BlazeMeter

LOAD TESTING AND APM

Accelerating Performance Tuning and Troubleshooting

Table of Contents

INTRODUCTION

It's November – and Trip2Trop*, a travel agency specializing in Caribbean vacations, has a big problem. After months of development, weeks of testing, a carefully scaled-up infrastructure, and a successful controlled launch to hundreds of users, the company's new customer-facing travel-planning mobile application has stumbled badly in its first days. Performance bottlenecks have cropped up to bring the application to its knees. Instead of easily booking trips to tropical destinations and checking reservations and confirmations, customers are encountering unacceptable delays and crashes as the app slows to a crawl. They're angry and Trip2Trop's revenue is in jeopardy.

Even if users are experiencing only minor delays, Trip2Trop's revenue could still be in jeopardy because customers bring high expectations. Today, users are increasingly intolerant of sluggish app behavior, especially as our connected lives evolve beyond traditional computers and smartphones. Apps are everywhere, and we expect them to provide real-time responsiveness. These demands have put new focus on load and performance testing. And, as the lines between development and production continue to blur (and in some cases, get obliterated), a comprehensive approach to application performance management in the DevOps era requires rethinking the outmoded load testing approaches of the past.

PERFORMANCE TESTING: FROM THE OUTSIDE-IN

Courtney and Kevin from Trip2Trop's IT team are assigned to get to the bottom of the problem – ASAP – before the Christmas selling season kicks in. Fortunately for Trip2Trop, each of them brings their own complementary perspectives to the challenge.

For Courtney, the app's performance is a matter of load. Load tests send virtual traffic to the back-end application that emulate client requests, such as a GET to the home page or a POST to "Add to Shopping Cart" page of the site.

Using virtual loads, strategies, and sophisticated load-testing tools, she can gradually scale up the number of simultaneous users to find tipping points and inflection points that show where performance degrades and at what volumes. She measures response times for the requests, the number of requests per second (throughput), the number of concurrent user sessions, and the number or percentage of errors coming back from the app.

It's an outside-in view of the application from the user's perspective, one that gives us the important real-world user experience.

For example, load testing can show that app launches will be unacceptably slow at scale, because the back-end response time for logging in and returning current promotions to the app goes from less than half a second under no load to more than 10 seconds when 1,000 users are logging in at once.

APPLICATION PERFORMANCE MANAGEMENT: FROM THE INSIDE-OUT

Kevin, however, approaches the challenge from a different perspective. Since most applications are becoming highly fragmented with numerous changing components, there are many potential sources of trouble. He views things from the inside-out using application performance management (APM) tools and strategies. With APM, developers can inspect the app stack to zero in on the root causes of the performance issues uncovered by Courtney's performance testing.

APM tools run inside the application stack, typically loading as a near-zero-footprint agent that injects a tiny amount of code into the running app to measure all of its behaviors. For example, a GET request to the home page will trigger the execution of code on the app server. APM agents record each method or function that's executed and the time spent running that code. APM agents also record code that calls other resources, such as data stores and caching servers and service calls to other apps.

While load testing tells us what conditions slow the application down, APM tells us why it's slow. It shows us, for instance, that the user login servlet calls a method that retrieves user-specific information from the Oracle database and the SQL statement is waiting for a free connection in the JDBC Connection Pool. That's information Kevin and Courtney can relay back to the team so that they know how to fix and optimize the app.

These APM issues won't appear during development and functional testing cycles. They only appear at scale. They require the ability to emulate real-world load in a controlled and repeatable way. That's why savvy development teams recognize that load testing and APM go together well in an optimized, complementary fashion that accelerate tuning and troubleshooting without waiting for real customers to encounter hidden bottlenecks.

CONTINUOUS DELIVERY

Today, in an era where app lifecycles are sometimes measured in hours, not weeks, many development teams have embraced some form of continuous delivery. The ability to deliver high-quality features quickly is making this an increasingly popular discipline.

In continuous delivery processes, the lines between traditional stages of the application lifecycle aren't just blurred – they're obliterated as code is written and tested in iterative, overlapping schedules. While some apps can require manual gatekeepers to promote builds to releases, most can actually be automated as long as the tests are complete and accurate. Automated tests help prevent buggy, vulnerable, or sluggish software from being pushed into production. Testing is the engine that makes continuous delivery possible.

