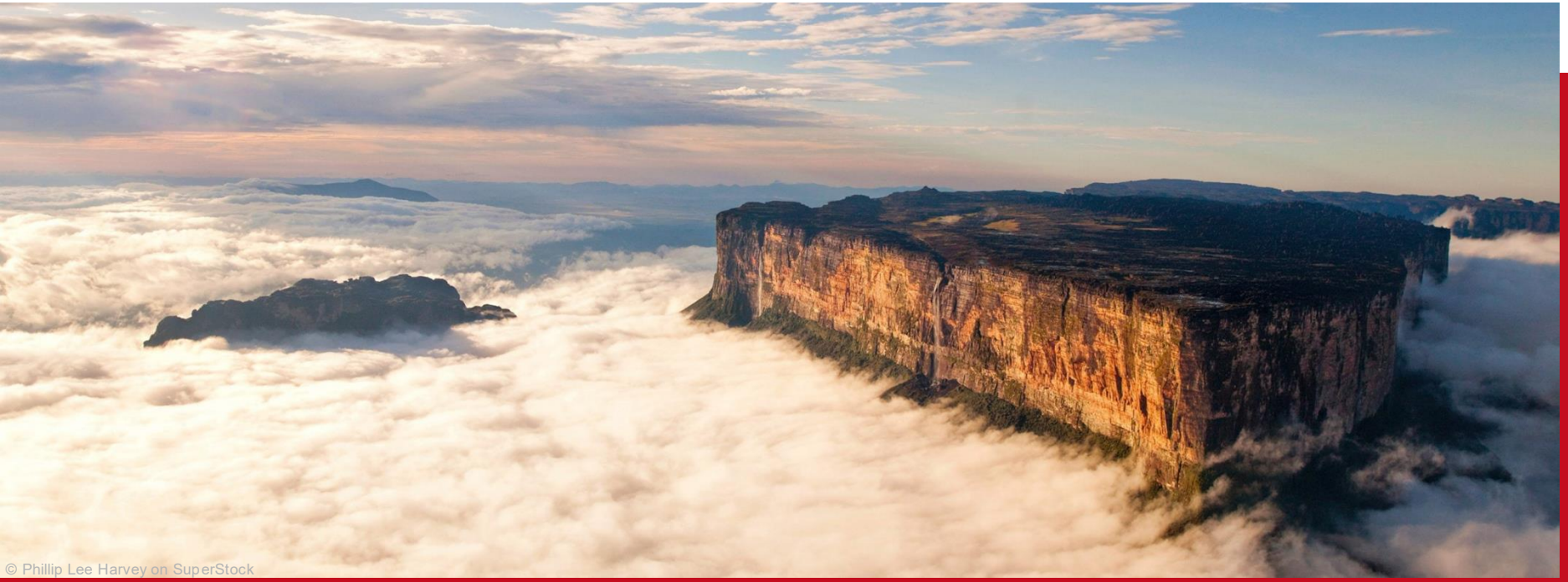
TECHNISCHE UNIVERSITÄT BERLIN

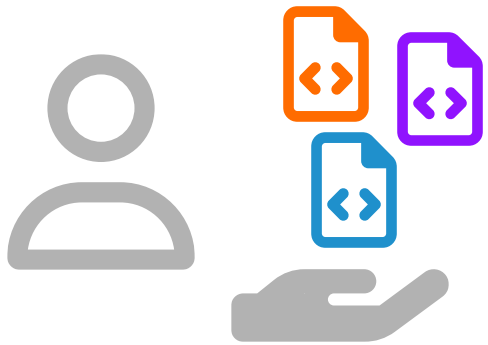


MINOS: Exploiting Cloud Performance Variation with Function-as-a-Service Instance Selection

