 2,000 weekday peak; however, you should suspect that there may be at least 200 catalog browses (10% of the weekday peak) during the purchase Saturday workload peak.

Here is the revised workload plan split out to represent the two workload peaks discovered so far.

Draft 3 Performance workload plan for weekday browsing

Transaction group	Response time requirement	Peak active users/Customers	day/time of peak activity
Menu navigation	Max 3 seconds	2,000	Mon–Fri 12–1 P.M.
Log on/Log off	Max 3 seconds	2,000	Mon–Fri 12–1 P.M.
Product detail display	Max 4 seconds	2,000	Mon–Fri 12–1 P.M.
Purchase steps	Max 7 seconds	50	Mon–Fri 12–1 P.M.
Catalog search	Max 10 seconds	2,000	Mon–Fri 12–1 P.M.
Credit card payment	Max 30 seconds	50	Mon–Fri 12–1 P.M.

Draft 3 Performance workload plan for saturday purchases

Transaction group	Response time requirement	Peak active users/customers	Day/time of peak activity
Menu navigation	Max 3 seconds	500	Sat 9 A.M.–11
Log on/Log off	Max 3 seconds	500	Sat 9 A.M.–11
Product detail display	Max 4 seconds	200	Sat 9 A.M.–11
Purchase steps	Max 7 seconds	500	Sat 9 A.M.–11
Catalog search	Max 10 seconds	200	Sat 9 A.M.–11
Credit card payment	Max 30 seconds	500	Sat 9 A.M.–11

9.3 WORKLOAD EXECUTION TECHNIQUES

Once you have identified the groups and volumes of transactions for each peak workload test, you need to develop the steps to create that peak in a test environment so that performance measurements can be taken. There are three traditional steps for executing performance test workload.

9.3.1 Workload Ramp-up to Peak

Workload ramp-up is the process of initiating enough user sessions to drive the peak workload. If the peak is 2,000 users, then 2,000 users must be logged on to the system in the test environment during the first part of the peak workload window. In real life, 2,000 users would not simultaneously log on within the first seconds of the lunch hour. The log ons would be staggered over a period of 5–10 min (more workload analysis information is needed here) as users put away their daily business and prepare to browse during their lunchtimes.

Successful ramp-up without any user transaction activity is the testing objective of the first step. Said another way, if the peak workload of users cannot even log on, then workload transaction response time is moot. The log on or launch or startup user activity may appear simple on the screen; however, this is normally the time when the largest number of processing resources (memory, files, and process threads) are allocated to the user. Complete resource depletion of the application's test operating environment occurs frequently during initial attempts to ramp up active users to peak.

The first time you attempt to ramp up any large number (hundreds) of users, the application is expected to fail before half of the intended users have been initiated. Try the ramp-up in small increments of 5 or 10 users to better isolate the ramp-up failure point. At this time in the development and testing cycle, only a small number (less than five) of users have ever been launched at any time by either the developers or testers. The ramp-up will reveal all kinds of resource problems such as not enough memory, not enough processing threads, or not enough file space.

Once the application has been able to successfully ramp up the peak workload log on without any user activity, the application is tested for successful ramp- up within a peak time window, say 10 min in our example. Again, expect the application to fail several ramp-up attempts within the required time interval. The resource starvation issues will shift from memory, files, and process units to raw CPU speed and communication bandwidth that really have not been taxed fully by slow ramp-up testing at this point. Expect some software under development to never achieve the required peak workload ramp-up in the required time interval due to inadequate performance design. If the peak workload within the time interval is a contractual requirement, then the developers are faced with either a major redesign effort and associated contract penalties or abandonment of the software and loss of the software development contract.

9.3.2 Workload Ramp-down From Peak

Workload ramp-down is the process of gracefully ending all the active user sessions that drove the peak workload. At the end of lunchtime, users log off the system, maybe over a 2–3 min period (more workload analysis information is needed here too).

With peak workload ramp-up successfully achieved, ramp-down testing commences in a similar way. During ramp-up testing, the ramp-down is not an issue. You close down the active users in the test environment any way you can as quickly as you can to clean up and get ready for the next ramp-up test. Now we need to focus on achieving successful ramp-down. As with the ramp-up, a successful ramp-down should be attempted first in increments of 5 or 10. While releasing resources should be faster and less error-prone than acquiring resources, ramp-down errors will be encountered. Continue to ramp up/ramp down in additional user increments. Once gentle ramp-down has been achieved from peak workload, perform the ramp-down again within a required ramp-down time interval if such a requirement exists. Expect more errors to be encountered during the timed ramp-downs.

9.3.3 Performance Measurement at the Peak

Having successfully achieved ramp-up and ramp-down for the peak workload, you are now ready to consider the timing aspects of performance measurement at peak workload. We are going to set aside the workload plans for a moment and gain a perspective on measuring the responsiveness of software. Then, we will marry that measurement perspective with the workload ramp-up/ramp-down testing that we have just completed.

