Production: Performance where it REALLY matters!

An excerpt from About:Performance blog

Best Practices

About the Author

Andreas Grabner has been helping companies improve their application performance for 15+ years. He is a regular contributor within Web Performance and DevOps communities and a prolific speaker at user groups and conferences around the world. Reach him at @grabnerandi

For the complete version of this blog, go HERE

"Production is where performance matters most, as it directly impacts our end users and ultimately decides whether our software will be successful or not. Efforts to create test conditions and environments exactly like Production will always fall short; nothing compares to production!" These were the opening lines of my invitation encouraging performance practitioners to apply for the recent WOPR24 (Workshop On Performance and Reliability).

Thirteen performance gurus answered the call and contributed to the event by providing their experience reports and participating in the workshops. Special thanks to organizers Eric Proegler and Mais Tawfik! My key takeaway of WOPR24 is that Performance Engineering as we know it is changing, turning away from traditional load testing and toward production and Continuous Integration, with performance the link between Dev and Ops in DevOps. Really, who would have thought?

STOP being a Load Tester – Become a DevPOps

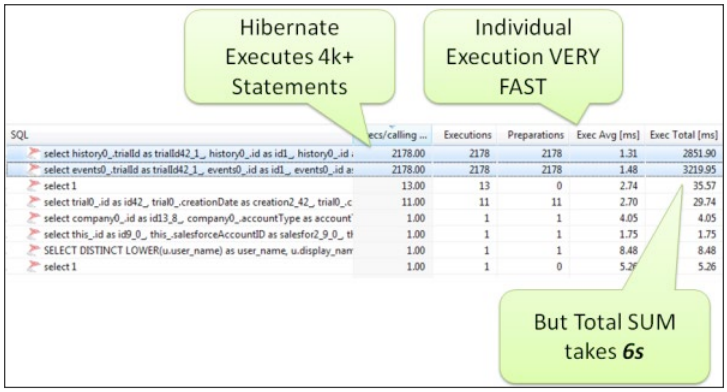
The most interesting observation for most of us attending the workshop was that the role and focus of a performance engineering team is changing toward continuous but shorter performance tests in continuous integration, operations monitoring, performance engineering in production, and as the link providing metrics-based feedback to the business and engineering. On the last day of the event we coined the term "DevPOps" with the Performance Team as the missing link to ensure that the software performs in production as it was intended to, based on all the work performed in engineering and the performance testing prior to deploy. DevPOps (like DevOps) features fast feedback loops at its core. Feedback loops need to not only flow back from production monitoring to testing, but also all the way back to engineering, to determine how the software is really behaving under actual load conditions. Therefore, the role of the DevPOps team includes a variety of new responsibilities in addition to traditional load testing:

- Automated Continuous Performance Engineering in CI
  - Shift-Left Performance Metrics into Jenkins, Bamboo and Co
  - Find regressions based on architectural metrics and stop the build
- Define Monitoring Metrics and Dashboards
  - Find relevant metrics for CxO, Engineering, Biz and Ops
  - Build monitoring infrastructure for both test and production

- Load and Performance Tests to test stability, scalability and monitoring
  - Run them in production or production like environments
  - Verify monitoring metrics with stakeholders
- Monitor Production, Compare with Test, Report to Stakeholders
  - Identify regressions between deployments and test environment
  - Communicate and discuss metrics to CxO, Engineering, Biz and Ops
- Continually optimize deployment configuration
  - Handle peak loads with scaling infrastructure
  - Identify and reduce BOT traffic (which, on average, accounts for about 70% of web traffic)

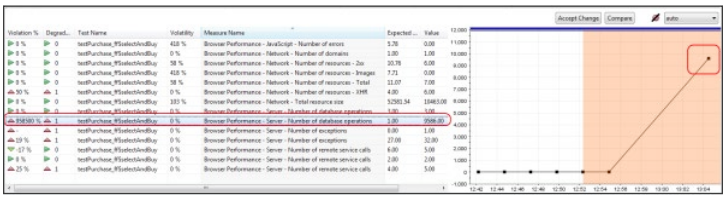
Automated Continuous Performance Engineering in CI