Trever Schirmer, V. Carl, N. Höller, T. Pfandzelter, D. Bermbach | IC2E' 25

3S

FaaS

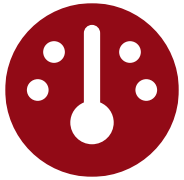


Just the Code

Serverless Platform

(Infinite) Scalability
& Elasticity

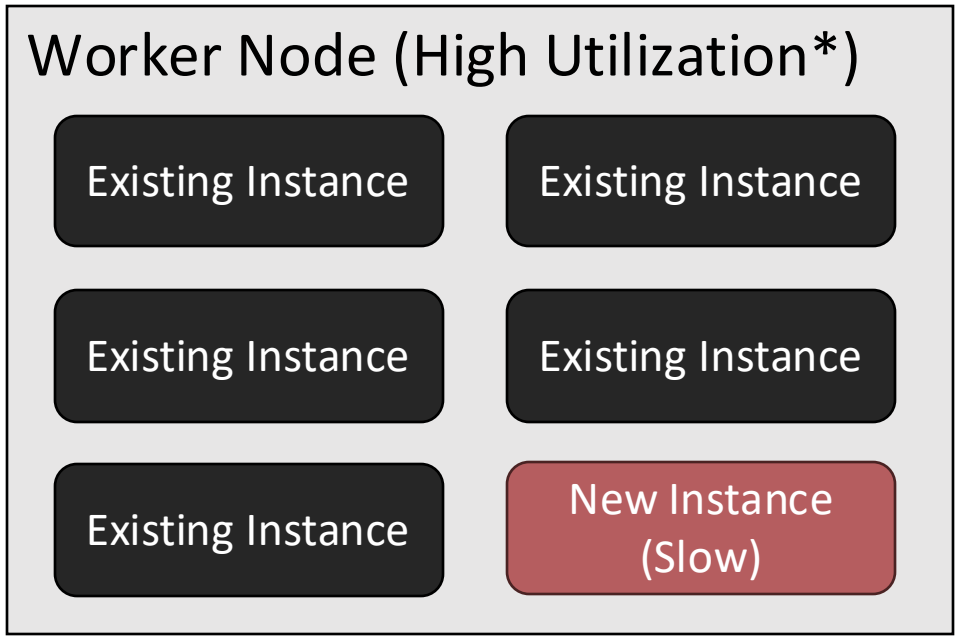
Hidden Abstractions

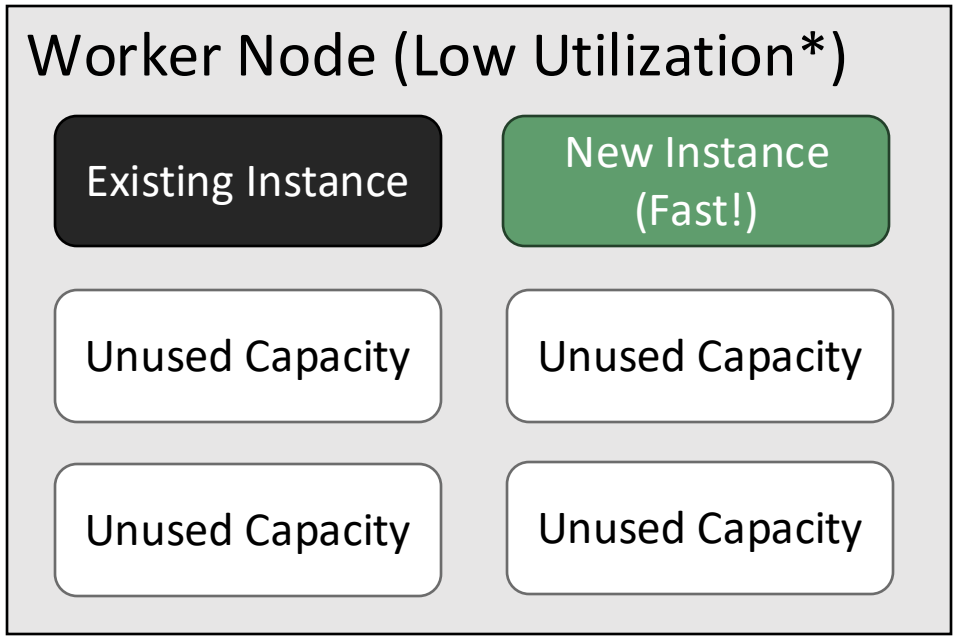


Unreliable Performance

Expensive?

