# The Eighth International Workshop on Load Testing and Benchmarking of Software Systems (LTB 2020)

http://ltb2020.eecs.yorku.ca

April 20, 2020
Edmonton, Canada
Co-located with ICPE 2020
The 11th International Conference on Performance Engineering

## [Keynote] Alexandru Iosup, Vrije Universiteit Amsterdam.

**Will It Rain Today? Understanding the Weather of Computing Clouds, before it Happens.**

Cloud computing services play an important role in today's modern society. They enable daily operation and advances in key application domains, from banking to e-commerce, from science to gaming, from governance to education. Combining technology developed since the 1960s (e.g., modes of resource sharing) with new paradigms that could only have emerged in the 2010s (e.g., FaaS), they promise to enable unprecedented efficiency and seamless access to services for many. However successful, we cannot take the cloud for granted: its core does not yet rely on sound principles of science and design, its engineering is often based on hacking, and there have already been worrying signs of unstable operation. In this talk, we posit that we can address the current challenges by focusing on the relatively large complex of systems (that is, systems of systems or even ecosystems), and by increasing and focusing the effort put into load testing and benchmarking. We contrast this to the current focus on single or relatively small systems, and not always principled testing and benchmarking. We show examples of how our approach could work in practice, presenting (i) results related to performance variability, (ii) discovery methods that feed into the engineering of future load testing and benchmarking frameworks, and (iii) processes that could improve the reproducibility and credibility of experimental results in this field. This leads us to formulate the vision of a community-wide effort to create the Distributed Systems Memex, to share and preserve operational traces collected from the distributed systems that currently underpin our society. Part of this work has been conducted in the international collaboration provided by the SPEC RG Cloud Group.

## [Experience Talk] Boris Zibitsker, BEZNext and Alexander Podelko, Oracle.

**Performance Testing and Modeling for New Applications.**

Traditional load testing (optimized for the waterfall software development process) was mostly focused on pre-production realistic tests, so the main goal was to make load and environment as similar to production as possible. Drastic changes in the industry in recent years, especially agile development and cloud computing, opened new opportunities for performance testing. Instead of a single approach to performance testing, we now have a full spectrum of different tests which can be performed at different moments. Deciding what and when to test is now a very non-trivial task. But the same industry trends that made performance testing easier - a working system on each iteration due to agile development, cloud infrastructure available for deployment - introduced new challenges.

Due to in increased sophistication and scale of systems supporting production workloads, full-scale realistic performance testing is not viable anymore in most situations (or, at least, not on each iteration). In many cases we may have different partial performance test results - for lower level of load, different parts of the system, different functionality, etc. As a result, analyzing and interpreting performance tests is more challenging and may require modeling to make meaningful conclusions about performance of the whole system.

Performance measurements collected during performance testing provide response times and resource utilization for specific workloads. Together with knowledge about architecture and environments, these measurements can be used to create a model to predict a system's performance (to be verified by larger-scale performance tests if necessary). This is a proactive approach to mitigating performance risks, but it requires significant skills and investments as well as proper implementation. For existing systems, this approach is often complemented (or even completely replaced) by reactive approaches of observing the production system ("shift-right" – monitoring, tracing, canary testing, etc.). However, this approach does not work for new systems. If you are creating a new system, you need proactive methods such as early performance testing ("shift-left") and modeling to make sure the system will perform as expected.

Modeling becomes more important at the DevOps design stage when performance and cost consequences of different design decisions need to be investigated. In this case, production data is not available until the system is fully developed and deployed and waiting for pre-production results in order to build the model is not acceptable.

The best examples of performance risk mitigation by a combination of performance testing and modeling are Big Data or Data Warehouse, On Prem and in the Cloud. The enormous size of the systems makes creation of full-scale prototypes almost impossible. However associated performance risks are very high – implementing a wrong design may be not fixable and can lead to complete re-design from scratch. So, building a model to predict a system's cost and performance based on early/partial prototype performance test results and knowledge about architectures and environments is the main way to mitigate associated risks.

In this presentation, we will review value and limitations of the available load testing tools and discuss how modeling and optimization technology can expand results of load testing. We will review a use case illustrating data collection and workload characterization; anomaly, root cause and seasonality detection; workload forecasting and predicting impact of new application implementation.

We will illustrate how modeling and optimization technology are used to find the appropriate platform for a new application and develop proactive recommendations reducing the risk of performance surprises by answering questions like:

-How will new application perform in a production environment with a large number of concurrent users accessing large volumes of data?

-How will implementing new application affect performance of existing applications?

-Can changing priorities, concurrency and resource allocation for different workloads help to meet Service Level Goals?

-Does the production environment have enough capacity to support expected workloads and volume of data growth?

-Should new applications be deployed in a Data Warehouse or Big Data, On Prem or in the Cloud environment?

