HeROcache: Storage-Aware Scheduling in Heterogeneous Serverless Edge

The Case of Intrusion Detection Systems

Vincent Lannurien,[*,†] Camélia Slimani,[†] Laurent D'Orazio,[*,‡] Olivier Barais[*,‡]
Stéphane Paquelet,[*] Jalil Boukhobza[*,†]

May 28, 2024

[*] b<>com Institute of Research and Technology
[†] ENSTA Bretagne, Lab-STICC, CNRS, UMR 6285
[‡] Univ. Rennes, Inria, CNRS, IRISA

Table of Contents

HeROcache: Storage-Aware Scheduling in Heterogeneous Serverless Edge – The Case of IDS

Vincent Lannurien*‡, Camélia Slimani‡, Laurent d'Orazio*†, Olivier Barais*†, Stéphane Paquelet*, Jalil Boukhobza*‡

*b<>com Institute of Research and Technology, †Univ. Rennes, Inria, CNRS, IRISA.

‡ENSTA Bretagne, Lab-STICC, CNRS, UMR 6285, F-29200 Brest. France

Email: vincent.lannurien@ensta-bretagne.org, stephane.paquelet@b-com.com, {laurent.dorazio, olivier.barais}@irisa.fr, {camelia.slimani, jalil.boukhobza}@ensta-bretagne.fr

***Abstract*—Intrusion Detection Systems (IDS) are time-sensitive applications that aim to classify potentially malicious network traffic. IDSs are part of a class of applications that rely on short-lived functions that can be run reactively and, as such, could be deployed on edge resources, to offload processing from energy-constrained battery-backed devices. The serverless service model could fit the needs of such applications, given that the platform allows adequate levels of Quality of Service (QoS) for a variety of users, since the criticality of IDS applications depends on several parameters. Deploying serverless functions on unreserved edge resources requires to pay particular attention to (1) initialization delays that could be significant on low resources platforms, (2) inter-function communication between edge nodes, and (3) heterogeneous devices. In this paper, we propose both a storage-aware allocation and scheduling policy that seek to minimize task placement costs for service providers on edge devices while optimizing QoS for IDS users. To do so, we propose a caching and consolidation strategy that minimizes cold starts and inter-function communication delays while satisfying QoS by leveraging heterogeneous edge resources. We evaluated our platform in a simulation environment using characterization data from real-world IDS tasks and execution platforms and compared it with a vanilla Knative orchestrator and a storage-agnostic policy. Our strategy achieves 18% fewer QoS penalties while consolidating applications across 80% fewer edge nodes.**

***Index Terms*—serverless, orchestration, scheduling, edge, cloud, IDS, cache, consolidation, heterogeneous computing**

I. Introduction

IDS, a time-sensitive and critical application: A wide range of embedded systems that operate in static and controlled (*e.g.* sensors in a factory) or dynamic and uncontrolled environments (*e.g.* moving drone swarms) can be temporarily or constantly exposed to critical attacks through network links. As these attacks might jeopardize their execution and seriously damage the related infrastructures, considering them is a critical issue. To mitigate these threats, Intrusion Detection Systems (IDS) are used to analyze network traffic and detect patterns of potentially malicious activities. Machine Learning (ML) models are particularly relevant for a timely classification of the traffic, but are computationally intensive. As a consequence, running them directly on the embedded platform is not a safe solution, as it can affect their lifespan if operating on a battery [8], interfere with other critical tasks, or even be downright impossible to run due to resource shortage.