FaaS Performance: Short Term





FaaS Provider wants this:
- Better hardware utilization
- More Money! (Due to Pay-per-*ms*)

FaaS Users want this:
- Lower utilization → higher performance
- Save money *at the same time*

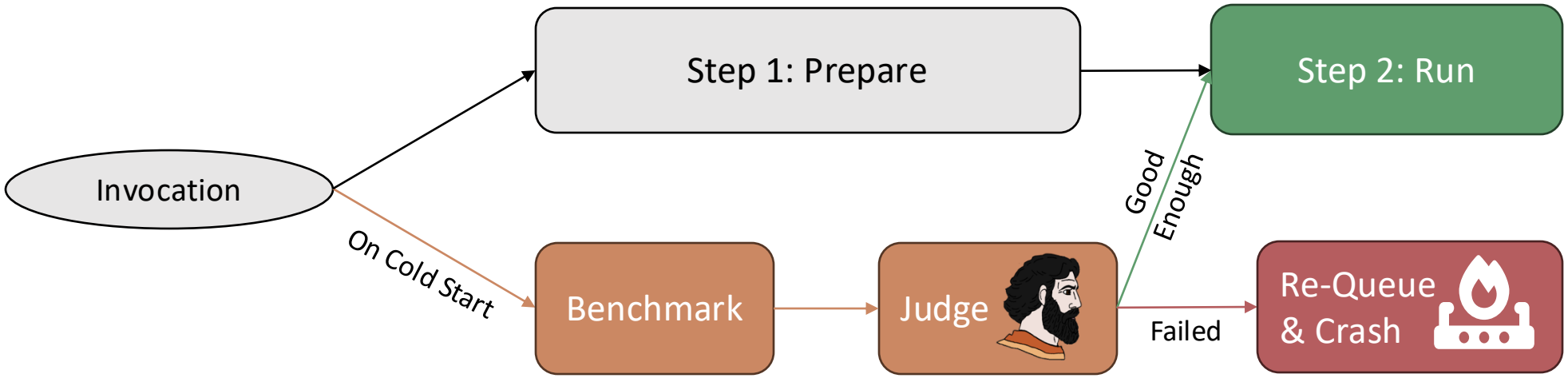
→ How to make sure our functions run on faster servers?

The Cloud Is Noisy

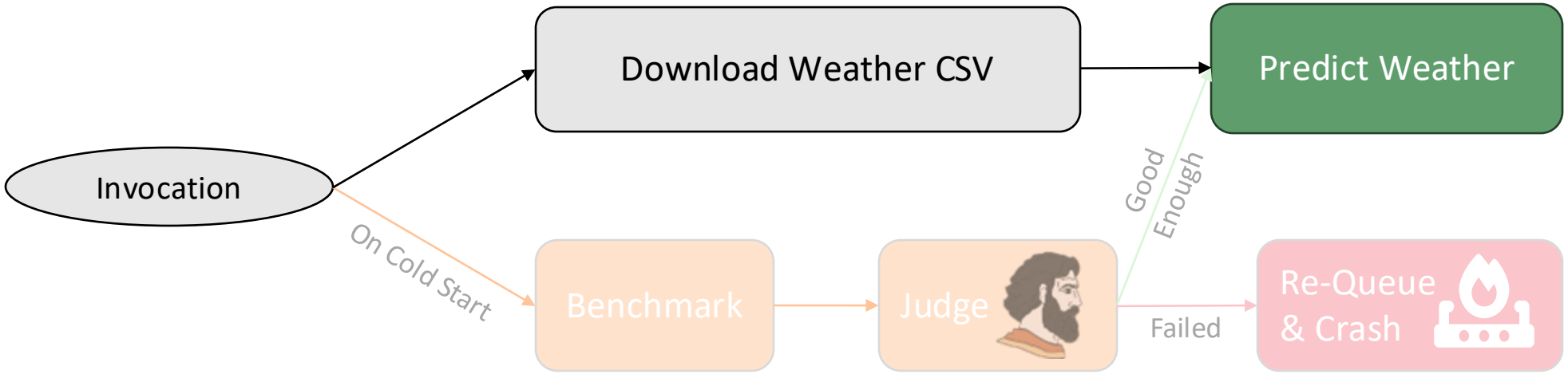
- Resource Contention
- Silicone Lottery
- Cooler Servers (Noisy Neighbor)
- Completely Different Hardware?
- …

