Serverledge: Decentralized Function-as-a-Service for the Edge-Cloud Continuum

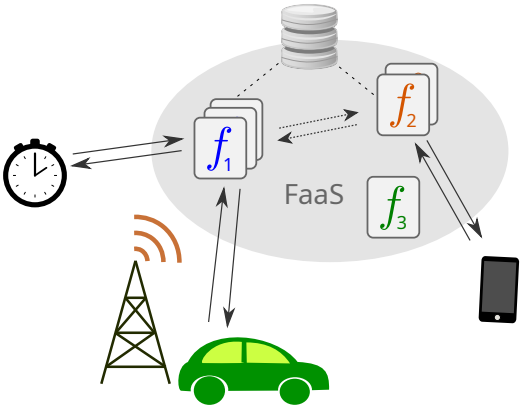
Gabriele Russo Russo Tiziana Mannucci Valeria Cardellini Francesco Lo Presti
University of Rome Tor Vergata, Italy

21st International Conference on Pervasive Computing and Communications (PerCom 2023)

TOR VERGATA
UNIVERSITY OF ROME

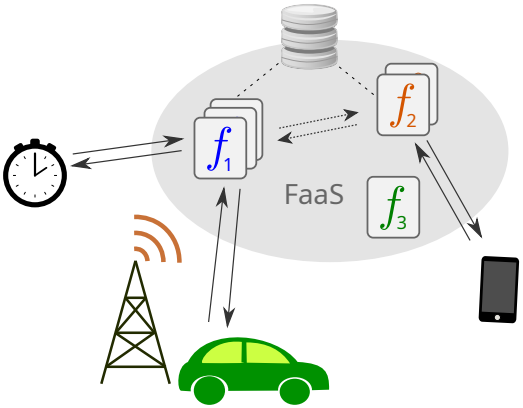
Function-as-a-Service

- Write functions, enjoy serverless execution
- Seamless scalability
- Fine-grained pricing
- Simplified app maintenance



Function-as-a-Service

- Write functions, enjoy serverless execution
- Seamless scalability
- Fine-grained pricing
- Simplified app maintenance
- Mostly Cloud-based!



FaaS Frameworks

- All the major Cloud providers have FaaS offerings, e.g.:
  - AWS Lambda
  - Google Cloud Functions
  - Azure Functions
  - IBM Functions
- Various open-source FaaS frameworks, usually deployed in Kubernetes clusters, e.g.:
  - OpenWhisk
  - OpenFaaS
  - KNative

Functions in the Edge-Cloud Continuum

- What about FaaS for pervasive services?
- Cloud execution not always acceptable
- Should we run functions across the Edge-Cloud Continuum?
  - Low-latency at the edge
  - Resource richness at higher layers, when needed
- But…methodologies, tools and platforms for FaaS in the Continuum still under development

Key Challenges

- Cloud-centric architecture
- Resource-constrained devices
- Mobility
- Performance unpredictability
- Privacy

Key Challenges

- Cloud-centric architecture
- Resource-constrained devices
- Mobility
- Performance unpredictability
- Privacy

+ FaaS-specific issues:

- Cold start
- Container image caching
- Stateful functions

State of the Art

- Major frameworks not suited for Edge environments
  - Cloud-oriented, centralized design
  - No support for service differentiation among users
- Two main directions pursued so far
  - Federating existing FaaS frameworks (e.g., through overlay networks)
  - Designing novel frameworks for the Edge

Frameworks for FaaS at the Edge

Framework	Geo. Distribution	Offloading	Execution Env.
OpenWhisk	No	No	Container
faasm	No	Horizontal	WASM-based
Sledge	No	No	WASM-based
tinyFaaS	No	No	Container*