Based on my personal experience you don't need to execute large scale load tests to find most of the problems that will later result in poor performance or scalability problems. Why? Because they are typically architectural in nature and can be found by executing either Integration, API-tests or a very small scale load test. The number one problem I find is inefficient access to the data store (Database, O/R Mapper, REST Service, Cache) resulting from querying too much data, using too many round trips to obtain the data, or not using optimized queries. Finding a data-driven problem, like the one illustrated here, in which wrong usage of Hibernate caused a feature in the software to execute thousands of individual SQL statements, was identified by looking at the # of SQL Statements executed by the integration test. If the same SQL statement is seen being executed more than once, you likely have a potential scalability issue.



Hook up your Integration or API Tests with Profiling or Tracing tools to capture access to your data layer

Once you determine how to capture these details per test, you can use these measure points to identify regressions across builds, as shown here:



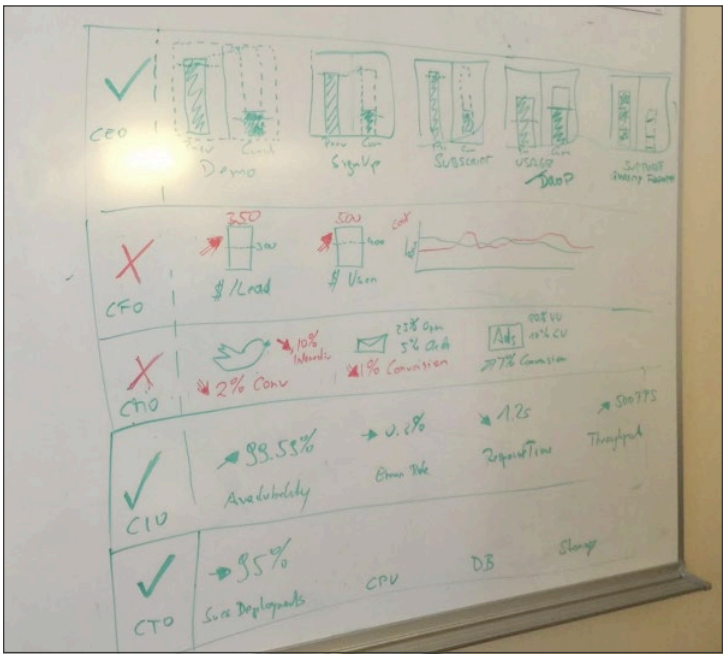
Identify changes in code behavior to spot bad code changes as soon as possible

Define Monitoring Metrics and Dashboards

We had one special exercise during WOPR where we divided participants into three teams to create "the perfect dashboard" showing important metrics for three fictitious businesses: eCommerce, SaaS and Enterprise Corporation (aka "Evil Corp" J). Interestingly, we all reached similar conclusions:

1) High-Level Business Metrics consumable for EVERYONE in the organization
2) Aggregated Status per team or business unit
3) More specific dashboards to review

Here is a rough sketch of the SaaS dashboard on which my team was working. We wanted to show high-level metrics that illustrate how successful we are as a company, focusing on the overall conversion funnel (interesting for CEO), costs involved to get people to our platform (CFO), lead generation and social media (CMO), as well as some technical metrics for the CIO & CTO on how the system operates and how much it costs to run it.



Our SaaS Company Dashboard created at WOPR24

Once you agree on which dashboards and metrics are relevant it's time for you to determine how to gather these metrics in your load testing, staging, and production environment. Why load testing and staging? Because that's the best way to see whether you are actually able to capture this data, and whether it really makes sense – or delivers value — to the stakeholders

Load and Performance Tests to test stability, scalability, monitoring

As we already found with most architectural issues in Continuous Integration, we can focus on the essential parts of load and performance testing:

- Stability: Uncover memory or connection leaks when putting the system under constant load for an extended period of time
- Scalability: Can the system handle spike load? Can you scale up by adding more servers or resources, and how does that impact overall system behavior? Does the system also scale down after peak load is over?
- Monitoring: As mentioned earlier, your testing environment is perfect to validate that monitoring dashboards will also work in production. You should also verify alerting, specifically, are the right people notified if the system slows down?