Core Idea

Core Idea

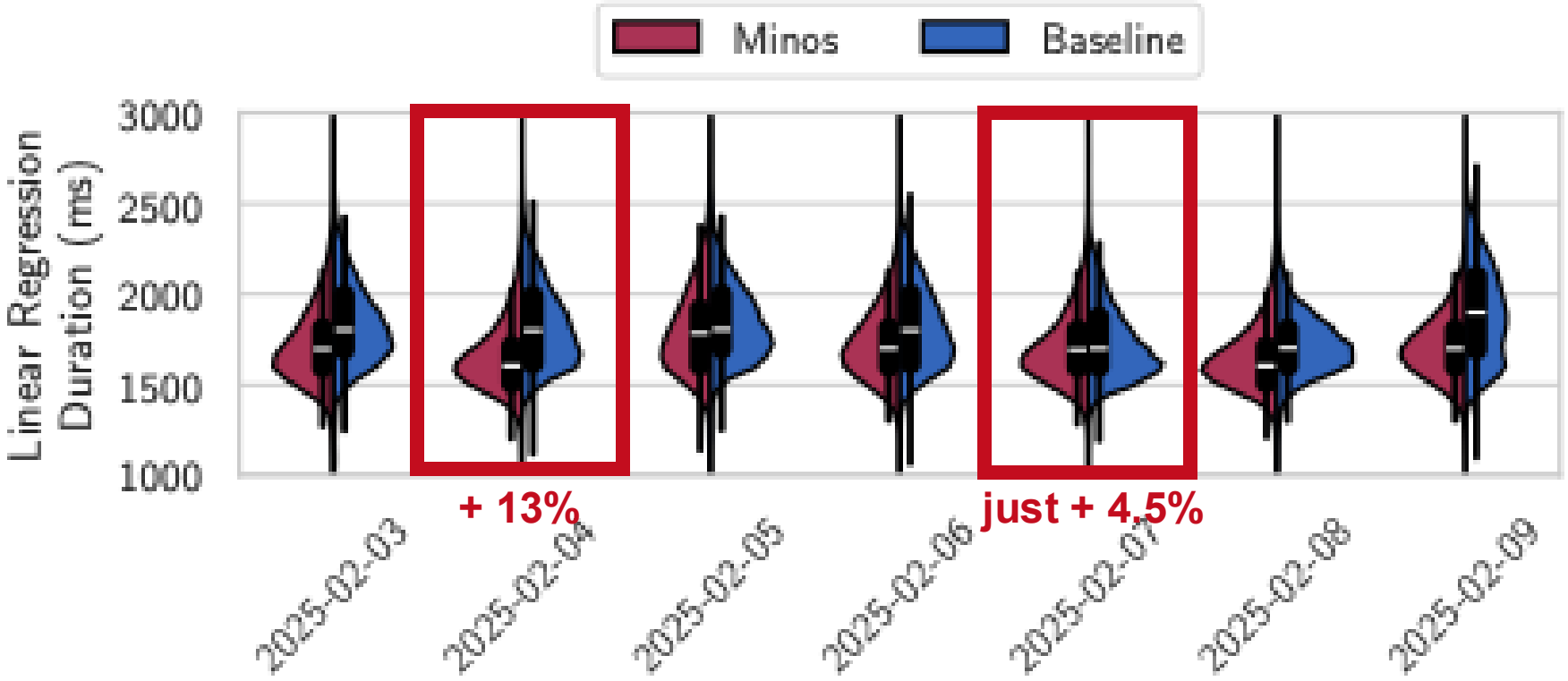

Step 1: Prepare
Step 2: Run
Invocation
On Cold Start
Benchmark
Judge
Good Enough
Failed
Re-Queue & Crash

