Performance Aware Cloud Computing

17th Symposium and Summer School On Service-Oriented Computing
June 27 2023

Sebastian Böhm, Johannes Manner, and Guido Wirtz

Distributed Systems Group
Otto-Friedrich-University Bamberg, Germany

Agenda

- **Introduction** - Research @DSG

  ... how we got to the topics to discuss today ...

- **Performance Aware Cloud Computing – Common Issues**
  - Documentation
  - Multi-Core Environments: Resource Config. ←→ Execution Behavior
  - Resource Unaware Open Source Research
  - Real-World Conditions, Reliability, …

- **Performance Aware Cloud Computing – Solution Approaches**
  - Use a Unified and Configurable Framework for Cloud-Edge Orchestration
  - Make Hardware Stacks and Experiments Comparable

- **Concluding Thoughts**

Research @DSG

RL — Evaluating the Quality of Cloud-native Software Architectures

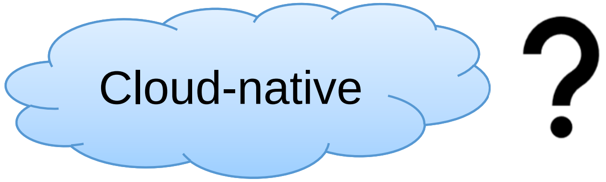
SW — Integration Testing of Serverless Applications

SB — Universal and Lightweight Cloud-Edge Orchestration

JM — A Simulation Framework for Function as a Service

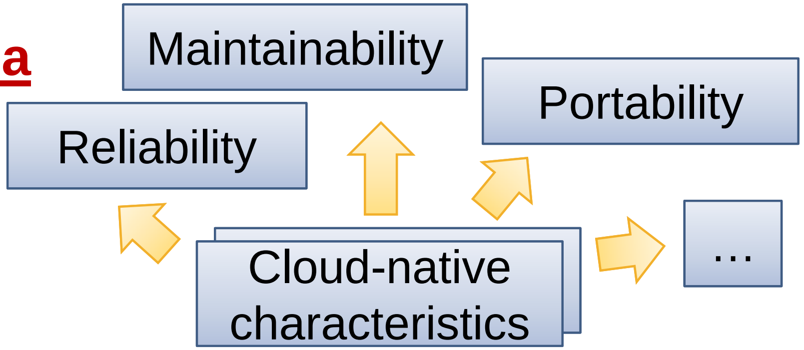
RL Evaluating the Quality of Cloud-native Software Architectures

Problem



- Popular and associated with many advantages (exploit all benefits of modern cloud environments)
- Covering a broad scope (from infrastructure to application characteristics & from implementation to operation)

Idea



A hierarchical, **multi-dimensional quality model**, that:

- explains how quality aspects are impacted by architectural characteristics associated with cloud-native
- enables quality evaluations of modeled software architectures based on the model elements

Challenges

- How can cloud-native characteristics be conceptualized and structured in a hierarchy of quality aspects?

  Lichtenthäler, R.; Wirtz, G.: Towards a Quality Model for Cloud-native Applications
  Service-Oriented and Cloud Computing. ESOCC 2022, Wittenberg, Germany (virtually), March 22-24, 2022

- How to model cloud-native software architectures to enable quality evaluations with the quality model?

  Dürr, K., Lichtenthäler, R.: An Evaluation of Modeling Options for Cloud-native Application Architectures to Enable Quality Investigations
  IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC) Companion, Vancouver, WA, USA, 6-9 December 2022

- How can the quality model be practically applied to application architectures? **work in progress ...**

  Lichtenthäler, R.; Fritsch, J., Wirtz, G.: Cloud-Native Architectural Characteristics and their Impacts on Software Quality: A Validation Survey. 17th IEEE Int. Conference on Service-Oriented Systems Engineering. SOSE2023, Athens, Greece, July 17-20, 2023

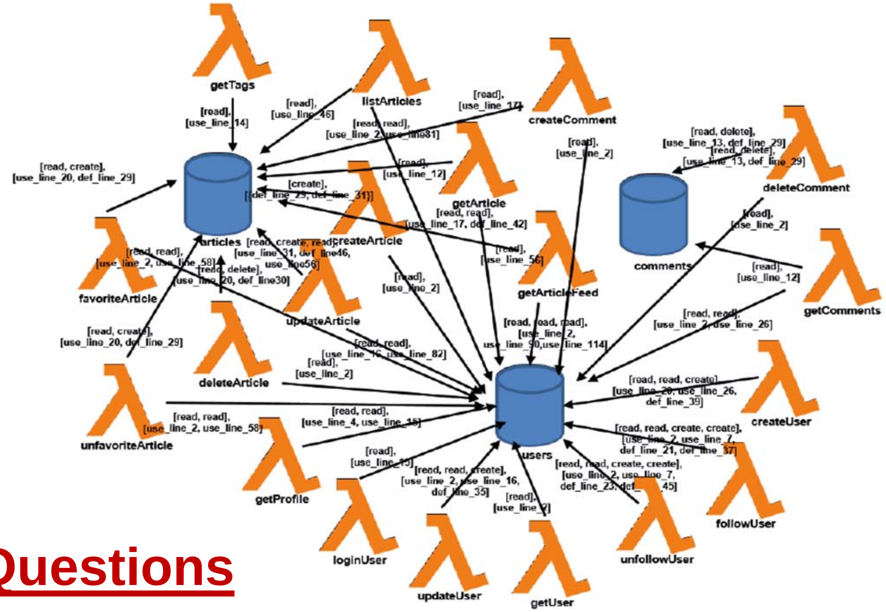
see https://r0light.github.io/cna-quality-model/

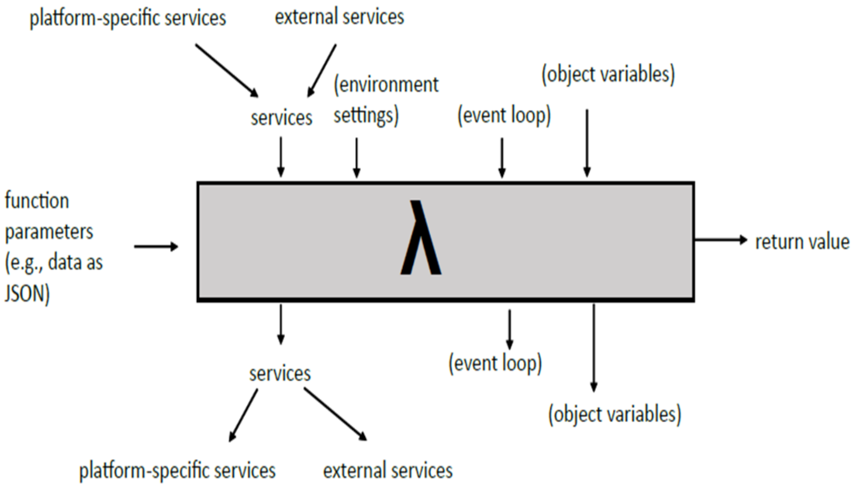
Integration Testing of Serverless Applications

Problem

- Testing single serverless functions is simple … BUT
- Combining various serverless functions with other cloud services ➔ complex serverless applications

How to test the integration of these applications?





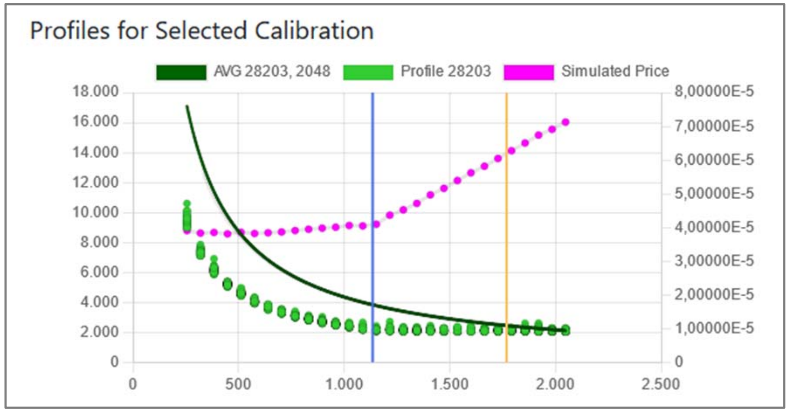
Research Questions

- Suitable model for integration testing?
- Useful coverage criteria and how to measure them?
- How to create test cases automatically?
- What test cases are most efficient?

Winzinger, St., Wirtz, G.: Data Flow Testing of Serverless Functions.
11th International Conference on Cloud Computing and Services Science. CLOSER 2021: 56-64.

