Load & Performance Testing in Production

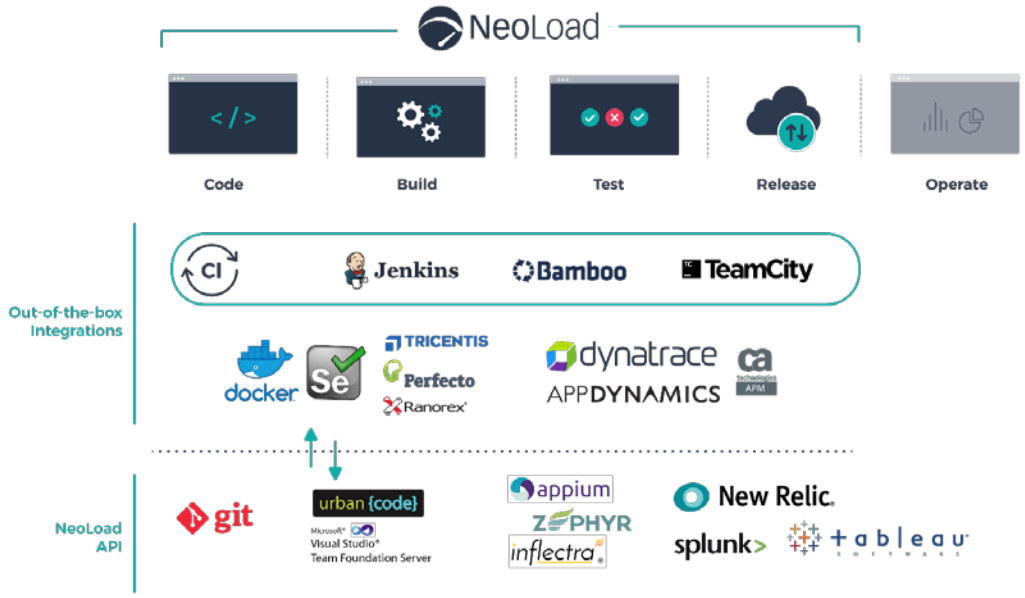
A Neotys Whitepaper

Table of Contents

Introduction

One of the biggest challenges testers face is establishing and managing the testing ecosystem. Although most load and performance testing activities are conducted in pre-production or QA environments, seasoned testers accept an exasperating truth—no test lab can completely emulate a production environment. The reasons for this are many. Securing budget to purchase identical production hardware for testing is often untenable. The process of cloning production data, server hardware, and configuration into the test lab is almost always resource prohibitive. Simulating the interactions between hardware, power supplies and networks is unrealistic.

The reality is that testers can only approximate their production environments by setting up testing ecosystems comprised of a complex assortment of best-of-breed solutions stitched together to form a coherent testing ecosystem. It is the role of the performance engineer to overcome the limitations of the test lab and deliver as high quality an application as possible. Performance teams understand that there is a need for both preproduction and production tests. In fact, there is a class of tests that simply do not make sense to execute in any environment other than production.

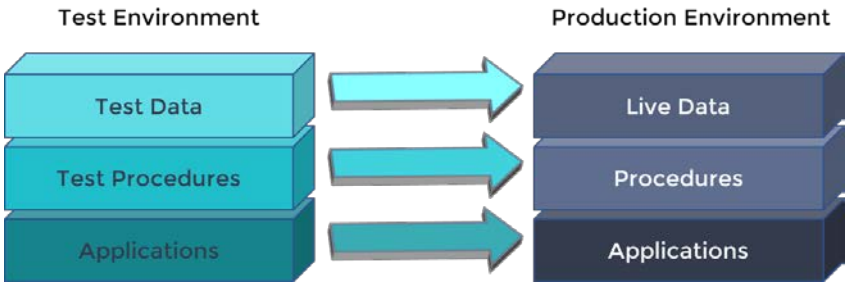


Differences between test environments and production environments can span a multitude of factors that make testing in a QA environment entirely suboptimal. Such differences include:

- Number of servers
- Type of network equipment
- Integration with third party tools
- Utilization of Content Delivery Network (CDN)

- Bandwidth utilization by other users or applications
- Latency between systems within respective environments
- Performance variances that result from different database sizes
- Load balancer performance issues