Experiments



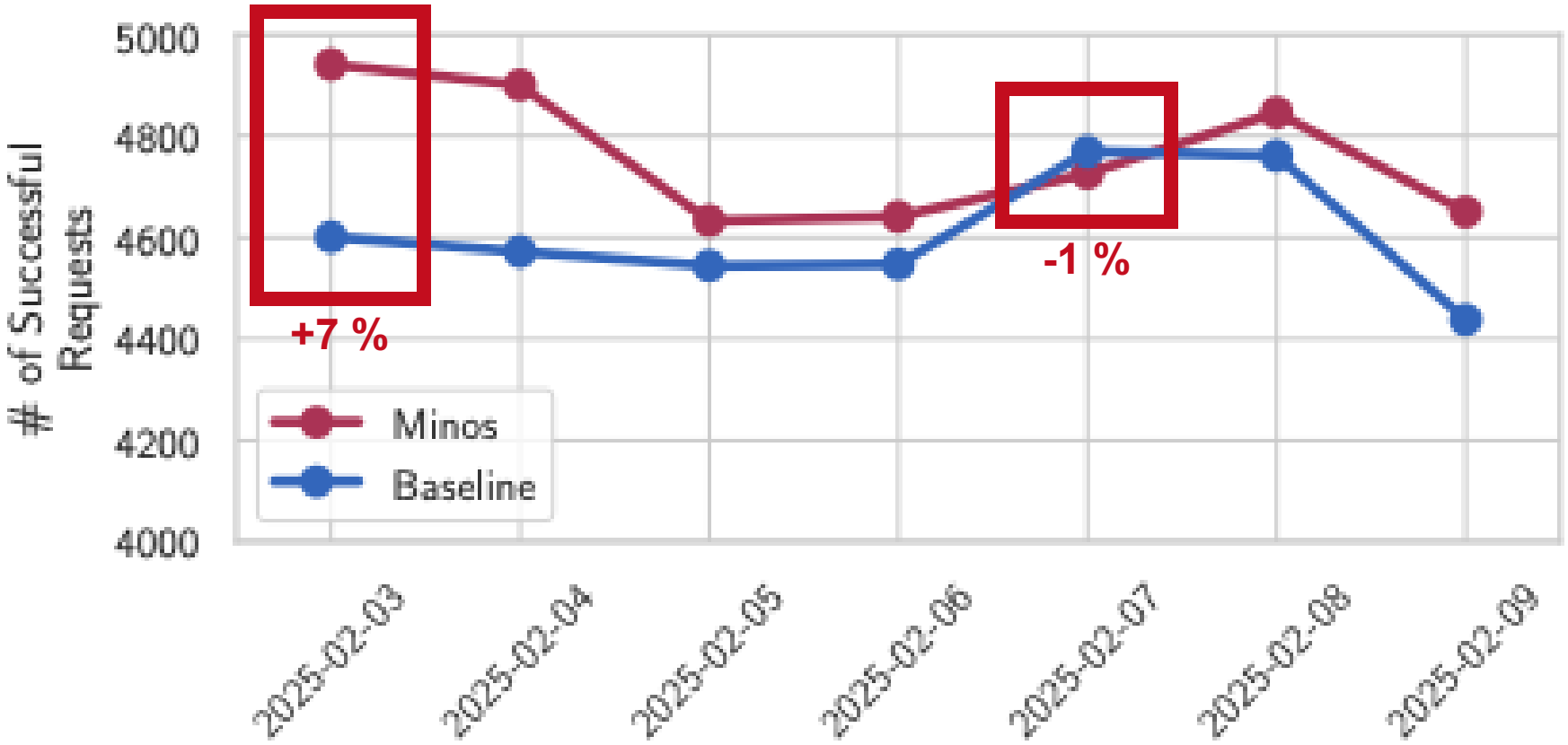
- 10 Virtual Users that wait one second between requests for 30 minutes
- Once a day for one week
- *Run a short (10 VU for one minute) test beforehand to figure out the 60th percentile*

The resource-intensive part is faster!