9.4 COMPONENT PERFORMANCE TESTING

The term “performance testing” has come to mean a variety of different software testing techniques where the common characteristic is the measurement of some kind of response time from the software. Performance testing can mean that you measure the response time of some component of the software that is suspected of contributing a large delay in completing a task. An example of performance testing at the component level would be database searches on various criteria in database files of various sizes regardless of what is done with the search results. Another example of performance testing at the component level would be rendering (painting) a screen with results from some processing such as a name search or a product search. To do this kind of performance testing, the software development need not necessarily be finished, but the candidate components must have already passed the tests for functional correctness. A really fast search that provides an incorrect result is not a good performance.

The purpose of doing performance testing on components is to get an early idea of whether the sum of the individual component response times can be expected to come anywhere close to the performance response time maximum we have recorded in our peak workload plans. If, for example, the worst database search response time for our invoice item display is 6 seconds and the worst screen rendering for that item's description is 5 seconds, then you can alert the developer that the current version of the Invoice Item Detail Display code has an 11 seconds total response time. If the business performance requirement for this display is 7 seconds, you have just given the developer an early warning that his or her code as is will not meet the performance requirements.

9.5 ROUND TRIP PERFORMANCE

Other meanings of the term "performance testing" tend to encompass more and more of the application activities to be included in response time measurement. At the far end of the performance testing definition spectrum is the measurement of the application's response time from when the "enter" key or "submit" button is pressed until the results have been fully rendered on the screen or until the report has been printed out or when the product is delivered to your door. Although there are several terms in the literature for this kind of performance testing, we will use the term "round trip performance testing." When we use this term, we will explicitly mean the time that it takes for one transaction or activity to be initiated by the user pressing Enter or Submit until the complete result such as a screen, report, or file is returned to the user. This overall timing will include all intervening processing done on the user's computer, all necessary communications with other support computers via networks, and all secondary processing done on other support computers for a given transaction.

To demonstrate what round trip performance testing might look like, consider putting timing points at various strategic points in the test environment platform of the application such as on the client computer, on the network, and on the server computer. Then, consider running just one purchase step transaction in the empty test environment. For the purposes of discussion, say the completed round trip took a total of 5.8 seconds and was comprised of the following individual timings:

Purchase step transaction round trip performance instrumentation

0.5 seconds	client—purchase screen data input record generation
0.2 seconds	network—transmission of new purchase record to server
2.4 seconds	server—new database purchase record insert into database
1.3 seconds	server—generate warehouse order record
0.7 seconds	server—generate purchase order confirmation record
0.1 seconds	network – transmission of confirmation record to client
0.6 seconds	client – display confirmation record / successful completion
=====	
5.8 seconds	total round trip performance response time in an empty test system

This is less than the 7 seconds maximum in our peak workload plans. The conclusion is that this transaction is ready for workload testing where we know the round trip performance will get slower as the transaction competes for resources in a busier system. Maybe it will stay below the 7 seconds maximum when the peak workload number of purchase steps are executed together, maybe not.

Carrying the thought one step further, it has been found to be very beneficial from a test management perspective to plan for the round trip performance testing of all transactions or representative transactions of all groups individually in an empty system. After recording the results on the peak workload plans, you should alert the developers as to which transactions did not perform under the maximum response time allowed by the software requirements. There is no need to start workload testing of these transactions yet because they already exceed their performance boundaries. In contrast, the transactions that did perform under the maximum response time allowed by the software requirements are ready for workload testing. Draft 4 shows what the extended workload plans might look like with the round trip performance results in an empty system. Round trip performance numbers with a (green) notation indicate that the transaction group on that row is ready for peak workload testing. Round trip performance numbers with a (red) notation indicate that the transaction group on that row is not ready for peak workload testing.

Draft 4 Performance workload plan for weekday browsing

Transaction group	Response time requirement	Round trip performance (red/green)	Peak active users/ customers	Day/time of peak activity
Menu navigation	Max 3 seconds	4.5 seconds (red)	2,000	Mon–Fri 12–1 P.M.
Log on/Log off	Max 3 seconds	2.0 seconds (green)	2,000	Mon–Fri 12–1 P.M.
Product detail display	Max 4 seconds	15.3 seconds (red)	2,000	Mon–Fri 12–1 P.M.
Purchase steps	Max 7 seconds	3.0 seconds (green)	50	Mon–Fri 12–1 P.M.
Catalog search	Max 10 seconds	1.6 seconds (green)	2,000	Mon–Fri 12–1 P.M.
Credit card payment	Max 30 seconds	103 seconds (red)	50	Mon–Fri 12–1 P.M.