The list above provides only a partial sampling of the variables that may be entirely absent from the pre-production environment. Test environments may lack support for many of the newer, emerging technologies, protocols and platforms present in production. Consider applications that exist in the Internet of Things ecosystem. These ecosystems are incredibly complex with many moving parts; are fueled by innovation and change; and swirl in a vortex of proprietary approaches, technologies, and protocols. For these reasons, it's important to remember that when QA tests applications in pre-production, testers can't test against real-world load patterns and real-world data. They must make educated guesses, guided by experience, and by the testing tools themselves.
Furthermore, many applications utilize third-party integrations for things like extending marketing capabilities, user tracking, social media connections, etc. These integrations, however, are often excluded in the load testing approach, because they are not deployed in testing environments.
Testing environments are structured with fewer servers and less hardware; it's left to the performance engineers to determine a ratio of performance between the testing and production environment. At first glance it might appear easy to approximate a meaningful ratio if both environments use the same types of machines with the only difference being a reduced number of servers per layer. We disagree.



Determining an accurate performance ratio is both hard and risky. We've already established that a 1:1 ratio between environments is unrealistic and cost prohibitive. So how do testers approach the math? What if the CPU capability in the test environment is half of that available in production? How can testers account for load balancing in production when it is absent in the test environment? As the performance ratio between the environments increases, risk increases, because the accuracy of the benchmarking is diluted.

In this context, the performance engineer could be tempted to proceed with load testing followed by an effort to extrapolate the results for the production environment. Unfortunately, extrapolation is usually an unrealistic approach due to the notable differences between the environments. For example, testers may assume that 50% of the system resources can support 50% of the actual production load. The assumption itself is risky, because it presumes that everything scales in a linear fashion. How realistic is that?

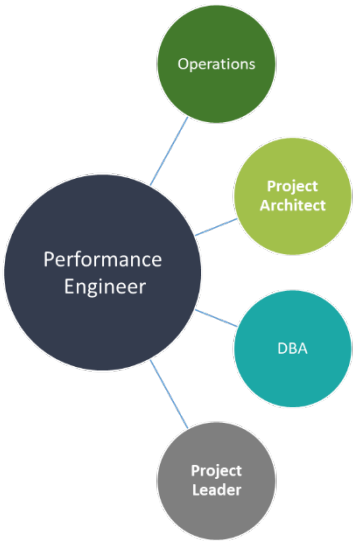
Given these challenges with the testing environment itself, more companies and consultants are Testing in Production (TiP). Although TiP provides a solution to the environment problem, it requires that testers adhere to best practices to test successfully while managing the many risks associated with this approach.

The Challenges of Testing in Production

Launching a Test without Knowing the Consequences

Measuring the performance of applications tested only in the production environment is very risky. To avoid costly performance issues, we recommend testing in the QA environment as well. In many companies, launching a load test in the production environment without having any idea of the released application's level of performance is tantamount to a crime against productivity. Depending on the results, it could also be disastrous.

Even when the application is tested against a testing environment, TiP requires several teams to augment the performance engineer and necessitates the participation of:



- Operations - To prepare for the test, this team can disconnect from the current production environment. They will alert the performance engineer of any impacts of tests on the rest of the datacenter.
- Project Architect or Technical Leader - This person should be able to identify potential issues by looking at the logs or the database activity.
- DBA - The DBA is responsible for finding blocking points on the database and connecting to/disconnecting from the production database
- Project Leader -
    - Defines the timeslots for load testing operations
    - Alerts the user of the potential disruptions
    - Defines the process of disconnecting from and reconnecting to the production environment

Teams must be prepared to react immediately, depending on the impact the test has on the production environment.

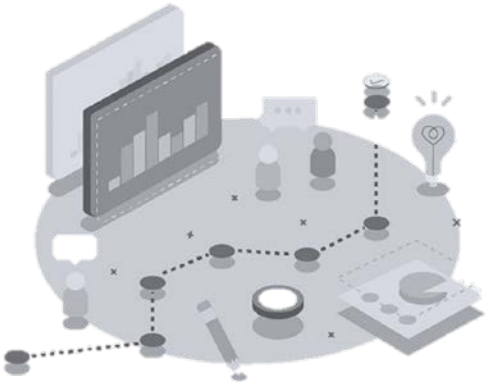
Testing in Production Can Affect Real Users

We all agree that load testing in production can affect users or business processes already working in the environment. Of course, the impact is linked to the main test objectives. For example, there would be greater impact on real users with a limit test than with a constant load model. Whatever the test objectives, the potential impact must be considered.