Winzinger, St., Wirtz, G.: Automatic Test Case Generation for Serverless Applications.
16th IEEE Int. Conference on Service-Oriented Systems Engineering. SOSE2022, Newark, CA, USA, August 15-18, 2022

Winzinger, St., Wirtz, G.: Comparison of Integration Coverage Criteria for Serverless Applications.
17th IEEE Int. Conference on Service-Oriented Systems Engineering. SOSE2023, Athens, Greece, July 17-20, 2023



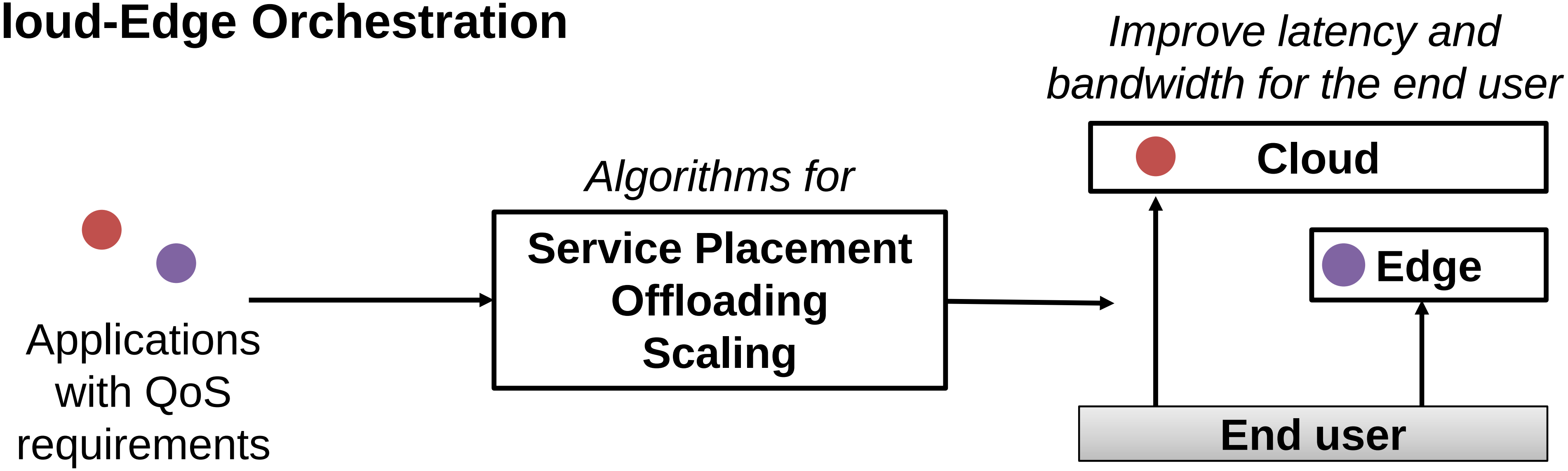
Overall Aim

Run your cloud function in various resource settings to simulate expected behavior at FaaS platforms.

Research Questions

- ❑ How can dev-prod parity be achieved?
  - Two distinct virtualized environments
  - Public cloud offerings vs open-source solutions
- ❑ How can developers be supported to make reasonable decisions?

Cloud-Edge Orchestration



Idea: Fostering the general applicability and ease of use of sophisticated algorithms for service placement, offloading, and scaling

- Research Question -

How can a universal and lightweight platform for cloud-edge orchestration be designed that fosters general applicability, reproducibility, and comparability?

Performance Aware Cloud Computing - Problems

Performance in cloud environments is directly related to utilization, latency and cost. Often, results of experiments are overrepresented whereas the configuration of machines, applications, workloads etc. is not documented but essential for a proper interpretation of results.

Awareness for some Common Issues

I1 — **Experiments are not documented in a self-contained way to reproduce results**

I2 — **Multi-Core environments and their impact on execution behavior are neglected**

I3 — **Experiments with open-source software are resource unaware**

I4 — **Experiments do not consider real-world conditions and reliability requirements**

MJ1

Performance Aware Cloud Computing - Problems

Performance in cloud environments is directly related to utilization, latency and cost. Often, results of experiments are overrepresented whereas the configuration of machines, applications, workloads etc. is not documented but indispensable for a proper interpretation of results.

Don’t Get Blindsided by Overlooking these Blind Spots

I1 — **Blind Spot: Reproducible Documentations**

I2 — **Blind Spot: Multi-Core Environments**

I3 — **Blind Spot: Open-Source Machine Configs**

I4 — **Blind Spot: Real-world Conditions and Reliability Requirements**

Folie 9

MJ1 Ich habe mal versucht die 4 Issues etwas kürzer, prägnanter zu fassen und finde das blinde Flecken eine schöne Analogie wären - sicherlich sind die Sachen allen irgendwie auch bewusst aber werden dann vielleicht einfach übersehen.

Manner, Johannes; 16.06.2023

Blind Spot: Reproducible Documentation

Several secondary studies confirm this issue:

1. 3 out of 26 experiments reproducible
   J. Kuhlenkamp and S. Werner, "Benchmarking FaaS Platforms: Call for Community Participation," Proc. of WoSC, 2018.
2. Majority of 122 papers not reproducible
   T. Kalibera and R. Jones, "Rigorous benchmarking in reasonable time," Proc. of ISMM, 2013
3. 26% of 315 data projects published raw data
   J. Couture et al., "A funder-imposed data publication requirement seldom inspired data sharing," PLOS ONE, vol. 13, no. 7, 2018
4. 35% of 98 experiments used proprietary components and test beds
   S. Smolka and Z. Á. Mann, "Evaluation of fog application placement algorithms: a survey," Computing, vol. 104, no. 6, 2022.

Solutions to this issue:

- ❑ Publisher guidelines to submit raw data and results
- ❑ Research data management platforms like Zenodo
- ❑ Guidelines for documenting research and experiments
- ❑ Standardization with Reference Architectures

Documentation Guidolines for Performing Benchmarks (for FaaS Benchmarks)

- Physical Machine Configuration (CPU model, RAM, OS version etc.), especially for local experiments
- VM/Container Identification
- Function Configuration (memory setting)
- Runtime, Programming Language
- Experiment Time, Duration and Number of Runs
- Data Measurement Procedure
  (Server Processing Time, Client Perceived Performance etc.)
- Workload over Time
- Function/Application Characteristics (e.g. Number of Database Pools, multithreaded implementations)

Manner, J.: SeMoDe – Simulation and Benchmarking Pipeline for Function as a Service
Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 105, University of Bamberg, November 2021.

Documentation Guidolines for Performing Benchmarks (for FaaS Benchmarks)

- **Physical Machine Configuration (CPU model, RAM, OS version etc.), especially for local experiments**
- VM/Container Identification
- Function Configuration (memory setting)
- **Runtime, Programming Language**
- Experiment Time, Duration and Number of Runs
- **Data Measurement Procedure**
  (Server Processing Time, Client Perceived Performance etc.)
- Workload over Time
- Function/Application Characteristics (e.g. Number of Database Pools, multithreaded implementations)

Manner, J.: SeMoDe – Simulation and Benchmarking Pipeline for Function as a Service
Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 105, University of Bamberg, November 2021.

Performance Aware Cloud Computing - Problems

Performance in cloud environments is directly related to utilization, latency and cost. Often, results of experiments are overrepresented whereas the configuration of machines, applications, workloads etc. is not documented but indispensable for a proper interpretation of results.

Awareness for Common Issues

I1 — **Blind Spot: Reproducible Documentations**

I2 — **Blind Spot: Multi-Core Environments**

I3 — **Blind Spot: Open-Source Machine Configs**

I4 — **Blind Spot: Real-World Conditions and Reliability Requirements**

Blind Spot: Multi-Core Environments – Misinterpretation

Situation

- CPU and other resources are scaled based on the selected memory setting
- Authors did not mention nor consider multi-threaded functions

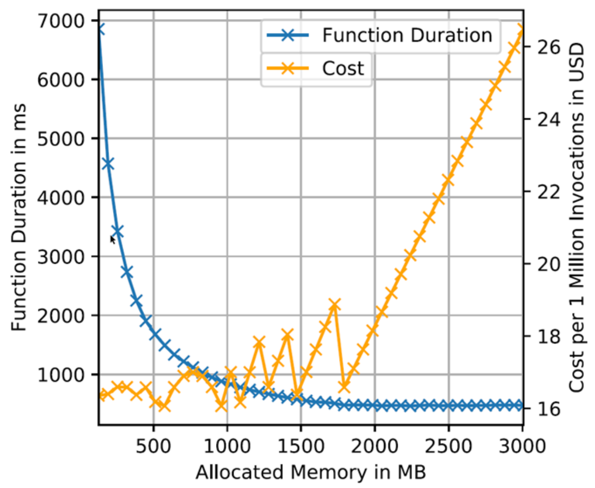
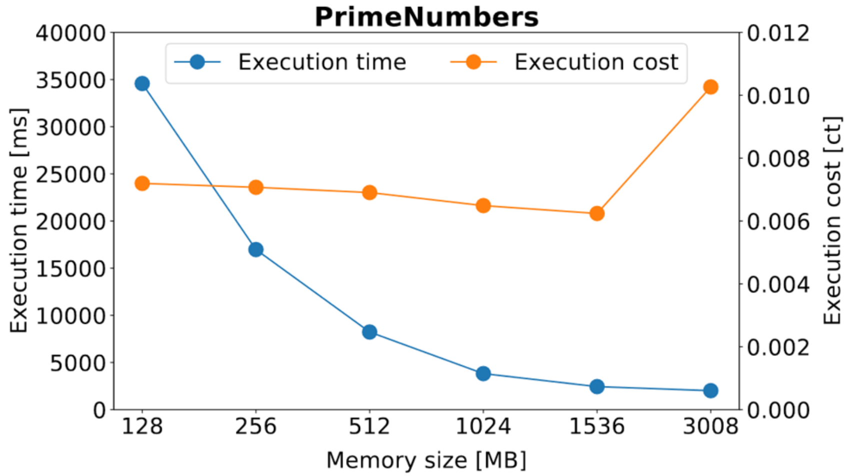


Figure 2 in C. Lin and H. Khazaei, “Modeling and optimization of performance and cost of serverless applications,” IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 3, pp. 615–632, 2021.