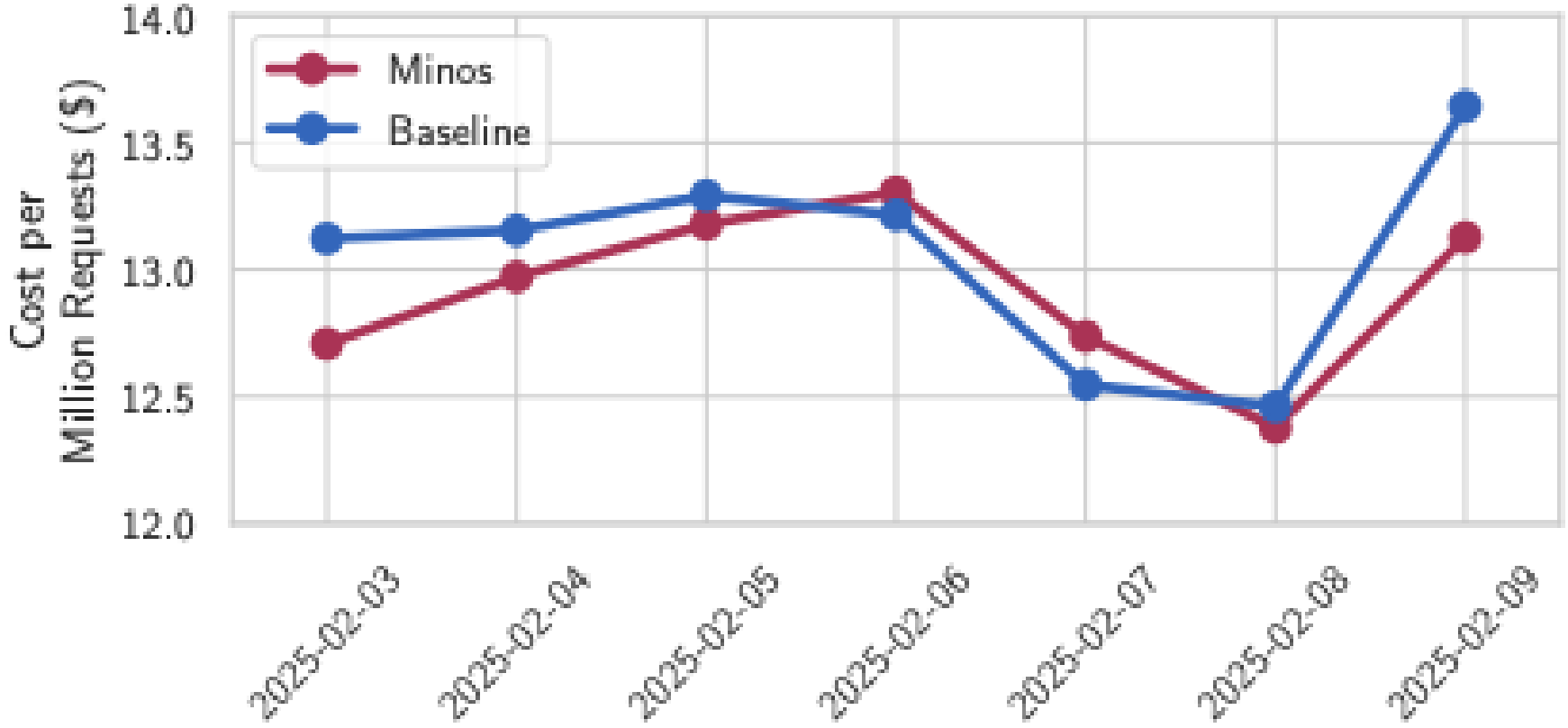


Duration of Linear Regression step: faster *every day*