Many studies demonstrate a correlation between user experience and profit (depending on the business area and the application). It's generally agreed that generating load during business hours endangers the:

- User experience
- Business opportunity for revenue capture
- Brand image of the company

Performance engineers must manipulate the load they apply to an environment. However, real users accessing the application during these tests add noise to the results, obfuscating performance test analysis and interpretation. With this mix of real and virtual traffic, it can be difficult to identify the root cause of the performance issue as it could be caused by:



- The load applied by the test
- The business process called by the production traffic
- A combination of both

Excessive noise in the environment will only make understanding the performance testing results that much more difficult. To avoid such issues, run load tests during low traffic hours or after deploying a new release of the application.

Generating Load on a Third-party Application Can Be a Legal Issue

Generating load on an application involving a third-party can indirectly generate load on that partner's environment. Legally, a testing project is not allowed to generate load on a third-party website without informing said party. Even then, the third-party can block or black list the traffic during the test, generating errors that affect load testing objectives.

Therefore, most testers remove requests that direct to the third-party. Keep in mind, this workaround will slightly alter all response times retrieved during the test.

Depending on how the web page is built, the usage of a third-party can slow page rendering. To work-around this issue, testers can emulate the response of the partner using service virtualization. Although this is technically the best solution, it can be expensive to implement and require extensive changes to the production environment. Therefore, it may be easier to remove all the requests pointing to third party websites.

Creation of Testing Data on the Production Environment

Load testing usually requires a large dataset to generate representative traffic (i.e., login, products, etc.) and there are often challenges around generating this data for use in a production environment. Some business transactions will generate data in the back-office systems of the company. If we look at an e-commerce website, validating orders can feed the back office with testing data and could connect to/from back office services of the company.

To utilize test data in the production environment:

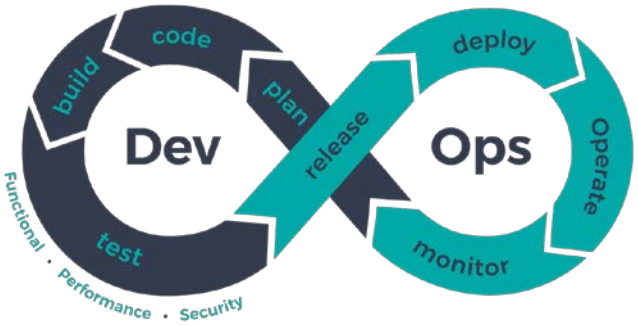
- Disconnect from production and connect to a testing database instead. (This is possible when the environment is not connected to several applications).
- Create a specific testing account in production that is dedicated to testing. Note that this can

sometimes be difficult and may not even be possible in the production environment.
- Avoid test steps that generate records in the back office (i.e., avoid validating a test order.)

Best Practices

Even when testers possess an in-depth understanding of their production environments, the unpredictability of real-world user behavior can introduce surprises and unwelcome ramifications in production. Assuming that certain application facets are thoroughly tested when they are not can wreak havoc in production. To avoid such pitfalls, we recommend that QA teams leverage team-based and process-based best practice approaches to their performance tests.

Applying best practices to TiP offers many benefits. Obviously, it promotes collaboration between DevOps teams by elevating the frequency and discussion of topics relating to code quality and testing strategies, encouraging development and testing teams to interact and communicate regularly. By sharing insights and proven techniques, these discussions help to alleviate the pressures of "going it alone"—those cultural stressors found in many companies that press developers into testing code they wrote "just to see if it works." The philosophies around "you wrote it, you test it" can circumvent the need to share collective wisdom, because those new to development may lack the experience to apply readily available and proven testing strategies. These situations increase the risk that thorough testing isn't applied to every facet of the application.



With that said, in addition to applying team-based best practices, how can testing teams apply process-based best practices to their testing strategy and mitigate the risk of testing in production?

Select the Right Time Window to Apply the Load

When running tests on the production environment, testers should designate timeslots that impact fewer real users. For example, test:

- During night hours (i.e., 11PM – 5AM ET)
- After deploying a new release
- During maintenance hours

This method provides a very short time window during which performance engineers can conduct all their tests.

Monitor Infrastructure and Have the Proper Team on Standby

Testing in production requires constant monitoring of the entire architecture. Performance engineers must have a clear status on the health of the production environment to:

- Stop the test and avoid major production issues
- Correlate and identify bottlenecks in the application

Moreover, setting up proper monitoring won't be enough to secure the test in production; specific resources are required to make the right decisions. Such resources encompass compute/CPU cores, storage at all tiers, networking bandwidth, and so on. The point is that testing environments are often far smaller than their real production counterparts. Making the assumption that testing on a couple of white box servers running the same operating system and database versions, a few switches, and no-load variance other than what the load test software generates, equates to being confident that the application will perform under real-world loads in a real-world production environment is fallacious.