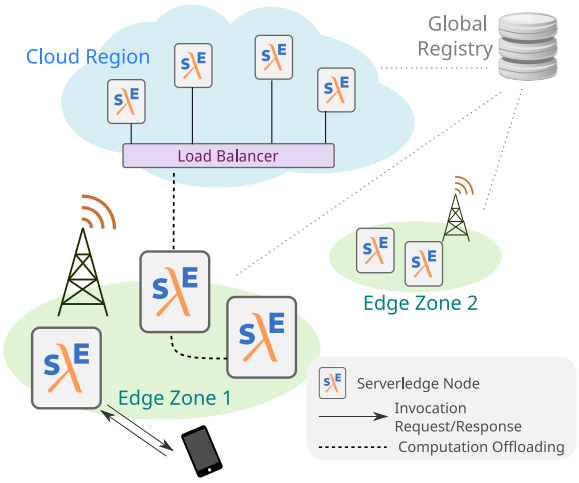
* = static pool

Goals

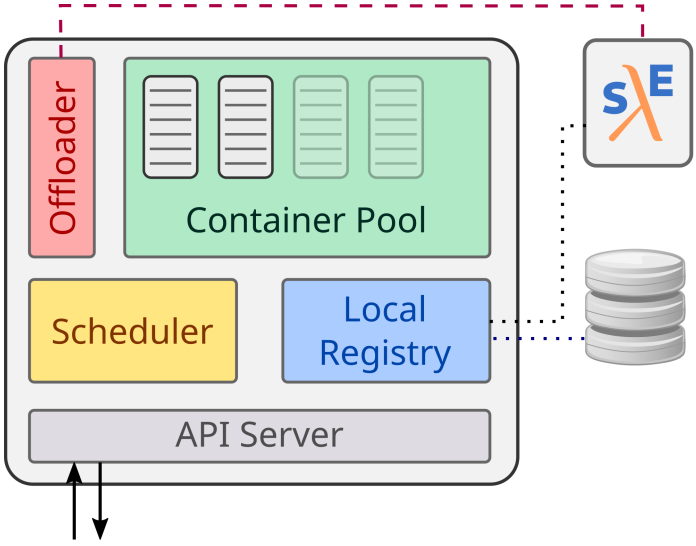
- Designing a FaaS framework with a decentralized architecture
- Horizontal and offloading mechanisms
- Support for QoS-aware scheduling
- Comparison against existing frameworks for FaaS at the Edge

Serverledge

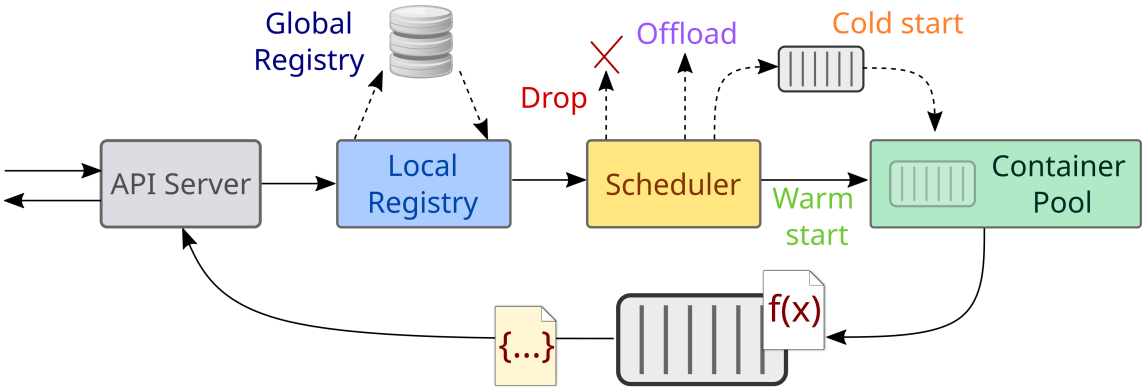
- A framework for the Continuum
- Nodes organized into Cloud regions and Edge zones
- (Replicated) Global Registry
- No centralized entry point!
- Vertical and horizontal offloading mechanisms



