University of Stuttgart
Institute of Software Technology
Reliable Software Systems

Performance Engineering for Microservices

Research Challenges and Directions

André van Hoorn

(and 7 more)

WOSP-C @ 8th ACM/SPEC International Conference on Performance Engineering,
April 22, 2017, L‘Aquila, Italy

The Authors

Robert Heinrich

KIT
Germany

André van Hoorn

U Stuttgart
Germany

Holger Knoche

Kiel University
Germany

Fei Li

Siemens AG
Austria

Lucy E. Lwakatare

U Oulu
Finland

Claus Pahl

Free U Bozen-Bolzano
Italy

Stefan Schulte

TU Wien
Austria

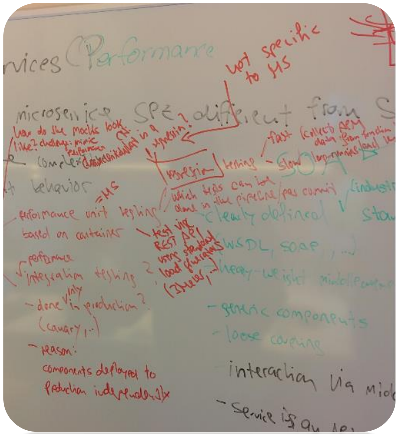
Johannes Wettinger

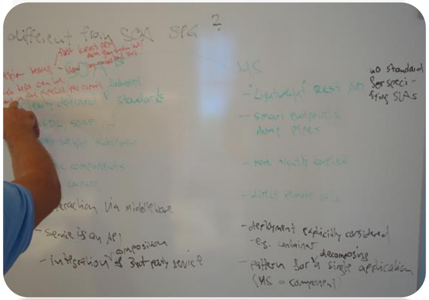
U Stuttgart
Germany

Result of Break-Out-Group @ GI Dagstuhl Seminar 16394

Software Performance Engineering in the DevOps World

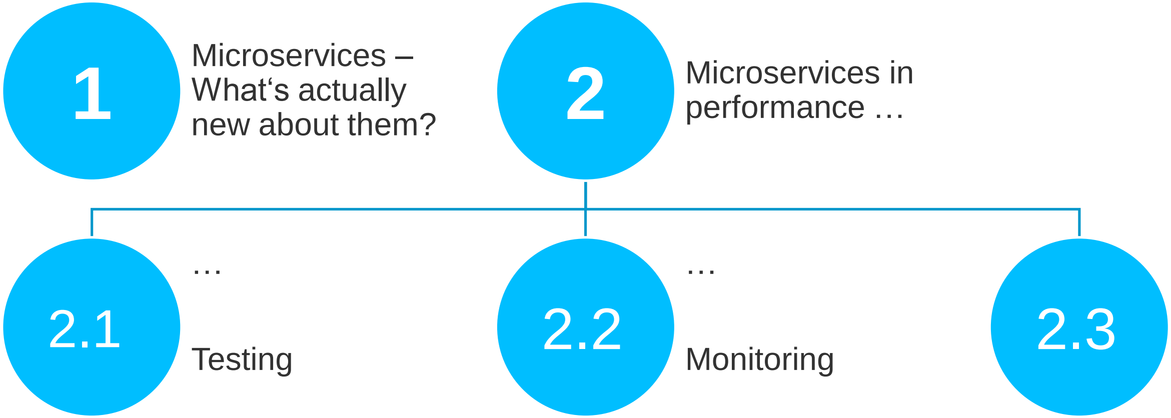
http://www.dagstuhl.de/16394


vices (Performance
not specific to MS
microservice SPE, different from S
how do the mocks look like? challenge: mimic performance
complex
behavior
=MS
performance unit testing
based on containers
regression testing
fast (collect APM data from production)
slow
SOA
which tests can be done in the pipeline/per commit
clearly defined
(industry
Stan
test via REST API using standard load generators (JMeter, ...)
WSDL, SOAP, ...)
heavy-weight middleware
performance integration testing?
- done only in production? (canary, ...)
- reason: components deployed to production independently
- generic components
- loose coupling
interaction via mid
- service is an API


different from SOA SPE ?
SOA
clearly defined
standards
SOAP
components
interaction via middleware
- service is an API
- integration / composition of 3rd-party service?
MS
- Lightweight REST API
no standard for specifying SLAs
- smart endpoints dump pipes
- more loosely coupled
- deployment explicitly considered
- e.g. container
- pattern for decomposing a single application (MS = component)