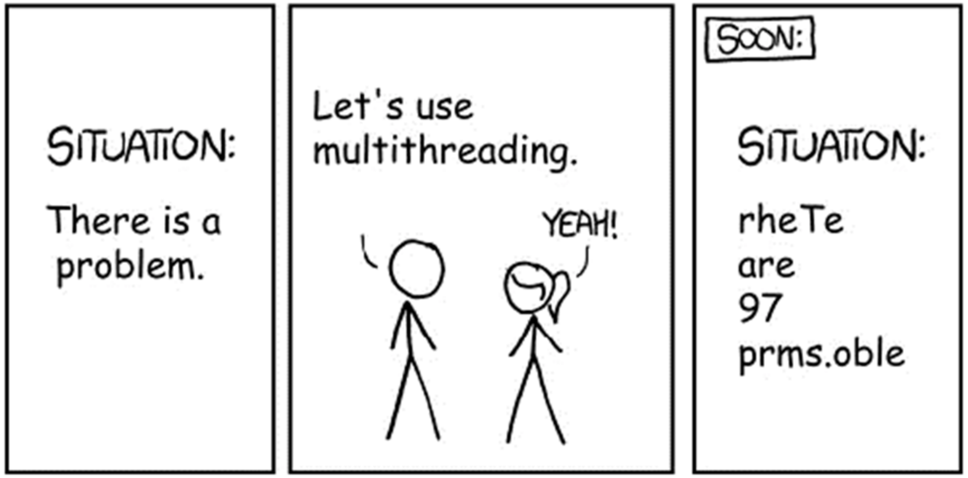


Part of Figure 1 in S. Eismann et al., “Sizeless: Predicting the optimal size of serverless functions,” arXiv e-Prints: 2010.15162, 2020.

Effect: Authors ‘Wonder’ why they face a plateau in execution time and a rapid increase in cost

HINT: Documentation at AWS states that 1769MB is 1 CPU equivalent

Blind Spot: Multi-Core Environments – Attempted Explanation



https://www.reddit.com/r/ProgrammerHumor/comments/dtiufv/multithreading_fixing_a_problem/

- Inherent complex, hard to track over time – therefore neglected
- Computer science education – single threaded per default
- Middleware frameworks like SpringBoot hide multi-threading
- Application use cases – various functions running in parallel and comprise an application
  → making phenomena like in the previous slide not apparent

Manner, J., Böhm, S.: Lecture Notes: Concurrency Topics in Java
Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik Nr. 106, University of Bamberg, April 2022.

Performance Aware Cloud Computing - Problems

Performance in cloud environments is directly related to utilization, latency and cost. Often, results of experiments are overrepresented whereas the configuration of machines, applications, workloads etc. is not documented but indispensable for a proper interpretation of results.

Awareness for Common Issues

I1 — **Blind Spot: Reproducible Documentations**

I2 — **Blind Spot: Multi-Core Environments**

I3 — **Blind Spot: Open-Source Machine Configs**

I4 — **Blind Spot: Real-World Conditions and Reliability Requirements**

Blind Spot: Open-Source Machine/Experiment Configs – Situation

Feature comparisons of open-source and cloud platforms exist, but…

… they use the preconfigured defaults of the tools

→ No fair comparison of cloud and on-premise hosted solutions possible

→ „assume cost of executing a function in the private cloud to be zero“

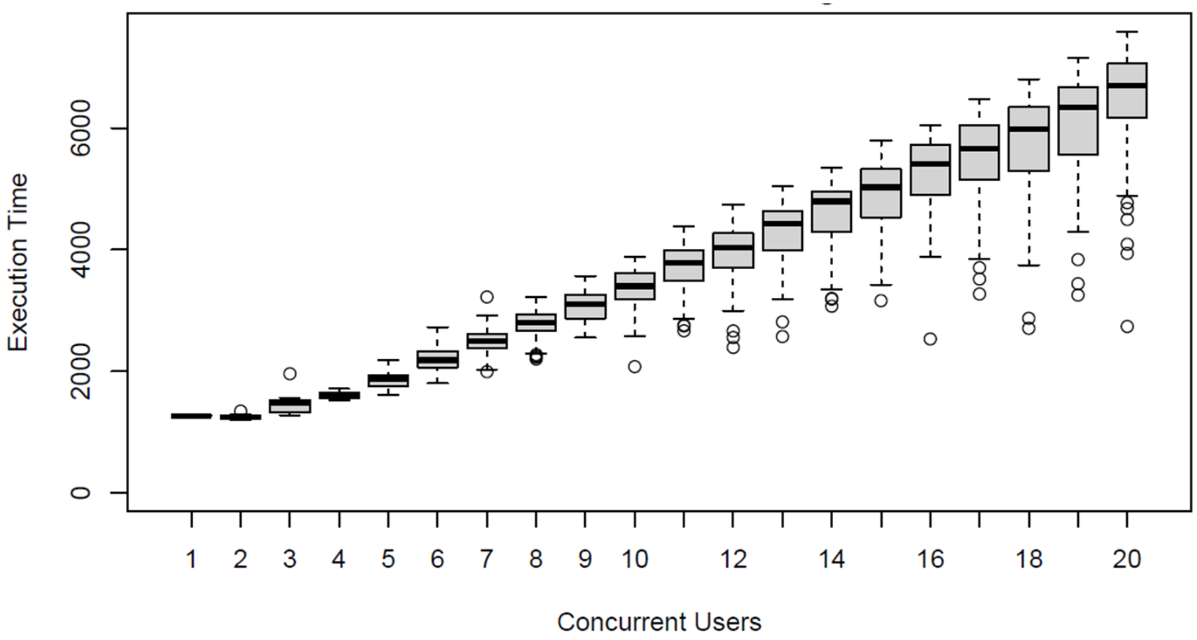
[Das2020, p.610]

→ K8s often used as highler level abstraction and foundation for open-source tools

A. Das et al., "Skedulix: Hybrid cloud scheduling for cost-efficient execution of serverless applications".
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), 2020.

Manner, J., Wirtz, G.: Resource Scaling Strategies for Open-Source FaaS Platforms compared to Commercial Cloud Offerings
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), 2022.

Blind Spot: Open-Source Machine/Experiment Configs – Problem



- Deployment: Single Node K8s cluster
- No open source publication does state explicit resource restrictions
- Noisy Neighbour already addressed for public cloud FaaS platforms

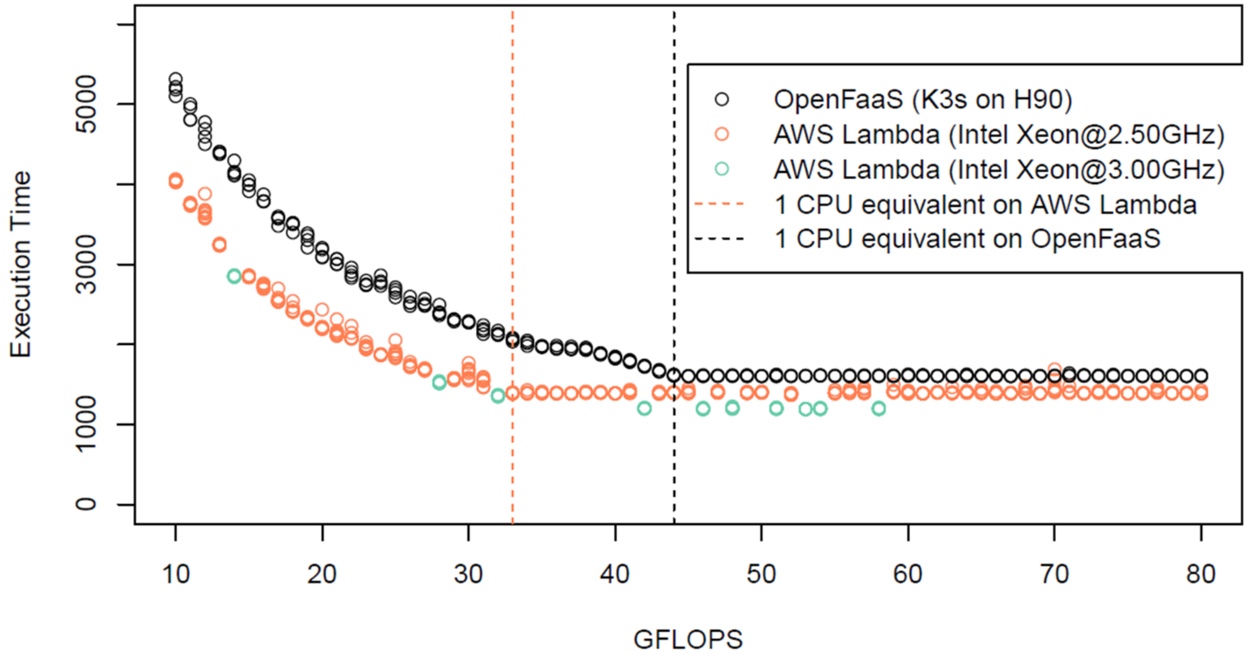
  D. Barcelona-Pons and P. Garcia-Lopez, "Benchmarking parallelism in faas platforms," 2020.

Effect: Pods compete for the resources -> performance degradation

Manner, J., Wirtz, G.: Resource Scaling Strategies for Open-Source FaaS Platforms compared to Commercial Cloud Offerings
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), Barcelona, Spain, July 11 - July 16, 2022.