## [Keynote] Alberto Avritzer, eSulabSolutions.

**Integrating Automated Scalability Assessment into DevOps.**

DevOps is an emerging software engineering paradigm that aims for fast feedback cycles between software changes in development and bringing these changes into production. Apart from cultural and organizational changes, DevOps employs a high degree of automation, cloud technologies, and tailored architectural styles such as microservices. The increased development speed and complexity impose various challenges to scalability assessment. However, this context also provides great opportunities, such as enabling access to extensive operational data obtained from continuous monitoring in production, which is a core DevOps principle.

The goal of this keynote is to provide an overview of challenges and approaches for scalability assessment in the context of DevOps and microservices. Specifically, we present scalability assessment approaches that employ operational data obtained from production-level application performance management (APM) tools, giving access to operational workload profiles and architectural information. We use this data to automatically create and configure scalability assessments based on models and load tests. The focus of this keynote is on approaches that employ production usage, because these approaches provide more accurate recommendations for microservice architecture scalability assessment than approaches that do not consider production usage.

We present an overview of (1) the state-of-the-art approaches for obtaining operational data from production systems using APM tools, (2) the challenges of scalability assessment for DevOps and microservices, (3) selected approaches based on operational data to assess scalability.

## [Experience Talk] David Daly, MongoDB.

**How to Waste Time and Money Testing the Performance of a Software Product.**

It is important for developers to understand the performance of a software project as they develop new features, fix bugs, and try to generally improve the product. While it is simple to state that requirement, it can be hard to do in practice. There are a lot of choices an organization faces when trying to understand the performance of the software, with a lot of opportunities to waste money (execution resources), or worse, time. We have run full speed into a number of those opportunities, and we like to think we have learned from those opportunities. The challenges include false positives, false negatives, reproducibility of tests, noise in the results, tests execution time, insufficient workload coverage, insufficient configuration coverage, cost, identifying regressions in the presence of noise, diagnosing those regressions, one regression hiding another, etc.

In this talk we describe our performance testing environment at MongoDB, and what we have built into that environment in order to address those challenges. The core of our environment is a focus on automating everything, integrating into our continuous integration (CI) system (Evergreen), controlling as many factors as possible, and making everything as repeatable and consistent as possible. We have three levels of performance tests (system level distributed tests, single node tests, unit level), and each level has three main areas: provisioning and setting up the system under tests, running the tests, and analyzing the results. Our performance tests have five main use cases: detecting changes as they are committed into the source repository, allowing developers to test their changes before committing to the source

repository, release management, exploring the performance of the system,and developing and adding new tests.

Our performance testing completely integrates with our continuous integration system for detecting performance changes as code change sets are submitted to the repo. The performance tests do not run on every commit. Instead, we specify a minimum period between runs. For our end to end distributed tests, that is commonly 1 day, while the single node focused tests run every 4 hours. The results from these tests are visualized, automatically processed to identify changes in performance (good and bad), tracked in our ticket-based system (JIRA), and assigned out to appropriate developers for follow-up. Developers may use the same system for submitting “patch tests” on change sets before committing the changes to the source repository, enabling them to optimize code and avoid introducing regressions into the official repository. Additionally, the results are tracked over the entire release cycle to understand how the upcoming release compares to the previous stable release. There may be several open performance regressions and improvements -- the system allows us to compare across all of those in order to monitor the net performance changes and make informed decisions about whether to release and where to focus our efforts.

The flexibility of our performance infrastructure also enables more exploratory work. All tests are controlled by configuration files controlling nearly every aspect of the system. That allows us to run experiments to ask questions such as “What happens if we change knob X to Y?” or “What happens as we scale to larger server instances?”, while maintaining the control and observability baked into the system.

Finally, the last use case is adding in new workloads. We have found we get the best coverage when we leverage the entire development organization to test performance. We actively encourage developers to add new workloads to test their new features. As such, the tools need to be flexible, robust, and easy to use for all the developers.

We have built our performance testing infrastructure over the last 5-6 years and it has had a huge impact on the product we release. We are able to quickly identify changes in performance, triage them, and assign them out to developers. The developers are able to exactly reproduce those regressions and are eager to fix them. And at the end of the day, we have a much better understanding of the performance of the software we release and ship to customers.

## [Experience Talk] Paul Bruce and Henrik Rexed, Neotys.

**Performance Testing Automation for CI/CD Pipelines.**

The conditions for professional testing of applications are becoming more and more challenging. On the one hand, there is a trend towards microservice architectures, which requires frequent testing of many small components and requires many integrations and regression tests. Another trend is towards a dynamic IT infrastructure: outsourcing to the cloud or on-premise supplemented by cloud resources and all this dynamically scaling to ensure performance on the one hand and to minimize costs and maintenance on the other. And then there is the expectation of us end users that we always expect the speed of Google Search under whatever conditions.

To do justice to all of this, we have to adapt our processes and methods, and often our tools and the way we work together. It would be nice if we could test extensively at a component level during the development of software, ideally a Test as Code approach is ideal, which can be done by developers. Without media discontinuity and with the continued use of all test activities, the integration tests, end-user experience tests, security tests, performance tests, etc. The test specialists can integrate them with the speed of agile development and with direct feedback into development. Since we constantly release and commission new software components, we also need a direct connection to production. The more dynamically we optimize our IT infrastructure, the greater the difference to our test and development environments. This is another reason why QA and Operations have to work very closely together. One of our customers - a well-known German bank - says: "Today we not only test individual applications such as the eBanking application but also an entire banking day with all applications and the full range of users."