A. van Hoorn et al.: Performance Engineering for Microservices – Research Challenges and Directions

Flow of Discussion – Here and There



1 Microservices – What‘s actually new about them?

2 Microservices in performance …

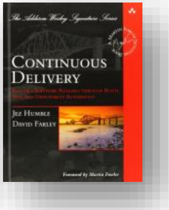
2.1 … Testing

2.2 … Monitoring

2.3 … Modeling and prediction

DevOps and Microservices

DevOps and Continuous Deployment



*“DevOps is a **set of practices** intended to **reduce the time** between committing a change to a system and the **change being placed into normal production**, while **ensuring high quality**.”* – Bass et al., 2015

- Practiced in different magnitudes
  - cars.com => 700 deployments/Year (~ 2 deployments/Day)
  - flickr => 10+ deployments/Day
  - amazon.com => Every 11.6 seconds (~ 7500 deployments/Day)
- Benecifial properties to reduce the time to production
  - No or very few coordination with other teams is required
  - Multiple versions of the same service can be run in the production environment simultaneously
  - Roll back/forward changes made to a running service in the event of errors

Adequate Architectural Style: Microservices

O'REILLY

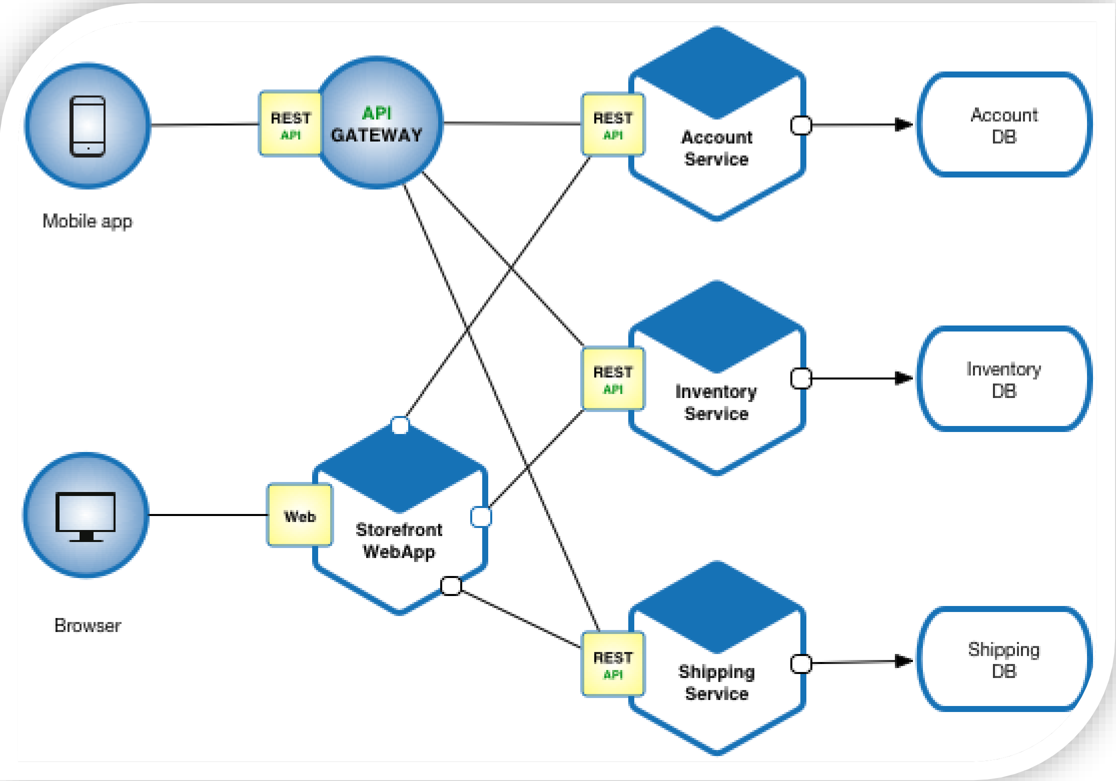
Building Microservices

Sam Newman

Release It!

Design and Deploy Production-Ready Software

Michael T. Nygard



http://microservices.io/patterns/microservices.html

Common Platform: Container-based Virtualization

- **Docker**
  - Open platform for building, shipping and running distributed applications