Node Architecture



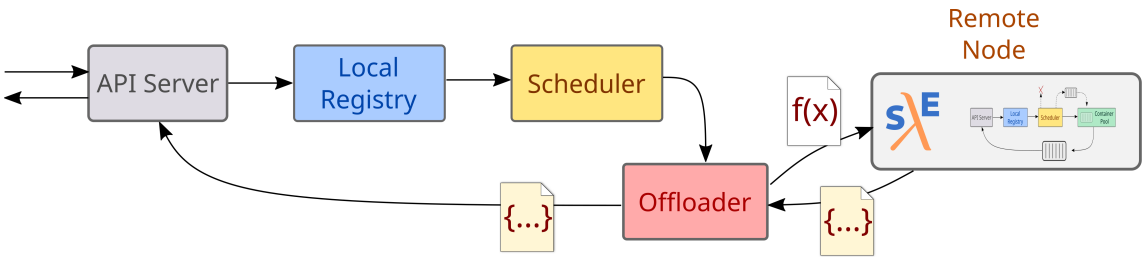
Journey of a Request



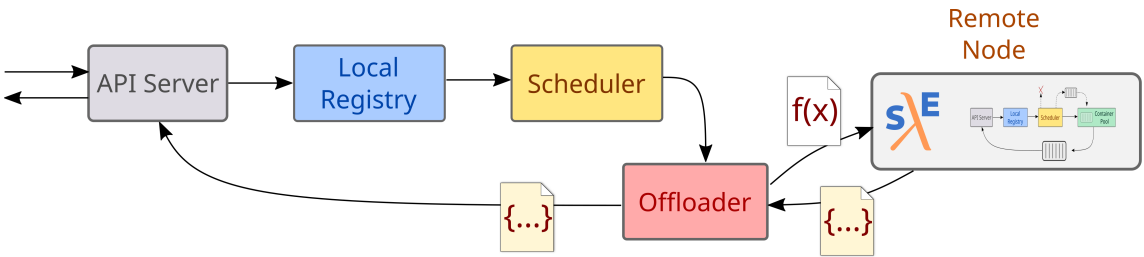
Offloading Mechanism

- Edge devices are likely to become overloaded (at least temporarily)
- Idea: offloading some requests
  - save local resources for “critical” invocations
  - exploit Cloud nodes when latency doesn’t matter
- Vertical/Horizontal offloading
- Each node acts as a reverse proxy in case of offloading

Journey of an Offloaded Request



Journey of an Offloaded Request



How to choose a destination node for offloading?

- Edge: send to a neighbor, based on proximity and current load
- Cloud: send to a load balancer

Implementation

- Implemented in Go
- Each node runs as a single process, which interacts with a Docker daemon for container management
- Global registry implemented on top of Etcd
- Additional components:
  - `serverledge-cli`: CLI interface
  - a load balancer for Cloud nodes

Evaluation

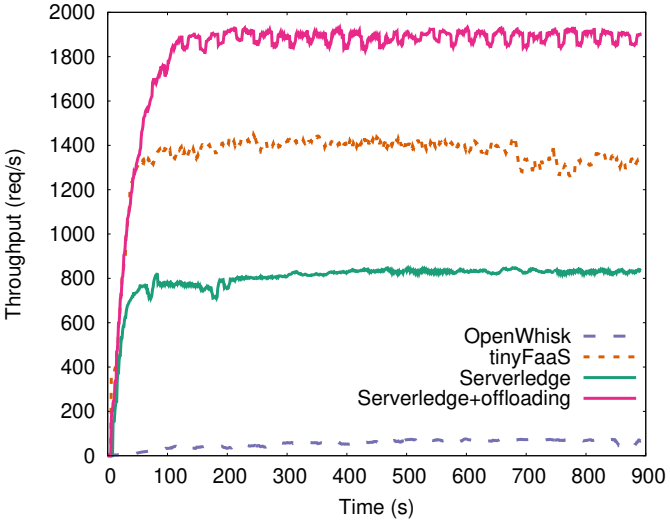
Goal 1: Performance comparison against SOTA alternatives

- Single-node deployment
- Load generated using Locust
- Comparison against faasm, tinyFaaS and OpenWhisk