Blind Spot: Open-Source Machine/Experiment Configs – Solution

- Restricting resources for pods based on K8s limits



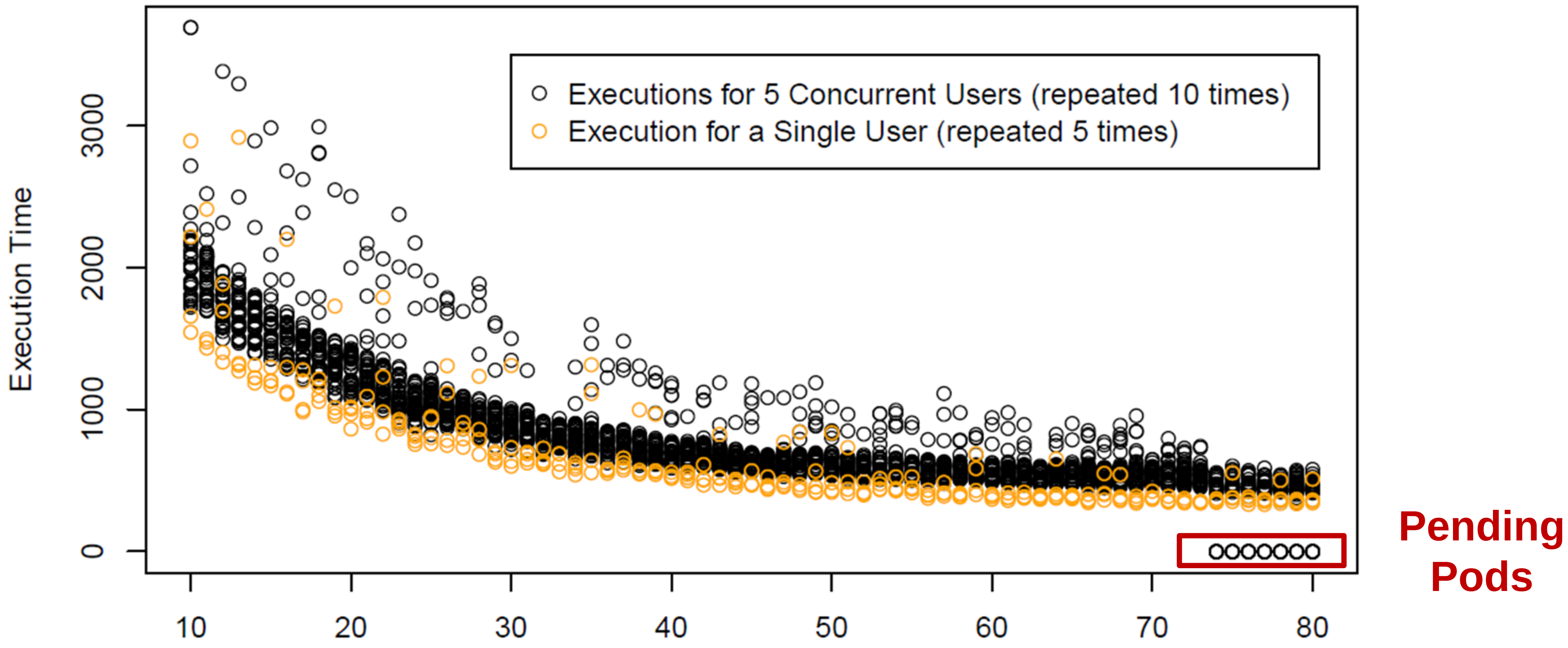
Problems

- Hardware heterogeneity on AWS Lambda (excluded 3GHz executions)
- One CPU equivalent is different (pay attention between the vertical lines)

Manner, J., Wirtz, G.: Resource Scaling Strategies for Open-Source FaaS Platforms compared to Commercial Cloud Offerings
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), Barcelona, Spain, July 11 - July 16, 2022.

Resource Aware Open-Source Research
Noisy Neigbour Solution

- Use our QoS layer (K8s resource limits)



→ enforce a resource aware scheduling of workloads

Manner, J., Wirtz, G.: Resource Scaling Strategies for Open-Source FaaS Platforms compared to Commercial Cloud Offerings
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), Barcelona, Spain, July 11 - July 16, 2022.

Performance Aware Cloud Computing - Problems

Performance in cloud environments is directly related to utilization, latency and cost. Often, results of experiments are overrepresented whereas the configuration of machines, applications, workloads etc. is not documented but indispensable for a proper interpretation of results.

Awareness for Common Issues

I1 — **Blind Spot: Reproducible Documentations**

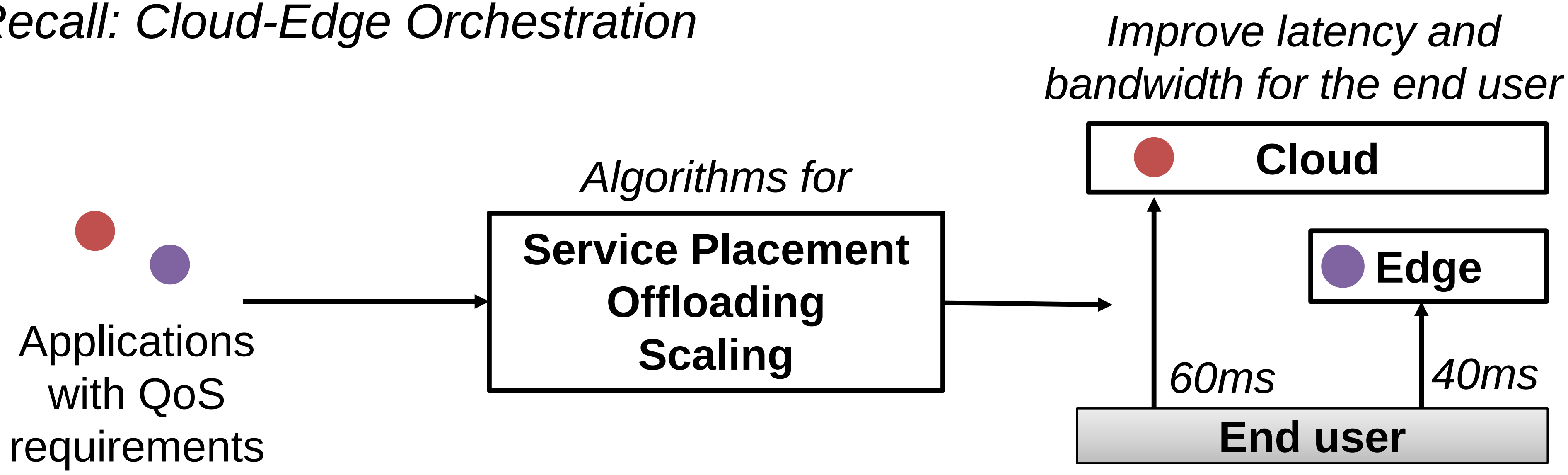
I2 — **Blind Spot: Multi-Core Environments**

I3 — **Blind Spot: Open-Source Machine Configs**

I4 — **Blind Spot: Real-World Conditions and Reliability Requirements**

Blind Spot: Real-World Conditions and Reliability Requirements

Recall: Cloud-Edge Orchestration



- Applications require a particular QoS (*e.g., deadline for a response*)
- Heterogeneous and resource-constrained nodes work in **unstable** environments and are faced by **volatile network conditions**

Proper monitoring of resources and network conditions is inevitable!

Cloud-Edge Orchestration

Frequent:

- Static resource management
- Static *latency* and *bandwidth*

Missing consideration of

- Dynamic resource utilization
- Real-time *latency* and *bandwidth*
- Energy

Table 1: Comparison of Resource-Awareness in Different Kubernetes Implementations for Cloud-Edge Architectures.

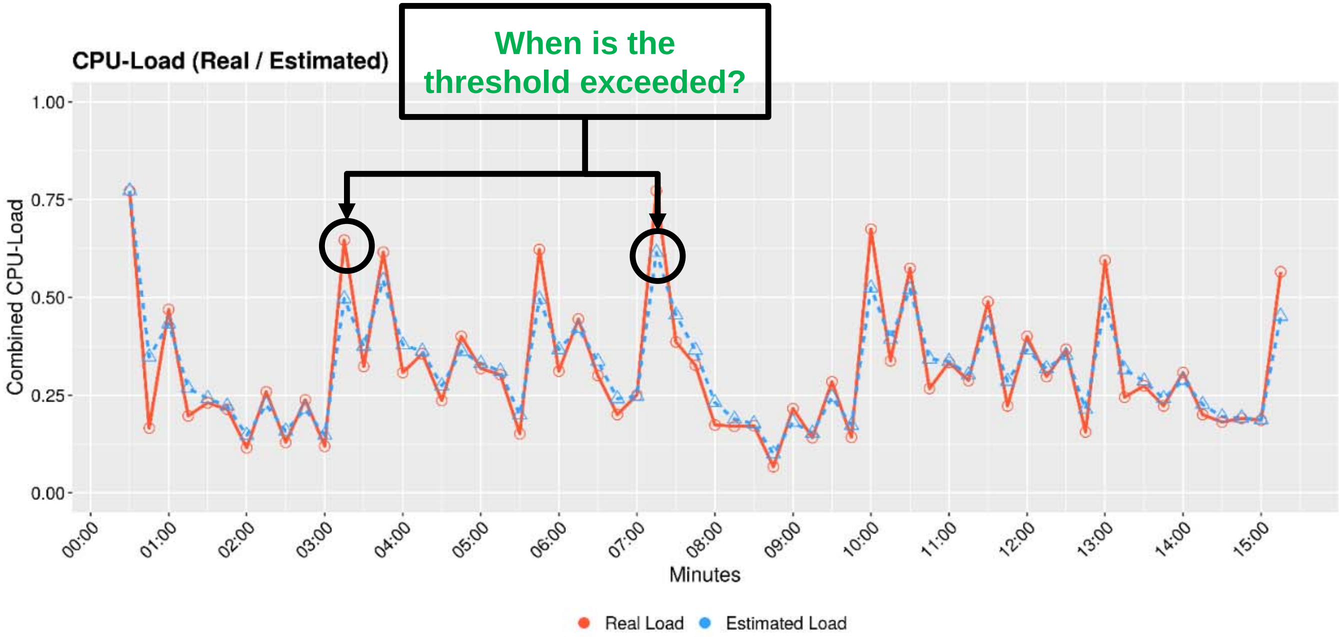
Authors	Year	Layer: *Cloud*	Layer: *Edge*	Layer: *IoT*	Scheduler: *Recompiled*	Scheduler: *Native*	Scheduler: *Extender*	Scheduler: *Custom*	Resources: *CPU*	Resources: *Memory*	Resources: *Disk*	Resources: *Energy*	Network: *Latency*	Network: *Bandwidth*	Metrics: *K8s API*	Metrics: *Custom*
[17] Han	2021	●	●					●	●	●	●		●			●
[22] Kayal	2020		●		●				❍	❍	❍		❍		❍	
[21] Kaur	2020	●	●					●	●	●		●		●		✲
[12] Eidenbenz	2020		●			❍		●	❍	❍	❍		●		●	
[15] Goethals	2020	●	●					●	❍	❍	❍		●		❍	●
[27] Ogbuachi	2020		●					●	●	●	●		●			●
[33] Santos	2019	●	●			❍	●	●	❍	❍	❍		❍	❍	❍	
[11] Casquero	2019		●					●	❍	❍	❍				❍	
[16] Haja	2019	●	●					●	❍	❍	❍		●		❍	●
[20] Javed	2018	●	●			❍			❍	❍	❍				❍	
[44] Wöbker	2018	●	●			❍			❍	❍	❍				❍	

