Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

Fog Computing

Bermbach | Part 4: Platforms and Applications

Maybe: Think about going for a PhD…?

#1: It‘s fun. You get to explore your area of choice. You get to play around with cutting edge technology. In fact: you build technology BEYOND the cutting edge.

#2: You have a lot of personal freedom in how you organize your days. Most of the clichées about working crazy hours are plain wrong.

#3: Depending on your career goals, you can aim for academia. Most PhD graduates don‘t – but you collect a lot of things valued in industry (depending on concrete job: detail knowledge, GitHub projects, experience as a speaker, a fancy title, writing skills, …)

#4: Payment is not bad. Salary: 52k (1st year), 56k (2nd + 3rd year), etc.

Agenda

Technische Universität Berlin

Lectures

- **From Cloud to Fog Computing**
- **Data Distribution**
- **Data Management**
- **Platforms & Applications**
- **Testing & Benchmarking**
- **LEO Edge Computing**

Assignments

- Prototyping Assignment
- Reading Assignment

Wrap-up

- **Q&A**
- Final Test

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Platforms and applications

How do we build compute platforms and applications for fog environments with their node hetereogeneity, geo-distribution, and resource limits at the edge?

Technische Universität Berlin

BASIC DESIGN PRINCIPLES

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

State-of-the-Art: Cloud Systems

Technische Universität Berlin

Microservice-based design
Infrastructure automation
Fault-tolerance through replication
…

But: Cluster-based deployment in only a few data centers

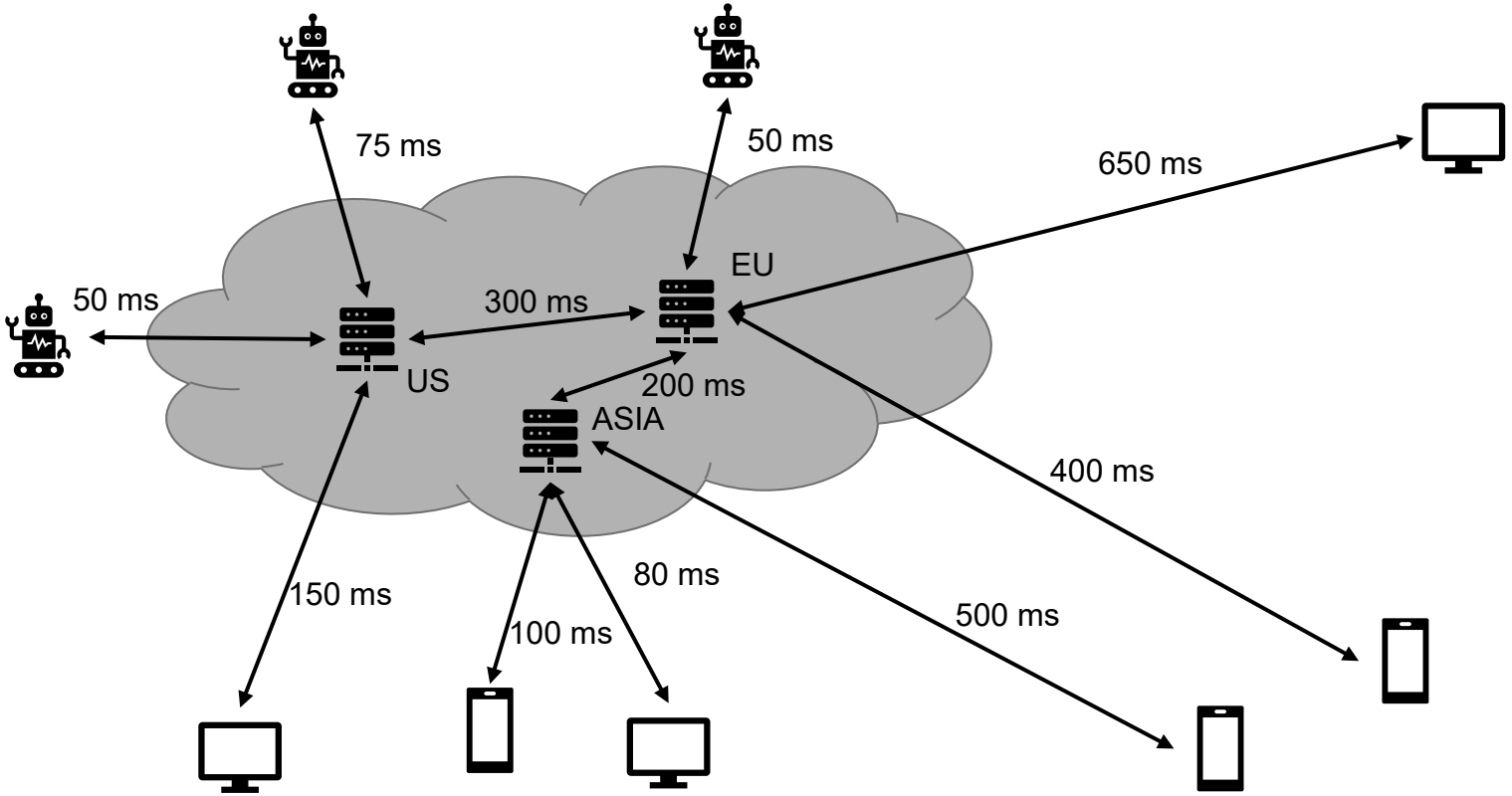
Fog: Single-node to cluster-sized deployment on millions of sites
=> More of the same; but go beyond and handle geo-awareness and fault-tolerance differently

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

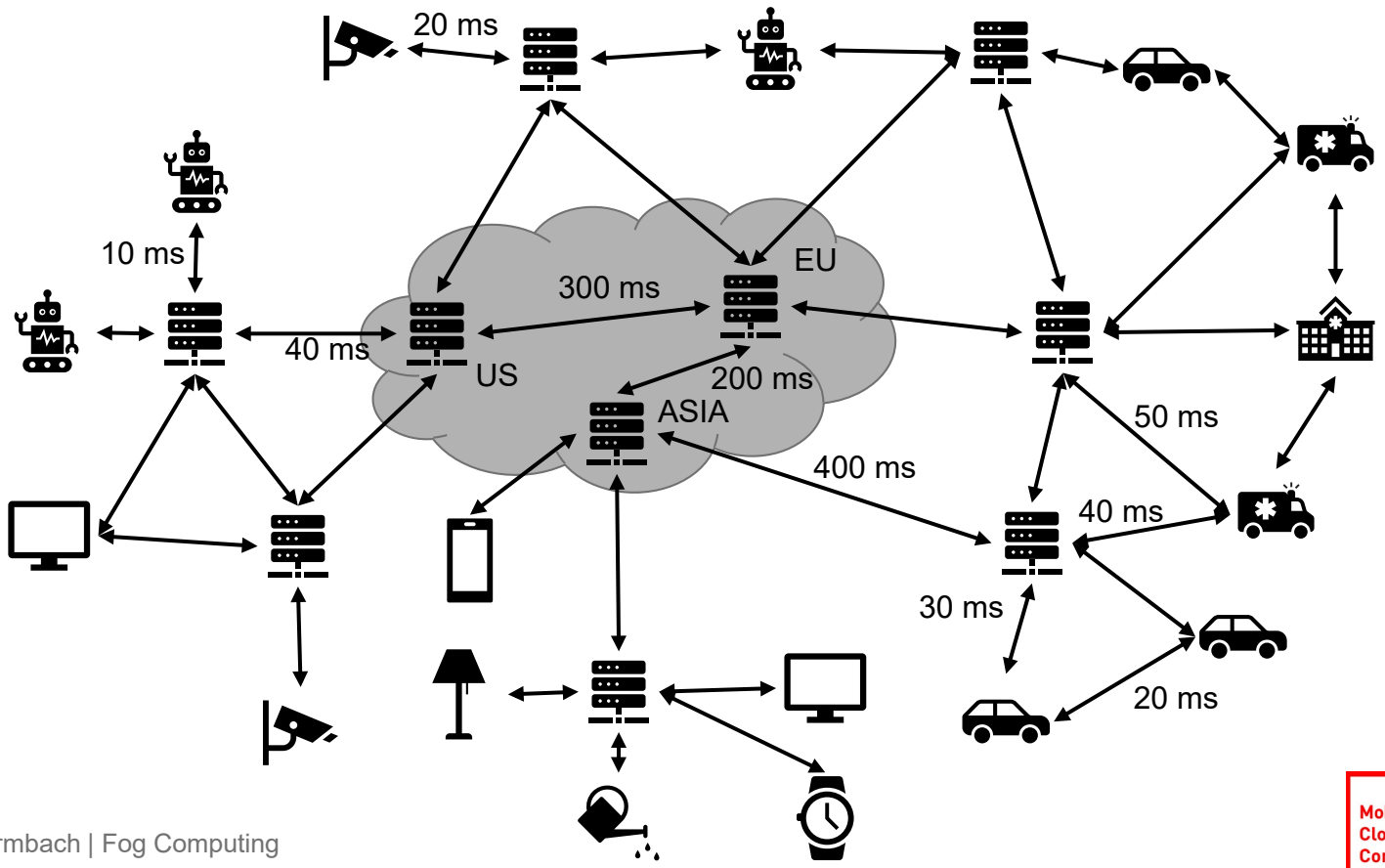
Geo-Awareness in the Cloud

- Limited to large regions (e.g., countries)
- High latency if the closest data center is quite far



Introducing Fog Nodes

- Fast connection to nearby fog nodes but limited bandwidth to cloud
- Access point(s) of mobile devices must be adapted based on its location



Geo-Awareness

Technische Universität Berlin

Infrastructure needs to expose location and network topology explicitly (beyond regions)

