TABLE OF CONTENTS

Introduction

The Challenges of Testing in Production
- Launching a Test without Knowing All of the Consequences 1
- Testing in Production Can Affect Real Users 1
- Generating Load on a Third-party Application Can Be a Legal Issue 2
- Creation of Testing Data on the Production Environment 2

Best Practices
- Select the Right Time Window to Apply the Load 2
- Monitor the Infrastructure and Make Sure to Have the Proper Team Available During the Test 3
- Use Service Virtualization or a Testing Database 3
- Slightly Alter the Scenarios to Avoid Interaction with the Back Office or Third-Party Services 3

Testing in Production and Proactively Monitoring 3

Conclusion 3
INTRODUCTION

One of the biggest challenges in performance testing is the testing environment itself. Although most load and performance testing activities are done in a pre-production or a QA environment, many projects don’t have the budget to recreate an environment that closely represents the production environment.

These suboptimal test environments often differ from the production environment in of the following ways:

- Number of servers
- Type of network equipment
- Integration with third party tools
- Utilization of CDN
- Bandwidth utilization by other users or applications

Furthermore, many applications utilize third-party integrations for things like extended 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.

Typically, testing environments are also structured with fewer servers or less hardware, and 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 come up with a meaningful ratio if both environments use the exact same types of machines and with the only difference being a reduced number of servers per layer. But in reality, testers often face environments with less hardware, fewer servers, a mutualized database and other differences as well.

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.

Given these challenges with the testing environment itself, more and more companies and consultants are testing in production (TiP). This activity is indeed a solution to the environment problem, but it requires dedication to best practices to test successfully while managing the many risks that go with this approach.

THE CHALLENGES OF TESTING IN PRODUCTION

Launching a Test without Knowing All of the Consequences

Measuring your application’s performance in the production environment alone is very risky. In order to avoid costly performance issues before moving on to the production environment, testing in the QA environment is recommended. Launching a load test on the production environment without having any idea of the released application’s level of performance is considered akin to a crime against productivity in some companies. Depending on the results, it could also be a good reason to fire you.

Even if you have tested your application against a testing environment, TiP requires several resources in addition to the performance engineer:

- Operations: This team can “unplug” the current production environment for the test. They are in charge of alerting the performance engineer of the impact of the tests on the rest of the datacenter.
- Architect or Technical leader of the project: Someone involved with the project should be able to identify potential issues by looking at the logs or the database activity.
- DBA: The DBA will be in charge of finding blocking points on the database and replugging the production database.
- Project leader:
  - Defines the timeslots for load testing
  - Alerts the user of the potential disruptions
  - Defines the process of unplugging and replugging the production environment

Teams will need to be prepared to react depending on the impact the test has on the production environment.

Testing in Production Can Affect Real Users

We all agree on one point: Load testing in production can affect users or business processes working in the environment. Of course, the impact is linked to the main objectives of the test. For example, there would be greater impact on real users with a limit test than a constant load model, but whatever the objectives, the potential impact must be considered.
Many studies show a correlation between user experience and profit (depending on the business area and the application). Generating load during business hours endangers:

- The user experience
- The business
- The brand image of the company

Performance engineers love to control the load they apply on an environment. However, real users accessing the application during tests add noise to the results, thus eliminating the performance engineers’ chance at total control. Additionally, with a mix of real and virtual traffic, it becomes 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
- Or a combination of both

Having too much noise on the environment will only make understanding the performance testing results that much more difficult.

To avoid these issues, run your load test 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 would indirectly generate load on the partner’s environment. Legally, the project is not allowed to generate load on a third-party website without informing said party. The third-party could block or black list the traffic during the test, generating errors that could affect your load testing objectives.

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

Depending on how the web page is built, the usage of third-party could slow down the whole rendering of the page. You can work around this issue by emulating the response of the partner using service virtualization. This is technically the best solution, but it can:

- Be expensive to implement
- Require a lot of changes to the production environment

The only alternative is 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 (login, products, etc.) and there are often problems with 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 could feed the back office with testing data and could connect to/from back office services of the company.

You can utilize test data in the production environment by:

- Disconnecting the production and plugging in a testing database instead (This is possible when the environment is not connected to several applications).
- Creating a specific testing account in production (dedicated to testing). Note that this can sometimes be difficult and or impossible in the production environment.
- Avoiding steps that generate records in the back office (avoid validating the order).

Removing specific actions on the scenarios to avoid creating data in the back office will affect the test objective and the testing efficiency. Whether you choose to test in production or not, considerable risk is involved.

**BEST PRACTICES**

How can you mitigate the risk of testing in production?

**Select the Right Time Window to Apply the Load**

When running tests on the production environment, select timeslots that impact fewer real users like:
During night hours (out of business hours)  
After deploying a new release  
During maintenance hours

This method provides a very short time window during which the performance engineer will need to conduct all the tests.

**Monitor the Infrastructure and Make Sure to Have the Proper Team Available During the Test**

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 to avoid any major production issues  
- Correlate and identify the bottlenecks of 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.

**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. Therefore, to guarantee the efficiency of the load test by including these features, you’ll need to utilize service virtualization. Doing so allows you to replace the third-party with a service emulating the response of the third-party or back office.

On the other hand, you could simply remove all the actions involving a third party from the scenario. This solution, however, will affect 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 the Scenarios to Avoid Interaction with the Back Office or Third-Party Services**

In order to avoid impacting the business data, remove all interactions with:

- Third-party services  
- The backend

If using service virtualization is impossible or too difficult on the production environment, you can employ this method by removing these interactions.

**TESTING IN PRODUCTION AND PROACTIVELY MONITORING**

Testing in production is necessary to ensure:

- The expected load is supported by the live environment  
- The end-to-end user experience will be acceptable  
- The 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 you to identify and monitor all performance issues in the production environment.

When it comes to TiP, you will need to also 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 you to run one single user-journey from several locations all the while alerting you of abnormal response times. This will help operations identify and resolve production issues without these issues having to be detected by real users.

**CONCLUSION**

Creating a test environment that mirrors your production environment 100% is practically impossible. Therefore, by definition we are retrieving unrealistic performance results from our tests due to the key differences between the testing and production environments.
Most companies avoid testing in production due to its potential impact on actual users’ activities and their data. You can reduce the impact of production system testing on actual application users by following best practices like 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 the test execution.

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