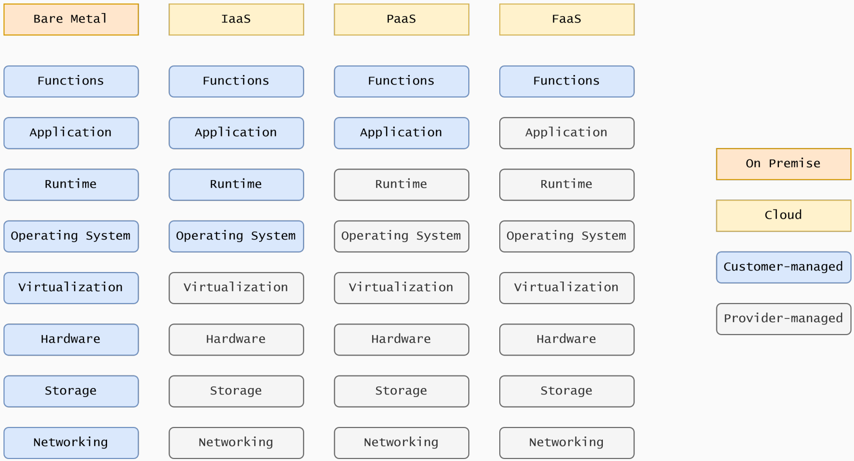
IDS on the edge: A solution to offload these resources-hungry algorithms from deployed embedded systems while keeping the system reactive to attacks is to run IDS in the cloud, and in particular on edge devices [9]. IDS must satisfy variable Quality of Service (QoS) requirements and might be needed only during critical periods, identified beforehand. As a consequence, running IDS on reserved edge devices could be inefficient from a cost perspective. In fact, different types of attacks might have different impacts on the underlying infrastructure. In addition, the risk of attack could change in time and place (according to application domain). We argue that deploying IDS on unreserved low-energy resources on the edge could provide the benefit of a cost-effective solution for running such applications, while keeping the latency lower than when relying on the cloud.

Serverless computing for IDS on the edge: One of the main cloud computing paradigms that makes it possible to run event-driven applications on unreserved resources with fine resource allocation granularity is serverless computing [10]. Deploying serverless computing on the edge for IDS, and more generally for time-sensitive and critical applications, is cost-effective as it opens up optimization opportunities for service providers: dynamic scaling of resources following load peaks in interactive applications, as well as fine and measured allocation granularity for limited edge resources.

Challenges of serverless on the edge for time-sensitive and critical applications: To deploy time-sensitive applications composed of short-lived functions in heterogeneous serverless edge computing, three challenges should be addressed: (1) reduce initialization delays, (2) avoid high communication delays, and (3) leverage heterogeneous resources to satisfy variable QoS. **Initialization delays.** IDS functions are short-lived, and serverless computing relying on unreserved resources implies a higher rate of function initializations, each requiring pulling the function image from a dedicated image storage node for deployment on the edge nodes [11]. Edge devices expose low-capacity, low-performance storage devices behind network links limited in reliability and speed, hence this issue needs to be considered closely to satisfy users' QoS. **Communication delays.** In a

This work was supported by the Institute of Research and Technology b<>com, dedicated to digital technologies, funded by the French government through the ANR Investment referenced ANR-A0-AIRT-07.

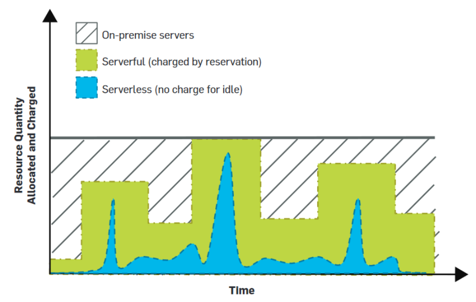
Context – Cloud Service Models



Context – Serverless Cloud Challenges

- Dynamic resources allocation: Rightsizing? Scaling from zero?
  - Instantiating a function = *cold start* delay
- Dynamic function scheduling: Mapping requests?
  - Per-request QoS requirements
  - Various levels of performance across heterogeneous hardware

💡 We proposed a cost-aware policy for private cloud serverless platforms that allowed reduced energy consumption while achieving SLA [6]



Serverless platforms dynamically (de)allocate hardware resources following load variations on applications [8]

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic allocation (following load variations)
    - Dynamic placement (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of performance
  - Various levels of cost
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of performance
  - Various levels of cost
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of performance
  - Various levels of cost
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of performance
  - Various levels of cost
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – Serverless Cloud Challenges

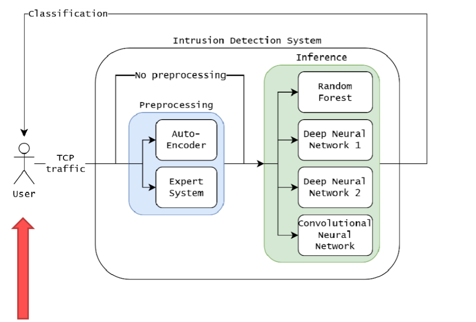
- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