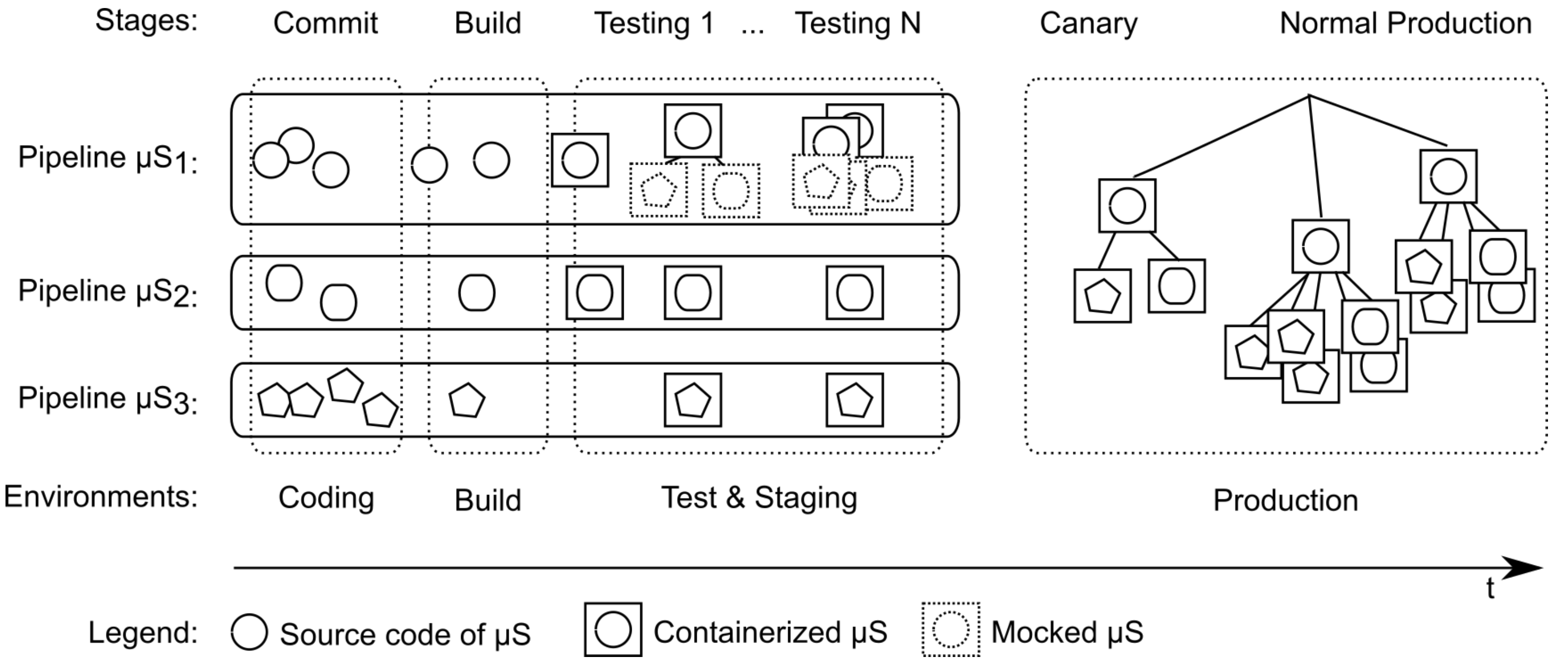
```
$ docker run account
```

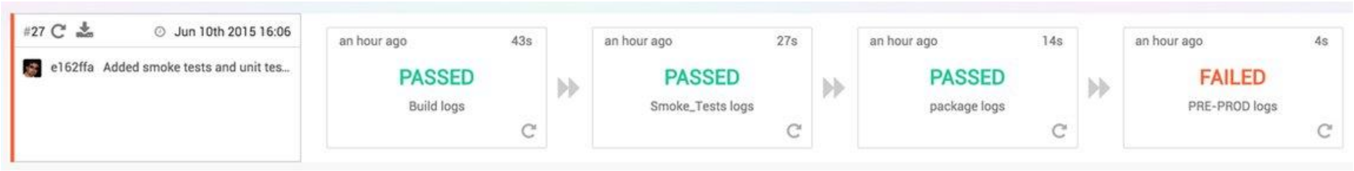
- **Kubernetes**
  - Container cluster manager
  - Orchestrates deployment of containers
  - + much more:

    Load-balancing, auto-scaling, fault-management, monitoring, service discovery, …