Draft 4 Performance workload plan for saturday purchases

Transaction group	response time requirement	Round trip performance (red/green)	Peak active users/ customers	Day/time of peak activity
Menu navigation	Max 3 seconds	4.5 seconds (red)	500	Sat 9 A.M.–11
Log on/Log off	Max 3 seconds	2.0 seconds (green)	500	Sat 9 A.M.–11
Product detail display	Max 4 seconds	15.3 seconds (red)	200	Sat 9 A.M.–11
Purchase steps	Max 7 seconds	3.0 seconds (green)	500	Sat 9 A.M.–11
Catalog search	Max 10 seconds	1.6 seconds (green)	200	Sat 9 A.M.–11
Credit card payment	Max 30 seconds	103 seconds (red)	500	Sat 9 A.M.–11

9.5.1 A Typical Performance Response Curve

Starting with an empty test environment, you should launch more and more copies of the same transaction and watch a predictable pattern of round trip performance emerge under increasingly higher workload conditions. Figure 9.1 illustrates what the plot will look like if you keep adding more transactions.

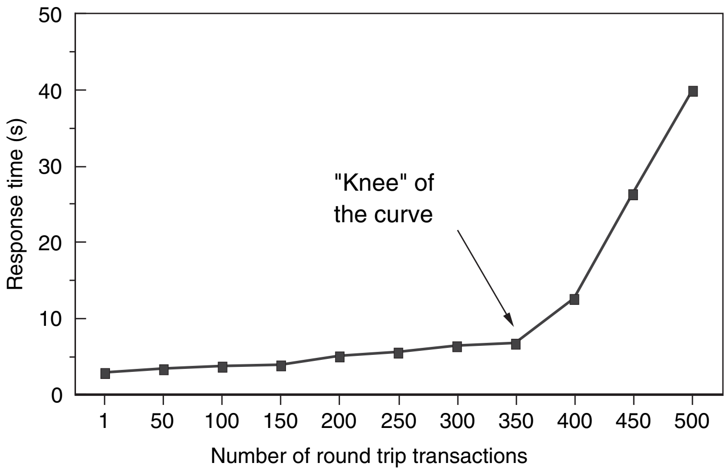

Figure 9.1 Round trip performance for catalog browse transactions

The *x*-axis is the number of transactions active at the same time. The *y*-axis is the slowest response time measured for any of the active transactions. Using either the average response time or the fastest response time for comparison with the performance requirement will, in effect, mask the worst response times. A reasonably well-designed transaction process will exhibit linear performance up to a point, in this case 350 transactions. More than 350 transactions exhibit an exponential response time. The point at which the trend changes from linear to exponential is traditionally called the "knee" of the curve. This curve inflection represents some kind of bottleneck arising in the transaction process path. The plot does not tell you the nature of the bottleneck, just the circumstances. Currently, the only way to discover the location of the knee is to execute the workload and plot the results.

Your first impression of using the performance curve might be that you must push your workload till you encounter the knee of the curve, but that may not be necessary. Consider the plot in Figure 9.2 and our business performance requirement of 10 seconds maximum per catalog browse for a peak workload of 250 active transactions.

Box A represents the peak workload round trip response time of 250 active transactions. At 250 transactions, the worst transaction response time was about

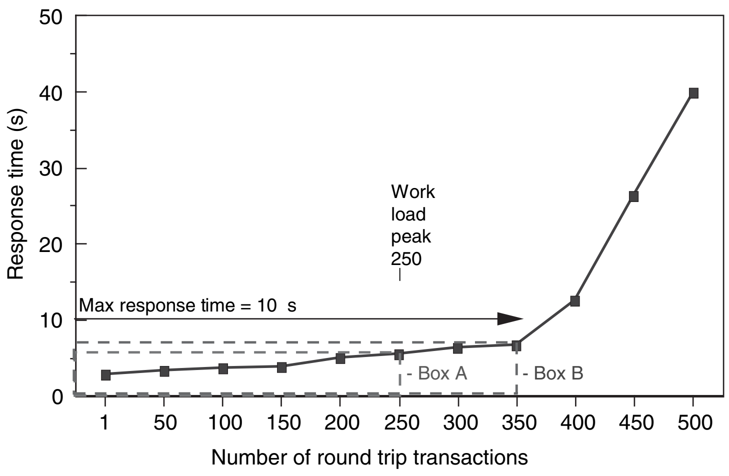

Figure 9.2 Round trip performance for catalog browse transactions

5.6 seconds, well below the 10 seconds maximum response time (arrow). Knowing that the knee of the curve is lurking somewhere out there, you extend your peak workload test to 350 transactions per Box B and see that the plot is still linear and still below the 10 seconds maximum. So you have just discovered that you have at least a 40% margin of safety with the performance requirement without seeing the knee of the curve.

Consider the plot in Figure 9.3 with the same 10 seconds response time requirement maximum for a peak workload of 350 active transactions.