An application

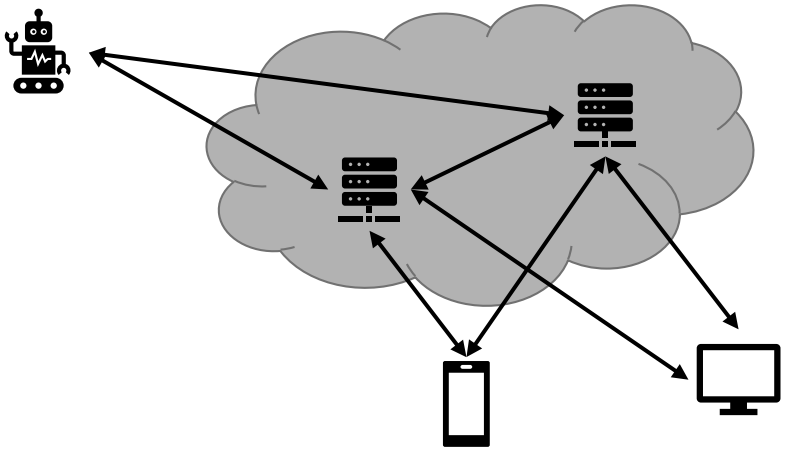
- Must to be aware of its deployment location
- Needs to handle client movement (handover to other edge device)
- Must be prepared to move components elsewhere (=> stateless application logic)
- Must move data when necessary
- May not rely on the availability of remote components

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Fault-Tolerance in Cloud Applications

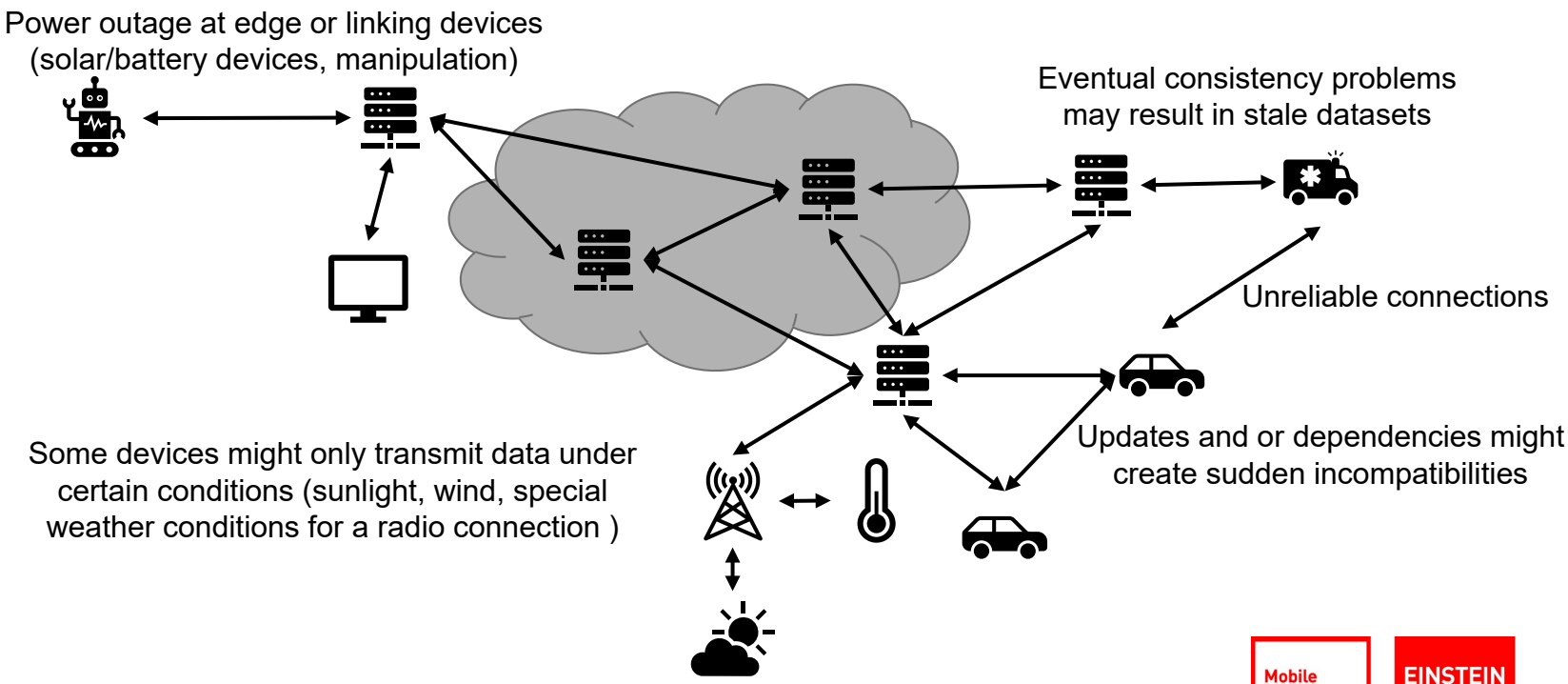
- Redundant Servers
- Retry-on-error principle (with other service instance)
- Monitoring services and its workload, auto-scaling
- Chaos-Monkey which randomly shuts down services to check if the system adapts and catches the outage



Fault-Tolerance in Fog Applications

Prevalence of faults depends on the number of nodes:

- Systems and/or its components fail continuously
- Connecting infrastructure fails or operates with reduced quality



Fault-Tolerance in Fog Applications

Technische Universität Berlin

- Buffer messages until its receiver is available again
- Expect data staleness and ordering issues
- Cache data aggressively
- Compress data items with checksums as much as possible on unreliable connections (small time frames must suffice to transmit the entire message)
- Plan with incompatibility, constantly monitor software versions on devices
- Design for loose coupling
- …

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

Platforms for the edge

FUNCTION-AS-A-SERVICE

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Recap: Function-as-a-Service

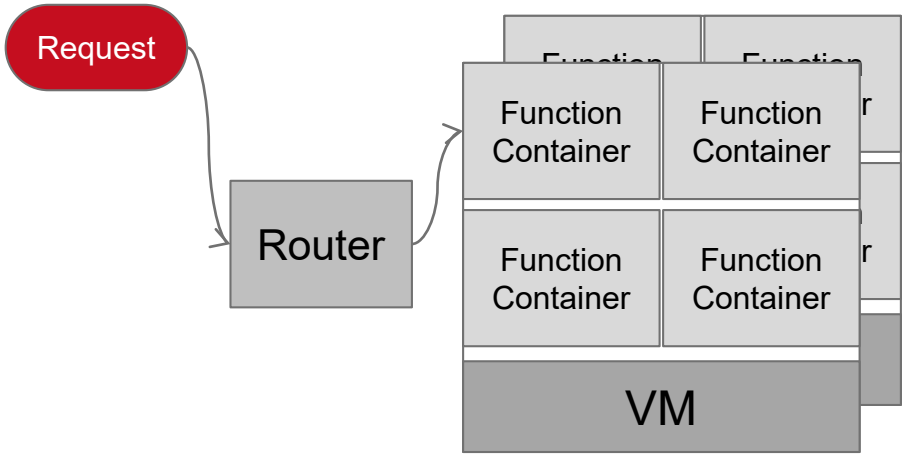
Technische Universität Berlin

According to Martin Fowler:

> *"FaaS is about running backend code without managing your own server systems or your own long-lived server applications."*
> (https://martinfowler.com/articles/serverless.html#unpacking-faas)

…and we're doing this at the granularity level of single functions.

AWS Lambda



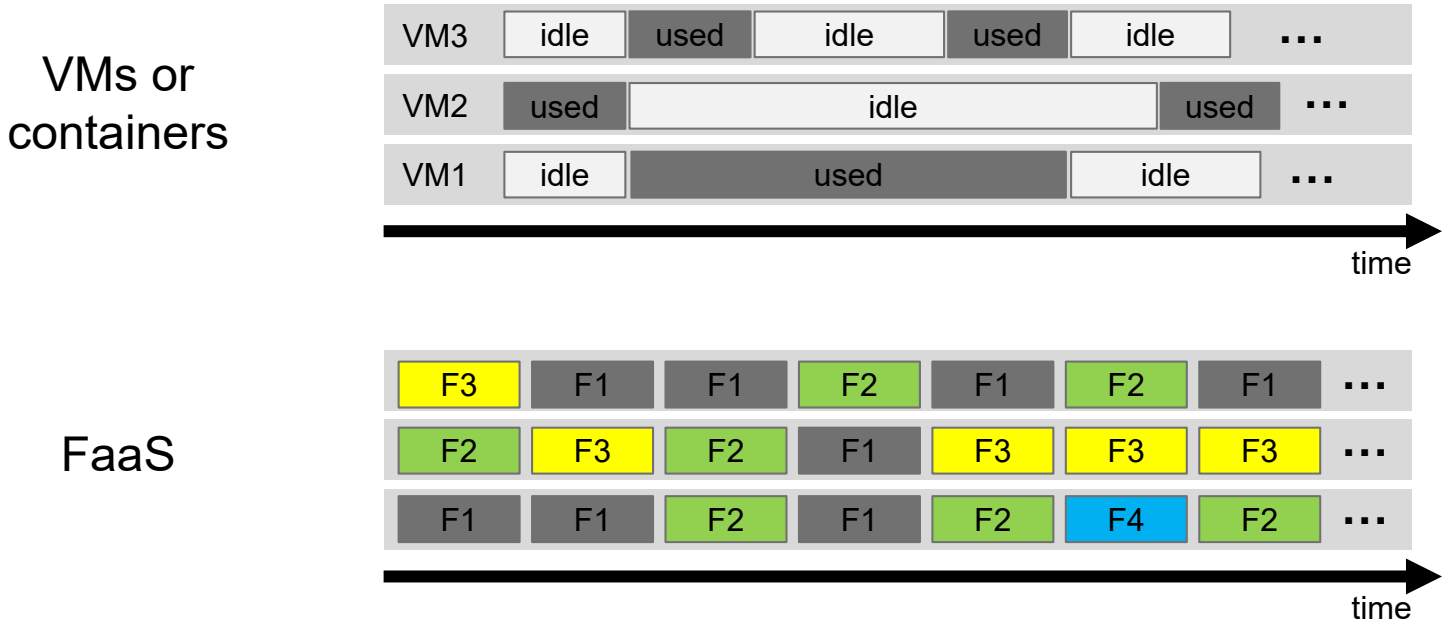
Google Cloud Functions

APACHE OpenWhisk

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

FaaS is very promising for the edge



1. Higher utilization of scarce edge resources
2. Stateless functions can be moved as needed

FaaS systems for fog environments

Existing FaaS platforms work well in the cloud but have problems on smaller fog nodes towards the edge.