```
$ kubectl scale rc account –replicas=3
$ kubectl –rolling-upgrade inventory inventory:5.3
```

Microservices in the CD Pipeline





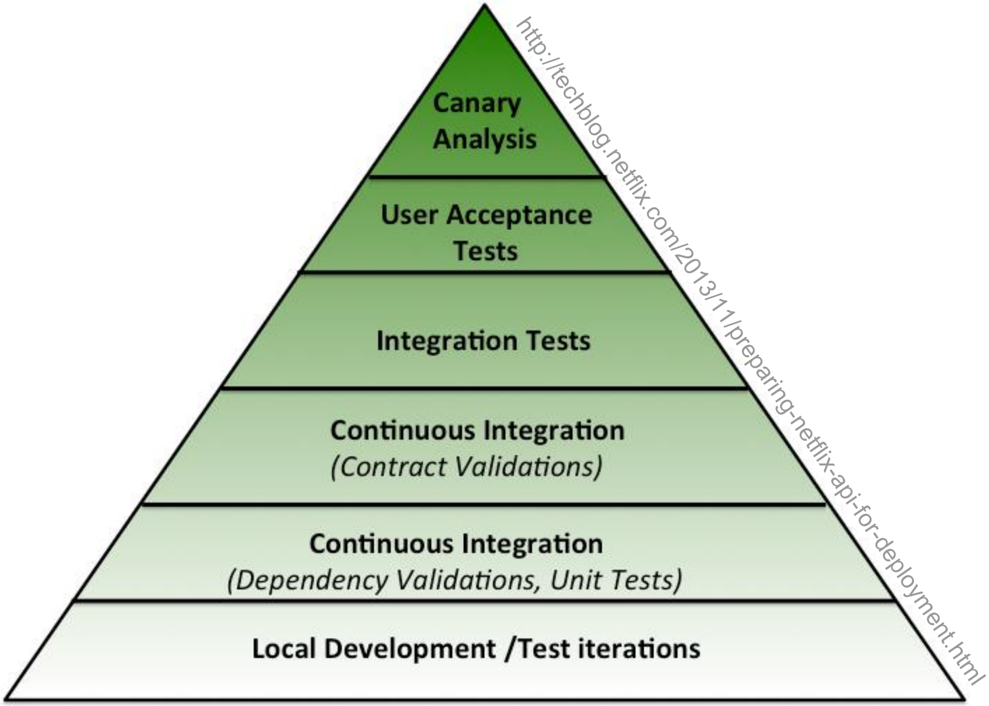
#27 Jun 10th 2015 16:06

e162ffa Added smoke tests and unit tes...