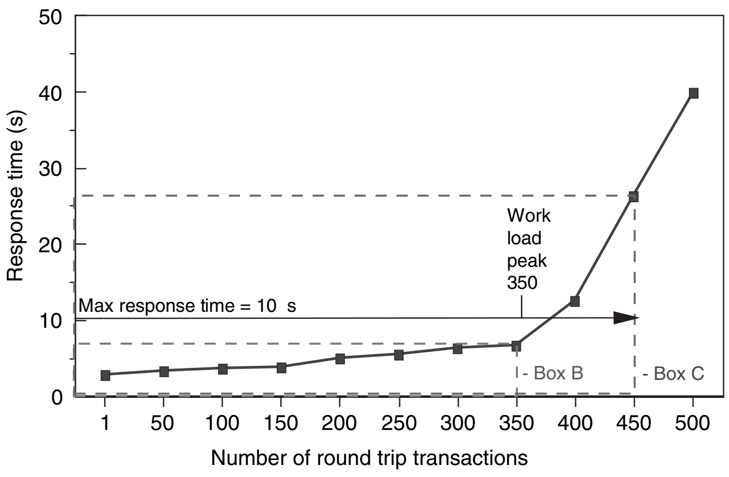

Figure 9.3 Round trip performance for catalog browse transactions

Box B still represents the peak workload response time for 350 active transactions. At the new 350 transaction peak, the worst transaction response time is about 6.8 seconds, still well below the 10 seconds maximum response time (arrow). Knowing the knee of the curve is lurking somewhere out there, you extend your peak workload test to 450 transactions as illustrated with Box C and, in fact, see the knee of the curve push your transaction response time to 26.4 seconds, well above the 10 seconds maximum response. So you have just discovered that there is no margin of safety in the 350 peak workload. As soon as something in the business forces more than 350 transactions active at the same time, the transaction will begin to slow down unacceptably.

What do you do with this information ? You should discuss your findings with the development team. They may want to analyze the code for the apparent response bottleneck, make some programming changes, and attempt to push the knee well to the right by some agreed safety margin. They may want to cap the system just to the left of the knee. Because the knee is 350 transactions in this case, the developer might put a governor on the transaction with a 10% safety margin so that when 315 transactions have become active, the user attempting to submit the 316th transaction will get a message like "The system is responding to the maximum number of catalog browse requests. Please wait a moment and try your request again." Regardless of which alternative the developers choose to implement, you will need to retest their solution to confirm that the changes keep the system off the knee of the performance curve.

Recapping your performance testing activities so far, you have developed the needed peak workload plans; you have successfully ramped your workloads up and down, and you have run your round trip performance tests for each transaction that will be in the peak workload mix. The next step is to run the round trip performance tests with the peak workload mix.

9.5.2 Saturday Peak Workload in an Empty Test System

We return to our performance workload Draft 4 plans to demonstrate what can happen when you test a workload mix. We will focus first on the workload mix for Saturday purchasing. For simplicity of discussion, we will place just three transaction groups in our demonstration: log on/log off, purchase steps, and catalog search. We choose the Saturday workload to mix first because, of the two workload plans, the Saturday plan requires fewer transactions at the testing peak. This is what we know about these three transaction groups for the Saturday workload:

Transaction group	Response time max	Peak workload
Log on/log off	3 seconds	500
Purchase steps	7 seconds	500
Catalog search	10 seconds	200

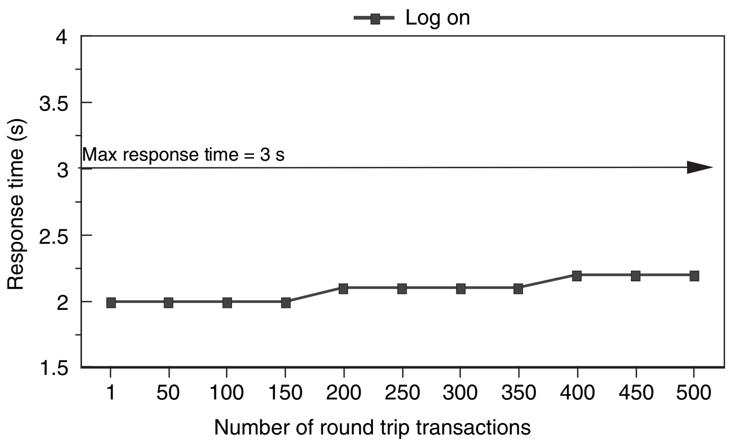

Figure 9.4 Round trip performance for peak logon workload

Next we test each transaction group with its Saturday workload volume. Figure 9.4 shows the first results of testing those transactions individually at peak workload before we mix them into one workload. First we test and confirm that the log on/log off response times are well below the requirement maximum for the Saturday peak workload for that transaction.

Next, in Figure 9.5 we test and confirm that the purchase steps response times are well below the requirement maximum for the Saturday peak workload for that transaction.

