FaaS@Edge

Bringing Function-as-a-Service to Voluntary Computing at the Edge

Catarina Gonçalves, José Simão, Luís Veiga

PRR Plano de Recuperação e Resiliência
REPÚBLICA PORTUGUESA
Financiado pela União Europeia NextGenerationEU

PT Smart Retail

Portugal como referência para a nova geração de retalho autónomo e inteligente

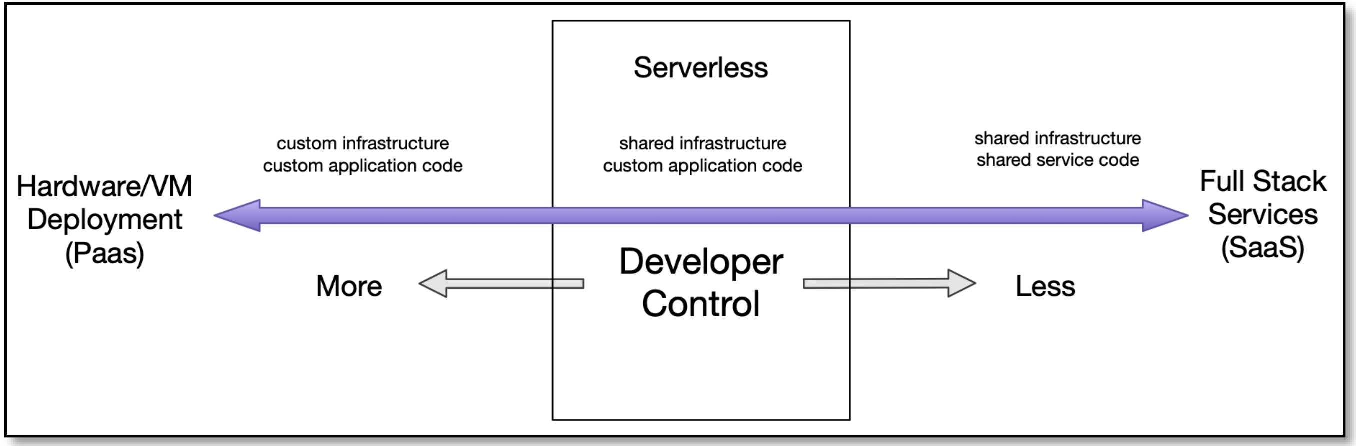
CLOUD STARS

With the support of EU Horizon Europe research and innovation programme under grant agreement No 101086248 (CloudStars).

GECON 2024 - 20th International Conference on the Economics of Grids, Clouds, Systems, and Services
Rome, Italy, 2024/09/26

Motivation

- The Functions-as-a-Service movement
  - Event-based functions executing mostly stateless operations

