Microservices: A Performance Tester’s Dream or Nightmare?

Simon Eismann
University of Würzburg
@simon_eismann

Cor-Paul Bezemer
University of Alberta
@corpal

Weiyi Shang
Concordia University
@swy351

Dušan Okanović
University of Stuttgart
@okanovic_d

André van Hoorn
University of Stuttgart
@andrevanhoorn

https://research.spec.org/
working-groups/rg-devops-performance.html
What is Performance Regression Testing?

Performance Regression testing

1. Deploy Application
2. Perform Load test
3. Compare to previous commit

DevOps Pipeline

Github

commits changes

triggers

Build
Unit test
Regression test

Microservices: A Performance Tester’s Dream or Nightmare?
@simon_eismann
Requirements for Performance Testing

R1 A stable testing environment which is representative of the production environment

R2 A representative operational profile (including workload characteristics and system state) for the performance test

R3 Access to all components of the system

R4 Easy access to stable performance metrics

R5 Sufficient time
Microservice traits

- **T1** Self-containment
- **T2** Loosely coupled, platform-independent interfaces
- **T3** Independent development, build, and deployment.
- **T4** Containers and Container Orchestration
- **T5** Cloud-native
Microservices - A Performance Testers Dream or Nightmare?

Benefit 1: Containerization
- Containers package environment
- Simplifies setup of test environment

Benefit 2: Granularity
- Individually testable services
- Dependencies via HTTP calls
- Dependencies easily mocked

Benefit 3: Easy access to metrics
- Orchestration frameworks simplify metric collection
- Application-level metrics common

Benefit 4: Integration with DevOps
- Size reduces performance test duration
- Performance testing within pipeline
Too good to be true? – Let’s test it!

**RQ1**
How stable are the execution environments of microservices?

**RQ2**
How stable are the performance testing results?

**RQ3**
How well can performance regressions in microservices be detected?
Case Study

TeaStore Benchmarking Application

Scenario | #Nodes | Cores/Node | Memory/Node |
---------|--------|------------|-------------|
Default   | 20     | 1          | 6.5 GB      |
Balanced  | 20     | 1          | 6.5 GB      |
LargeVMs  | 5      | 4          | 26 GB       |
Autoscaling| 5     | 4          | 26 GB       |
Regression (baseline)| 5 | 4 | 26 GB |
Regression | 5    | 4          | 26 GB       |

Table 1: Cluster size in the different scenarios.

Deployment Platform

docker, Kubernetes, HashiCorp Terraform, Google Cloud Platform
Research Question 1 – Selected Findings

How stable are the execution environments of microservices across repeated runs of the experiments?

**Finding 1:** The non-deterministic behaviour of the autoscaler results in different numbers of provisioned microservice instances when scaling the same load.

**Finding 2:** Even when fixing the number of provisioned instances of a microservices, their deployment across VMs differs.

<table>
<thead>
<tr>
<th>Load</th>
<th>Service</th>
<th>Experiment run</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>Auth</td>
<td>1: 4, 2: 5, 3: 6, 4: 7, 5: 8, 6: 9, 7: 10</td>
</tr>
<tr>
<td></td>
<td>WebUI</td>
<td>1: 8, 2: 8, 3: 8, 4: 8, 5: 8, 6: 8, 7: 8</td>
</tr>
<tr>
<td></td>
<td>Recom.</td>
<td>1: 2, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1</td>
</tr>
<tr>
<td></td>
<td>Persist.</td>
<td>1: 8, 2: 7, 3: 6, 4: 5, 5: 4, 6: 3, 7: 2</td>
</tr>
</tbody>
</table>

Table 2: Number of provisioned service instances after twenty minutes of warmup across ten experiment repetitions in the Autoscaling scenario.

(a) Deployment during experiment run 6/10

(b) Deployment during experiment run 1/10

Figure 3: Deployment from two repetitions of LargeVMs scenario with 700 requests/second. The differences in deployment are indicated by thick red borders (A = authentication service, I = ImageProvider service, P = Persistence service, R = Recommender service, W = WebUI service).
Research Question 2 – Selected Findings

How stable are the performance testing results across repeated runs of the experiments?

**Finding 1:** There exist statistically significant differences between the performance testing results from different scenarios.

**Finding 2:** The total CPU busy time may not be statistically significantly different between scenarios.

![Box plots showing mean response time for four scenarios and three load-levels each.](Figure 2: Mean response time for four scenarios and three load-levels each (all distributions consist of ten elements, one for each repetition of the scenario).)

![Box plots showing CPU busy time for four scenarios and three load-levels each.](Figure 4: CPU busy time for four scenarios and three load-levels each (N=10).)
Research Question 3 – Selected Findings

How well can performance regressions in microservices be detected?