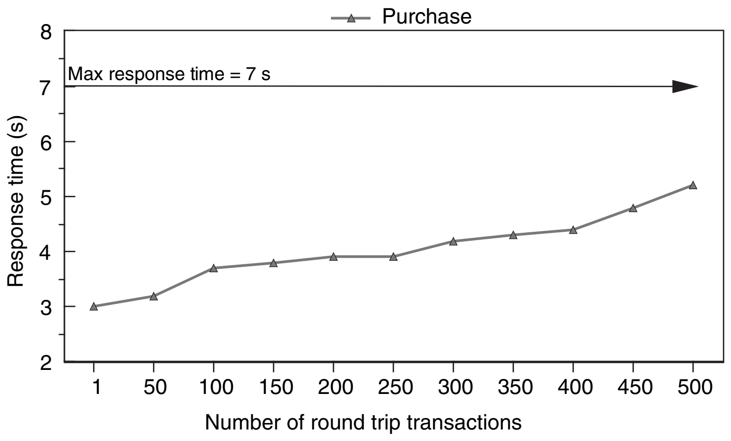

Figure 9.5 Round trip performance for peak purchase workload

Last, in Figure 9.6 we test and confirm that the catalog browse response times are well below the requirement maximum for the Saturday peak workload for that transaction.

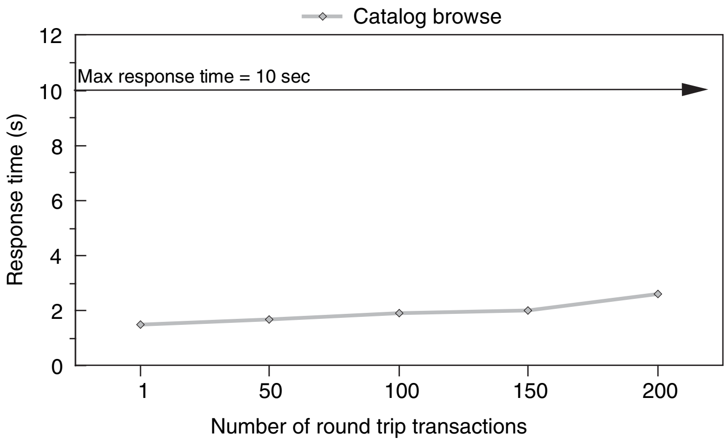

Figure 9.6 Round trip performance for peak catalog workload

9.5.3 Saturday Peak Workload Mix

Because all three transaction groups perform below (faster than) their required response times at peak workload, it is appropriate to attempt a workload mix of the same volume of transactions. Your first thought might be to throw all three transaction groups into the same performance measurement window and see if they continue to demonstrate performance below their requirements. The problem with this approach is that if one or more of the transaction groups' response times go above the requirement maximum, you do not have many clues as to what in the mix changed the performance.

Consider a deliberate ramp-up of each transaction group into the mix in an order that reflects the business priority of the workload. For example, the business priority of the Saturday workload is purchasing. So the suggested transaction group mix sequence for the Saturday workload is (1) log on/log off, (2) purchase steps, and (3) catalog search.

Figure 9.7 shows the results of ramping up the log on/log off to peak workload, then launching purchase steps to peak workload.

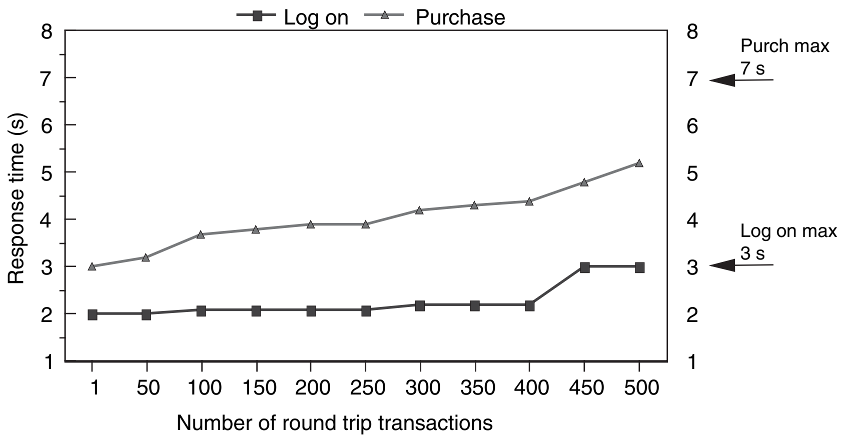

Figure 9.7 Round trip performance for peak logon + purchase workload mix

The purchase step transaction group is behaving well (not slowing down much) throughout the workload. The log on/log off transaction group behaved well until the workload reached 400 users. At that point, the log on/log off performance jumped to the requirement maximum. The performance tester should use this information to alert the developer that the log on performance becomes marginal when the purchase step workload exceeds 400 active transactions.

Figure 9.8 shows the results of ramping up the log on/log off along with launching purchase steps *and* catalog browser transactions to peak workload.

