CAPITA

Whitepaper

The challenges of performance testing

In our recent whitepaper "Ensuring Value from Performance Testing" we explored the factors that need to be taken into consideration in deciding when to test. However, there is no escaping the fact that effective performance testing is complex and time consuming. Making a strong case for appropriate testing requires finding a balance between the risk of a potential system failure and the resource required to carry out the testing.

When development is successful and testing does not identify any defects, it can look as if the testing was unnecessary. Whilst it is unlikely that functional testing will complete without detecting a defect, the same is not true in performance testing. Typically, only a few defects may be found and yet the effort required to detect those defects is comparatively greater. It is only when testing is inadequate and an unmitigated risk becomes an issue that the true value of testing is appreciated.

Thankfully the prevailing attitude towards performance testing has shifted to where it is now recognised as both necessary and beneficial. But performance testing still presents a number of distinct challenges that require a structured approach to overcome. If these challenges are not addressed, there is the risk that testing will be inadequate and potential flaws in the system remain hidden.

Challenge 1: Planning

It is still surprisingly common for a project plan to provide only a two-week window for performance testing, with limited or no contingency to fix and retest any detected defects. This reflects a traditional view of testing whereby it is the last activity before go-live. There is little tolerance for the go-live date to move, so testing gets squeezed.

The issue here is that the performance testing was planned without the involvement of a performance test specialist who can assess the performance test requirements of the project and create a realistic estimate for the required activities. This results in insufficient time to conduct effective performance testing, meaning that any effort will be purely on a best endeavour basis and may fail to identify performance risks. The reputational risk of this approach is huge.

The remedy for this challenge is to engage performance test resource as early as possible, allowing the work to be estimated properly and a test plan to be constructed in advance of scripting and execution activities. Early planning will also identify tasks that are not directly related to performance testing but will have an interaction or dependency. This will help keep performance testing off the critical path to production and ensure that required resources are engaged at the appropriate time for the project.

CAPITA

Ideally, a performance specialist should be involved even earlier, at the requirements stage, to make sure that requirements are unambiguous and testable. By articulating how the application should perform in formats that are meaningful for both developers and testers, this specialist can add a useful level of precision and definition. This helps to avoid modelling problems throughout the development lifecycle.

Challenge 2: Modelling

The objective of performance testing is to evaluate what performance will be in production and, therefore, the test environment needs to have a similar capability to the production environment. The challenge is to create a realistic model of the environment; one that takes into account both the types of transaction that will be carried out and the volume of likely transactions.

It is not possible (or realistic) to script everything that users might do on a system, and so testing often falls back onto scaling models, which apply an estimate of the difference between the production and testing environments to the test results. These are difficult ratios to calculate, with the added pitfall that environments do not scale evenly. This leads to the risk that an issue in one area goes undetected because testing did not stress that particular component.

To avoid this issue, we recommend a structured approach towards identifying and prioritising the various factors that will affect how an application performs. This begins with gathering input from a number of key areas within the business to understand how they need the application to work. Then, a detailed analysis of all the requirements and the impact of potential failures are used to create a weighted set of priorities for testing. In this way, the test results will be approximately correct and realistic as opposed to precisely wrong.

Challenge 3: Data

Related to the modelling challenge is the issue of test data, where a performance test involves processing a high volume of transactions in a short period of time.

To give an example of the quantity of test data required: an authorisation system running at 10 transactions per second will consume 36,000 authorisation requests in a typical one hour test. If each of those is required to be unique in some way then it is a considerable task to create sufficient 'dynamic' data to drive the test.

This data creation issue is compounded when we consider 'static' data held within the system. Suppose each authorisation request in this above example relates to an individual user and the test requires each individual user to have an account set up in the target system. Having insufficient 'static' user account data within the system will falsely improve test results, because the load upon the system will be reduced and the time taken to search through user records will be unrealistically short.

In effect, creating 36,000 user accounts can become a performance test in itself. This can be further complicated if the process of registering a user requires confirmation via a link sent over email, or if the part of the system that handles requests for new users is only designed to process 100 users per hour – meaning data creation could take 360 hours to be able to run a one hour test.

One solution to this can be to use production data, if it exists, but this has challenges in itself. Often this data is covered by the Data Protection Act, and so it has to be obfuscated. Too simple an obfuscation algorithm can favourably improve search results, giving a false positive result under test conditions that cannot be replicated in production.

A further complication can be introduced via caching, whereby the system remembers recent results to speed up providing the same information again in the short term.
If insufficient 'dynamic' data is used to drive the test then another false positive result is possible, as the search results quickly become cached unrealistically. The aim should be to keep the cache at a reasonable level so that it is returning as little data as possible.

To address these challenges, performance testing specialists use a variety of tools that are available for creating data models. These tools, combined with breadth of experience, can streamline the process so that it need not drain resources and instead becomes another stepping stone to success.

Challenge 4: Tooling

The ability to mimic hundreds of end users requires specialist load test tools and a specialist to operate them, both of which can be expensive. Free tools are available as alternatives but these tend to require considerably more effort to script and, as these tools command a considerably lower market share, performance test specialists skilled in using those tools are rare.

Challenge 5: Security

The final challenge is that of security which, like test data, has many facets. Performance testers need to conduct a realistic test which means including security. However that same security can greatly inhibit the ability to test.

For instance, performance test scripts require the process they are mimicking to be repeatable. The presence of a Captcha or other mechanism to ensure human interaction makes the scripting process significantly more complex.

Similarly, the test environment should be similar to the production environment, including access privileges. This restricts the performance tester's access which, at a simple level, can restrict the tester from monitoring the target application or system whilst under load. This lack of monitoring makes it harder to assess how the target application or system performed and also makes it harder to identify any performance bottlenecks.

At a more complex level, security can also prevent server access, which makes it difficult for the tester to install tools or use robot-created accounts. These challenges are not insurmountable, but they require time, effort and skill to address.

CAPITA

Whitepaper

Conclusion

Performance testing delivers important benefits:

- It reduces the risk of losing customers or revenue owing to badly performing software or applications
- It reduces the likelihood of costly reworks
- It ensures the best configuration of software, reducing the need to spend budget on more costly hardware
- It allows you to make comparisons between releases to understand performance trends and plan for future architecture, infrastructure and capacity
- It can help you meet your regulatory requirements

Performance testing is a challenging discipline. At Capita, we have considerable experience working with clients to deliver successful performance test engagements.

To help you address the challenges of performance testing, we recommend:

- Planning performance testing at the earliest opportunity to ensure useful results and prevent critical path disruption
- Budgeting for performance testing and for the specialist resource to support it
- Ensuring you have a suitable test environment to support performance testing
- Using appropriate load test tools and the expertise to use them
- Allowing sufficient time for data creation
- Providing performance testers with access to monitor the environment

Author

Alistair Gerrard, Head of Automation and Performance

For further information about these services please contact:

Email: **marketing.itps@capita.co.uk**
Tel: **+44 (0) 8456 077466**
www.capita.co.uk/itprofessionalservices