Finding 1: Using only a single experiment run results in flaky performance tests

Finding 2: Using ten experiment runs results in stable performance tests

Table 4: Comparing the distributions of the response time and total CPU time between different scenarios with regression between the LargeVMs and 10% Regression scenario (all distributions consist of ten elements, one for each repetition of the scenario—hereafter identified as N=10).

<table>
<thead>
<tr>
<th>Load [Req/s]</th>
<th>Response time p-value</th>
<th>Response time Eff. size</th>
<th>CPU Utilization p-value</th>
<th>CPU Utilization Eff. size</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>0.00</td>
<td>1.00 (L)</td>
<td>0.00</td>
<td>1.00 (L)</td>
</tr>
<tr>
<td>800</td>
<td>0.00</td>
<td>1.00 (L)</td>
<td>0.00</td>
<td>1.00 (L)</td>
</tr>
<tr>
<td>900</td>
<td>0.00</td>
<td>1.00 (L)</td>
<td>0.00</td>
<td>1.00 (L)</td>
</tr>
</tbody>
</table>

Figure 6: ROC curve showing the detection accuracy for the 10% and 30% regression.
Microservices - A Performance Testers Nightmare?

**Nightmare 1**

**Stability of the environment**
- Autoscaling/container orchestration is not deterministic
- Execution environment can not be expected to be stable

**Nightmare 2**

**Reproducibility of the experiments**
- The repeated experiments may not result in the same performance measurements
- Multiple measurements required for regression testing

**Nightmare 3**

**Detecting small changes**
- Variation between measurements can be quite large
- Detecting small changes is challenging
Research Directions

Research Direction 1
Variation reduction in executing performance tests

Research Direction 2
Studying the stability of (new) performance metrics

Research Direction 3
Creating a benchmark environment for microservice-oriented performance engineering research
Replication Package

Performance measurements

- Wrapped in docker container for platform independent execution
- Requires only Google Cloud access keys as input
- Fully automated performance measurements

Available online at: https://doi.org/10.5281/zenodo.3588515

Data set and analysis

- Measurement data of over 75 hours of experiments
- Scripts to reproduce any analysis, table or figure from the manuscript
- 1-click reproduction of the results as a CodeOcean Capsule

Available online at: https://doi.org/10.24433/CO.4876239.v1
Summary

Microservices - A Performance Testers Dream or Nightmare?

Benefit 1: Containerization
- Containers package environment
- Simplifies setup of test environment

Benefit 2: Granularity
- Individually testable services
- Dependencies via HTTP calls
- Dependencies easily mocked

Benefit 3: Easy access to metrics
- Orchestration frameworks simplify metric collection
- Application-level metrics common

Benefit 4: Integration with DevOps
- Size reduces performance test duration
- Performance testing within pipeline

TeaStore Benchmarking Application

Deployment Platform

Table 1: Cluster size in the different scenarios.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>#Nodes</th>
<th>Core/Node</th>
<th>Memory/Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>20</td>
<td>1</td>
<td>6.5 GB</td>
</tr>
<tr>
<td>Balanced</td>
<td>20</td>
<td>1</td>
<td>6.5 GB</td>
</tr>
<tr>
<td>LargeVMs</td>
<td>5</td>
<td>4</td>
<td>26 GB</td>
</tr>
<tr>
<td>Autoscaling</td>
<td>5</td>
<td>4</td>
<td>26 GB</td>
</tr>
<tr>
<td>Regression (baseline)</td>
<td>5</td>
<td>4</td>
<td>26 GB</td>
</tr>
<tr>
<td>Regression</td>
<td>5</td>
<td>4</td>
<td>26 GB</td>
</tr>
</tbody>
</table>

Finding 1: Using only a single experiment run results in flaky performance tests
Finding 2: Using ten experiment runs results in stable performance tests

Table 6: Comparing the distributions of the response time and total CPU time between different scenarios with regression between the LargeVMs and 10% Regression scenario (all distributions consist of ten elements, one for each repetition of the scenario – hereafter identified as N=10).

<table>
<thead>
<tr>
<th>Load [Req/s]</th>
<th>Response time p-value</th>
<th>EF size</th>
<th>CPU Utilization p-value</th>
<th>EF size</th>
</tr>
</thead>
<tbody>
<tr>
<td>700</td>
<td>0.00 1.00 (L)</td>
<td>0.00 1.00 (L)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>0.00 1.00 (L)</td>
<td>0.00 1.00 (L)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>900</td>
<td>0.00 1.00 (L)</td>
<td>0.00 1.00 (L)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: ROC curve showing the detection accuracy for the 10% and 90% regressions.
Microservices: A Performance Tester’s Dream or Nightmare?

Simon Eismann
University of Würzburg
@simon_eismann

Cor-Paul Bezemer
University of Alberta
@corpal

Weiyi Shang
Concordia University
@swy351

Dušan Okanović
University of Stuttgart
@okanovic_d

André van Hoorn
University of Stuttgart
@andrevanhoorn

https://research.spec.org/
working-groups/rg-devops-performance.html