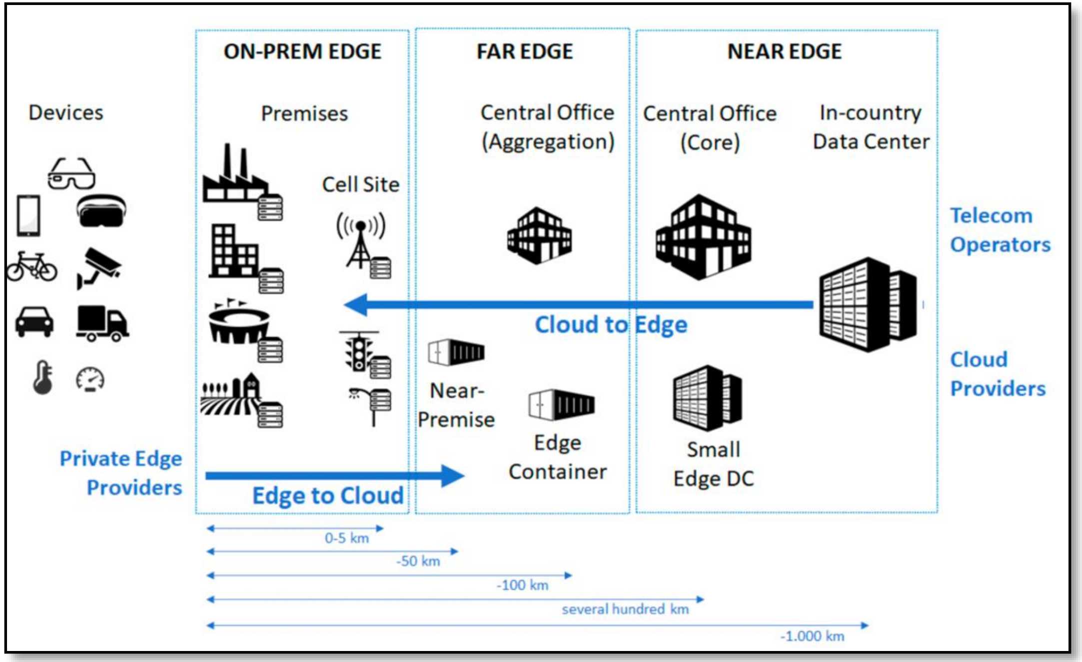


Serverless Computing: Current Trends and Open Problems, *Research Advances in Cloud Computing. Springer (2017)*

  - Available as fully managed Cloud Services and Open-source Middlewares
  - Usually short lived computations that are more sensitive to memory usage

Motivation

- The Edge Computing movement
  - Heterogeneous set of devices available for computation, particularly on-prem
  - Reduces data transmitted to the cloud, saves bandwidth, enhances privacy



Study on the Economic Potential of Far Edge Computing in the Future Smart Internet of Things, European Commission (2021)

Challenges

Centralized Architectures

- Cloud platforms mostly on centralized architectures

Cloud services depend on resource-intensive environments

- Cloud services are not designed to operate on resource-constrained environments

Heterogeneous Devices

- Edge systems encompass a variety of heterogeneous devices

Distributed data visibility and sharing

- Moving away from cloud relies distributed forms of data sharing

Related work

Distribution of computing power

Grid Computing

Consumption-based business model

(pay-per-use elasticity)

Cloud Computing

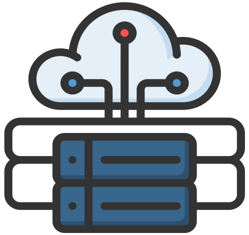
Leverage users' resources (e.g., in **SETI@home** [2], **nuBOINC** [3])

Edge Computing

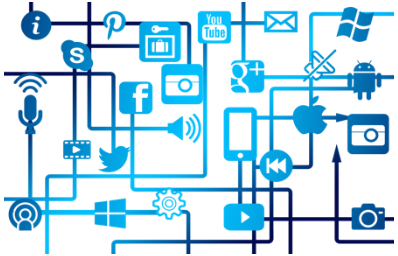
Closer to the source of data (e.g., in **Cloudlets** [4])

Related work

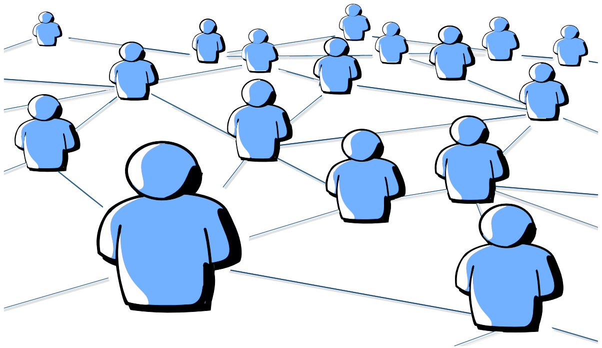
- Centralized vs Distributed data management



Cloud Storage



Content Delivery Networks



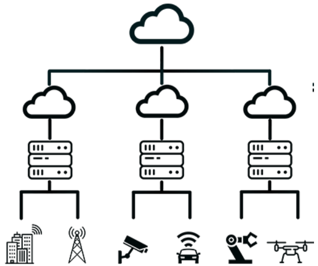
P2P Data Networks

- Overlay networks where peers can autonomously share their resources
- Efficiently locate and transfer files across peers (often final users, e.g., **IPFS** [6])