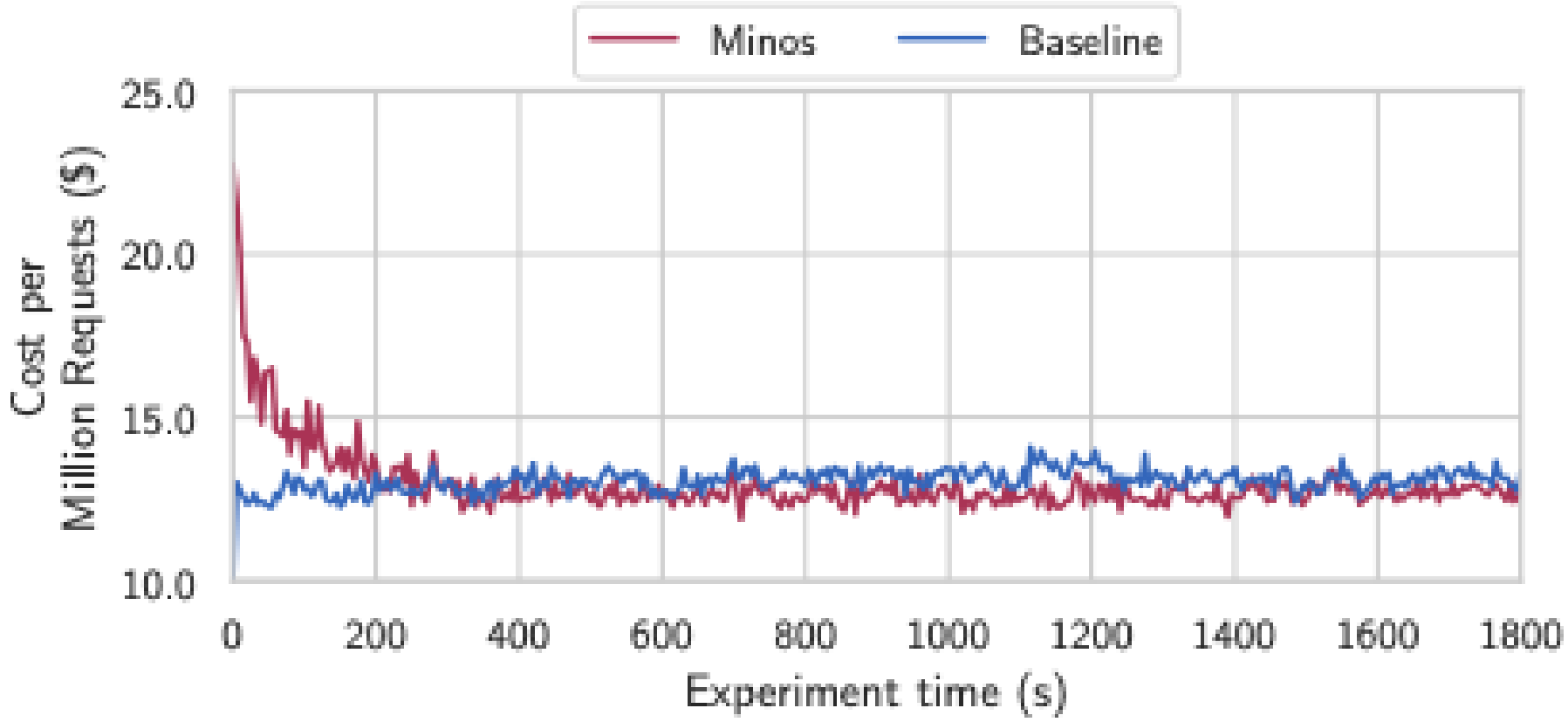
More requests almost every day!



Cheaper almost every day!