●=fully supported; ❍=partially supported; *no circle* = not supported; ✲ = n/a

S. Böhm and G. Wirtz, "Towards Orchestration of Cloud-Edge Architectures with Kubernetes," in *Science and Technologies for Smart Cities*, LNCS, Springer, 2022, pp. 207–230.

Neglected Real-World Conditions and Reliability Requirements

- Measured real-time resource utilization (*e.g., CPU, memory, and I/O utilization*) are subject to variation over time
- Most approaches for orchestration activities do not clarify how they *obtained*, *transformed* and *smoothed* the data



Example CPU-Load:

- **Raw data** may lead to unwanted operations
- **Update interval** is necessary to reproduce results
- Important for Offloading and Scaling – **When** should the **activities be triggered**?

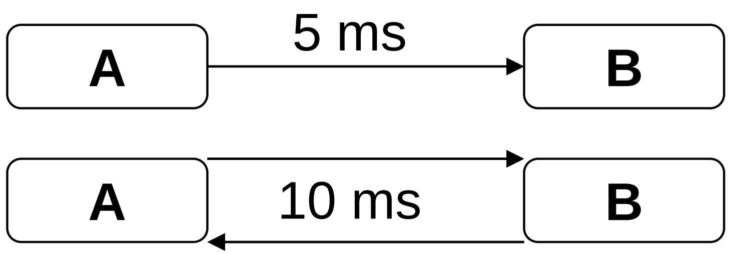
S. Böhm and G. Wirtz, “A Quantitative Evaluation Approach for Edge Orchestration Strategies,” in *Service-Oriented Computing*, Springer, 2020, pp. 127–147.

I4 Neglected Real-World Conditions and Reliability Requirements

- Latency (=delay) is a manifold term
- Example: *The average latency between device A and device B is (set to) 5 ms.*

Problems:

- Different types of delays:
  - End-to-end delay
  - Round-trip-delay
  - End-to-end delay half the round-trip delay is **only an approximation!**

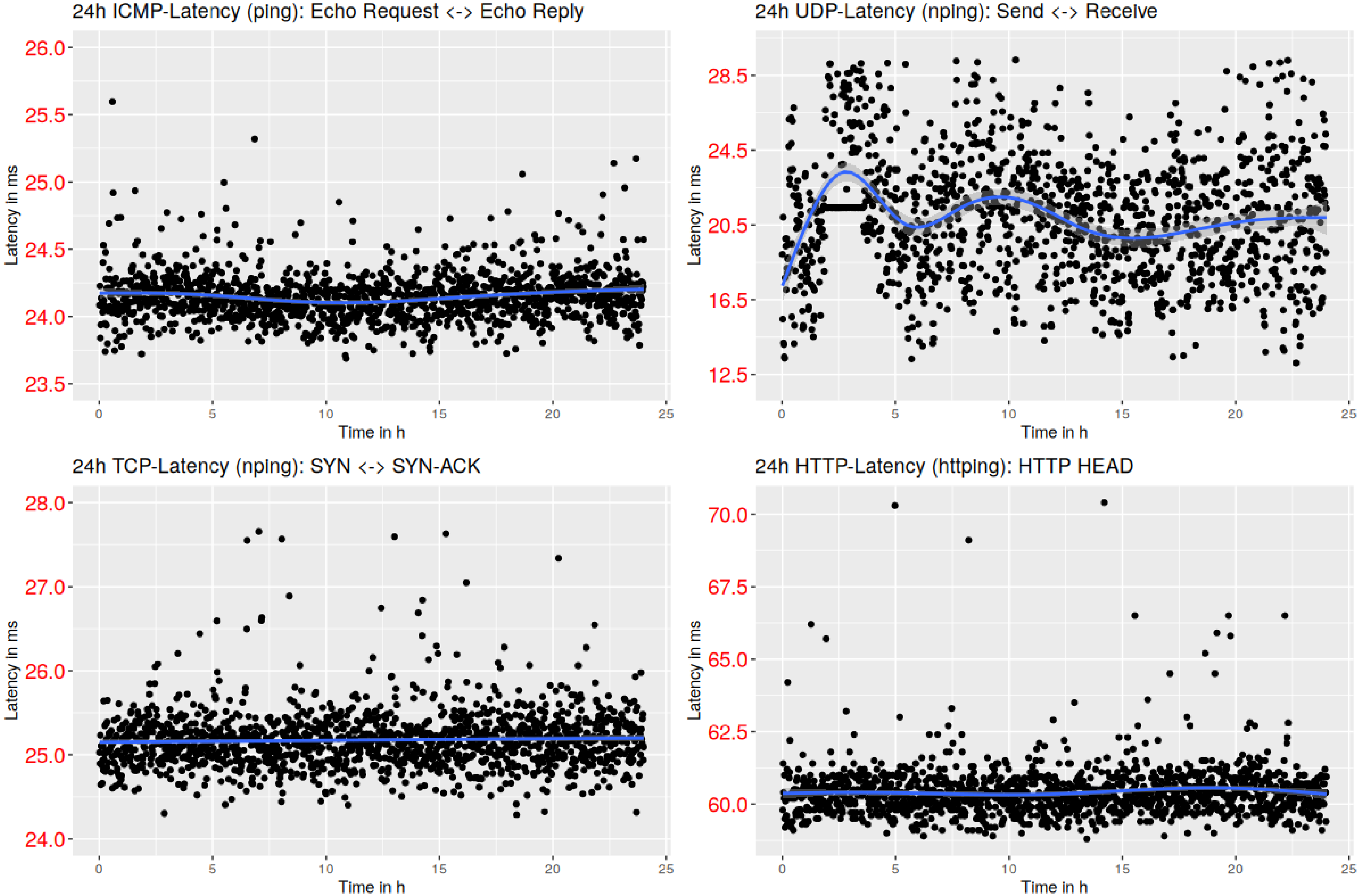


- Different ways to measure:
  - Ping via **ICMP**, **UDP**, or **TCP**? **=> application-aligned monitoring!**
  - Even ping via **HTTP** is possible (HTTP HEAD)
  - End-user perceived latency (=time between request and response)

➔ Clearly define on how the latency between nodes is measured!

Neglected Real-World Conditions and Reliability Requirements

Overview: Different RTDs between *Bamberg, DE* and R niijsr jjwßSQ



Neglected Real-World Conditions and Reliability Requirements

- High availability and fault tolerance are poorly supported
- Even Kubernetes-based orchestration architectures lack full fault tolerance (*e.g., on-premises without cloud*)
- Missing replication of container registries impact orchestration activities

Table 2: Comparison of Architectural Capabilities in Different Kubernetes Implementations for Cloud-Edge Architectures.

Authors	Year	Topology			Provision model			Fault tolerance			Container registry		
		Cloud	*Edge*	*IoT*	*Scaling*	*Offloading*	*Edge-only*	*Cluster*	*Control plane*	*Cluster storage*	*Cloud*	*Edge*	*Replicated*
[17] Han	2021	●	●		●	●	●	●			●		
[22] Kayal	2020		●				●				●		
[21] Kaur	2020	❍	❍		❍	❍	●	●				✲	
[12] Eidenbenz	2020		●				●					✲	
[15] Goethals	2020		●				●					✲	
[27] Ogbuachi	2020		●				●				●		
[33] Santos	2019	●	●		●	❍	●				●	●	●
[11] Casquero	2019		●				●				●		
[16] Haja	2019	●	●			●	●					✲	
[20] Javed	2018	❍	❍		❍	❍	❍		●	❍		✲	
[44] Wöbker	2018	●	●		●	❍	●					●	

●=fully supported; ❍=partially supported; *no circle* = not supported; ✲ = n/a

S. Böhm and G. Wirtz, "Towards Orchestration of Cloud-Edge Architectures with Kubernetes," in *Science and Technologies for Smart Cities*, LNCS, Springer, 2022, pp. 207–230.