an hour ago 43s	an hour ago 27s	an hour ago 14s	an hour ago 4s
PASSED	PASSED	PASSED	FAILED
Build logs	Smoke_Tests logs	package logs	PRE-PROD logs

Testing

Common Testing Practices for DevOps/Microservices



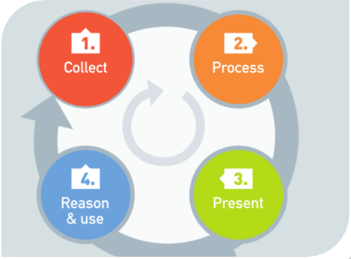
http://techblog.netflix.com/2013/11/preparing-netflix-api-for-deployment.html

Performance Testing Microservices

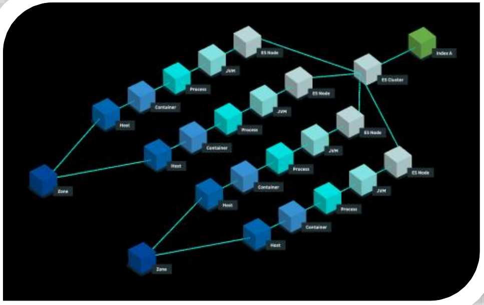
- Conflict with CD practices:
  extensive performance tests vs. pace of stages
- Concerns
  - Which tests are run for each commit?
  - Which tests are run for a consolidated set of commits?
  - Which tests to run in which situations/environments?
- Research directions
  - Splitting test suites by different scopes + (dynamic) test case selection
    - Decide which tests should be performed
  - Innovative online performance testing strategies (canary, …)
  - Use of input from monitoring data, operational failures, and the actual application status
    - Extraction and refinement of tests

Monitoring

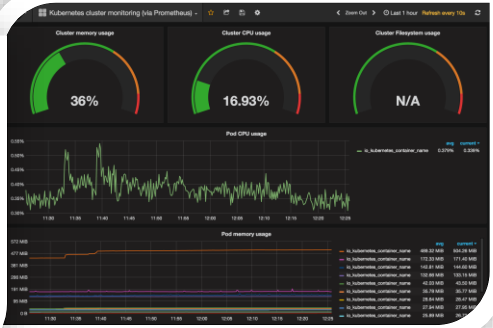
Monitoring Microservices



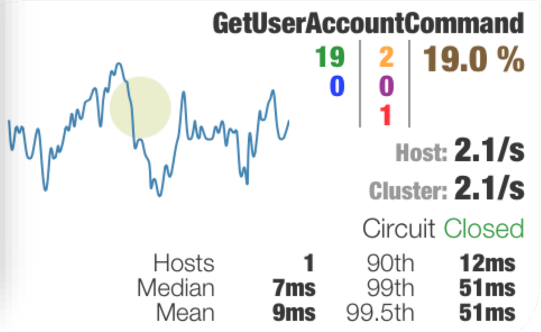
- Monitoring is a first-class property of microservices/DevOps
- Extensive monitoring facilities available
  - Application, infrastructure, and system level:
    - mature APM tools
    - monitoring capabilities in frameworks (Hystrix, …)
  - Container level: built-in monitoring (Docker, Kubernetes, …)