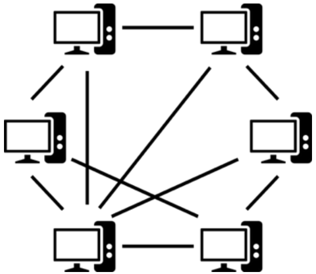
Contributions

- A distributed middleware architecture (algorithms and protocols) leveraging volunteer resources for FaaS deployments on Edge Computing nodes

Function-as-a-Service

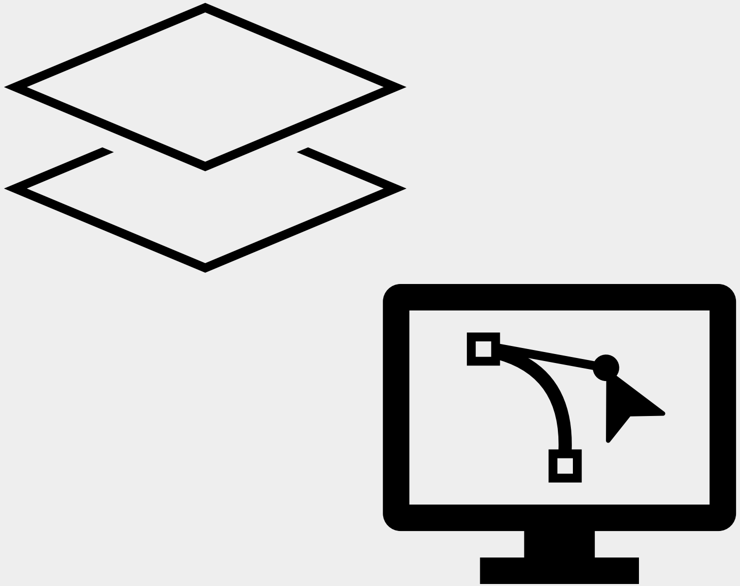


Edge Computing

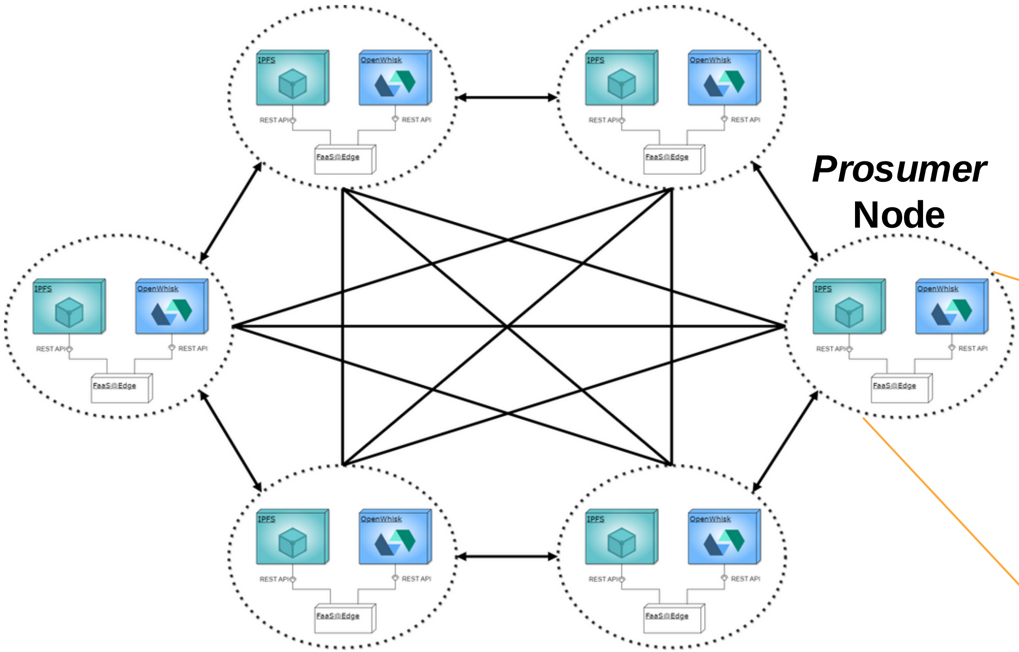


Peer-to-Peer Content Storage and Distribution

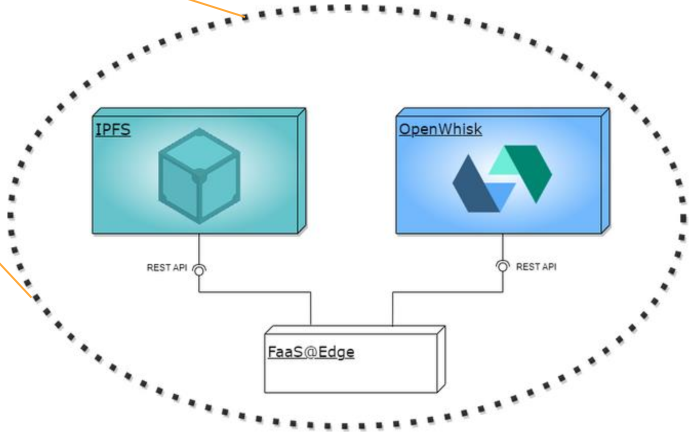
Architecture and Algorithms



Architecture



Nodes act as *prosumers*, can be producers and consumers of resources



FaaS@Edge sits on top of InterPlanetary File System (IPFS) peer-to-peer architecture and its content-addressing data mechanism

Supplying resources

- Supplier node runs OpenWhisk and calculates a set of *offers* based on free memory and how much it is willing to share
