THE BENEFITS OF SHIFT-LEFT PERFORMANCE TESTING

Performance issues discovered in production can lead to larger issues down the road, where resolution can be expensive and time-consuming. They can also negatively impact customer satisfaction due to slow-responding requests. The good news is that a shift left to involve development teams in performance testing can help alleviate these issues.

Imagine this scenario: A production issue has occurred that has direct impact on customer response time. The issue needs to be addressed by development. This is a very time-consuming and high-cost activity that impacts current development activities and in turn slows innovation.

Could this performance issue be prevented? A staggering 86 percent of participants in a recent webinar stated that they were aware of performance issues that came back to get resolved by development. Why does this continue to happen? Can shift-left testing relieve these scenarios?

Monitoring Performance

Performance deals with resource use and availability. Failure to take performance into account when introducing code changes invites the risk of inefficient processing, which can lead to lower quality, increased costs, slower innovation, and unhappy customers. Performance issues can lead to dissatisfied customers who may decide to take their business somewhere else—and once you've lost a customer, it is very difficult to get them back. And don't forget that the time spent resolving these issues in production, rather than earlier in the software delivery lifecycle (SDLC), decreases time developers could be innovating, giving your competitors a frustrating advantage. Shift-left

testing helps avoid these issues by pushing more testing activities to development, thus decreasing the chance of performance issues getting deployed into production.

Collaboration with Operations

Most developers have little insight into what happens to code once it is deployed. To maximize the effectiveness of performance tests and ensure that testing is producing useful data, developers should be empowered with the knowledge of how their code interacts with assets already in production.

Performance analysts should share experiences with developers so they can become acutely aware of what the applications do, and operations teams should explain batch jobs that compete for resources for online transactions. For example, in order to avoid increases in cost dictated by service level agreements (SLAs), a set of long-running jobs that approach the batch window limit could be identified and performance tested after code changes. Candidates may include batch jobs with high CPU, high I/O, high usage count of IBM® CICS® transaction, or high usage of IBM® Db2® SQL.

Once you've determined the candidates for performance testing, a downsized version of the test should be prepared. You certainly don't want to be executing full-blown performance tests on a database with 20 billion rows, as this would be expensive and limit-excessive. You need a downsized version that is small enough to run in a short timeframe but still able to catch resource changes from run to run. Additionally, a baseline should be established against which you can compare metrics such as CPU time, MSU, wait time, average service time, etc. to spot performance issues.

Automation

Shift left and automation go hand in hand. By automating the execution of tests, you now have a process in place to watch for performance issues introduced by code changes and address them while the developer is still familiar with these changes. This will increase the quality and efficiency of the changes and prevent resource-draining code from being promoted into production.

Note that it is not just code changes that can affect performance. There are system changes that could affect results. Whenever system software or hardware changes are made, the performance test suite should be executed. I have seen significant positive changes just from hardware upgrades. Be sure to establish a new baseline for your performance tests after any such change.

For example, BMC AMI Strobe now supports performance testing of 64-bit applications, enabling teams to estimate potential improvements in performance before transitioning programs to 64-bit. Performance in production can then be tested to verify that the expected improvements have been realized.

In summary, a shift left of performance tests combined with automation will result in the following benefits:

- Higher velocity
- Higher quality
- Happy customers
- Faster innovation
- Reduced costs

Automating performance testing and shifting left allow you to pinpoint inefficiencies earlier in the process and reduce the time spent on identifying and correcting issues that would otherwise be found later in the SDLC. With better results produced more quickly, you'll save money, pave the way for your next great innovation, and most importantly, satisfy your customers.