Towards Elastic Memory Allocation of Serverless Functions in Disaggregated Memory Systems

4th Workshop on Heterogeneous Composable and Disaggregated Systems

Co-located with ASPLOS/EUROSYS 2025
Rotterdam, Netherlands
March 30, 2025

Achilleas Tzenetopoulos
ECE NTUA Ph.D. Student

Dimosthenis Masouros
ECE NTUA Ph.D.

Dimitrios Soudris
ECE NTUA Professor

Sotirios Xydis
ECE NTUA Ass. Professor

HCDS 2025
ASPLOS 2025
H.F.R.I. Hellenic Foundation for Research & Innovation

Overview

1. Introduction
   1.1. Serverless Computing
   1.2. Memory Disaggregation
2. Elastic Memory for Serverless Functions
   2.1 Impact on latency
   2.2 Memory footprint Pitfall
3. Preliminary Results
4. Conclusion & Future Work

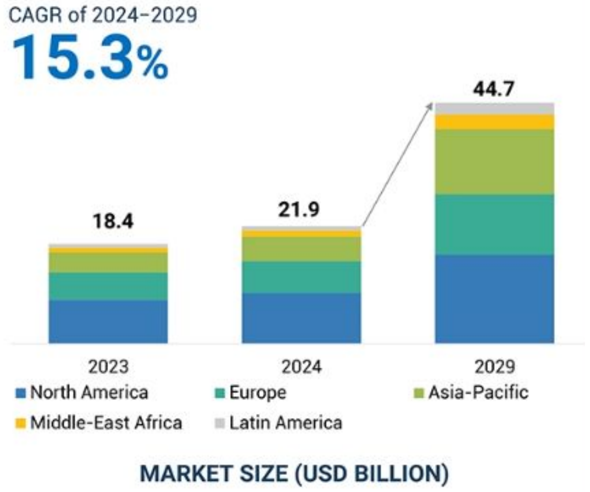
Introduction - Serverless Computing

An emerging Cloud Computing paradigm

A **growing** market: 44.7 USD Billion by 2029

A step closer to the promises of Cloud:

- **Fully-managed** by Cloud providers (AWS, Google, MS Azure).
- **Pay-per-use** (ms-scale)
- **Elasticity** (Demand-driven scale-out)



Azure Functions

AWS Lambda

Serverless Computing Report, Markets and Markets Report
https://www.marketsandmarkets.com/Market-Reports/serverless-computing-market-217021547.html

Introduction - Serverless Computing

When a request arrives:

1. **Initialization:** Execution environment is initialized (If no function instance is available).

2. **Function execution:** Functions usually receive/send their inputs/outputs from/to remote storage. Memory footprint temporarily increases..

3. **Keep-Alive Phase:** Function will remain idle (to avoid cold starts)

4. **Graceful Termination**

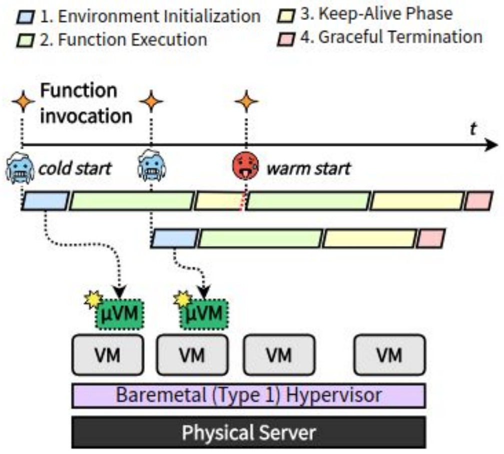


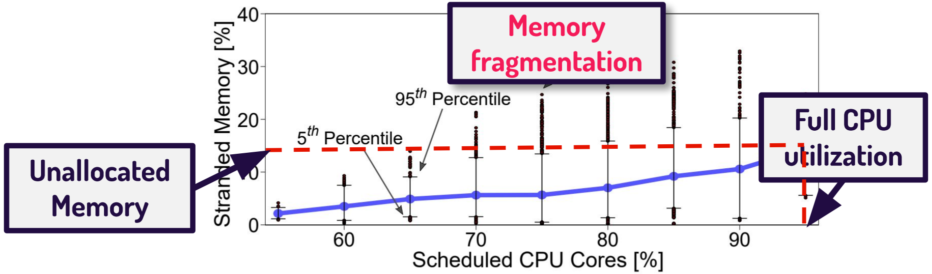
Figure 1: Overview of a serverless function's lifetime.