- Announces its memory resources to the network through the IPFS, using Content Identifiers (hash-based labels used to point to material in IPFS)

Algorithm 1 Supply Resources algorithm

```
function SupplyResources(freeRes, maxRes):
    usedRes ← ResourcesInUse(freeRes, maxRes)
    RemoveAllOffers()
    offerCount, offerSize ← CalculateOffers()
    foreach offerCount, offerSize do
        newOffer ← CreateOffer(offerSize)
        supplierActiveOffersMap.Add(newOffer)
```

Discovery and Scheduling

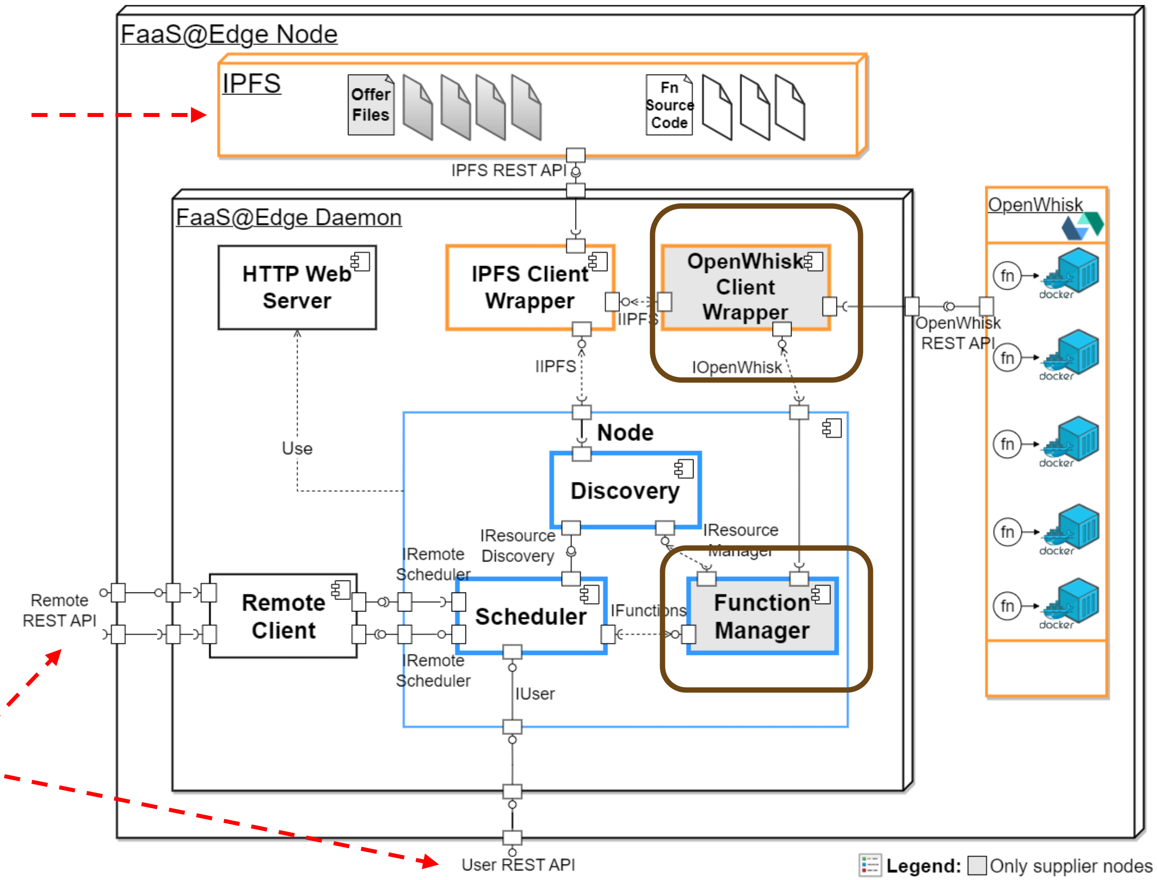
- Consumer nodes (running the client interface to FaaS@Edge) *search* the network for potential supplier nodes with compatible offers
- A supplier node confirms or rejects the acceptance to execute the function
- Metadata is updated in the IPFS network
- Concurrency in the process can lead to failures