Performance Aware Cloud Computing - Solutions

S1 **Use a unified and configurable framework for cloud-edge orchestration**

S2 **Calibrate testbeds and document it**

Performance Aware Cloud Computing - Solutions

S1 **Use a unified and configurable framework for cloud-edge orchestration**

S2 **Calibrate testbeds and document it**

S1 Generic Cloud-Edge Orchestration Architecture

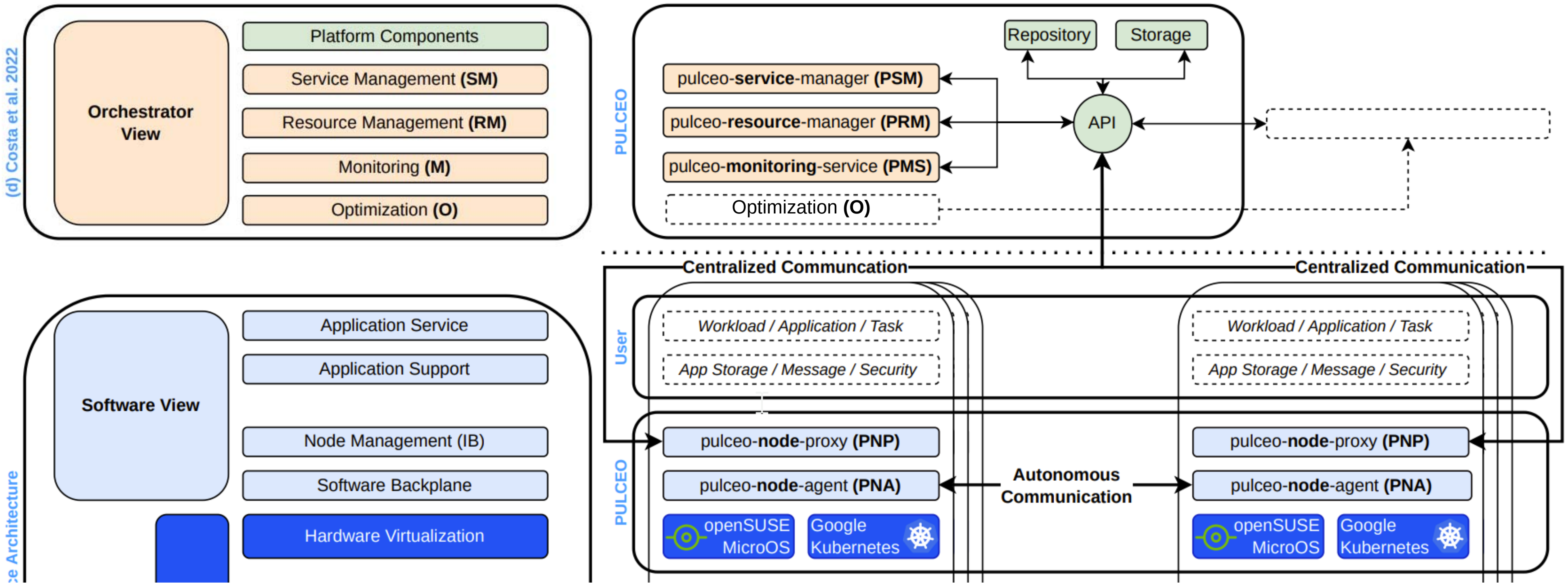
- **Context:** The majority of solutions uses **custom** and **proprietary components** to realize edge-agnostic orchestration capabilities
- **Symptoms:** Custom and **repetitive implementation** complicates **benchmarking** and **comparing** solution approaches
- **Idea:** Implementation of **configurable** and **reusable** components

Component	**Service Management**	**Resource Management**	**Monitoring**	**Optimization**
Subcomponents	**QoS**	**Inventory**	**Infrastructure**	**Placement**
Example	*8 ms application response time*	*Cloud and Edge Nodes*	*Continuous Monitoring*	*Constrained Satisfaction Problem*

B. Costa, J. Bachiega, L. R. de Carvalho, and A. P. F. Araujo, “Orchestration in Fog Computing: A Comprehensive Survey,” *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–34, Feb. 2023.

Decoupling the Orchestration Logic via an API

- **Fact:** The solution approaches for orchestration activities follow different techniques, partly with high complexity and long running operations
- **Example:** Genetic algorithms, machine learning (CNNs), greedy approaches, exact solutions, etc.
- To enable 3rd-party orchestration techniques, decoupling is necessary!

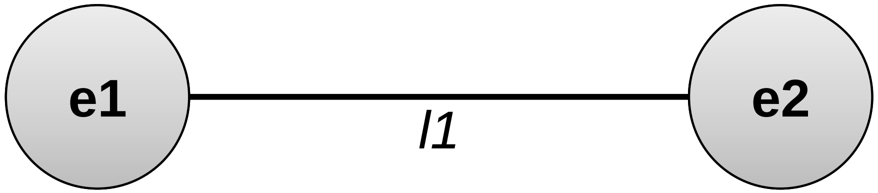


Böhm S., Wirtz, G.: PULCEO - A Novel Architecture for Universal and Lightweight Cloud-Edge Orchestration
Proceedings of the IEEE International Conference on Service-Oriented System Engineering (SOSE), 2023.

Example (1): API-Driven Cloud-Edge Orchestration

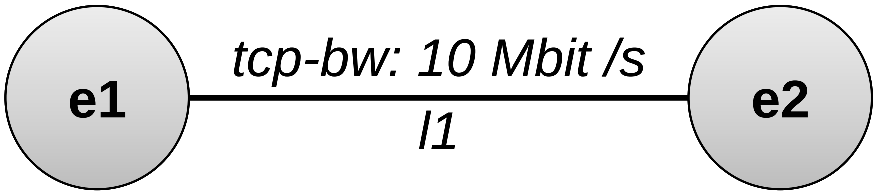

```
POST /api/v1/nodes
{ id: e1, type: edge, sku: B2S, location: frankfurt }
```

200 OK `{id: e1}`...same for e2…



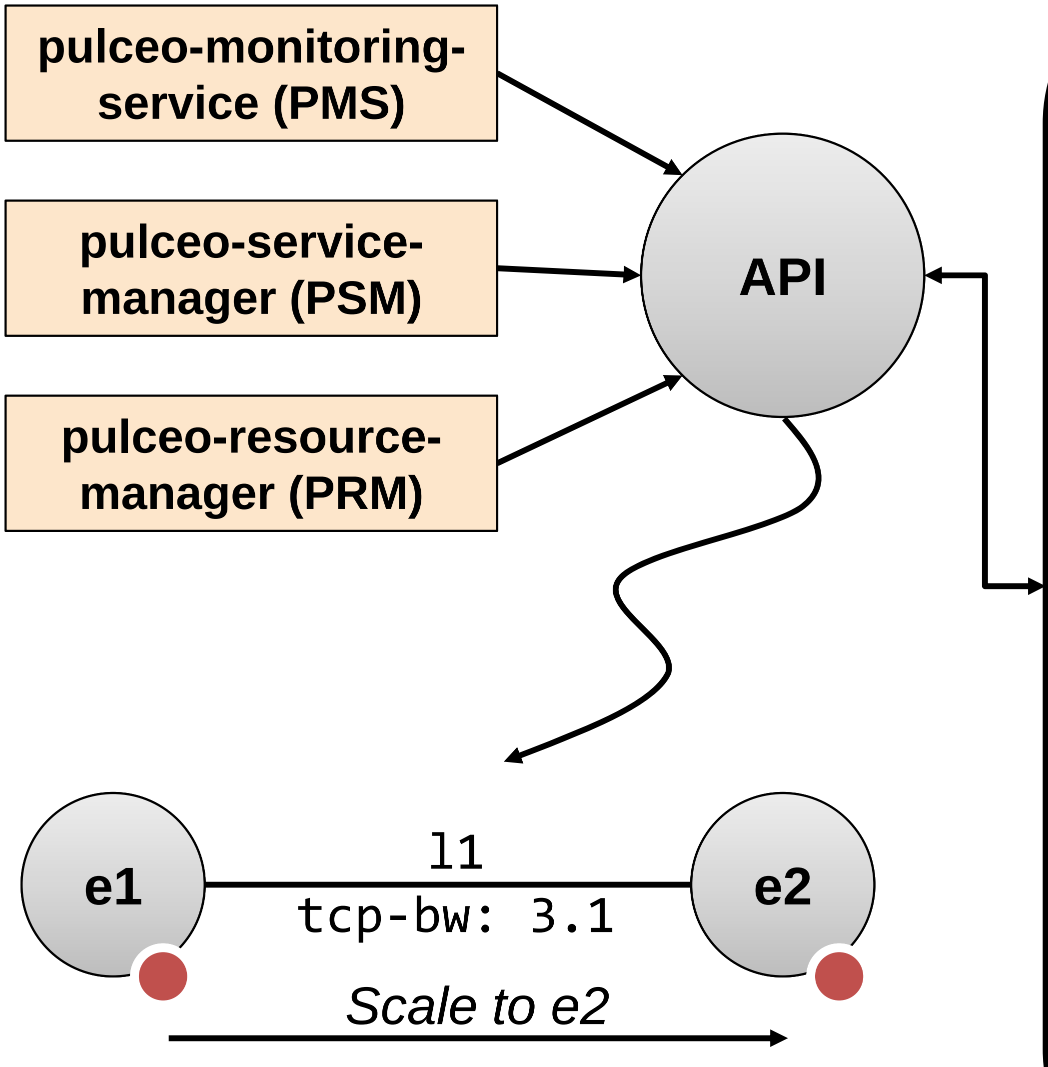

```
POST /api/v1/links
{ type: logical, src: e1, dest: e2 }
```