Using the topic of performance testing as an example, which in the past very often only came into play in a silo of its own shortly before a release, we want to show how development accompanying component tests can be designed and almost completely automated, from integration tests and fully comprehensive system-wide tests through to recording and feedback from production. We will show how different tools such as GIT, Jenkins, Selenium, NeoLoad, Dynatrace, and Jira can be integrated in a meaningful way. We show how automated this can be and how suitable such a toolchain can be for DevOps and the speed of Agile. And last but not least, we show that the whole thing is also possible with a highly dynamic and flexible IT infrastructure. The whole arc is completed on the aspect of collaboration because without this it would be difficult.

## [Experience Talk] Gopal Brugalette, SoFi.

**Machine Learning Applications to the Analysis of Performance and Load Testing Results.**

Managing the performance of complex systems requires more than simply running load tests. You need to perform a careful analysis of test results and production metrics. The sheer amount of data generated makes analysis a huge challenge that is often left wanting. With machine learning (ML) and the application of data science techniques, you have the opportunity to derive valuable and actionable information from big data. Gopal Brugalette shares the basic concepts behind ML, covering clustering, predictive analysis, and neural networks. He shows you how to implement algorithms using open source tools and languages like Python, R and AWS cloud services. With real-world examples, Gopal demonstrates the big data platforms Hadoop, Elasticsearch and AWS Sagemaker and illustrates performance and load testing problems like performance monitoring, test result comparisons, error message analysis, and user insights. Join Gopal to learn about machine learning and how you can start solving your performance and load testing challenges.

## [Experience Talk] André van Hoorn, University of Stuttgart.

**Architecture-based Resilience Testing of Microservice-based Software Systems.**

Microservices are an emerging architectural style for distributed software systems. Microservice-based software systems are expected to be resilient to environmental changes such as spikes in workloads as well as software and hardware failures. Resilience can be achieved by applying respective application-level and platform-level design patterns, e.g., circuit breakers and bulkheads. Resilience testing aims to assess the effectiveness of the applied resilience patterns by conducting experiments under meaningful workloads and fault loads. These experiments can be conducted in staging and production environments. The latter is becoming popular under the notion of chaos engineering, e.g., via tools such as the Simian Army and the Chaos Toolkit. Current resilience testing practice for microservices is ad-hoc, and faults are injected at random locations, e.g., shutting down virtual machines and delaying service replies. It is an open challenge how to best select promising sets of experiments to execute.

In this talk, I will give an overview of our ongoing research on architecture-based resilience assessment of microservice-based software systems. The main content of the talk will be split into two parts, covering two complementary approaches:

Part 1: First, I will introduce our ORCAS approach that aims to automate the efficient search for resilience vulnerabilities by incorporating architectural knowledge, as well as knowledge about the relationship between resilience anti-patterns, patterns, and suitable fault injections. The approach builds on existing works on model-based and measurement-based quality evaluation of software systems, combined with machine-learning to generate resilience experiments. In addition to the overview of the approach, I will present experimental results on its effectiveness.

Part 2: Second, I will report on our experiences with adopting well-known architecture and risk analysis techniques, such as ATAM (Architecture Tradeoff Analysis Method), FTA (Fault Tree Analysis), HAZOPS (Hazard and Operability Study), and FMEA (Failure Mode and Effects Analysis) to analyze system resilience and identifying resilience experiments. We have obtained the experiences from industrial case studies.

## [Keynote] Carey Williamson, University of Calgary.

**Things That Go Bump in the Net.**

We live in a world in which we are very dependent on the Internet for our personal and professional activities. This dependence on network-based applications, and Internet connectivity, makes us vulnerable when things go wrong, or don't behave as expected. Furthermore, diagnosing network-related performance problems is often a challenge, with some of the reasons being data volume, geo-distributed applications, transient behaviours, and encryption. This talk shares some of our experiences on the benchmarking, performance evaluation, and debugging of network-based applications over the past decade. It will discuss some of the tools and methods that we use, and the many lessons we have learned along the way. Examples will include Web, video streaming, online social network, email, and LMS applications.

## [Research Paper] Dheeraj Chahal, Ravi Ojha.

**Migrating a Recommendation System to Cloud Using ML Workflow.**

Inference is the production stage of machine learning workflow in which a trained model is used to infer or predict with real world data. A recommendation system improves customer experience by displaying most relevant items based on historical behavior of a customer. Machine learning models built for recommendation systems are deployed either on-premise or migrated to a cloud for inference in real time or batch. A recommendation system should be cost effective while honoring service level agreements (SLAs).

In this work we discuss on-premise implementation of our recommendation system called iPrescribe. We show a methodology to migrate on-premise implementation of recommendation system to a cloud using ML workflow. We also present our study on performance of recommendation system model when deployed on different types of virtual instances.