We need leaner options for the edge

Case studies:

- Lean OpenWhisk
- tinyFaaS
- NanoLambda
- AuctionWhisk

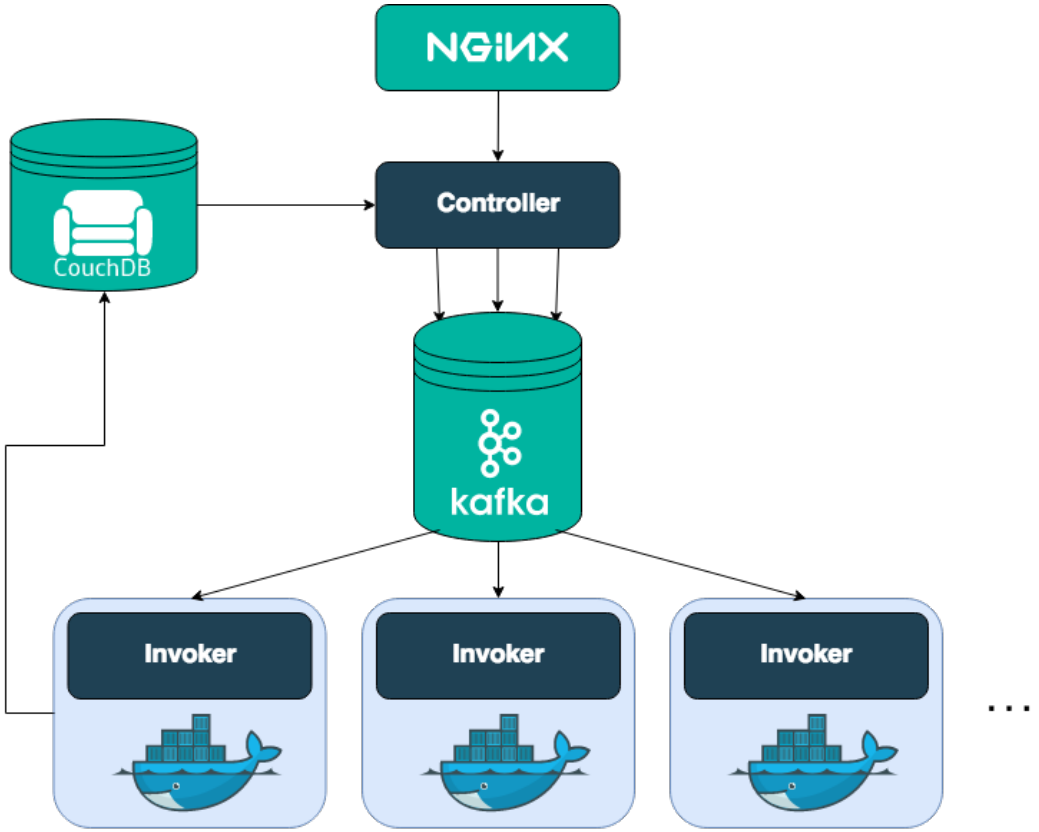
FaaS for the edge

LEAN OPENWHISK

https://medium.com/openwhisk/lean-openwhisk-open-source-faas-for-edge-computing-fb823c6bbb9b

OpenWhisk

Released in 2016 by IBM, basis of IBM Cloud Functions

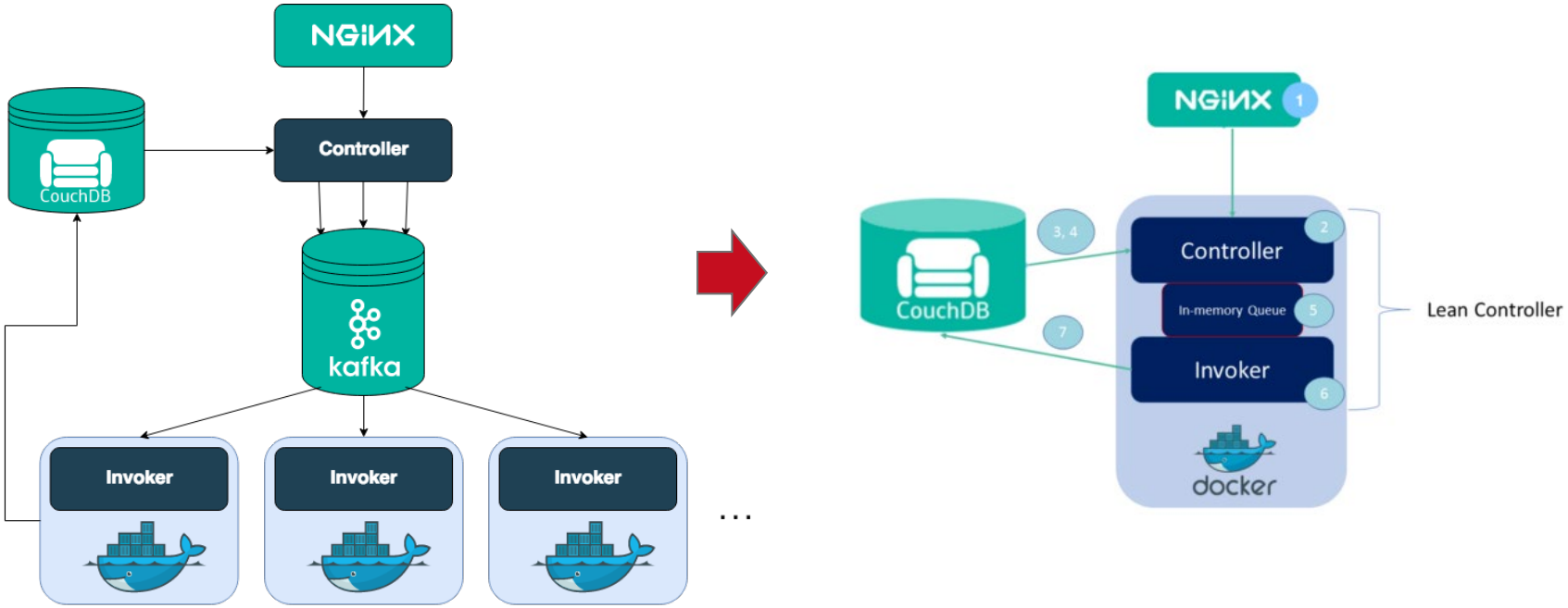


Designed for cluster-based deployments

Too heavy for the edge

Source and further reading:
https://github.com/apache/openwhisk/blob/master/docs/about.md

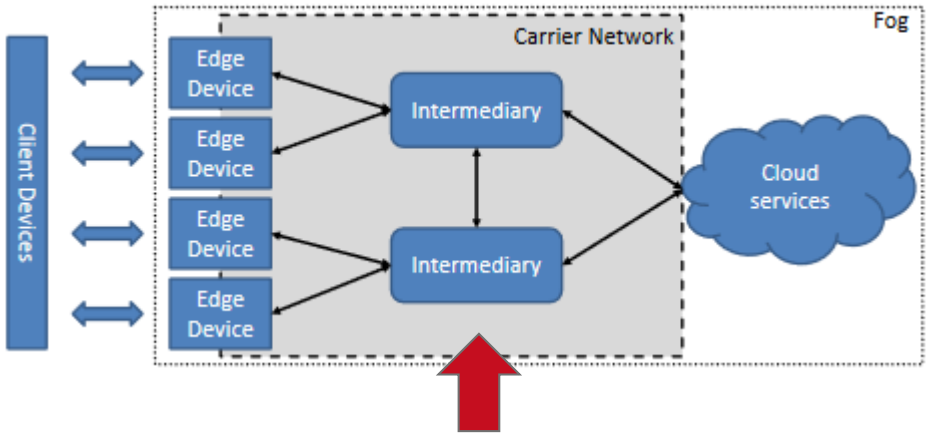
OpenWhisk vs. Lean OpenWhisk



Does it suffice for the edge?

+ Lean OpenWhisk replaced the heaviest components
+ Lean OpenWhisk is fully compatible with OpenWhisk and part of OpenWhisk releases

- Lean OpenWhisk still has a lot of unnecessary code that was written for cloud-based cluster deployments



Technische Universität Berlin

FaaS for the edge

TINYFAAS

Pfandzelter, Bermbach. tinyFaaS: A Lightweight FaaS Platform for Edge Environments. IEEE ICFC 2020.

Mobile Cloud Computing

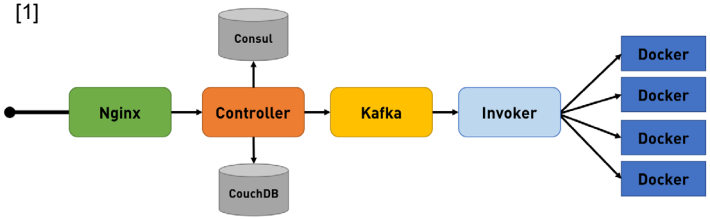
EINSTEIN CENTER Digital Future

Running FaaS on the edge?

Lean OpenWhisk is still rather heavy-weight and more suited for intermediary nodes than for the edge.