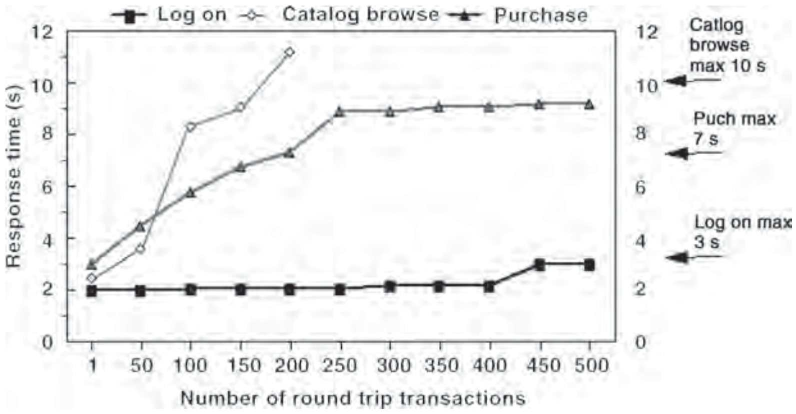

Figure 9.8 Round trip performance for peak logon + purchase + catalog browse workload mix

9.5.4 Saturday Workload Showstoppers

Clearly, there is some kind of resource conflict between the purchase steps and the catalog browse. The performance requirement max for the catalog browse was exceeded when the ramp-up achieved only 100 active users. These results are sufficiently severe as to halt further performance testing of the Saturday workload until the development team diagnoses and implements some performance solutions. This kind of testing discovery is referred to as a “showstopper” from the testing perspective. Furthermore, because we chose the lower volume workload peak to test first, we know that there is no benefit to starting the weekday workload testing that demands an even higher volume of the same poorly performing transactions.

While the performance testers await development solutions to these response time problems, other types of testing such as functional and structural testing can continue. Notice that testers doing the other types of testing will be affected when the developers do find solutions to the response time problems. Specifically, the development solutions to the performance problems will need to be retested (regression testing) to verify that the performance solutions affect only the performance aspect of the software and not inadvertently impair any functional aspects of the software. Regression testing here will add time and effort to the test schedule that may not have been anticipated in the original test execution schedule.

9.5.5 Saturday Workload Showstopper Corrections, We Think

At this point in our discussion, we receive the corrected, functionally regressed software ready for performance workload retesting. We learn from the code correction log and discussions with the developers that the logon code modules experienced interference from the purchase steps at 400 active users because the memory area allocated for user IDs and passwords began to "leak" into the purchase step execution area. We also learn from the same sources that the purchase steps and catalog browse code modules shared a set of utility routines in a common dynamic library that began to degrade the performance of both modules by the way the utility routines were loaded for execution.

Our next step is to rerun the individual transaction groups in an empty test system to verify that the performance fixes did not, in fact, slow down the code. Figures 9.9–9.11 show the results of our empty system ramp-up performance measurements in an empty test system.

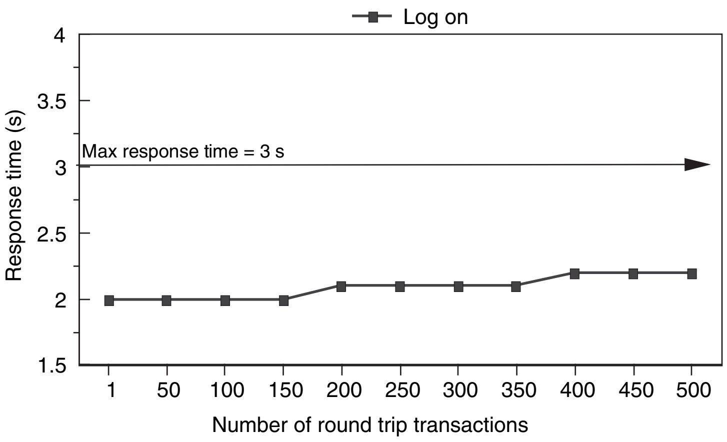

Figure 9.9 Round trip performance for peak logon workload after performance corrections have been applied

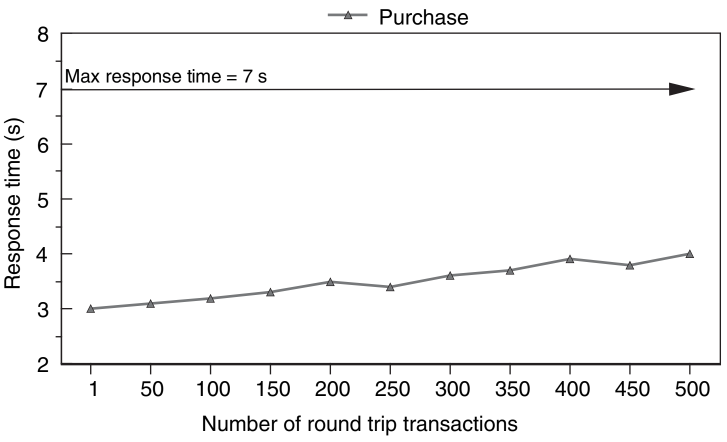

Figure 9.10 Round trip performance for peak purchase workload after performance corrections have been applied