Monitor Production, Compare with Test, Report to Stakeholders

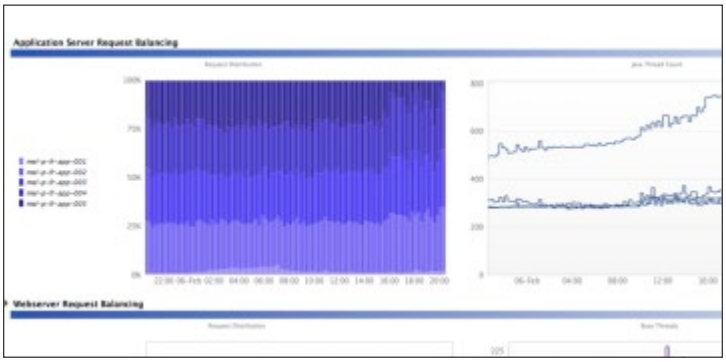
Production: Where Performance Really Matters. That was our title for WOPR and is where we are now. Use your metrics and dashboards to see whether the production environment behaves just as in testing. If there is a difference find out why:

- Load Behavior: Different spikes during the day? Caching and bots not considered?
- End User Behavior: Users use different features in a different way? Real devices behave differently than in the simulation?
- System Behavior: Other applications on same hardware impacts your application? Virtualized or Cloud Instances show different behavior?

Spotting these differences allows you to impact future product development in the event certain features are not used at all or are used in a different way than previously expected. You can update your load test scenarios for your "Go Live" tests to obtain a more realistic simulation. It also lets you build a better test environment that shows similar resource characteristics found in production.

Constantly optimize deployment configuration

If real-use traffic and behavior are changing in your test environment, you may not always have time to test changes to the deployment and configuration. Sometimes must react fast and reconfigure load balancers, add more virtual instances, or block/redirect certain request types. The following example shows a classical monitoring dashboard designed to validate how load is balanced across web and application servers. Changing to a different load balancing algorithm, or redirecting certain requests to a certain server farm, might be necessary to keep the system performing well. Make sure that all of these changes make it back to the pre-production setting for future testing:



Monitoring Load Distribution behind a Load Balancer. Do we need to change the algorithm? Do we need to re-route or block certain requests?

Other WOPR Highlights and Acknowledgements

In the two and one half days at the BlazeMeter office in Mountain View (thanks for offering us your location), we had several great experience reports and group exercises, all dealing with different aspects of performance engineering in production. We discussed the influence of container and virtualilzation technologies, and how to best tackle real-user monitoring while considering that most real users will engage with software or apps through their mobile device. The great thing about WOPR is that participants received direct feedback to work in which they are currently engaged, as well as to the experience report that was presented. I am confident that some of the attendees returned to their offices ready to change how they traditionally monitor performance. I want to say a big "thank you" to all attendees, as my writing of this blog was influenced by everyone: Eric Proegler, Mais Tawfik, Ben Simo, Oliver Erlewein, James Davis, Doug Hofmann, Andy Hohenner, Yuri Maked

Dynatrace Digital Performance Platform — it's digital business... transformed.

Successfully improve your user experiences, launch new initiatives with confidence, reduce operational complexity and go to market faster than your competition. With the world's most complete, powerful and flexible digital performance platform for today's digital enterprises, Dynatrace has you covered.

Learn more at dynatrace.com

Dynatrace has redefined how you monitor today's digital ecosystems. AI-powered, full stack and completely automated, it's the only solution that provides answers, not just data, based on deep insight into every user, every transaction, across every application. More than 8,000 customers use Dynatrace to optimize customer experiences, innovate faster and modernize IT operations with absolute confidence.

02.03.17 2042_PerformanceWhereItReallyMatters-BlogReprint_jg https://www.dynatrace.com/blog/production-performance-where-it-really-matters/