How can we run

[1]



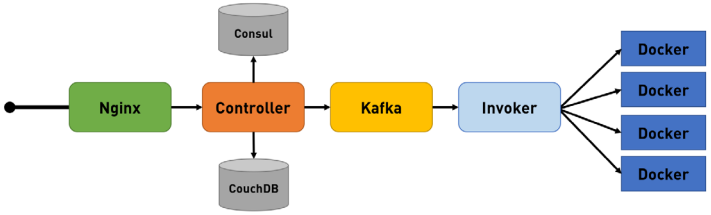
on

[2]

?

[1] https://thenewstack.io/behind-scenes-apache-openwhisk-serverless-platform/
[2] https://commons.wikimedia.org/wiki/File:Raspberry_Pi_3_(24914484549).png

Basic idea



tinyFaaS

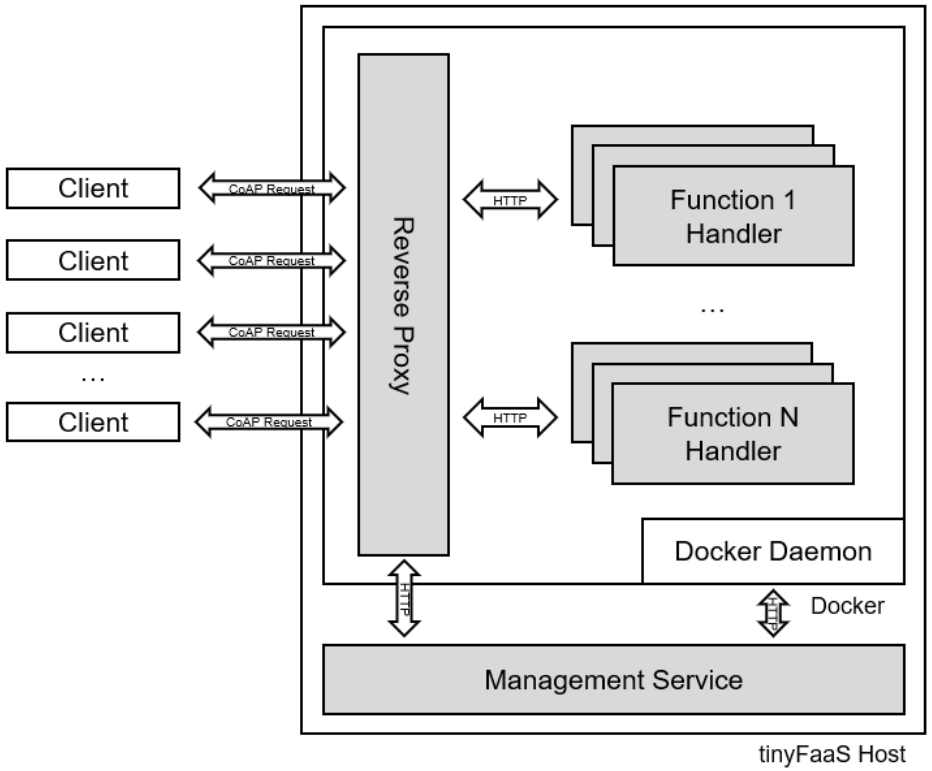
Goals: lightweight, extensible, http or http compatible

Key mechanisms

Remove as many components as possible

CoAP as application protocol

Parallel execution of requests within a container
=> one container per client or per function

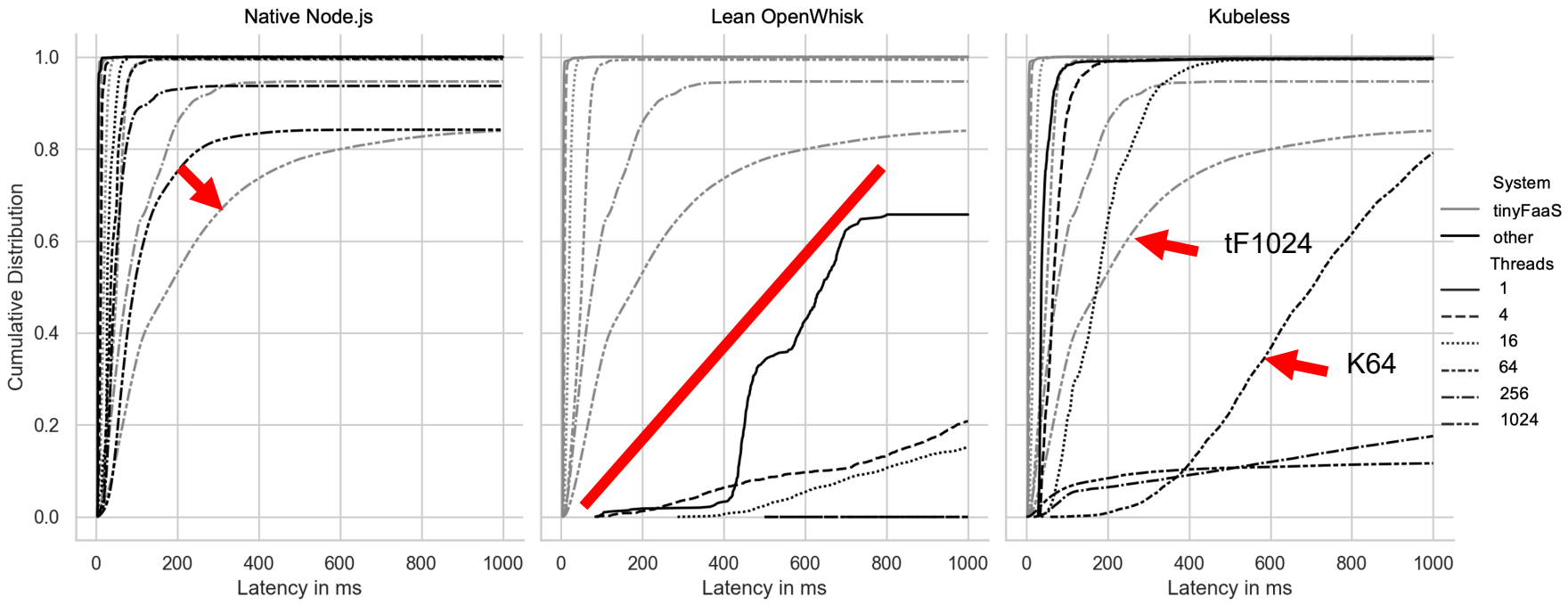


Experiments

Prototype implemented in Go and Python for node.js runtimes

- Compare tinyFaaS to: native node.js, Lean OpenWhisk, Kubeless
- Infrastructure: Raspberry Pi 3 B+
- Measure latency at different load levels with hard SLA

Does it work?

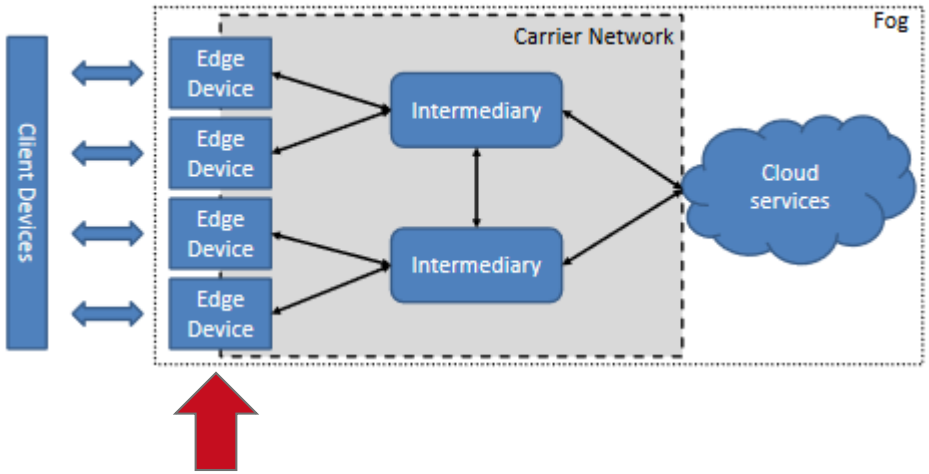


- Native node.js: very low overhead of tinyFaaS
- Lean OpenWhisk: OW does not work on Raspberry Pi (at least 35% error rate, latency = 100x tinyFaaS)
- Kubeless: comparable at very low load but does not scale

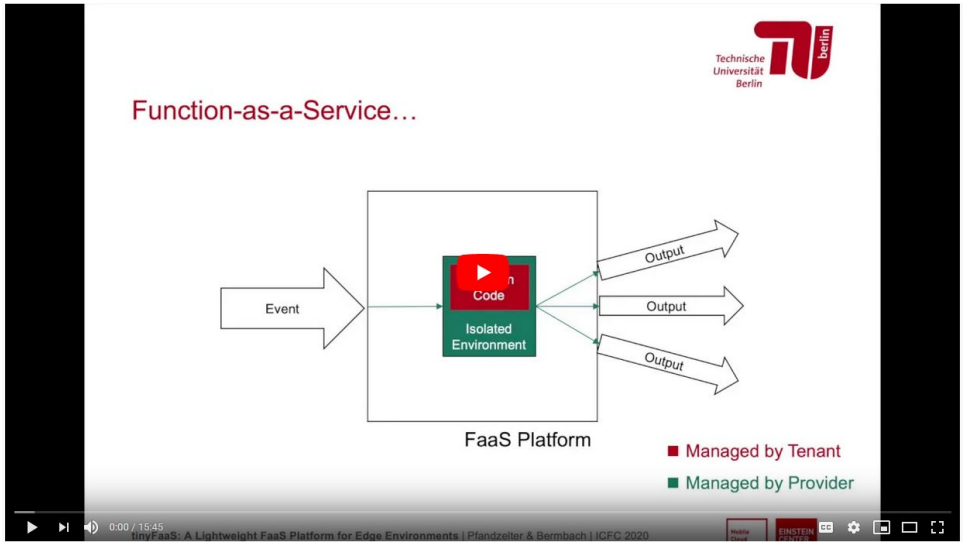
Does it suffice for the edge?