Algorithm 4 Function Submission Scheduling algorithm

```
function Schedule(fnConfig):
  resNeeded ← fnConfig.Resources
  availOffers ← DiscoverResources(resNeeded)
  availOffers ← RandomOrder(availOffers)
  foreach offer in availableOffers do
    fnStatus ← SubmitFunction(fnConfig, offer, self.IP)
    if fnStatus = ok then
      deployedFunction ← NewDeployedFunction(fnConfig, offer.SupplierIP)
      functionsMap.Add(deployedFunction)
      return fnStatus
  return Error("Unable to schedule function")
```

FaaS@Edge node Architecture

Resource offers and source code references



Local and Remote interfaces

Using the platform

START faasedge **start** -m <memory> [-w]

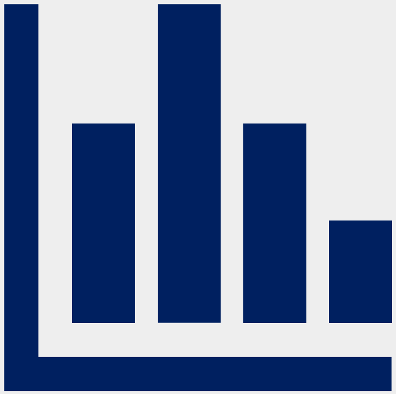
EXIT faasedge **exit**

SUBMIT faasedge **submit** <cid> -m <memory> -n <name> -k <kind>

INVOKE faasedge **invoke** <name> -result -args <json args>

Evaluation

- Workloads characterization
- Testbed
- Latency
  - The time taken to select a node
- CPU and memory
  - Usage at the supplier node
- Request success rate



Example workloads

- Example functions written in Go language, implemented as examples for common use cases of deployments on FaaS@Edge (based on recent research [7])
- *Content Hashing*: Receive data contents and generate SHA256^R hash. Resulting hash returned to user if requested.
- *Database Query*: Request data from a database storing information of books in JSON format. User queries database for specific book using ISBN.
- *Image Transformation*: Get image data using HTTP call and do flip image vertically and returning image data in base64 format.