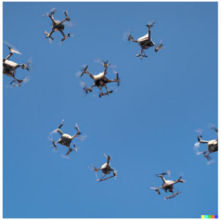
Context – Serverless Cloud Challenges

- Serverless resources are *not reserved* [8]
  - Increased provider's responsibility
    - Dynamic **allocation** (following load variations)
    - Dynamic **placement** (mapping requests to resources)
- Cloud resources are *heterogeneous* [5]
  - Various levels of **performance**
  - Various levels of **cost**
- Load is *unpredictable* [9]
  - Stochastic barrier
  - Need for an online solution
- Users have various *QoS requirements* [4]
  - Some use cases are throughput-centric (batch jobs)
  - Others need lower latency (interactive jobs)

Context – IDS Application

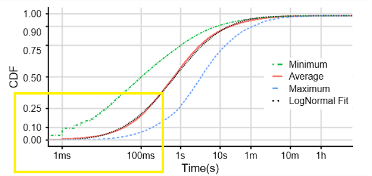
- Use case: Intrusion Detection Systems
  - Intermittent use of resources
    - IDS is only useful during drone missions
  - IDS relies on Machine Learning algorithms
    - Random Forests, Neural Networks
    - Leverage hardware accelerators
- Challenges:
  - Scheduling functions chains
  - Heavyweight function images (CUDA...)
  - Very short execution times (hundredths of milliseconds)
  - Intermediate data communication and storage



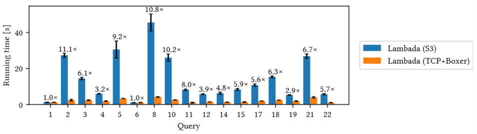


?

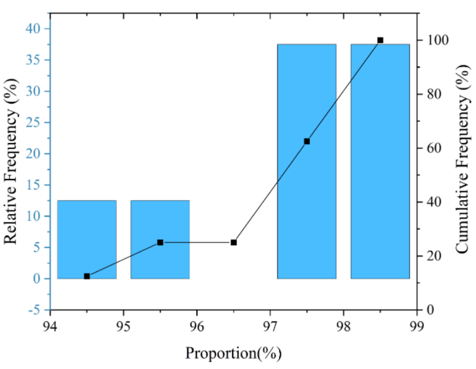
Context – Problem Justification



25% of functions at Microsoft Azure Functions are executed in 100 ms or less [9]



Remote storage communications induce critical slowdowns [11]



Pulling function images accounts for more than 80% of total response time [12]

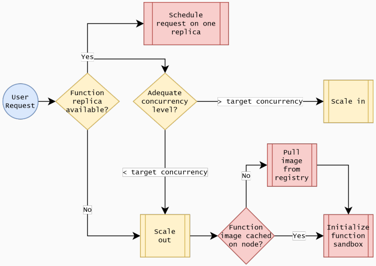
Contribution – Problem Statement

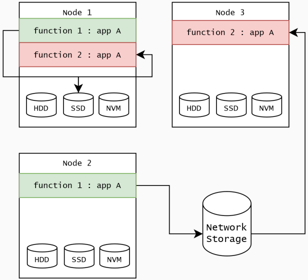
How to account for **initialization and communication delays** when deploying **chains of short-lived serverless functions** on **edge cloud**, leveraging **heterogeneous hardware** to optimize time-sensitive applications that require **variable QoS**, while limiting the number of edge nodes used?

Table 1: Breakdown of storage impacts on cost

	Impact	Cost
Resources allocation	Function response time	I/O bandwidth (Gbps)
	Resource contention	I/O capacity (GB)
Function scheduling	SLA penalties	I/O latency (ms)
	Tasks consolidation	I/O capacity (GB)
Application execution	Inter-function communications	I/O latency (ms)
	Output data storage	I/O capacity (MB)

Contribution – Function Cache and Function Communications



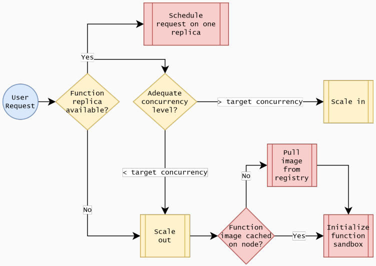


Policy to manage node function images cache and minimize cold start delays

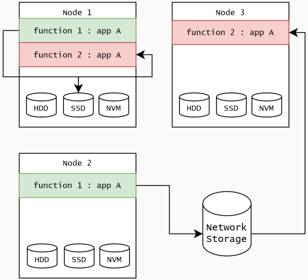
Policy to consolidate functions and maximize node-local communications