+ tinyFaaS is designed for small edge nodes
+ much more efficient than alternative solutions

- No support for cluster-based deployments
- No support for on-device deployments



Further details



https://www.youtube.com/watch?v=mte3fUAagm4

FaaS for the edge

NANOLAMBDA

George et al. NanoLambda: Implementing Functions as a Service at All Resource Scales for the Internet of Things. ACM SEC 2020.

Figures and text reused with the authors kind permission.

Approach

Technische Universität Berlin

Targeted at extremely resource constrained devices

- ESP8266 with 96KB of RAM and 512KB of program flash storage
- CC3220SF with 256KB of RAM and 1MB of program flash storage

Subset of standard libraries

Implements AWS Lambda API

Builds on CSPOT [1] which provides a low-level FaaS framework

[1] R. Wolski, et al, CSPOT: Portable, Multi-scale Functions-as-a-Service for IoT, ACM Symposium on Edge Computing (SEC 2019)

Mobile Cloud Computing

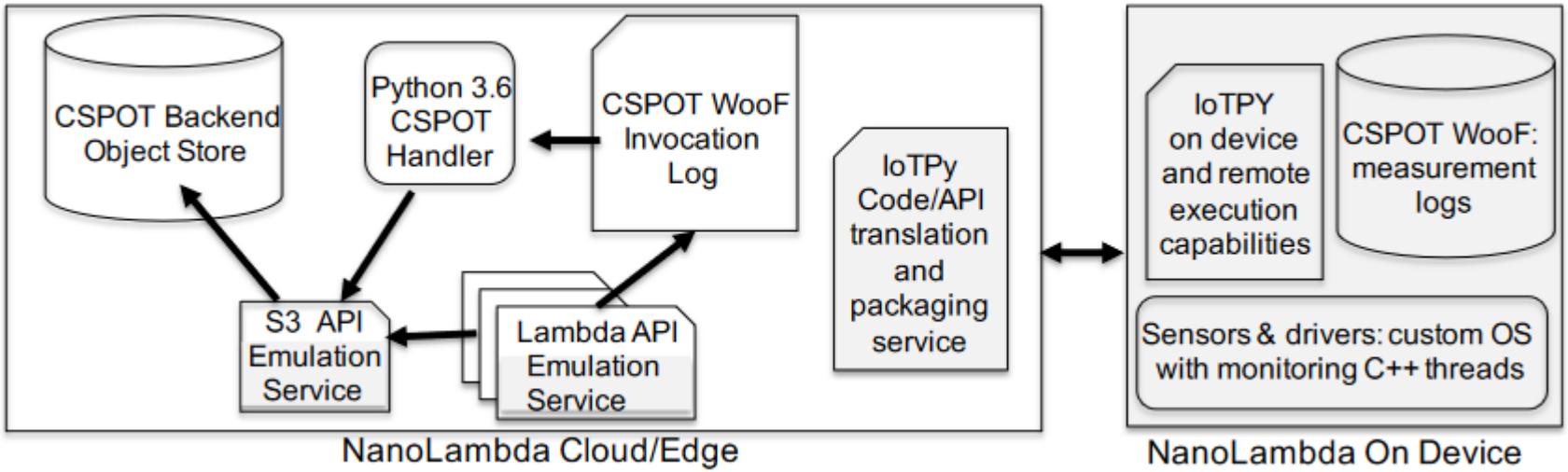
EINSTEIN CENTER Digital Future

Approach (cont.)

Lightweight python VM for bytecodes (IoTPy): saves code space by omitting non-core language features

Remote compilation (cloud/edge): code is never delivered to device

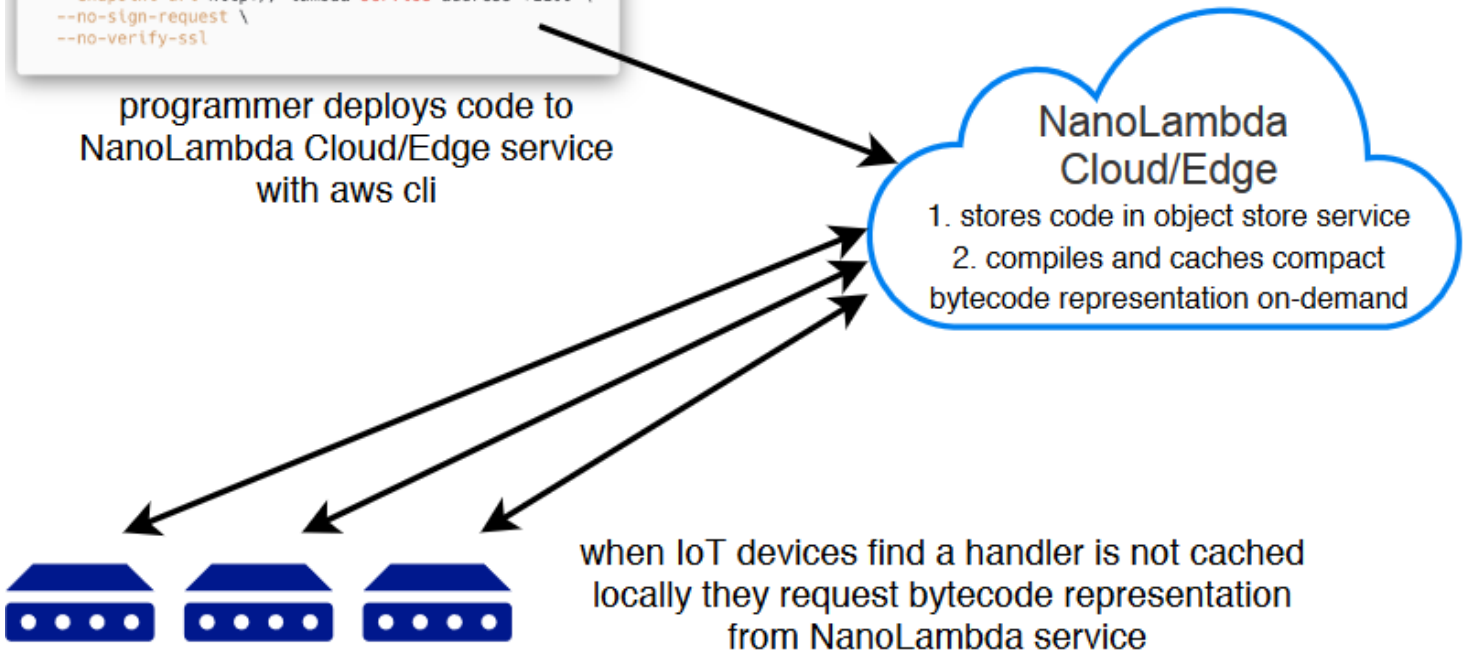
Compact representation is serialized with flatbuffers and sent over network



Life cycle of functions

```
# create zip file with python application
zip edgelambda.zip edgelambda.py
# upload it to open source lambda service
#     for compilation and deployment
aws lambda create-function \
  --function-name scheduled_pred_main \
  --zip-file fileb://edgelambda.zip \
  --handler edgelambda.new_accel_sample \
  --runtime python3.6 \
  --endpoint-url http://<lambda service address>:1100 \
  --no-sign-request \
  --no-verify-ssl
```

programmer deploys code to NanoLambda Cloud/Edge service with aws cli

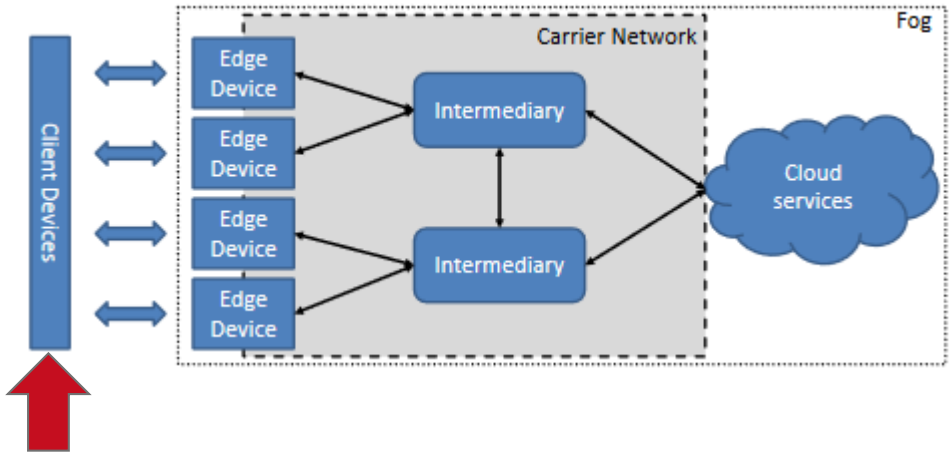


when IoT devices find a handler is not cached locally they request bytecode representation from NanoLambda service

