

Running the right way

Alan Mouldale, Technical Director and lead LoadRunner trainer at StarBase, warns against underestimating the difficulty of effective performance testing

For the uninitiated there are many pitfalls in performance testing and at StarBase we refer to “*first performance test syndrome*”. Put simply, organisations believe (perhaps because they have been told by sales departments) that performance testing is simple and then discover that it is not.

StarBase is often called in to mount rescue missions for performance tests that have gone wrong. The scene is usually that an application has been deployed and it doesn't scale to the demands placed on it; the business is losing money or face or both. The implementation may have even been backed out. The application has scaled to the anticipated number of users in the test but when put live has failed with a fraction of that number. Often the client is adamant that the performance testing tool is at fault. From experience we know this is unlikely. The real cause is often found by asking one of the following four questions.

Were the right business processes emulated?

It is a fact of life that, unless you have very deep pockets, you can't test every business process that will be executed against your application. You have to be selective, and often this selection is made based on “gut feeling” or because a business process has performed poorly in the past. This introduces a risk; a business process that has performance problems, for example inefficient code, may not be included in the performance test. Only when you release your application will the performance issues with this business process become apparent. These performance issues may be limited to the business process or may impact the whole application.

The good news is this risk is easy to mitigate. Start by listing all of the possible business processes for the application and then select those to be emulated with the following criteria:

- how frequently will the business process be executed?
- what is the anticipated impact on the application's infrastructure when the business process is executed?
- what is the impact to the business if the business process does not perform under load?
- how much will it cost to emulate the business process?

When you are certain you have selected the right business processes, you need to ensure that the actions a user takes to perform the business processes are correctly documented in a *click path*.

Click paths are best developed by observing a real user of the application and noting their interactions (clicks) with the application. Before documenting the click path you should understand the type of user that you wish to emulate. A new or inexperienced user, who is likely to be slower, using drop downs and following wizards, will probably navigate the business process differently than an experienced user, who is likely to be quicker, knowing shortcuts and quick entry codes. These different user types will place a different load on the application.

When you've finished this part, it's useful to list the business processes you *are* going to emulate in a table, including priority (eg high, medium, low) and user type (eg customer, system admin) for each.

Was the right level of load generated?

The business should be able to tell you how many users they are expecting. Be warned, this information could be, and often is, overly optimistic. This would lead to the performance test overstressing your application's infrastructure.

Although overstressing the application is not a problem *per se*, it may lead to unfounded recommendations to upgrade the applications infrastructure to enable it to cope with this excessive load. This in turn could lead to costly and unnecessary hardware and software acquisitions.

Of course the business may be more conservative in its predictions and this could lead to the test not placing enough load on the system, which is a far bigger problem than overstressing it. If you understress your application, you may not expose bottlenecks present in its infrastructure. When you go live these bottlenecks may materialise, leading to unanticipated poor performance. Establishing accurate business volumes is a complex process and involves understanding the number of users that will access the application and the frequency at which business processes will be executed by these users.

To ensure thorough testing you should establish business volumes to emulate several different periods. Common ones include a typical busy hour, so that you can benchmark your application's usual performance characteristics, and a peak hour, so that you can ensure there is acceptable degradation when the application is at its busiest. You may also want to include a seasonal peak, recognising that in most businesses there are periods through the month or year where the application will be run at loads far higher than usual; the Christmas rush for example. When you have calculated your figures, add them to the table.

You are now building a business volumes model your business users will understand, which will ultimately form the basis of your scenario creation.

Was the load generated in the right way?

Most performance testing tools offer record/replay technology; that is you record the actions of a real user into a script and then you replay them. Usually the tools are sold as being as simple as that to use, allowing you to concentrate on performance testing your application. However, in reality this is almost never the case.

We have witnessed many examples of performance tests that have only tested the application's error page. This is because unless the testing tool is explicitly told to do so it will not recognise these unexpected error pages and assume that the correct page was returned.