Policy to prevent contention on node storage between function cache and function communications

Contribution – Function Cache and Function Communications



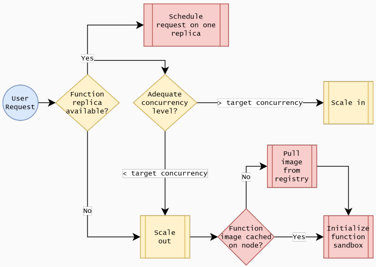
Policy to manage node function images cache and minimize cold start delays



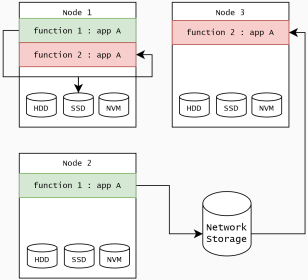
Policy to consolidate functions and maximize node-local communications

Policy to prevent contention on node storage between function cache and function communications

Contribution – Function Cache and Function Communications



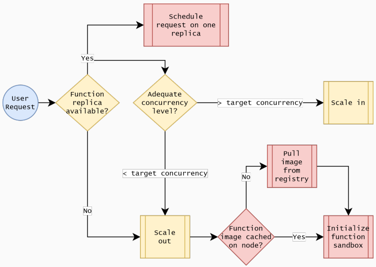
Policy to manage node function images cache and minimize cold start delays



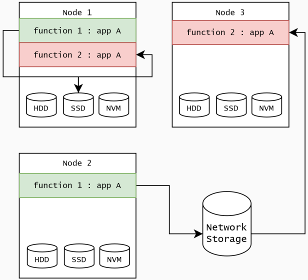
Policy to consolidate functions and maximize node-local communications

Policy to prevent contention on node storage between function cache and function communications

Contribution – Function Cache and Function Communications



Policy to manage node function images cache and minimize cold start delays



Policy to consolidate functions and maximize node-local communications

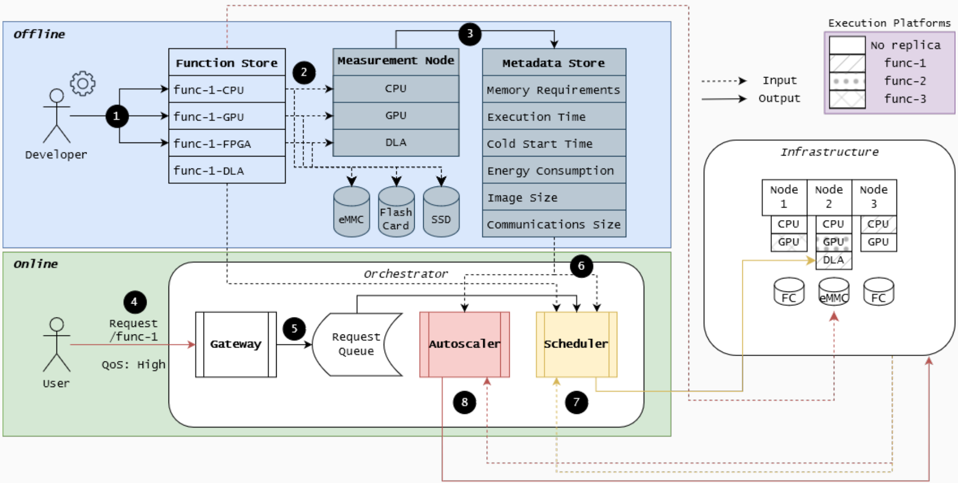
Policy to prevent contention on node storage between function cache and function communications