Use Service Virtualization or a Testing Database

To properly test the application, the performance engineer has to include all of the most important actions in the load which are important to effectively measure the impact on the business. Important actions often involve other systems or the back office. For example, testers should consider the impact of external service calls on their applications. These impacts can be significant depending on the number and utilization of the calls which may perform high demand, resource-consuming processing work. Understanding how these calls affect performance is critical.

Therefore, to guarantee the efficiency of the load test by examining features such as these, we recommend employing service virtualization. Doing so allows testers to replace the third-party with a service emulating the response of the third-party or back-office. On the other hand, testers could simply remove all the actions involving a third party from the scenario. This solution, however, will affect both the test objective and the testing efficiency. Testing in production is a balance between the risk of running the test and the risk of not running the test.

Slightly Alter Scenarios to Avoid Interactions with Back Office or Third-party Services

Because testing environments are inherently less-complex that real world environments, it isn't feasible to duplicate or simulate certain aspects of the production environment e.g., cost of licensing, compute resource and storage requirements for a minimal deployment, etc. Testers must work around that by removing references to those services and backend tiers.
In order to avoid impacting the business data, remove all interactions with:

- Third-party services
- The backend

For example, consider an application that makes calls to a huge ERP, such as SAP. Testers can set up an emulation for the SAP application by having a test server act as a stand-in for SAP services and

provide canned responses to mimic the necessary protocols, such as the SAP GUI.

Testing in Production and Proactive Monitoring

Testing in production is necessary to ensure that the:

- Expected load is supported by the live environment
- End-to-end user experience is acceptable
- Network equipment or CDN can adequately handle the expected load

The production environment has its own set of challenges due to the dataset, usage of third party, etc. As such, actions may be removed in order to enable a test in production.

While it definitely helps to measure the quality and user experience of real users, this approach won't enable testers to identify and monitor all performance issues in the production environment.
When it comes to TiP, testers need to proactively monitor the application. By monitoring, we are not referring to retrieving technical counters on the architecture but measuring the end-user performance on a regular basis. Synthetic monitoring, for example, has the advantage of allowing QA to run one single user-journey from several locations, all the while alerting testers about abnormal response times. Monitoring helps operations identify and resolve production issues without these issues having to be detected by real users.

Conclusion

In an ideal world, the more closely the testing environment matches the production environment, the more accurate the performance benchmarking results. However, creating a test environment that mirrors the production environment 100% is practically impossible. Therefore, by definition, testers glean unrealistic performance results from their tests due to key differences between testing and production environments.

Most companies avoid testing in production based on the potential impact on real-world user activities and data. Testers can reduce the impact of production system testing on actual application users by following team-based and process-based best practices, such as increased collaboration, testing during off-hours on off-days, testing before a release, testing during a maintenance window, performing read-only database operations or implementing service virtualization and carefully monitoring test execution.

Testing exclusively in production is better than nothing but limiting the testing effort to the production environment alone is not recommended, because it won't allow testers to properly tune or size the application prior to testing in a high-risk setting. Instead, testing in production should be viewed as the last validation to guarantee the user experience by including CDN in the testing approach. More than load testing, applying proactive application monitoring is recommended to detect performance issues before the real production users experience them.

About Neotys

The success of your digital strategy relies on your ability to deliver fast and reliable software, regularly. Creating great software quickly, using an optimized performance testing process is your competitive advantage – Agile and DevOps are part of the solution.

Neotys has over 12 years of development investment into NeoLoad – the performance testing platform designed to accelerate Agile and DevOps processes. It's built by engineers who recognized that in order to achieve their own Agile adoption objective, they needed to create a product that could facilitate superior load and performance testing continuously. The end result – up to 10x faster test creation and maintenance with NeoLoad.

The end result – up to 10x faster test creation and maintenance with NeoLoad.

We truly believe that the Performance Engineer can become the critical application performance partner providing the best testing coverage while respecting the cadence of the Continuous Delivery process. As performance becomes the responsibility of the wider team, continued delivery of an optimized performance testing platform is what drives our work every day.

For more information about Neotys and NeoLoad visit: www.neotys.com or contact sales@neotys.com