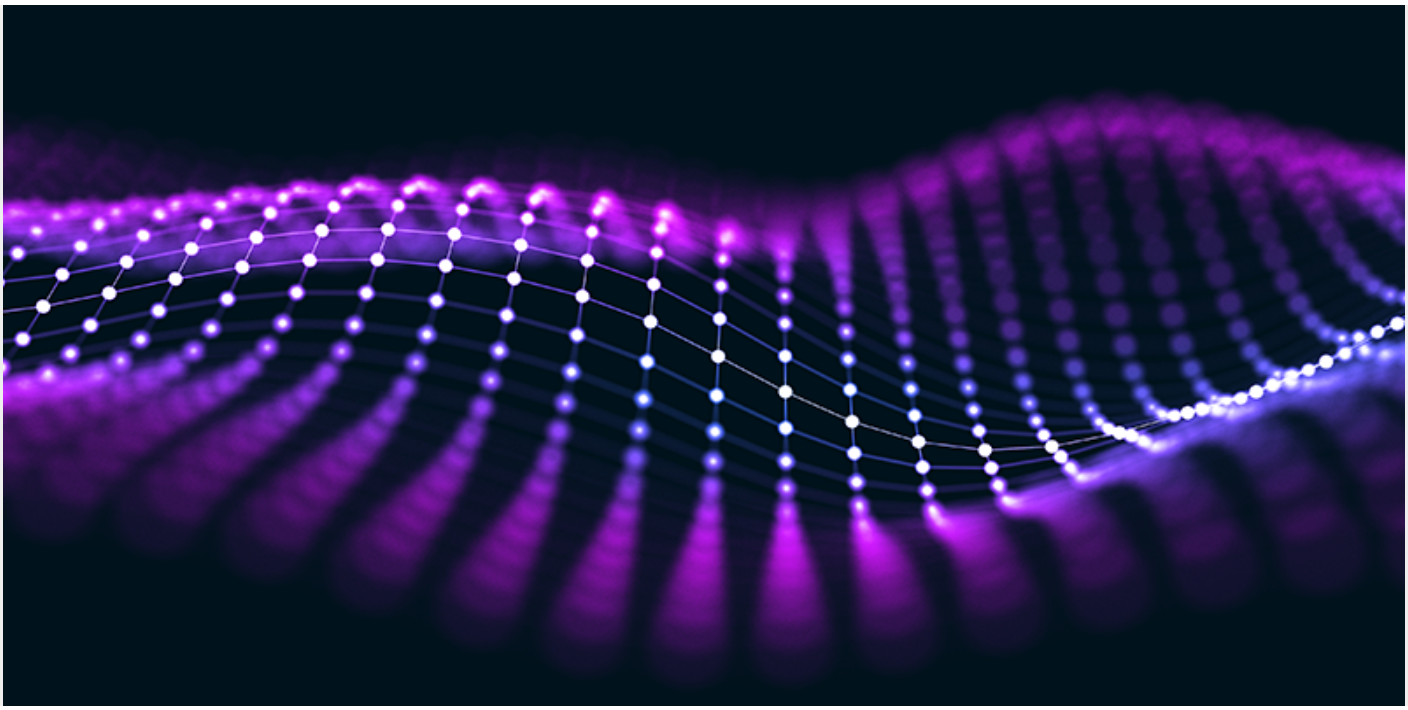
KEEP PERFORMANCE ISSUES FROM SLOWING YOUR INNOVATION



Overview: Shifting left in performance testing helps to identify issues before they make it into production, where it is more expensive and time-consuming to fix them. Automatically testing performance as changes are made improves the velocity and quality of code, in addition to giving your developers more time to innovate and provide value to your customers.

Imagine the following: You've executed many unit/functional/integration tests and proved the application logic works. The resultant output data and reports are correct. You're confident that you're ready for production deployment.

But what about the performance? What about the response time the end-user may experience?

In a recent internal survey of webcast viewers, less than 3% of developers indicated they believe there are no performance issues in production that require attention from their development group. A staggering 86% are aware of issues that came back to get resolved by development. Why does this continue to be an issue?

Monitoring Performance

Performance deals with resource use and availability. If performance is not taken into consideration before code is released to production you add the risk of inefficient processing, which can lead to

unhappy customers who may decide to take their business somewhere else. Once you've lost a customer, it is very difficult to get them back. Plus, the time required for development to resolve performance issues in production takes away from more innovation. This can allow the competition to pull away from you because you are busy resolving issues that should have been caught earlier in the software delivery lifecycle (SDLC). The effect of all this is lower quality, increased costs, slower innovation and unhappy customers.

Shift-left testing pushes more activities to Development resulting in fewer issues getting deployed into production. But shifting left is not enough. Performance testing should be automated so your developers can work on code versus contending with the drudgery of manually setting up performance tests. Automating shift left performance testing also enables developers to get critical feedback on all aspects of their code earlier in the development phase, leading to higher quality applications and better customer experiences.

Collaboration with Operations

Most developers have little insight into what happens to the 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 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, a set of long-running jobs that approach the batch window limit could be identified and performance tested after code changes. Candidates may include high CPU batch jobs, high I/O batch jobs, high usage count of CICS transaction, and high usage of Db2 SQL.

Once you've determined these 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. This would be expensive and limit-excessive. You need a downsized version that is small enough to run in a short timeframe but still be 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

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 these 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 occur, 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.

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. The resulting increase in quality and efficiency will lead to reduced costs, increased innovation, and, most importantly, satisfied customers.