Contribution – State of the Art

Table 2: State-of-the-Art work on data-aware autoscaling platforms

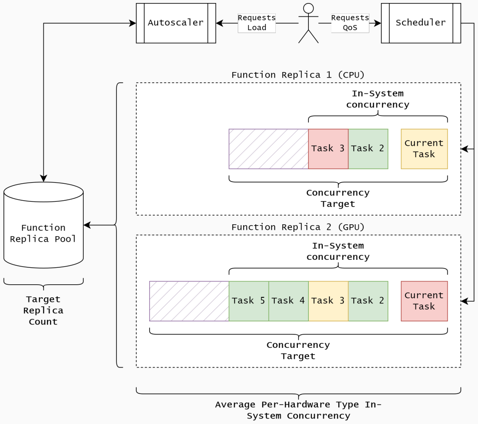
	Function chains	QoS-aware	Hardware heterogeneity	Programming constraint	Energy consumption	Function cache	Function communications
Cypress [2]	✓	✓	✗	✓	✓	✗	✓
FaDO [10]	✗	✗	✗	✓	✗	✗	✓
FaasFlow [7]	✓	✗	✗	✗	✗	✗	✗
FIRST [13]	✗	✗	✗	✓	✓	✗	✗
HeROfake [6]	✗	✓	✓	✓	✓	✗	✗
Netherite [3]	✓	✗	✗	✓	✗	✗	✓
Palette [1]	✓	✗	✗	✗	✗	✓	✓
Target solution	✓	✓	✓	✓	✓	✓	✓

Contribution – Overall System

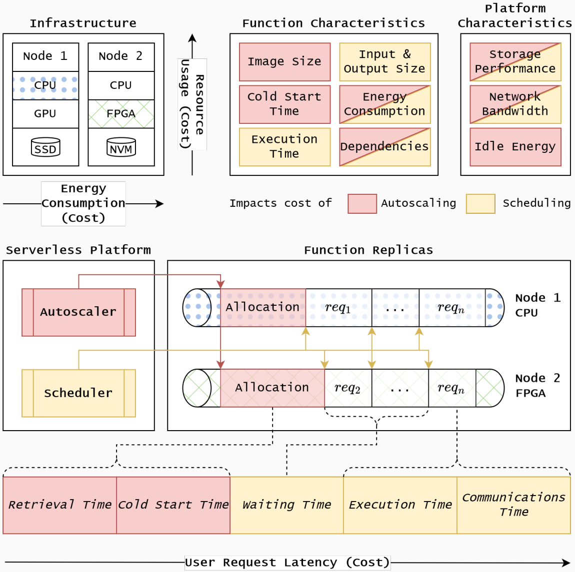


Contribution – Overview

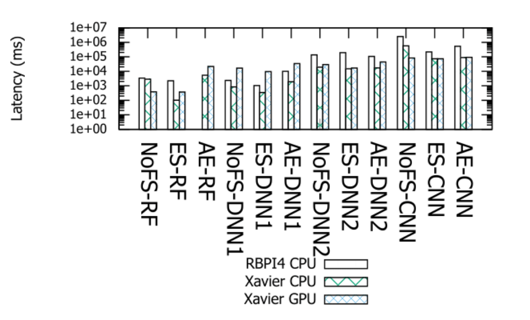
- Cost model
  - **Resources allocation**: how to rightsize the pool of function replicas?
  - **Tasks placement**: how to map user requests with different QoS levels to heterogeneous replicas?
- Orchestration policy
  - Minimize orchestration cost
  - Leveraging hardware heterogeneity and data locality
- Simulation environment
  - Observing a ”live” system to understand the moving parts
  - Evaluating and comparing different policies on QoS metrics



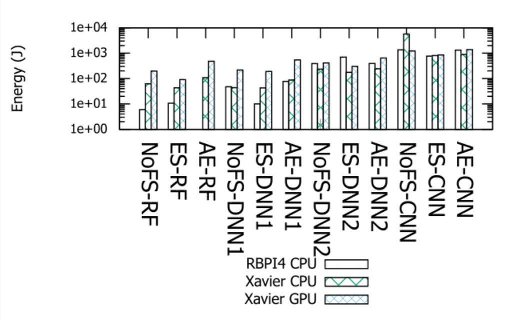
Contribution – Cost Model



Contribution – Characterization on Heterogeneous Hardware



Latency characterization of IDS models



Energy consumption characterization of IDS mdoels

Contribution – Cost Minimization Strategy

Autoscaling

→ increased **consolidation**
→ reduced **makespan**
→ reduced **energy consumption**
→ reduced **cost of ownership**

$$\begin{aligned}\forall N, \forall P \in N, scaleCost_a^{f_{i_{N,P}}} = \\ k_{CP} \cdot CP_{a_N} \\ +k_{TT} \cdot TT_{f_{N,P}} \\ +k_{EC} \cdot EC_{f_{N,P}} \\ +k_{HP} \cdot HP_{f_{N,P}}\end{aligned} \tag{1}$$