200 OK `{ id: l1 }`




```
POST /api/v1/metrics/links
{ id: l1, type: tcp-bw, recurrence: 5m, transformer: { type: exp, factor: 0.5 } }
```

Example (2): API-Driven Cloud-Edge Orchestration



Orchestration Logic

1) Read real-time data

```
GET /api/v1/metrics/l1
{ type: tcp-bw, value: 3.1,
unit: Mbit/s }
```

2) Execute orchestration logic

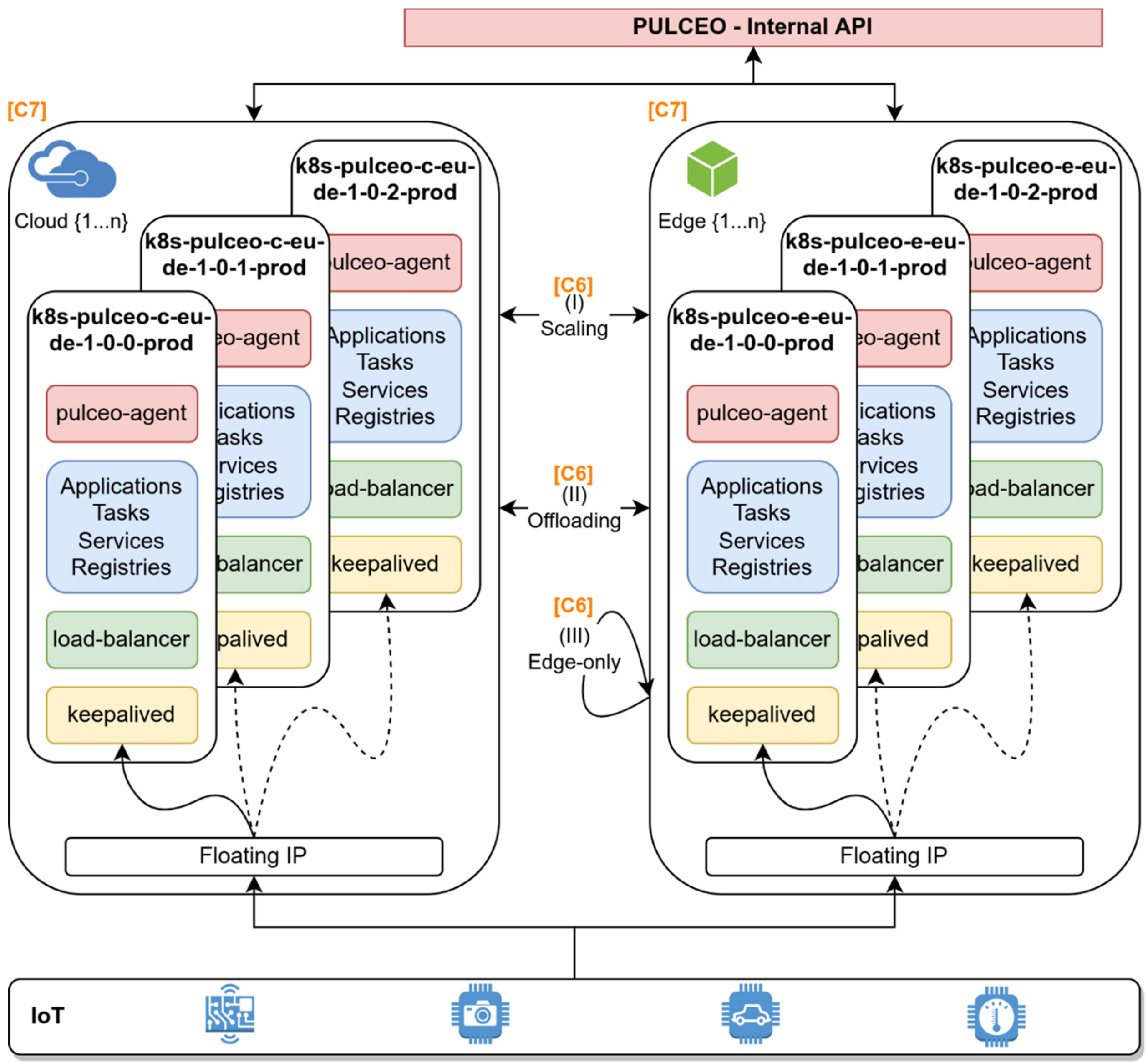
```
if tcp-bw < 5 then scale;
Calculate_deployment();
Scale to e2;
```

3) Perform operations

```
POST /api/v1/deployments
{ type: scale, application:
84df33, dest: e2 }
```

PULCEO – Reliability Considerations

- Built-in high availability for all cloud, fog, and edge nodes
- High-availability is completely transparent
- Limited autonomy if there is no connection to the cloud anymore.
- Replicated container registry