https://www.instana.com/product/



https://docs.giantswarm.io/



GetUserAccountCommand

19 | 2 | 19.0 %
0 | 0
1

Host: 2.1/s

Cluster: 2.1/s

Circuit Closed

Hosts	1	90th	12ms
Median	7ms	99th	51ms
Mean	9ms	99.5th	51ms

https://github.com/Netflix/Hystrix

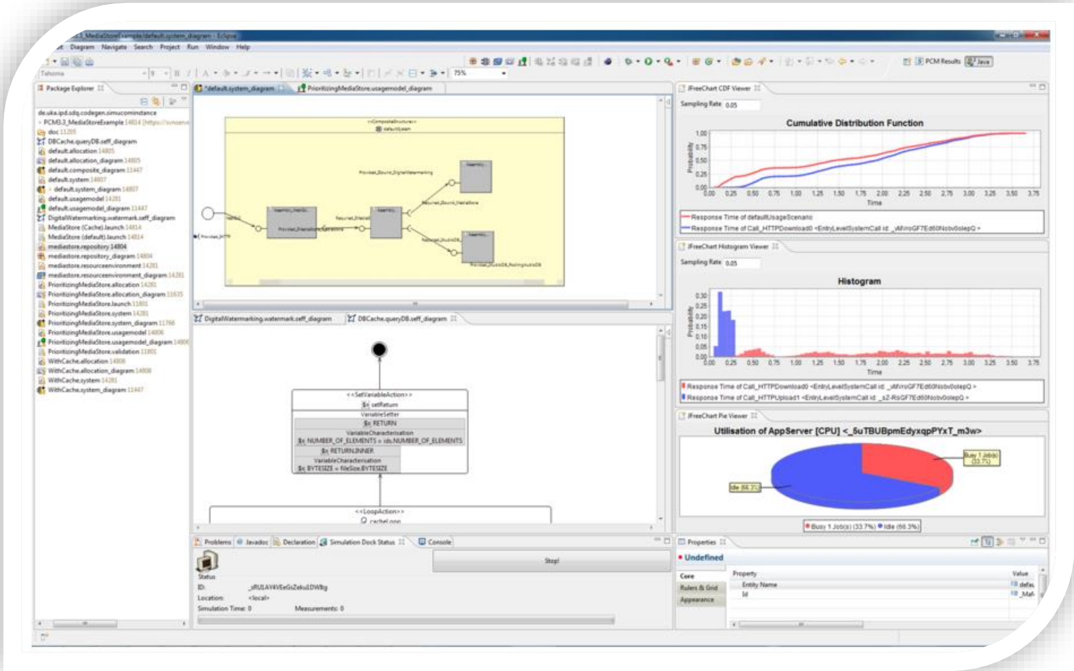
Monitoring Microservices – Challenges

1. Instrumentation challenge: polyglot technology stacks
2. Classic metrics meaningful? New metrics? For instance,
   1. Short-lived (stateles) containers: life time, capacity/elasticity
   2. State of resilience mechanisms (e.g., circuit breakers)
3. Accurate and precise anomaly detection and diagnosis under frequent change

Modeling and Prediction

Performance Modeling of Microservice Architectures

Performance modeling (and prediction) gained considerable attraction and maturity in SPE over the past two decades (UML SPT/MARTE, LQN, Palladio, …)

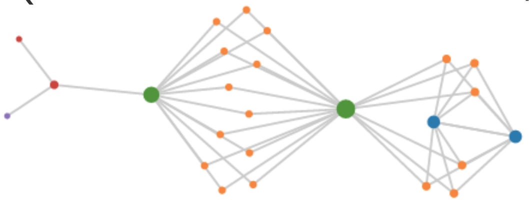

Package Explorer
*default.system_diagram
PrioritizingMediaStore.usagemodel_diagram
DigitalWatermarking.watermark.seff_diagram
DBCache.queryDB.seff_diagram
<<SetVariableAction>>
<<LoopAction>>
PCM Results
Java
JFreeChart CDF Viewer
Sampling Rate 0.05
Cumulative Distribution Function
Probability
Time
Response Time of defaultUsageScenario
Response Time of Call_HTTPDownload0 <EntryLevelSystemCall id: _vMroGF7Ed60NobvOolepQ >
JFreeChart Histogram Viewer
Sampling Rate 0.05
Histogram
Probability
Time
Response Time of Call_HTTPDownload0 <EntryLevelSystemCall id: _vMroGF7Ed60NobvOolepQ >
Response Time of Call_HTTPUpload1 <EntryLevelSystemCall id: _sZ-RsGF7Ed60NobvOolepQ >
JFreeChart Pie Viewer
Utilisation of AppServer [CPU] <_5uTBUBpmEdyxqpPYxT_m3w>
Busy 1 Job(s) (33.7%)
Idle (66.3%)
Busy 1 Job(s) (33.7%) Idle (66.3%)
Problems
Javadoc
Declaration
Simulation Dock Status
Console
Stop!
Properties
Undefined
Core
Rulers & Grid
Appearance
Property
Value
Entity Name
Id