Scheduling

→ avoid **missed deadlines**
→ use **less power**
→ enforce **high resource usage**

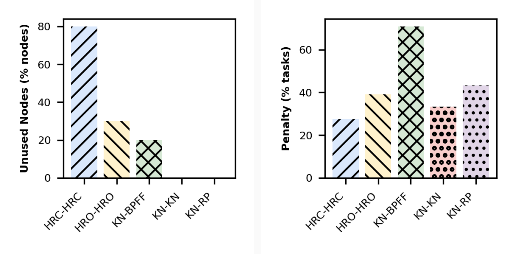
$$\begin{aligned}\forall (N, P) \in R_f, schedCost_{f_{i_{N,P}}} = \\ k_{QP} \cdot QP_{f_{N,P}} \\ +k_{EC} \cdot EC_{f_{N,P}} \\ +k_{TC} \cdot TC_{f_{N,P}}\end{aligned} \tag{2}$$

Evaluation – Simulation Environment

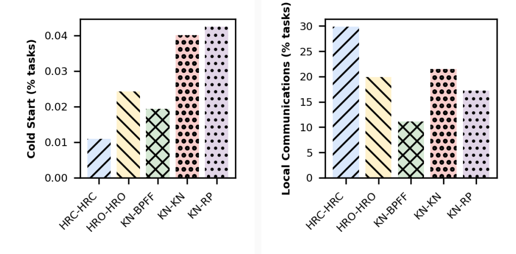
- HeROsim
  - In-house open source simulation tool
  - https://github.com/b-com/HeROsim
- Artifacts evaluated: ORO, ROR, ROR-R
  - Thank you, reviewers!
- Baseline policies:
  - Knative (KN) – Least Connected load balancing
  - Amazon Lambda (BPFF) – Bin-Packing First Fit consolidation
  - HeROfake (HRO) – Storage-oblivious, heterogeneity-aware policy
  - Random Placement (RP) – what could go wrong?

- Synthetic workload
  - Poisson process, $\lambda = 83$
  - Duration: 30 minutes
  - Uniform distribution of QoS levels and application requests
- 10 nodes in the infrastructure
  - 8 Raspberry Pi 4B
  - 1 Nvidia Xavier Jetson
  - 1 Xilinx Pynq Z2
- 100 Mbps network link between nodes

Evaluation - Against Baselines

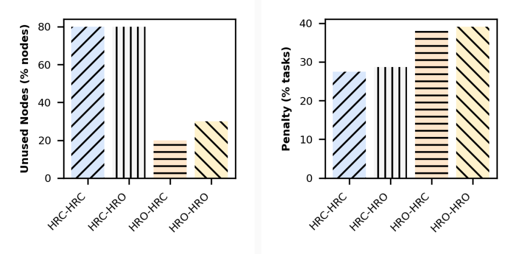


Consolidation across nodes and penalty proportions

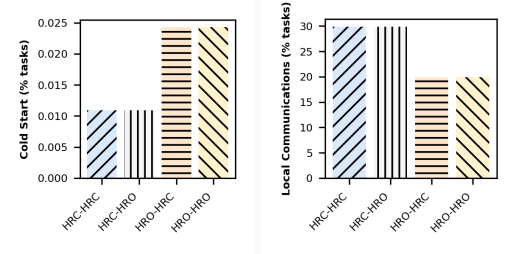


Cold start proportions and local communications

Evaluation - Individual Components



Consolidation across nodes and penalty proportions



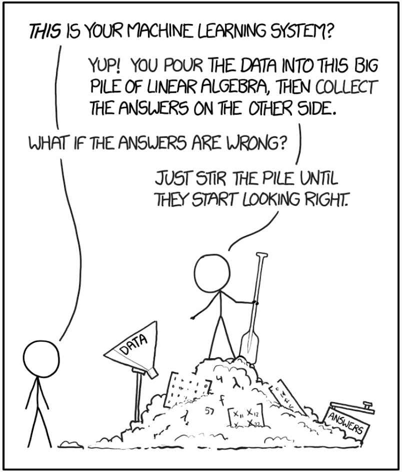
Cold start proportions and local communications

Conclusion

- HeROcache enforces applications consolidation:
  - reduces average initialization delays by 17.6%
  - cuts communication delays by 88.4%
- HeROcache enhances Quality of Service:
  - potential **reduction of static energy consumption by 80%**
  - maintains under 28% of QoS violations

Perspectives

- Limits of HeROcache:
  - Greedy algorithms!
  - Will not scale to large infrastructures...
- Machine Learning?
  - Duality between **prediction** and **reaction**
    - Proactive allocation (time series prediction)
    - Reactive scheduling (Q-Learning agent)



https://xkcd.com/1838/

Thank you!