Goal 2: Demonstrating advantages of QoS-aware offloading

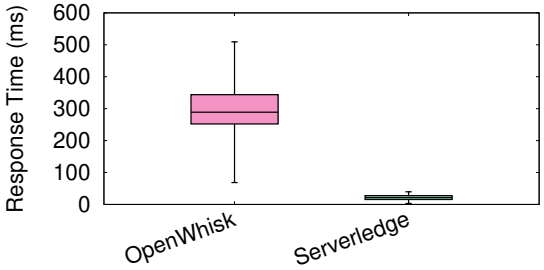
- 2 AWS Cloud regions act as “Cloud” and “Edge”
- 2 classes of users: best-effor and latency-sensitive

Results: Throughput

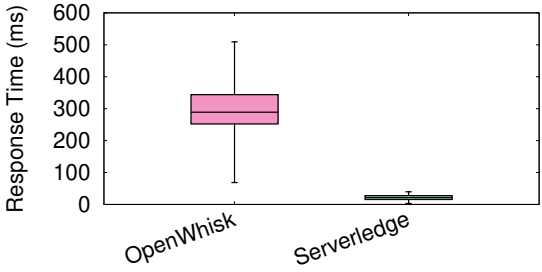


- tinyFaaS has the highest throughput (thanks to a static pool of containers!)
- OpenWhisk has the lowest one
- Serverledge can increase throughput via offloading

Results: Response Time

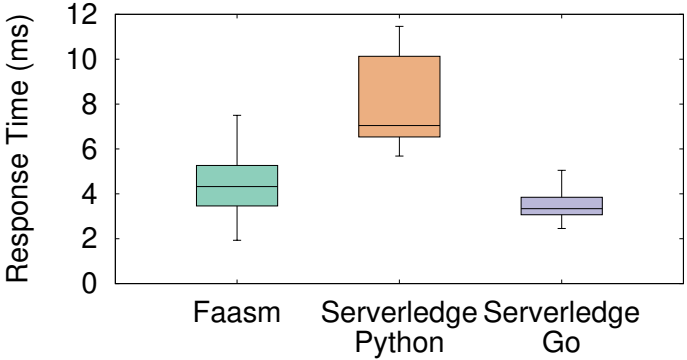


Results: Response Time



Results: Faasm Comparison

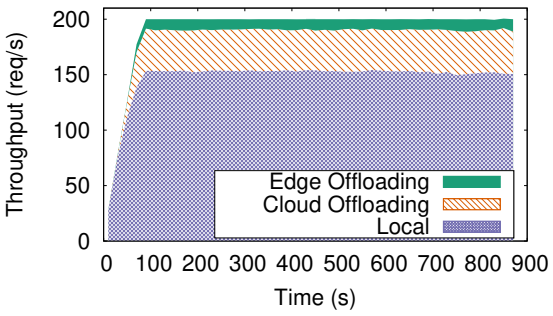
- We could not run faasm at high throughput because of a known issue causing crashes; focus on response time
- We use Python and Go implementations of *Fibonacci* computation



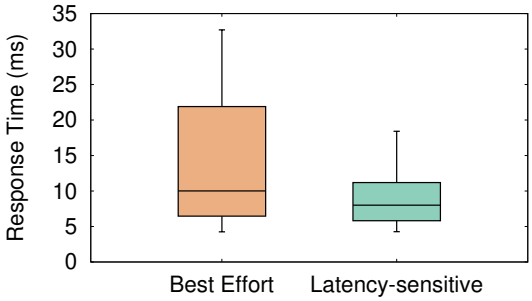
Results: Edge-Cloud Offloading

Proof-of-concept policy with 2 classes of users

Served requests



Response time



Conclusion

- A framework with a decentralized and easy-to-extend design
- Built-in support for offloading and service differentiation
- Open-source: `https://github.com/grussorusso/serverledge`
- …and “Artifact Evaluated”!

Already working on…

- Lightweight function execution environments
- Live function migration
- Adaptive offloading policies