An example of a worst case scenario is when an emulated user fails to logon (because of, for example, incorrect session information) and then proceeds through the rest of the actions in the business process, each in turn returning an application error page. As far as the tool is concerned the emulated user is running with no apparent error.

This kind of error has a double impact on the accuracy of the performance test:

- skewed timings: the tool is only measuring the time the application took to deliver the error page and not the step of the business process

No	Business process	Priority	User type	Typical load	Peak load
1	Login	High	Customer	150	300
2	Purchase item	High	Customer	100	200
3	Daily archive	Medium	SysAdmin	1	1

- disparate load: the load on the application will not be comparable to the real load as the test is not exercising the whole infrastructure, only the ability of the application to generate an error page

Make no mistake, this is very, very real and some major organisations have got it very, very wrong. It can be avoided by placing context checks in your emulated users. Context checks are tests that fail if the returned data or page didn't contain specific values or text, for example when you place an order, the order confirmation page is displayed.

As a general rule, each interaction with the application should be verified by at least one context check. This could be checking the title of a window or looking for specific data, such as an order number, in the returned page. Whatever context check you choose, it should be unique to the interaction to ensure that it has completed successfully.

Were the test results analysed correctly?

Analysis of performance test results is as much an art as it is a science and it requires experience to find the subtle trends that may be hidden by the vast amounts of collected data. It

is all too easy to look at the headline analysis figures, such as CPU usage and response time, and give your application a clean bill of health while missing warning signs that may be present in the results.

A good example of this is an online retailer who had performance tested a new version of their site. The results showed that throughout their test runs responses were stable and relatively good. In addition the CPU usage on the servers had been consistently low. Based on this analysis they deployed their new application. At first on launch day, the response times were good and the servers were coping well. However as the day progressed the site's performance degraded; so much so that eventually they had to back the new application out. Soon after, the cause of the problem was discovered – a memory leak. Reviewing the client's performance testing results clearly showed that the available memory was reducing throughout the performance test. This had not affected the response times or CPU usage in the test because the amount of available memory did not become critical during the test's 90 minute execution. The problem was there; the performance test had exposed it.

But because the client had only examined the CPU usage and response times they missed the memory leak in the application.

What else could be wrong?

Data is a key problem area for performance tests. If you are not supplying enough varied input data you may be running from cached data rather than exercising the database. For example, if you continue to supply the same search argument to a search engine, the results will be cached. Additionally, the quality of data within the database should be reviewed, especially if the database has been grown or sanitised.

It is worth reviewing the performance counters that are being monitored during the test. The trick here is not to monitor too much in the first instance as this can just obscure the facts. It is best to monitor high level counters for the CPU, memory, disk and network in the initial tests and then in subsequent tests focus in and introduce more counters for the areas that you perceive have problems.

Another major consideration is where in the application's infrastructure the load is generated. During initial application tuning it is best to generate the load directly against the application, excluding load balancers, firewalls etc. After the initial tuning is complete, you can then step back through the infrastructure. If the application wasn't performance tested from outside its firewall or from the Internet backbone, these could exhibit bottlenecks and be the cause of poor performance. PT



We don't need to lay claim to being the biggest, the strongest, the fastest or the most powerful Consultancy around – an example of what our clients have to say is more than enough:

"The key points about StarBase were its approach, the people on site were of the highest quality, and they worked closely with our team" Systems Manager, B&Q

Do you have a passion for quality and want to share our pride in being one of Mercury's Top Tier Business Partners?

We're searching for **talented junior and heavyweight Mercury Interactive Tools Consultants** to join our growing **Performance and Functional Testing teams.**

Would you like to work on exciting and varied Client based projects with a great team of people and enjoy career development that is second to none?

If you've got the talent to impress our clients then we'd really like to talk to you.

Call Samantha Davis on 0208 905 1120 or email your CV to careers@starbase.co.uk or visit www.starbase.co.uk



StarBase are a Mercury Interactive top tier business partner

(No agencies please)