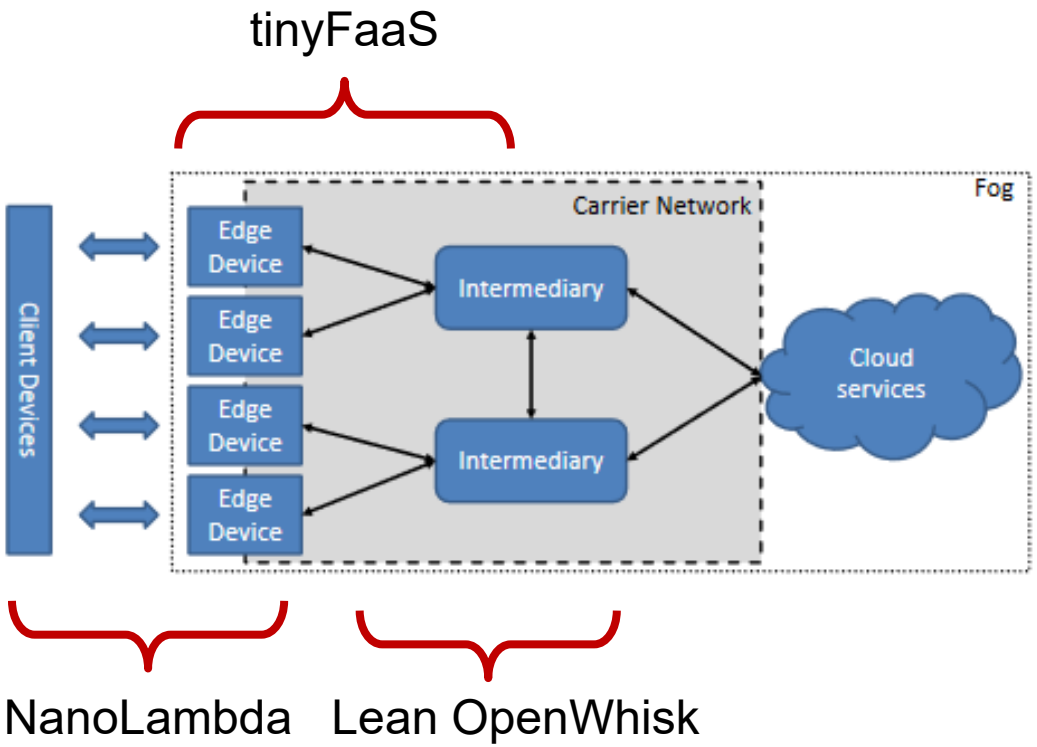
Does it suffice for the edge?

+ designed for small edge nodes and on-device
+ cloud-compatibility

- no support for cloud-based clusters



Faas for the edge



FaaS for the edge

AUCTIONWHISK

David Bermbach, Setareh Maghsudi, Jonathan Hasenburg, Tobias Pfandzelter. Towards Auction-Based Function Placement in Serverless Fog Platforms. IEEE ICFC 2020.

David Bermbach, Jonathan Bader, Jonathan Hasenburg, Tobias Pfandzelter, Lauritz Thamsen. AuctionWhisk: Using an Auction-Inspired Approach for Function Placement in Serverless Fog Platforms. Wiley SPE 2021.

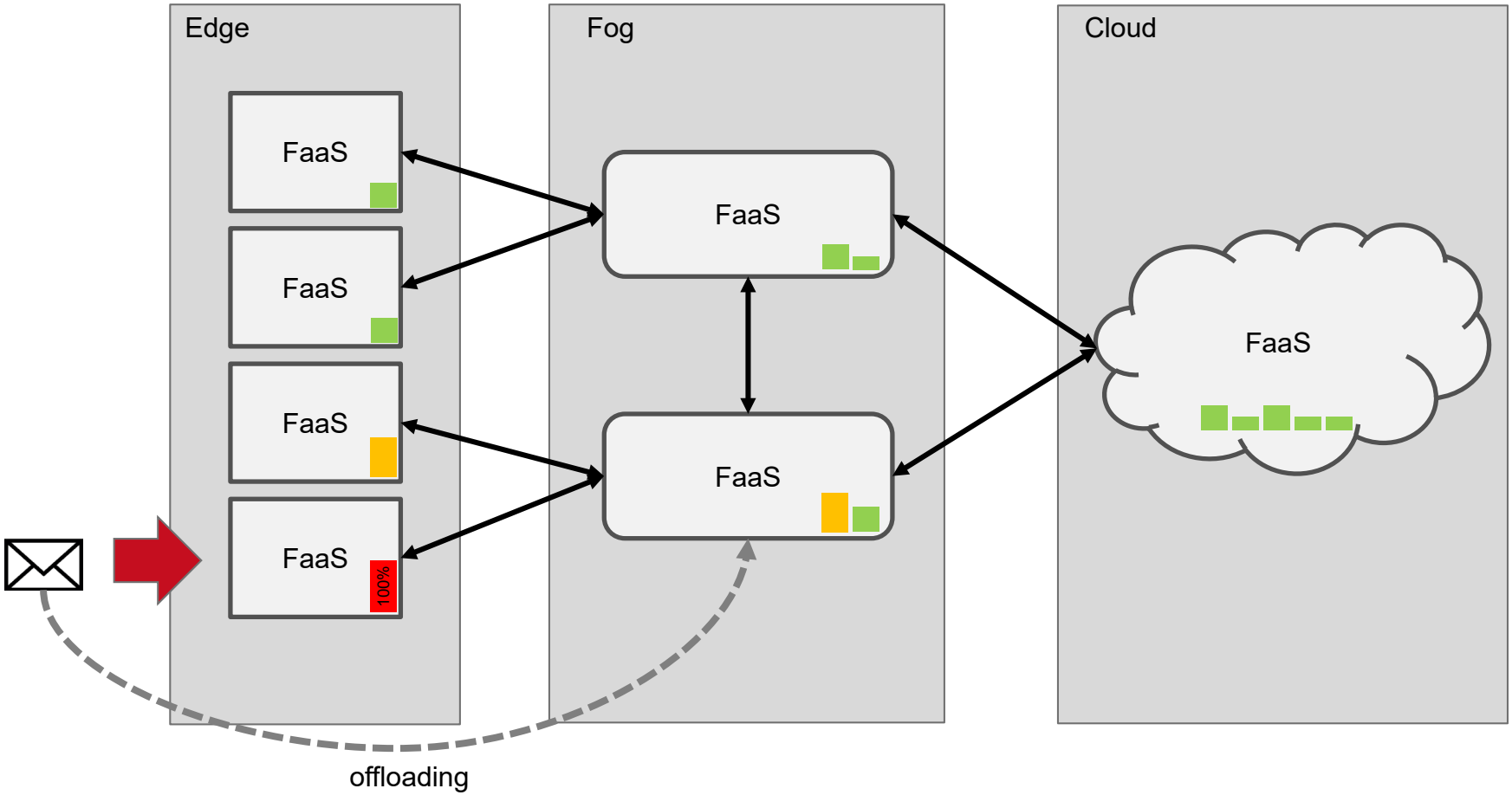
How to decide which function to execute where

Requests…
…arriving in the cloud => execute in the cloud
…arriving on the edge => execute on the edge

⇒ Optimal QoS (unless special data requirements or service dependencies)

⇒ But…

What happens when an edge node is overloaded?



How to pick a request for offloading?

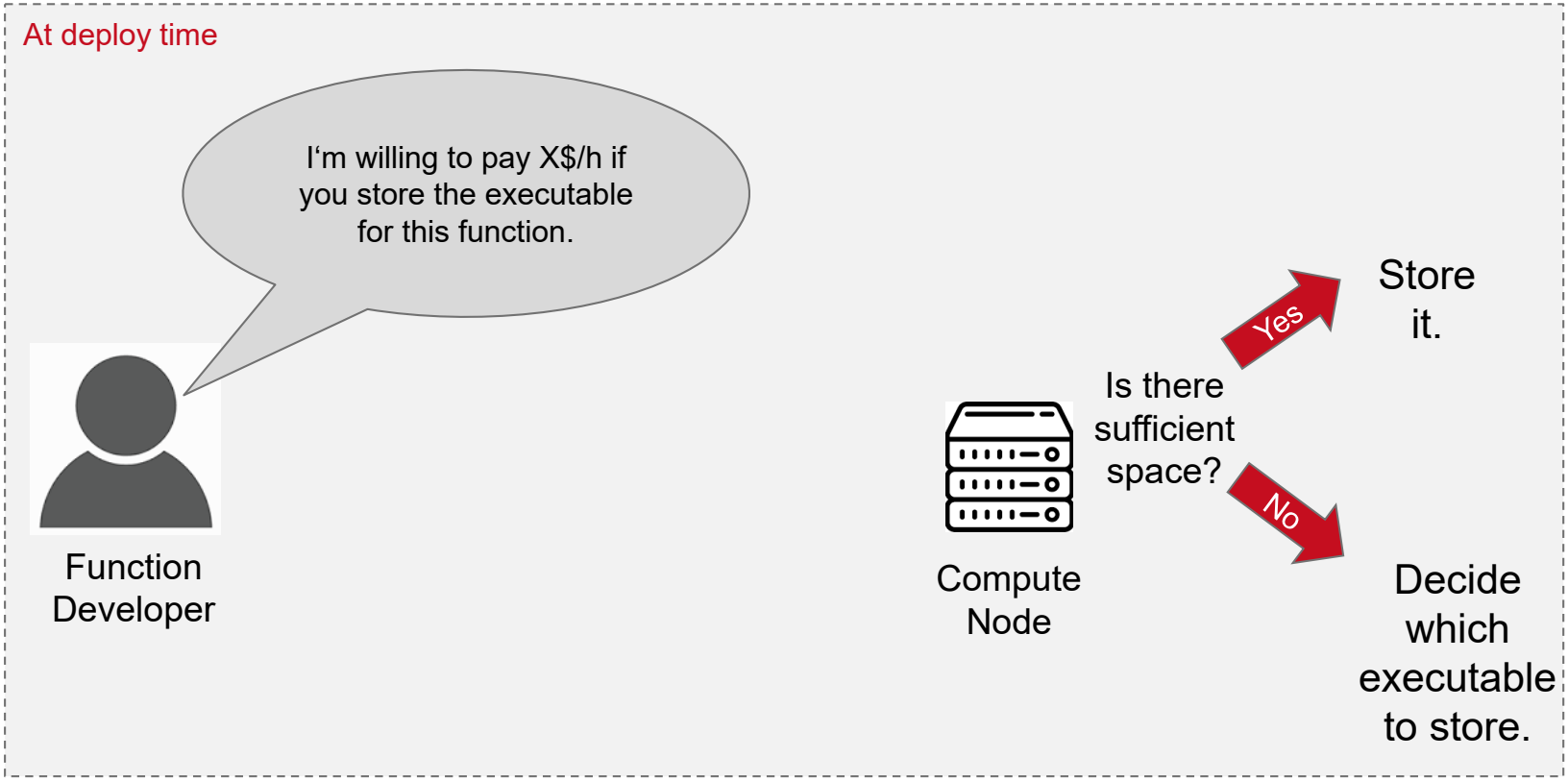
Idea:

- Let application developers bid on resources during deployment.
- When resources run low, nodes offload the request with the lowest bid.
- Separate bids for storage and execution offer fine-grained control

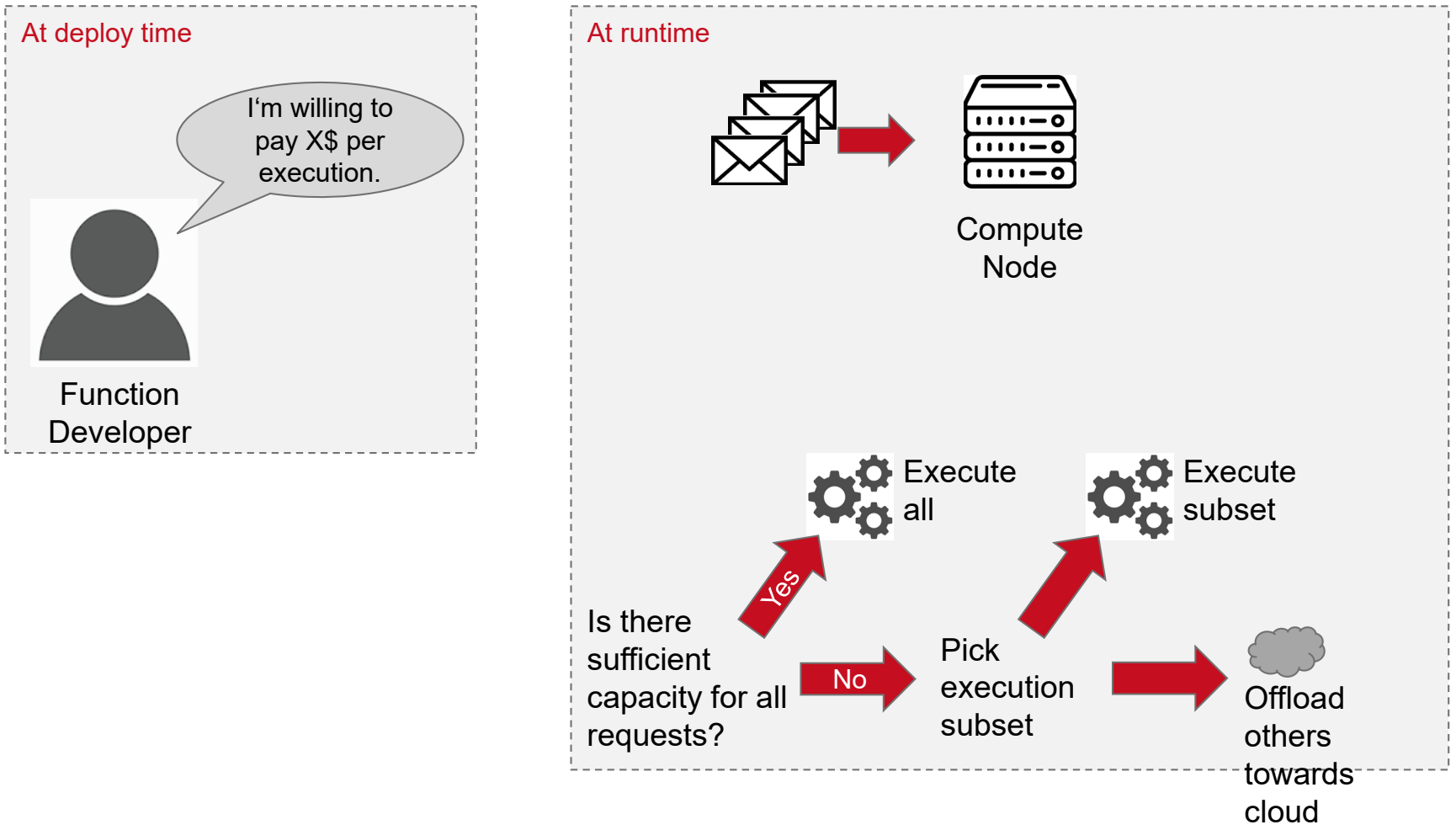
Advantages:

- Efficient resource allocation
- Decision is entirely local (=> it scales!)
- Easy to implement
- Maximizes profits for edge nodes (=> edge CapEx)

Bidding on storage



Bidding on execution capacity



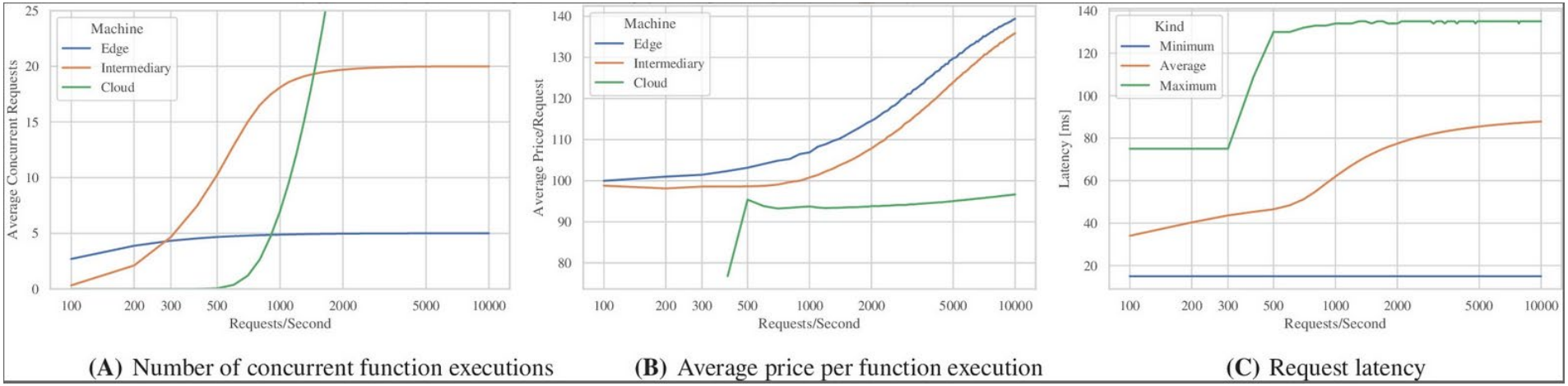
Implementation challenges

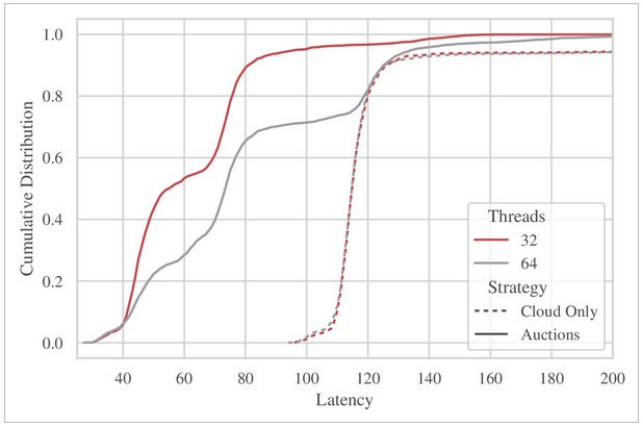
Requests don‘t arrive in batches

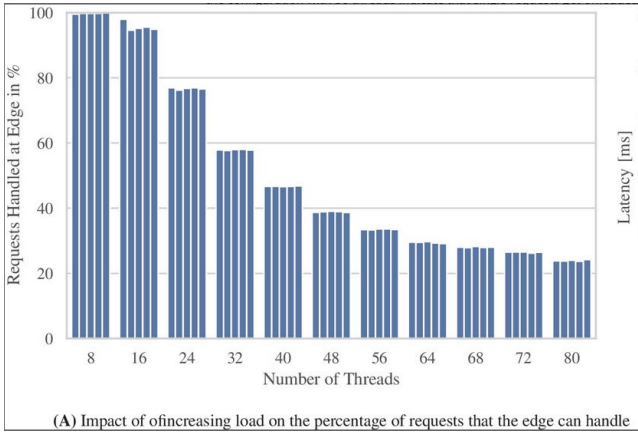
Estimating compute load and storage space

Deciding where to offload

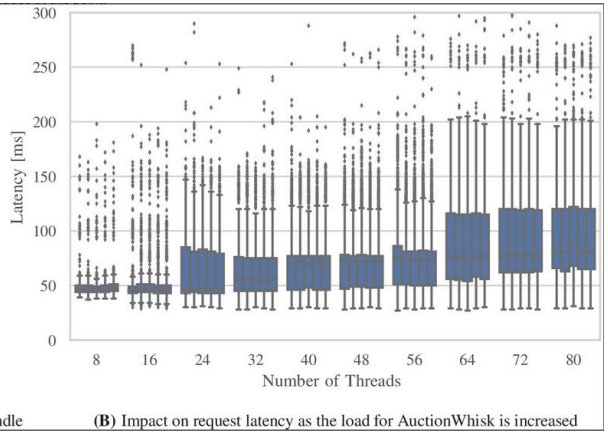
Simulation and experiment results







(A) Impact of ofincreasing load on the percentage of requests that the edge can handle