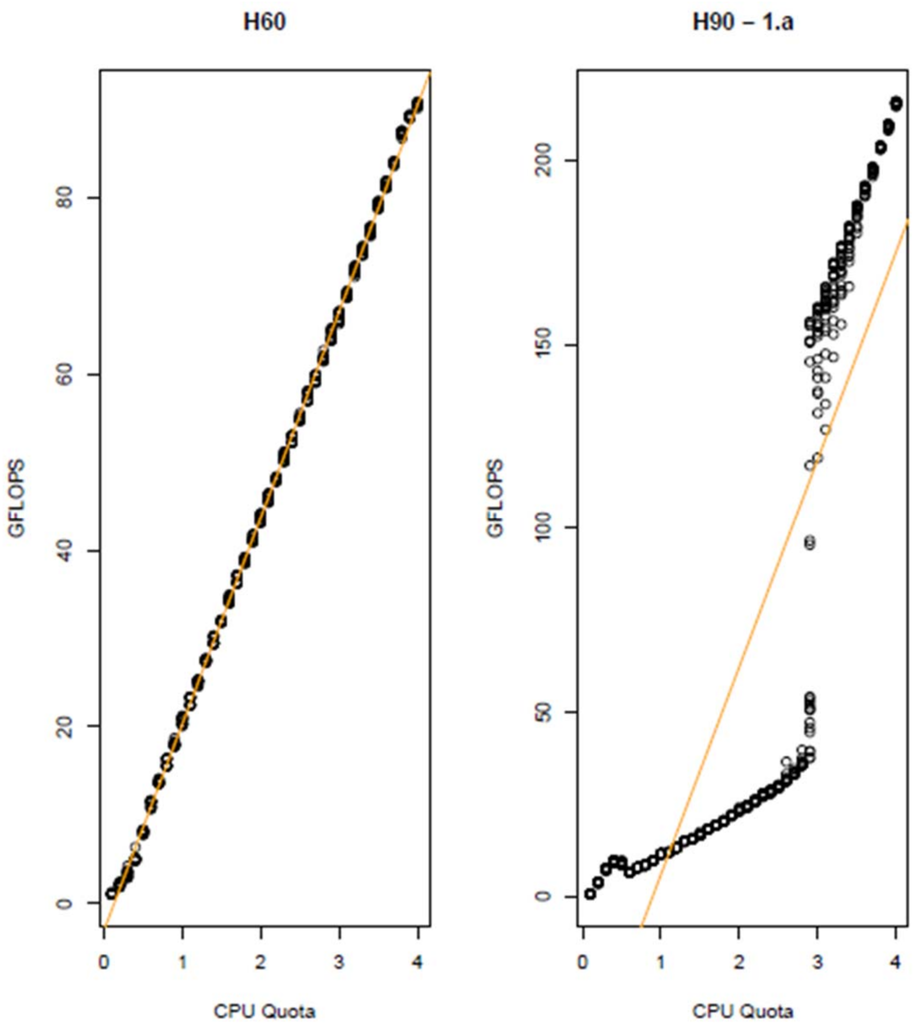


Performance Aware Cloud Computing - Solutions

S1 Use a unified and configurable framework for cloud-edge orchestration

S2 Calibrate testbeds and document it

S2 Local Testbed is often not calibrated nor comparable to other environments



Machine Config

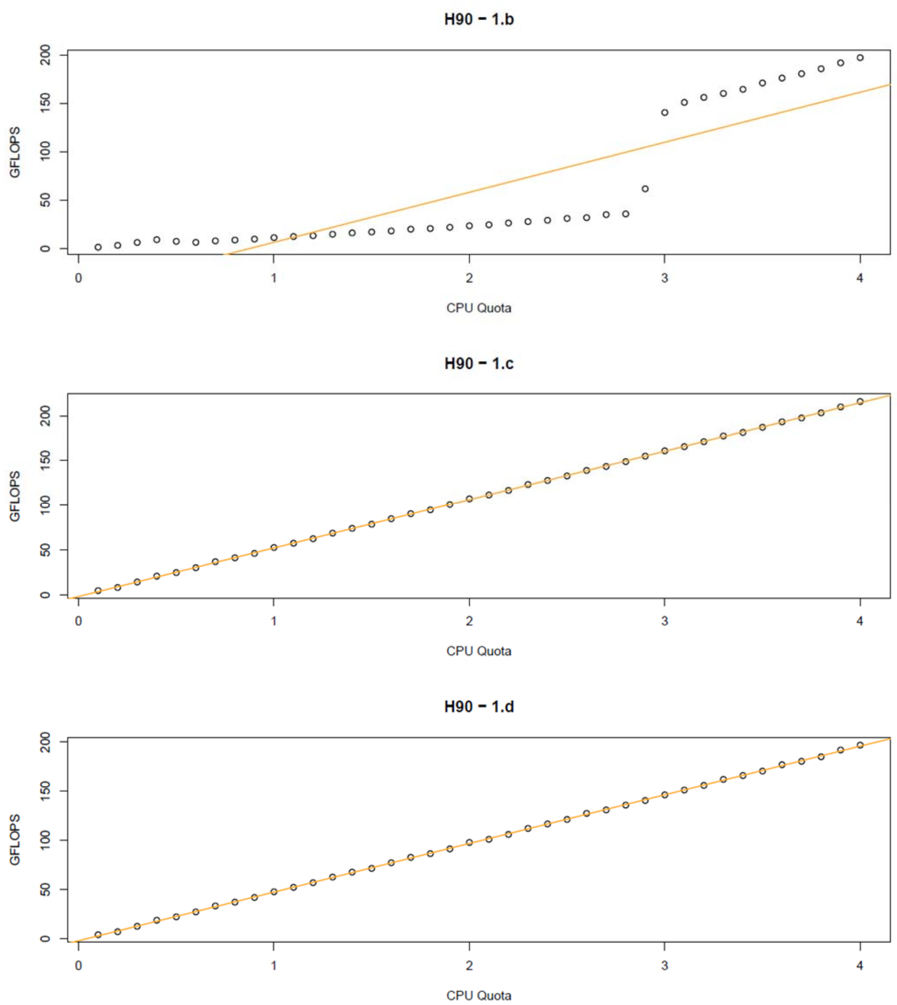
	H60	H90
Processor	i7-2600	i7-7700
Model	42	158
Base Freq.	3.4 GHz	3.6 GHz
Turbo Boost	3.8 GHz	3.9 GHz
Linux Kernel	5.4.0-65	5.4.0-70

Experiment

- Custom scaling driver responsible (`intel_pstate`)
- LINPACK used to report CPU performance
- Linux kernel virtualization features to limit resource consumption (cgroups)

Manner, J. and Wirtz, G.: [Why Many Benchmarks Might Be Compromised]
Proceedings of the 15th IEEE International Conference on Service-Oriented System Engineering, Oxford, UK (online), 23-26 August 2021

Iterative Calibration of Local Testbed



Methodology

- Changing scaling driver and turbo boost configuration
- Make scaling of resources visible by computing coefficient of determination (R^2)
- Compute linear regression model $f_{local}(x) = mx + t$

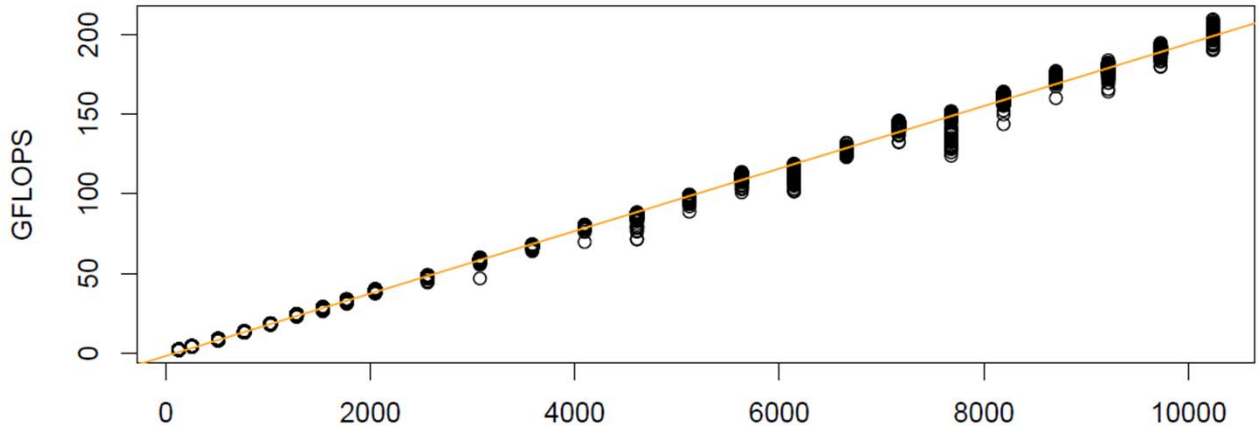
Manner, J. and Wirtz, G.: [Why Many Benchmarks Might Be Compromised]
Proceedings of the 15th IEEE International Conference on Service-Oriented System Engineering, Oxford, UK (online), 23-26 August 2021

Scaling of Resources in Public Cloud Environments

- Public Cloud Providers offer a variety of different Virtual Machines
- FaaS as an example for a compute service where user control is limited

→ only memory configuration

→ other resources are scaled proportionally



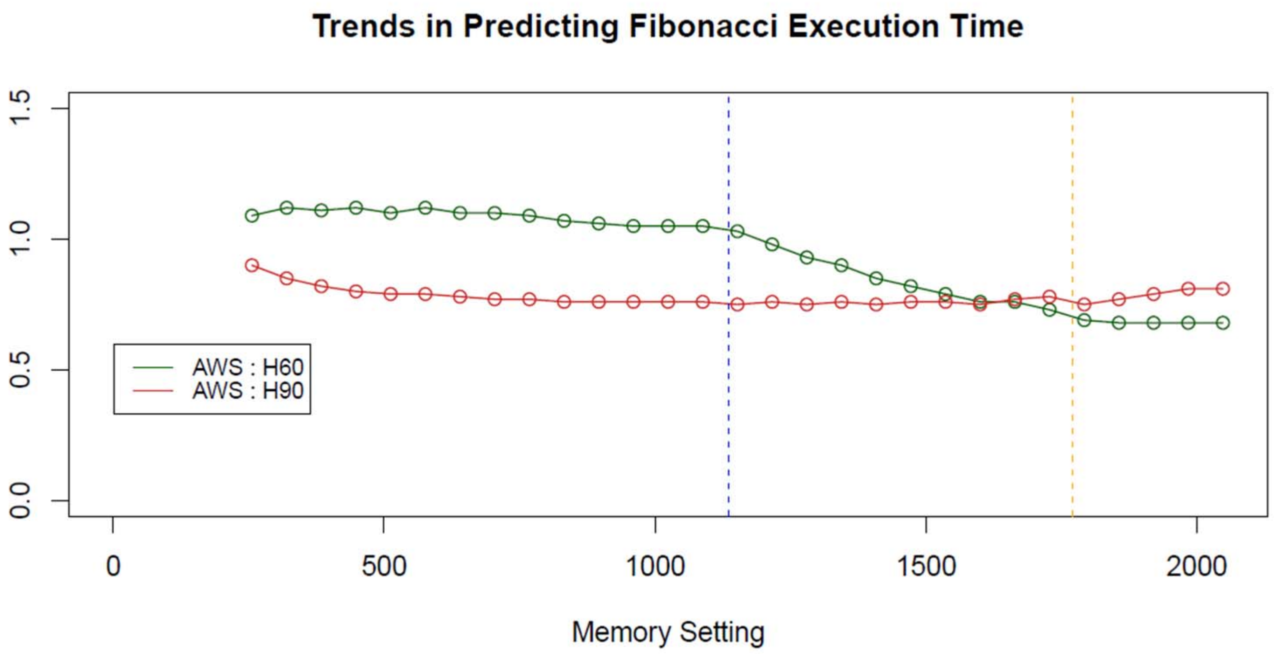
- Compute linear regression model: $g_{provider}(y) = my + t$

Manner, J., Endreß, M., Böhm, S. and Wirtz, G.: Optimizing Cloud Function Configuration via Local Simulations
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), online virutal congress, 5-10 September 2021

Making Environments Comparable

$$f_{local}(x) = g_{provider}(y)$$

- Prediction possible based on computed scalar factors
  (If cloud platform execution data is available)



Manner, J., Endreß, M., Böhm, S. and Wirtz, G.: Optimizing Cloud Function Configuration via Local Simulations
Proceedings of the IEEE International Conference on Cloud Computing (CLOUD), online virutal congress, 5-10 September 2021

S2

Testbed is often not calibrated nor comparable to other environments

Solution Ideas

- Documentation
- Abstract Computing Measure like Microsoft Azure Compute Unit (AZU)
- Perform calibration methods upfront before comparing different hardware stacks with each other

Limitation

- CPU performance currently only calibration option
- Isolated, single functions

Performance Aware Cloud Computing - Summary

I1 Experiments are not documented in a self-contained way to reproduce results

I2 Multi-Core environments and their impact on execution behavior are neglected

I3 Experiment with open-source software are resource unaware

I4 Experiments do not consider real-world conditions and reliability requirements

S1 Use a unified and configurable framework for cloud-edge orchestration

S2 Calibrate your testbed and make your hardware stack comparable to the hardware stack of cloud providers

Thoughts on Performance Aware Computing

T1

Documentation is a challenging part
Research community should come up with guidelines

T2

Multi-threading programming courses in computer science education (machine utilization)
Green Computing (CO_2 Budgets)

T3

Abstract computing measures like AZU or GFLOPS in S2 could support comparability of performance related research

T4

Algorithms and strategies for cloud-edge orchestration should be tested and evaluated under real-world conditions

Guido Wirtz
Head of Distributed Systems Group
guido.wirtz@uni-bamberg.de

PhD Students

Sebastian Böhm

Stefan Winzinger

Robin Lichtenthäler

Johannes Manner

Institution and department
University of Bamberg
Department of Applied Computer Sciences
Distributed Systems Group