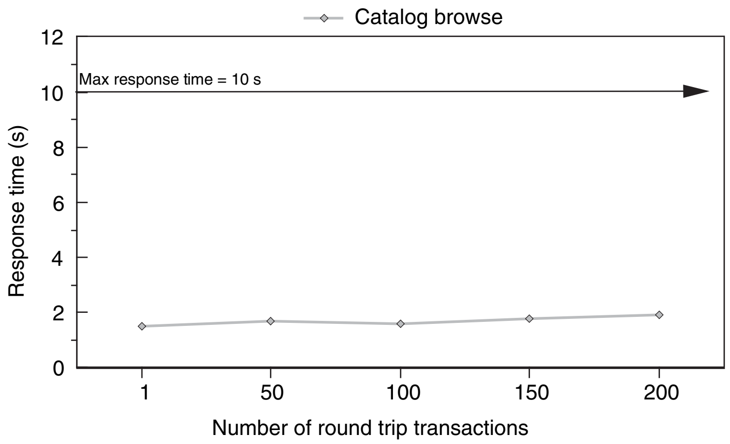

Figure 9.11 Round trip performance for peak catalog browse workload after performance corrections have been applied

The empty system performance of the log on/log off code has not been affected by the code changes. This response pattern is expected because the correction is supposed to make a difference only when other transaction groups are added to the workload mix. The purchase steps code and catalog browse code both show slightly improved performance in empty system workloads. This is a pleasant surprise because the code corrections were focused on cross-transaction performance degradation due to a poor common utility-routine-sharing strategy. The improved sharing strategy also helped each transaction group independently of the workload mix.

Because the individual transaction group workload rerun responses stay well below the requirement maximums, our final step for this workload plan is to rerun the transaction group mixes to verify that the performance fixes did speed up the transactions in competition for computing resources. In Figure 9.12 we see the first results of our transaction mix performance test reruns.

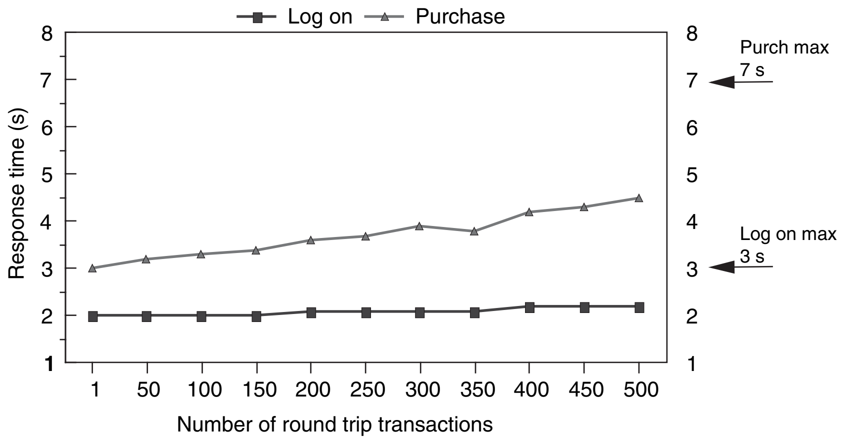

Figure 9.12 Round trip performance for peak logon + purchase workload mix after performance corrections have been applied

We can see that the log on/log off code performance correction did solve the problem. Now log on/log off response is not adversely affected after 400 purchase steps have been launched. Both transaction groups remain well below their requirement maximums.

Because the rerun of the first mix of transaction groups stay well below requirement maximums, we are ready to add catalog browse transaction groups to the mix to verify that the crosstransaction performance fixes did speed up the transactions in competition for computing resources. Figure 9.13 illustrates the results of our second transaction mix performance test rerun.

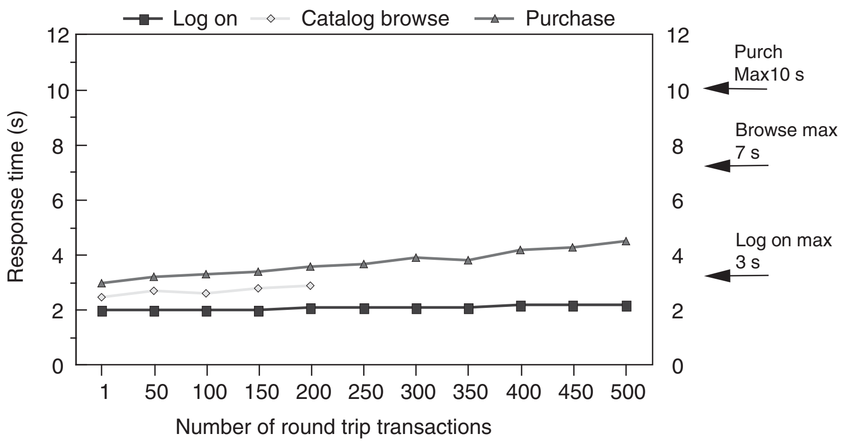

Figure 9.13 Round trip performance for peak logon + purchase + catalog browse workload mix after performance corrections have been applied

With these results, the performance corrections are now verified as lowering all mixed performance response times below the requirement maximums without adversely affecting functionality. The application under test passes the Saturday workload performance test. As more of the application is completed, the performance tests become a part of the regression test to prove that newly added functionality does not inadvertently degrade performance beyond the requirement maximums.