Status
ID: _sRULAY4VEeGsZeku1DWtg
Location: <local>
Simulation Time: 0 Measurements: 0

http://www.palladio-simulator.com/tools/screenshots/

Why Don‘t We Simply Use the Existing Ones?

1. Traditional use case – capacity planning – less relevant (auto-scaling)
   - But: resilience and design of runtime adaptation strategies
2. New abstractions needed to adequately capture the recent advances in deployment technology (container orchestration, scale, …)
   - More leight-weight techniques possible?



https://github.com/adrianco/spigo/

3. Model creation is challenging
   1. Application architecture can be extracted (e.g., from execution traces)
   2. How to learn the characteristics of the execution platform?

Conclusions

Summary of Observations and Challenges

- Observations
  - Microservices are an emerging architectural style enabling the efficient use of cloud technologies and DevOps practices
  - So far, performance engineering for microservices has not gained attraction in the relevant communities
  - Existing performance engineering techniques—focusing on testing, monitoring, and modeling—cannot simply be re-used
- Particular (activity-specific) challenges include
  1. strategies for efficient performance regression testing
  2. performance monitoring under continuous software change
  3. appropriate performance modeling concepts for shifted use cases

Future Directions – Combining Testing, Monitoring, Modeling

- The use of performance models to prioritize tests cases, including the decision whether and when (pipeline vs. production) tests are executed
- The use of monitoring data to create and refine performance tests, including the creation of representative usage profiles and performance-aware service mockups
- The use of tests and resulting monitoring data to automatically create performance models, including the combination of data from different environments
- The use of models to guide the diagnosis of performance problems
- Learning and assessing deployment strategies based on monitoring data

Advertisement / Call for Collaborations

- **SPEC RG DevOps Performance**

Mission: foster and facilitate research in combining **measurement-based** application performance management (APM) and **model-based** software performance engineering (SPE) activities for **business-critical application systems**.

http://research.spec.org/devopswg

- **CASPA: A Platform for Comparability of Architecture-based Software Performance Engineering Approaches**

  Düllmann et al., ICSA ‘17

CASPA:
A Platform for Comparability of Architecture-based Software Performance Engineering Approaches

Thomas F. Düllmann[1], Robert Heinrich[2], André van Hoorn[1], Teerat Pitakrat[1], Jürgen Walter[3], Felix Willnecker[4]

[1]University of Stuttgart, Germany
[2]Karlsruhe Institute of Technology, Germany
[3]University of Würzburg, Germany
[4]fortiss GmbH, Germany

Abstract—Setting up an experimental evaluation for architecture-based Software Performance Engineering (SPE) approaches requires enormous efforts. This includes the selection and installation of representative applications, usage profiles, supporting tools, infrastructures, etc. Quantitative comparisons with related approaches are hardly possible due to limited repeatability of previous experiments by other researchers.

This paper presents CASPA, a ready-to-use and extensible evaluation platform that already includes example applications and state-of-the-art SPE components, such as monitoring and [illegible]

To address these problems, we have developed an evaluation platform for SPE approaches, called CASPA.[1] It includes hooks for building blocks for typical SPE evaluations, comprising an AUT, a workload generator, a monitoring tool, and analysis approaches. Different implementations of the building blocks are already available and custom ones can be added easily. The platform comes with scripts allowing to setup the platform and start the experiments with ease. The platform is based on state-of-the-art technologies, such as container-based virtualization, which has been proposed as a promising approach for improving reproducibility in software engineering [4].

[illegible]

https://github.com/spec-rgdevops/CASPA-platform