Testbed

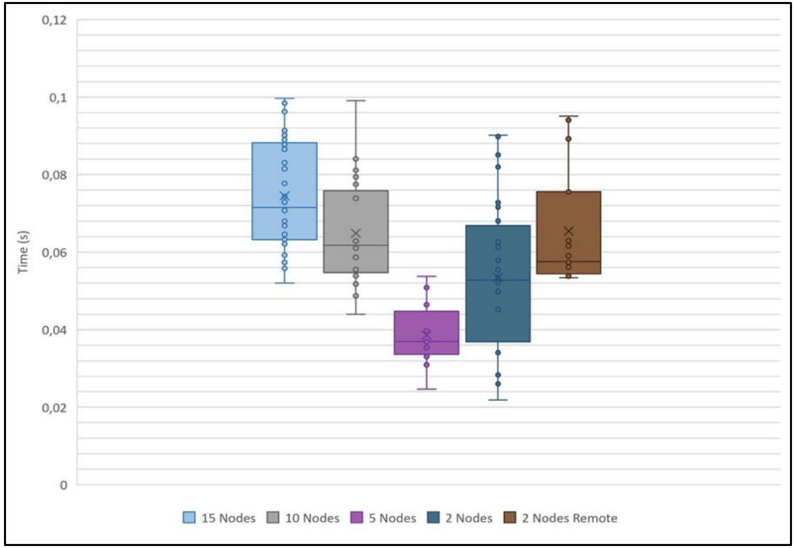
Nodes	Client Nodes	Supplier Nodes
2	1	1
2	1 Remote	1
5	3	2
10	4 + 1 Remote	5
15	6 + 1 Remote	8

Virtual Machine
each instance with
2 vCPUs + 2048MB RAM
exemplificative of edge
devices

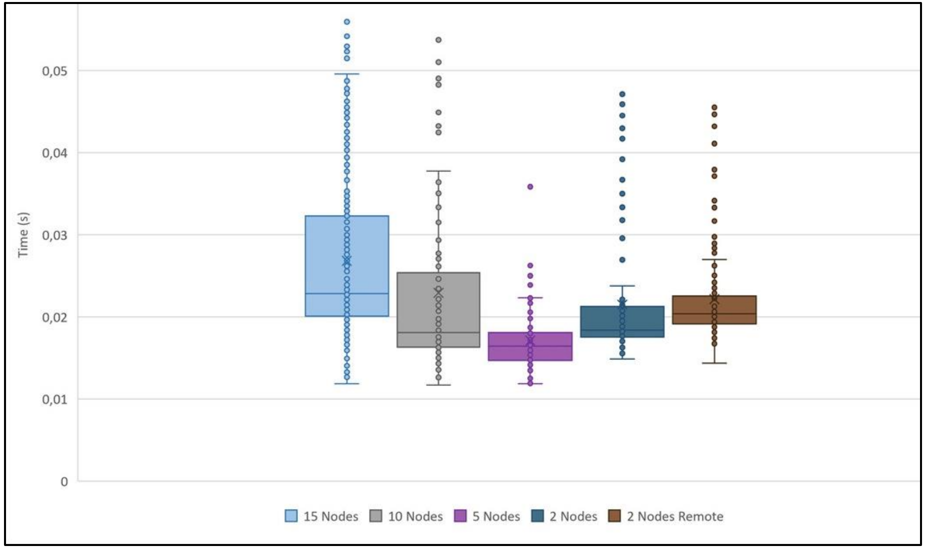
Submission delay/latency

Overall small latency for the set of functions and across different number of nodes