9.5.6 Weekday Workload Performance Plan Execution

You will approach the weekday workload testing in the same way that we performed the Saturday workload testing. Ramp up each transaction group in the workload to confirm that the transaction group response times are below the required maximums before any transaction mixing is attempted. If all transactions pass this first hurdle, then systematic mixing of workloads is attempted until all mixes are successful or until the testing hits a showstopper. The showstopper/correction/regression/retest cycle is repeated until the required performance is achieved. Rather than using more charts to demonstrate what might happen with the weekday workload testing, we will discuss the outcomes you might expect and why you might expect them. The associated charts should leap to mind.

The most noticeable difference between the Saturday and weekday workloads is the volume of work, almost four times more transactions. This would lead you to expect new volume-related issues to be revealed by the weekday performance testing. For example, we learned that the logon/logoff code was designed to operate primarily from lists (active user ID and password) in memory as opposed to lists on disk files. While it is true that everything runs faster in memory than from disk, it is also true that memory usually has a much smaller storage capacity. One possible outcome of this design strategy tradeoff is that the logon/logoff transaction group could become a memory exhaustion showstopper before the 2,000 weekday log on workload is achieved. Buying 2,000 memory chips costing $100 each may be a less desirable solution than revising the logon/logoff performance maximums to a slower response and revising the logon/logoff code to maintain some of its user ID/password information on disk files. The tradeoff faced by the application owner is the cost of additional hardware versus the cost of revising the logon/logoff code and attendant slower disk access performance expectation.

Because we will test fewer purchase step transaction groups during the weekday workload tests than we did during the Saturday workload tests, the purchase steps should rightfully be expected to stay below the performance maximums or at least not exceed them. The catalog browse transaction group is just the opposite situation from the purchase steps. The weekday workload will require many more catalog browse transaction groups than the Saturday workload tests. There is a distinct possibility that the catalog browse plot will reveal a curve knee before its workload peak is achieved. Aided by performance test results, the developers should be able to make the corrections that will ultimately allow the software to pass the weekday workload performance tests as well.

9.6 PUTTING PERFORMANCE TESTING IN PERSPECTIVE

Software performance is the ugly stepchild of software functionality. Both developers and testers spend significant time and effort to ensure that the required software functionality is delivered. Many development teams have discovered that if the software delivers correct functionality too slowly, the end-user is just as dissatisfied with the software as if the functionality were incorrect in the first place. The reason for this end-user attitude is because correct answers delivered too late lose business opportunity just as surely as if the correct answers were never delivered at all.

This chapter describes an approach to performance testing that will alert the development team about software performance shortfalls before the software is delivered to end-users. Although software delivery of business functionality has advanced significantly in the past 30 years, performance delivery continues to be problematic. Expect the first performance tests from any new software to reveal not "if" the software is performing to requirements but "how much" the software is missing the performance requirements.

9.7 SUMMARY

The objective of performance testing is to validate the software "speed" against the business need for "speed" as documented in the software requirements. Software "speed" is generally defined as some combination of response time and workload during peak load times. These peak load times may occur during lunchtime, at the opening of the stock market day, or after midnight when all online customers are in bed (overnight batch workload). Performance testing is achieved by a series of tests that introduces increasingly more workload of increasingly more complex business transaction mixes.

Performance testing occurs after the functional testing is mostly completed and the software has become quite stable (fewer and fewer changes or corrections). Functional defects in the software may be revealed during performance testing, but this is not the testing objective.

The performance tester's first workload challenge is to identify which business transactions and activities need to be measured for performance. In all but the simplest software applications, it is difficult to identify the most important transaction and activity candidates for performance testing. The difficulty lies partly with the software end-user's typical reaction, "everything must respond in less than 3 seconds." Somehow the tester must convince the end user that different groups of transactions and activities will have different performance requirements based on different business priorities.

The performance tester's second workload challenge is to determine peak usage of each group of transactions and the timeframes in which the peak usage occurs. Peak usage is normally measured in terms of active users. The third performance tester's challenge is to clarify just how many different workload peaks need to be tested.

Once you have identified the groups and volumes of transactions for each peak workload test, you need to develop the steps to create this peak in a test environment so that performance measurements can be taken. There are three traditional steps for executing performance test workload. First, you execute workload ramp-up to peak load. Then, you execute performance measurements at the peak. Finally, you execute workload ramp-down from the peak.

KEY TERMS

Response time
Response time requirement
Workload
Workload peak
Peak workload in an empty test system
Peak workload mix
Rule of 8
Ramp up to peak
Ramp down from peak
Performance measurement at the peak
Round trip performance
Response curve
Knee of the curve
Margin of safety
Testing showstopper