(B) Impact on request latency as the load for AuctionWhisk is increased

Technische Universität Berlin

Platforms for the edge

EDGE CONTAINER ORCHESTRATORS

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

(Cloud) Container orchestrators

Technische Universität Berlin

Cloud applications often build upon container orchestrators such as Kubernetes to alleviate infrastructure and container management efforts:

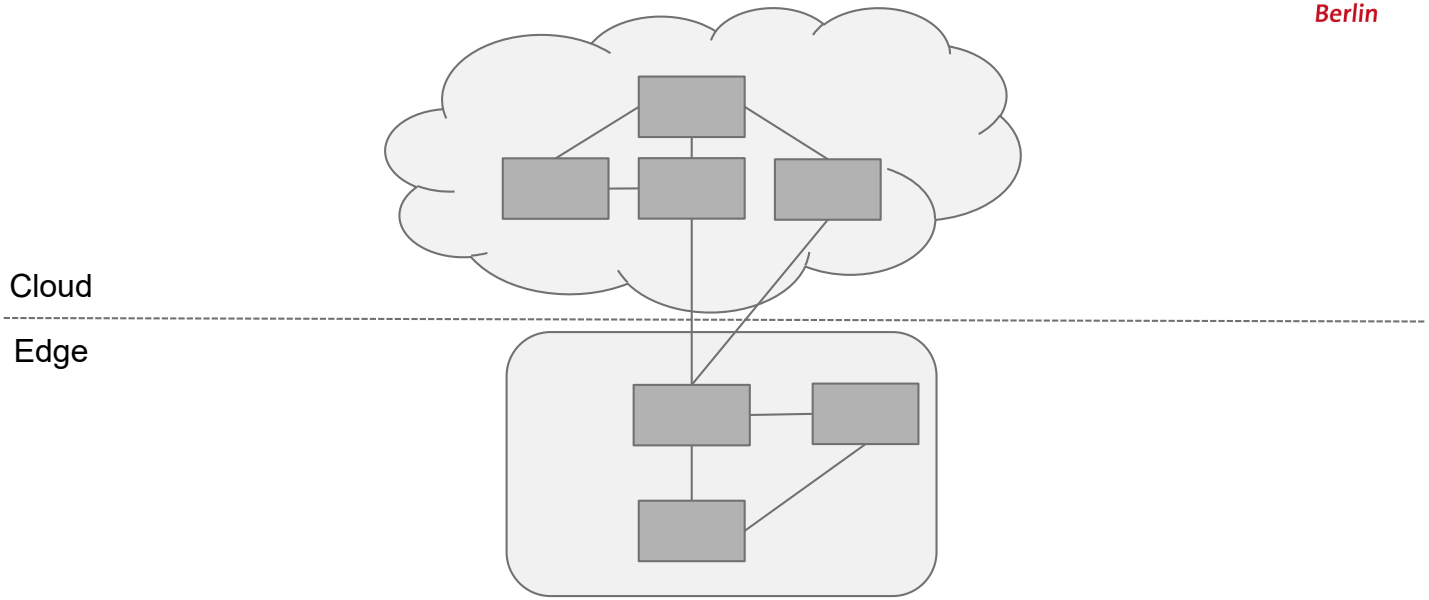
- Container scheduling
- Load balancing
- Resource limit control
- Health check
- Fault tolerance
- …

Is Kubernetes edge-ready?

- Large management overhead
- Small edge devices
- Unreliable connections
- …

How to “extend” Kubernetes to the Edge?

Edge container orchestrators



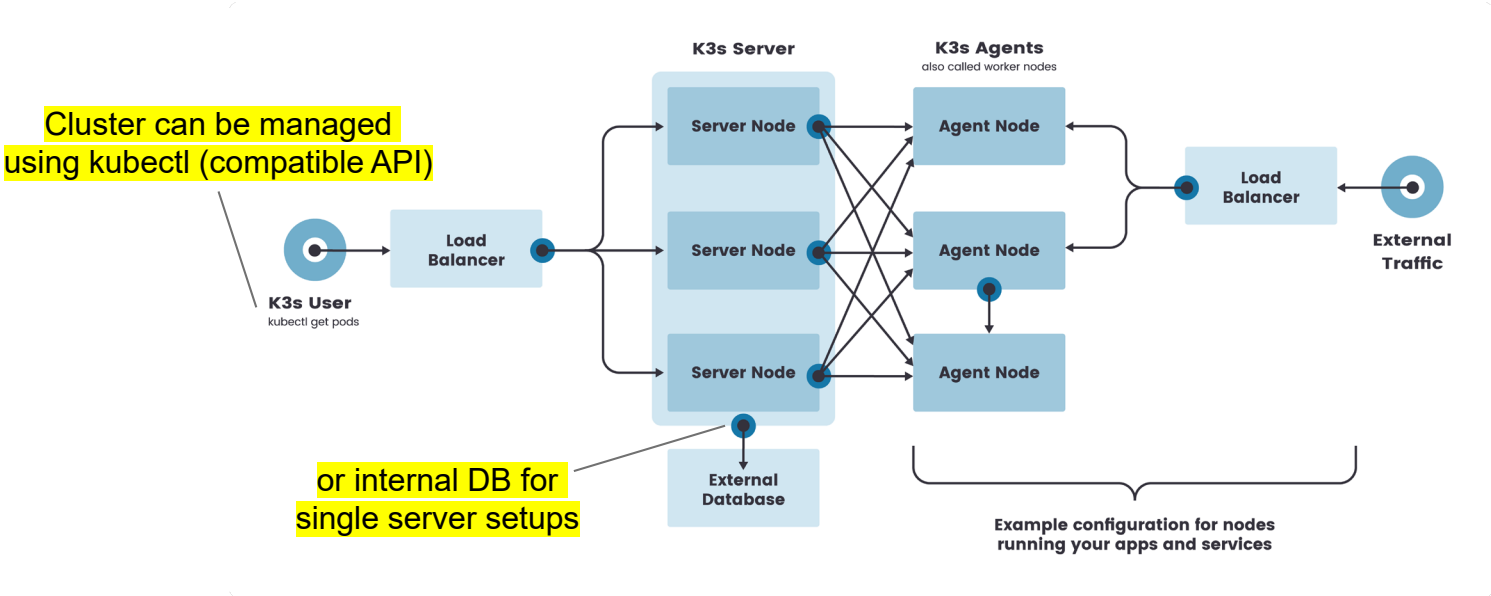
There are first proposals for edge container orchestrator (ECO) frameworks that are specifically tailored for the edge (e.g., K3s, MicroK8s, and OpenYurt)

Idea:

- Provide the same feature set as K8s but optimize it for the edge
- Lightweight and less resource demanding orchestrators

K3s

- Fully compatible with the K8s API
- Removes unnecessary/cloud-specific features (e.g., cloud storage plugins)
- Bundled into a ~100MB single binary (easy to install)
- K3s server for cluster management and K3s agents running services
- Runs on devices as small as a Raspberry Pi



Source: https://rancher.com/docs/k3s/latest/en/architecture/ [May 13, 2022]

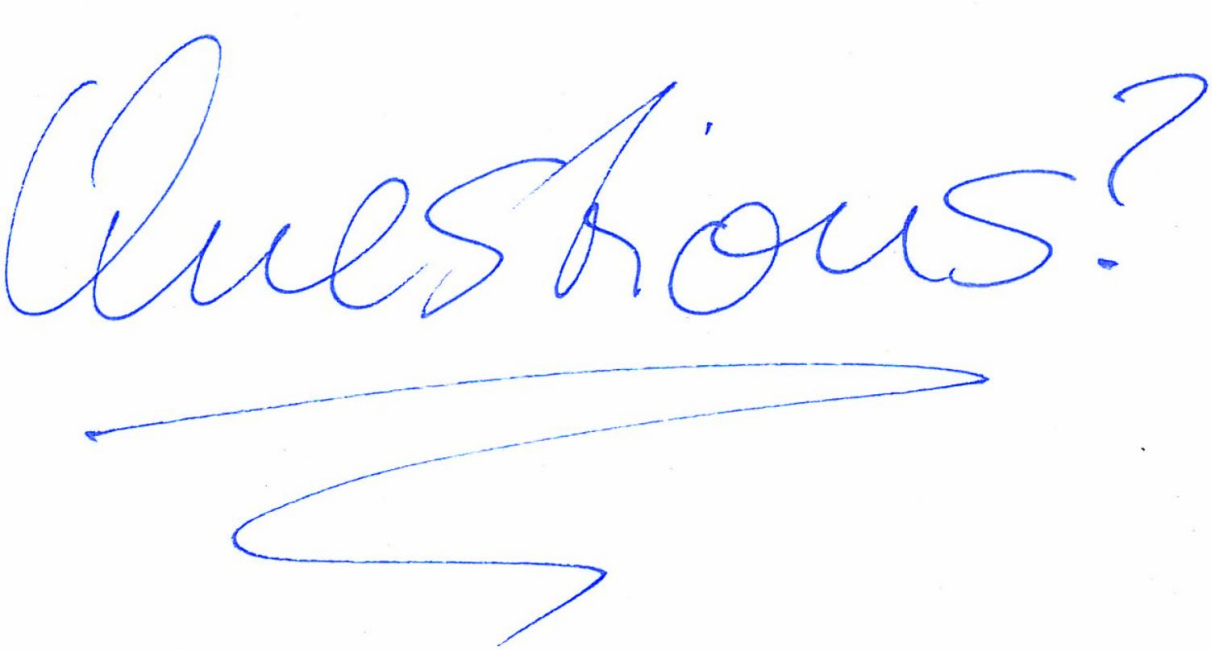
Summary

Addressing geo-awareness in applications is hard

Novel fault-tolerance challenges in fog environments

FaaS can be a good platform approach for the edge
- Better resource utilization
- Powerful programming model
- Flexibility
- Event-driven is a good fit for many edge/fog applications

There are also Kubernetes extensions for the edge


Questions?