Longer experiments are cheaper



Average cost over the whole experiment

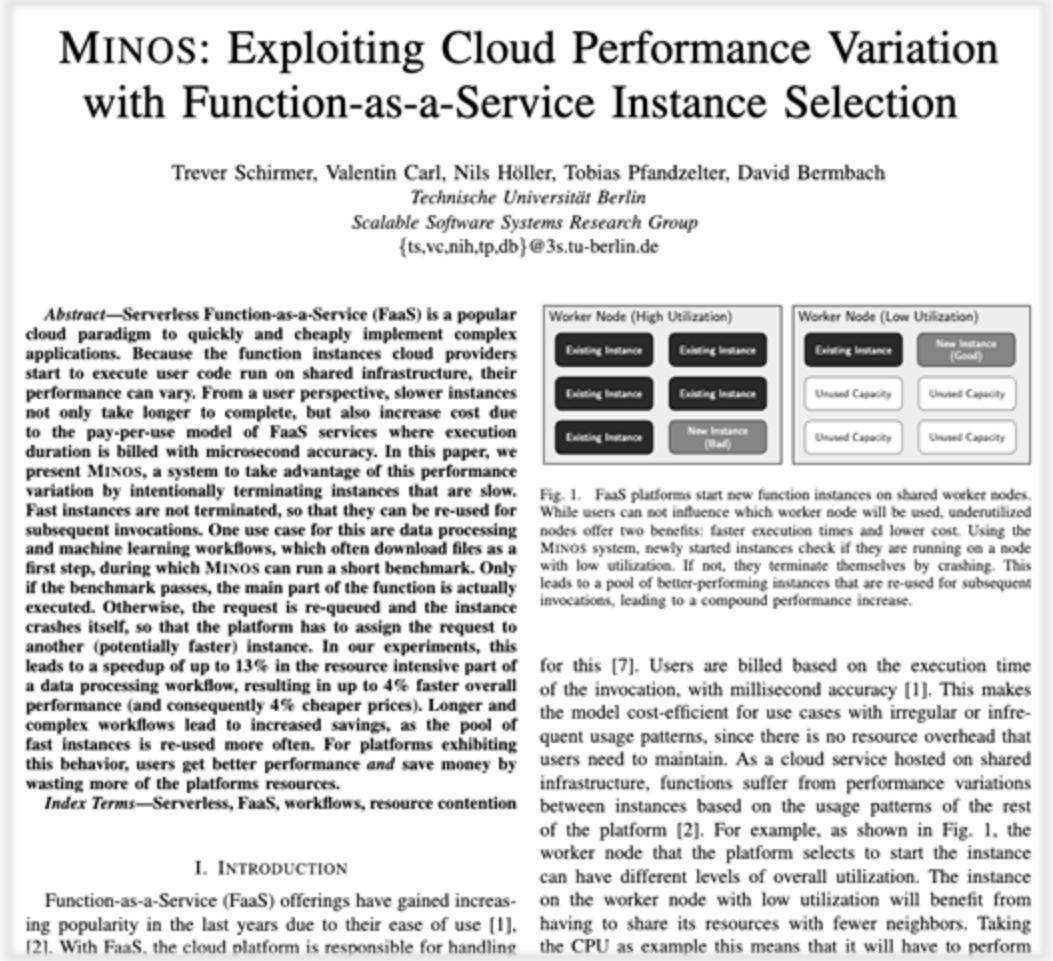
Conclusion

- Waste resources of the cloud platform!
- Better performance and cheaper for users!
- At the same time!

Contact

t.schirmer@tu-berlin.de

trever.eu // tu.berlin/3s



MINOS: Exploiting Cloud Performance Variation with Function-as-a-Service Instance Selection

Trever Schirmer, Valentin Carl, Nils Höller, Tobias Pfandzelter, David Bermbach
Technische Universität Berlin
Scalable Software Systems Research Group
{ts,vc,nih,tp,db}@3s.tu-berlin.de

***Abstract*—Serverless Function-as-a-Service (FaaS) is a popular cloud paradigm to quickly and cheaply implement complex applications. Because the function instances cloud providers start to execute user code run on shared infrastructure, their performance can vary. From a user perspective, slower instances not only take longer to complete, but also increase cost due to the pay-per-use model of FaaS services where execution duration is billed with microsecond accuracy. In this paper, we present MINOS, a system to take advantage of this performance variation by intentionally terminating instances that are slow. Fast instances are not terminated, so that they can be re-used for subsequent invocations. One use case for this are data processing and machine learning workflows, which often download files as a first step, during which MINOS can run a short benchmark. Only if the benchmark passes, the main part of the function is actually executed. Otherwise, the request is re-queued and the instance crashes itself, so that the platform has to assign the request to another (potentially faster) instance. In our experiments, this leads to a speedup of up to 13% in the resource intensive part of a data processing workflow, resulting in up to 4% faster overall performance (and consequently 4% cheaper prices). Longer and complex workflows lead to increased savings, as the pool of fast instances is re-used more often. For platforms exhibiting this behavior, users get better performance *and* save money by wasting more of the platforms resources.**

***Index Terms*—Serverless, FaaS, workflows, resource contention**

I. INTRODUCTION

Function-as-a-Service (FaaS) offerings have gained increasing popularity in the last years due to their ease of use [1], [2]. With FaaS, the cloud platform is responsible for handling

Fig. 1. FaaS platforms start new function instances on shared worker nodes. While users can not influence which worker node will be used, underutilized nodes offer two benefits: faster execution times and lower cost. Using the MINOS system, newly started instances check if they are running on a node with low utilization. If not, they terminate themselves by crashing. This leads to a pool of better-performing instances that are re-used for subsequent invocations, leading to a compound performance increase.

for this [7]. Users are billed based on the execution time of the invocation, with millisecond accuracy [1]. This makes the model cost-efficient for use cases with irregular or infrequent usage patterns, since there is no resource overhead that users need to maintain. As a cloud service hosted on shared infrastructure, functions suffer from performance variations between instances based on the usage patterns of the rest of the platform [2]. For example, as shown in Fig. 1, the worker node that the platform selects to start the instance can have different levels of overall utilization. The instance on the worker node with low utilization will benefit from having to share its resources with fewer neighbors. Taking the CPU as example this means that it will have to perform

Sources

[0] Trever Schirmer, Nils Japke, Sofia Greten, Tobias Pfandzelter, and David Bermbach. 2023. The Night Shift: Understanding Performance Variability of Cloud Serverless Platforms. In Proceedings of the 1st Workshop on SErverless Systems, Applications and MEthodologies (SESAME '23). Association for Computing Machinery, New York, NY, USA, 27–33. https://doi.org/10.1145/3592533.3592808