Submission



Invocation

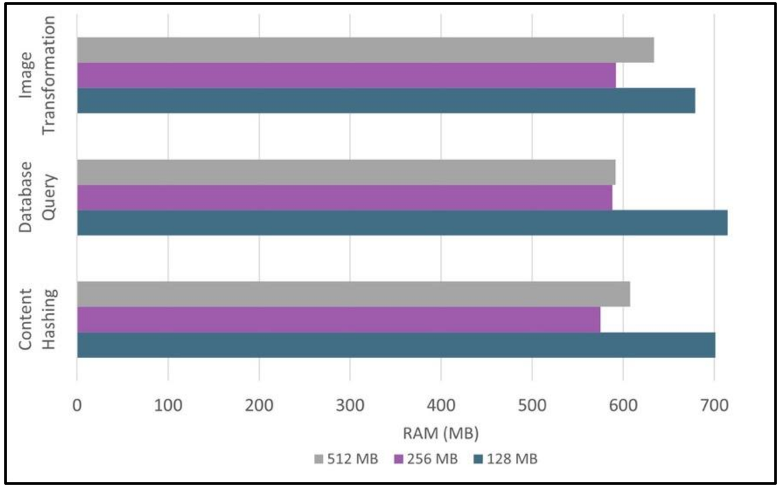


Usage of CPU and memory

Efficient load balancing with CPU usage decrease as more nodes are added to the system

	95th %ile	90th %ile	75th %ile	Median	Average	Client Avg
15 Nodes	2.60%	2.60%	2.40%	2.20%	2.31%	0.30%
10 Nodes	2.90%	2.90%	2.80%	2.70%	2.70%	0.30%
5 Nodes	4.40%	4.40%	4.30%	2.65%	2.91%	0.40%
2 Nodes	7.97%	7.67%	6.75%	5.20%	5.61%	0.80%

Memory system usage per workload and memory requirement



Success scheduling rate

	Request Success Rate	
Function Type	**Submission**	**Invocation**
Content Hashing	99.49%	100.00%
Database Query	95.16%	94.98%
Image Transformation	100.00%	100.00%
Function Memory		
128 MB	95.24%	97.28%
256 MB	99.49%	98.73%
512 MB	100.00%	100.00%
Total Requests	98.76%	98.69%

Overall high request success rates
for submission (node selection)
and invocations (node usage)

FaaS@Edge vs. Local OpenWhisk submission and invocation

	Total Time (s)					Latency (s)
Submission	**95th %ile**	**90th %ile**	**75th %ile**	**Median**	**Average**	**Average**
FaaS@Edge	0.1575	0.1483	0.1349	0.1124	0.1150	0.0653
Local	0.0924	0.0918	0.0696	0.0569	0.0602	NA
Invocation						
FaaS@Edge	0.2742	0.2606	0.2517	0.0485	0.1173	0.0224
Local	0.1707	0.1675	0.1610	0.0691	0.0934	NA

- FaaS@Edge results obtained with 10 node deployment using 256MB function memory and all function types requested by client nodes in cluster machines.
- Remote invocations to the edge nodes are possible at the cost of *slightly* higher invocation times when compared with local deployment

Conclusions and Future Work

- Introduced FaaS@Edge, decentralized system to implement FaaS model in Edge Computing environments
- Function Latency times of invocation requests almost equivalent to local deployment - low performance loss in FaaS@Edge
- Good balancing resulting, adding more nodes yields lower execution times of workloads
- Resources in nodes are well occupied; High request success rate

Future work

- Incentives to prioritize consumers and to reward producers
- Improve execution concurrency of function in each producer node

Thank you!

References

1. Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, et al. Serverless computing: Current trends and open problems. In Research advances in cloud computing, pages 1–20. Springer, 2017
2. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., and Werthimer, D. Seti@ home: an experiment in public-resource computing. Communications of the ACM, 45(11):56–61, 2002.
3. Silva, J.N., Veiga, L., and Ferreira, P. nuboinc: Boinc extensions for community cycle sharing. In 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pages 248–253. IEEE, 2008.
4. Verbelen, T., Simoens, P., De Turck, F., and Dhoedt, B. Cloudlets: Bringing the cloud to the mobile user. In Proceedings of the third ACM workshop on Mobile cloud computing and services, pages 29–36, 2012.
5. Pires, A., Simão, J., and Veiga, L. Distributed and decentralized orchestration of containers on edge clouds. J. Grid Comput., 19(3):36, 2021.
6. Benet, J. IPFS-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561, 2014.
7. Dukic, V., Bruno, R., Singla, A., and Alonso, G. Photons: Lambdas on a diet, in Proceedings of the 11th ACM Symposium on Cloud Computing, ser. SoCC ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 45–59.