🙏

Questions?

📧 vincent.lannurien@ensta-bretagne.org

📦 https://github.com/b-com/HeROsim

References i

M. Abdi, S. Ginzburg, X. C. Lin, J. Faleiro, G. I. Chaudhry, I. Goiri, R. Bianchini, D. S. Berger, and R. Fonseca.
Palette Load Balancing: Locality Hints for Serverless Functions.
In *EuroSys '23*, pages 365–380, Rome Italy, May 2023. ACM.

V. M. Bhasi, J. R. Gunasekaran, A. Sharma, M. T. Kandemir, and C. Das.
Cypress: Input Size-Sensitive Container Provisioning and Request Scheduling for Serverless Platforms.
In *SoCC '22*, pages 257–272, San Francisco California, Nov. 2022. ACM.

S. Burckhardt, B. Chandramouli, C. Gillum, D. Justo, K. Kallas, C. McMahon, C. S. Meiklejohn, and X. Zhu.
Netherite: Efficient Execution of Serverless Workflows.
Proc. VLDB Endow., 15(8):1591–1604, apr 2022.

References ii

R. Buyya, S. K. Garg, R. N. Calheiros, and B. Bla.
SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions.
In *CSC '11*. IEEE, 2011.

E. Horta, H.-R. Chuang, N. R. VSathish, C. Philippidis, A. Barbalace, P. Olivier, and B. Ravindran.
Xar-Trek: Run-Time Execution Migration among FPGAs and Heterogeneous-ISA CPUs.
In *Middleware '22*. ACM, 2021.

V. Lannurien, L. D'Orazio, O. Barais, E. Bernard, O. Weppe, L. Beaulieu, A. Kacete, S. Paquelet, and J. Boukhobza.
HeROfake: Heterogeneous Resources Orchestration in a Serverless Cloud – An Application to Deepfake Detection.
2023.

Z. Li, Y. Liu, L. Guo, Q. Chen, J. Cheng, W. Zheng, and M. Guo.
FaaSFlow: Enable Efficient Workflow Execution for Function-as-a-Service.
In *ASPLOS '22*, page 782–796, New York, NY, USA, 2022. Association for Computing Machinery.

J. Schleier-Smith, V. Sreekanti, A. Khandelwal, J. Carreira, N. J. Yadwadkar, R. A. Popa, J. E. Gonzalez, I. Stoica, and D. A. Patterson.
What Serverless Computing is and Should Become: The next Phase of Cloud Computing.
Commun. ACM, 2021.

- M. Shahrad, R. Fonseca, Í. Goiri, G. Chaudhry, P. Batum, J. Cooke, E. Laureano, C. Tresness, M. Russinovich, and R. Bianchini.
  Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider.
  USENIX ATC'20, 2020.
- C. P. Smith, A. Jindal, M. Chadha, M. Gerndt, and S. Benedict.
  FaDO: FaaS Functions and Data Orchestrator for Multiple Serverless Edge-Cloud Clusters.
  In *ICFEC 2022*, pages 17–25, Messina, Italy, May 2022. IEEE.
- M. Wawrzoniak, I. Müller, R. Fraga Barcelos Paulus Bruno, and G. Alonso.
  Boxer: Data Analytics on Network-enabled Serverless Platforms.
  2021.

B. Yan, H. Gao, H. Wu, W. Zhang, L. Hua, and T. Huang.
Hermes: Efficient Cache Management for Container-based Serverless Computing.
In *12th Asia-Pacific Symposium on Internetware*, Singapore, 2020. ACM.

L. Zhang, C. Li, X. Wang, W. Feng, Z. Yu, Q. Chen, J. Leng, M. Guo, P. Yang, and S. Yue.
FIRST: Exploiting the Multi-Dimensional Attributes of Functions for Power-Aware Serverless Computing.
In *IPDPS 2023*, pages 864–874, St. Petersburg, FL, USA, May 2023. IEEE.