Until recently, load testing was generally thought of as something that should be done only occasionally. The classic scene for many dev teams is the "war room" frenzy of load testing for key releases or in advance of big events or product launches. While this kind of testing still has its place, episodic testing prevents the team from achieving the benefits of fast feedback and rapid, iterative testing that is the raison d'être for continuous integration and delivery.

With solutions like BlazeMeter , it's now feasible to treat performance testing as we do other kinds of tests: running them locally with every commit and build in CI, and in staging or rehearsal environments – even in production. And we can do this from the cloud or from on-premises locations for testing behind the firewall.

LOAD TESTING AND APM

The heart of any good load test is the ability to find where constraints, bottlenecks, and other performance slowdowns are appearing in the underlying application or infrastructure. Secondarily, load testing also shows the correlations between those slowdowns and the fluctuating levels of traffic being sent to the app.

In this context, BlazeMeter drives synthetic requests – tens or hundreds of thousands of virtual users – to the app. As the user volume grows, BlazeMeter can measure response times, throughput, latency, and other key performance indicators from the client perspective. This shows us when the user starts to experience slowdowns, when performance starts to degrade, what the load conditions are when that slowdown occurs, and other indicators of that bottleneck.

For example, we might see that a product lookup transaction in a retail app exceeds its SLA expectation. That would require the development team to investigate further and learn exactly what is causing the problem. From there, we can leverage the deep inspection capabilities provided by **AppDynamics**, **New Relic** and **Dynatrace** to isolate the exact issue. Perhaps it's the call chain for a slow method or a database call that's holding things up. In addition, APM solutions expose user journey patterns, which in turn, help to design and prioritize better test plans. Used together, load testing and APM accelerate the journey to higher performing apps and infrastructure.

TEST TIPS

Testing should begin “at your desk.” To make this easier, familiar, and more convenient, coders can now create and edit performance tests using a straightforward text-based syntax that closely resembles the DevOps “infrastructure as code” design pattern. BlazeMeter’s YAML and JSON performance test language keeps DevOps teams in their preferred editor, minimizing cognitive context-switching. Most importantly, putting performance testing capabilities in developers’ hands gives them immediate feedback about the performance of new code and a baseline for performance as features are enhanced over time.

When that code is committed and built, the text-based performance test specifications, along with other relevant tests, can be automatically merged to create more complex load scenarios that further exercise the overall release candidate. BlazeMeter’s API and command-line tools further simplify this process and adapt to nearly any workflow and toolchain.

CONCLUSION

Given ever-rising user expectations for performance and responsiveness, website and mobile-app owners must make further commitments to understand their site's performance from both load-testing and application performance management (APM) perspectives. From both outside-in and inside-out, these complementary disciplines can unlock the causes of bottlenecks and zero-in on root causes.

As development teams move beyond episodic development paradigms and embrace continuous development methodologies where they create code in rapid-turn iterative cycles, testing is shifting from a periodic phase to a near-constant process at every commit and build. BlazeMeter and its APM partners, AppDynamics, New Relic, and Dynatrace, have recognized and responded to this new environment by identifying when performance breaks down and why it breaks down – the invaluable insights developers need to ensure the fastest-performing, highest-quality code.

If you want to learn more about how performance testing and APM can work so easily together, view the webcast recording, **Load Test Like a Pro**.

Ready to get started? **Create your free BlazeMeter account today.**

ABOUT BLAZEMETER

BlazeMeter is the leader in continuous performance testing for the continuous delivery era. We ensure the delivery of high-performance software by enabling DevOps teams to quickly and easily run open-source based performance tests against any mobile app, website or API at massive scale to validate performance at every stage of software delivery.

BlazeMeter recognizes that performance testing has shifted dramatically, and has introduced "Performance-Testing-as-Code" to make load and performance testing a natural part of every software delivery workflow. BlazeMeter tests websites, mobile applications and APIs under real-world conditions and behaviors. We do this through open source tools that work across platforms, vendors and tech stacks, such as Taurus, JMeter, Gatling and Jenkins, so the introduction of open source load testing into the enterprise is as smooth as possible. BlazeMeter is a centralized environment where all team members across the board – not just performance engineers – have what they need to design, run and analyze tests and communicate clearly with each other.