Introduction - Memory Disaggregation

- Traditional cluster setups lead to **resource fragmentation** at scale, where CPU and memory resources are **underutilized**.

- **Memory disaggregation** addresses this by **decoupling memory from compute nodes**, treating it as an **elastic, shared pool** that can be dynamically allocated and rebalanced across clusters.

Memory Stranding [1]



[1] Li, Huaicheng, et al. "Pond: Cxl-based memory pooling systems for cloud platforms." Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2023.

Elastic Memory for Serverless Functions

HCDS 2025

Related Work on Elastic Memory

FaaSMem [1]: Focus on initialization phase allocations

Memory Harvesting VMs [2]: Right-sizes the memory allocated by VMs

Adrias [3], Pond [4]: Select local or remote memory binding to improve resource efficiency

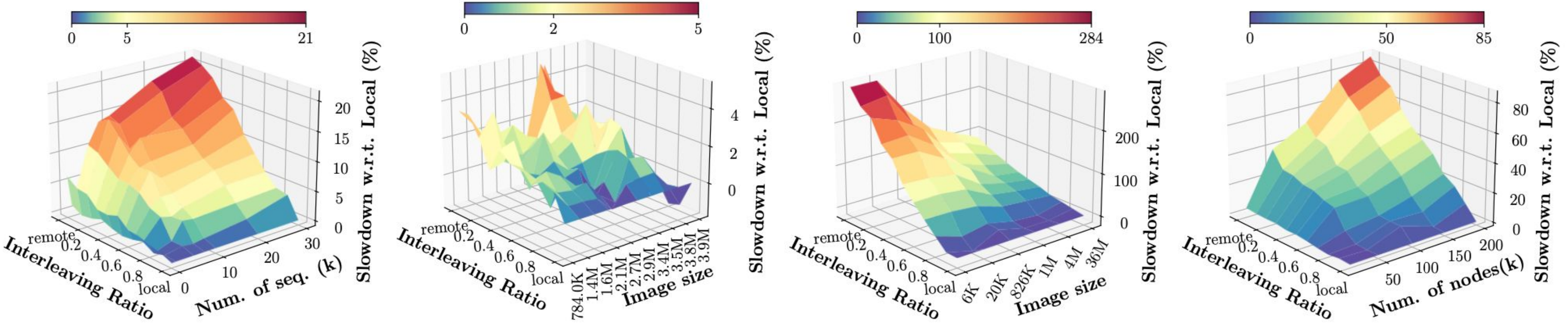
The goal of this work is to **explore**, and **leverage** an **expanded, per-function memory configuration space** to **increase** resource **efficiency** of **serverless** deployments in datacenters.

Specifically:
- **Allocate** different **portions** of **memory** locally.
- Consider the **entire** function **lifecycle**
- Use **footprint-latency Pareto** optimal solutions for **request-level** optimizations.

[1] Xu, Chuhao, et al. "Faasmem: Improving memory efficiency of serverless computing with memory pool architecture." Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. 2024.
[2] Fuerst, Alexander, et al. "Memory-harvesting vms in cloud platforms." Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. 2022.
[3] Masouros, Dimosthenis, et al. "Adrias: Interference-aware memory orchestration for disaggregated cloud infrastructures." 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2023.
[4] Li, Huaicheng, et al. "Pond: Cxl-based memory pooling systems for cloud platforms." Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2023.

Elastic Memory for Serverless Functions

- Measure **impact** of remote memory allocations on the **performance** of 4 serverless functions from SeBs [1]
- Our system uses Intel Optane Persistent Memory, configured as a zero-CPU NUMA node.
- Varying **input size** and **local/remote memory allocation ratio** between 0 (fully remote) and 1 (fully local).



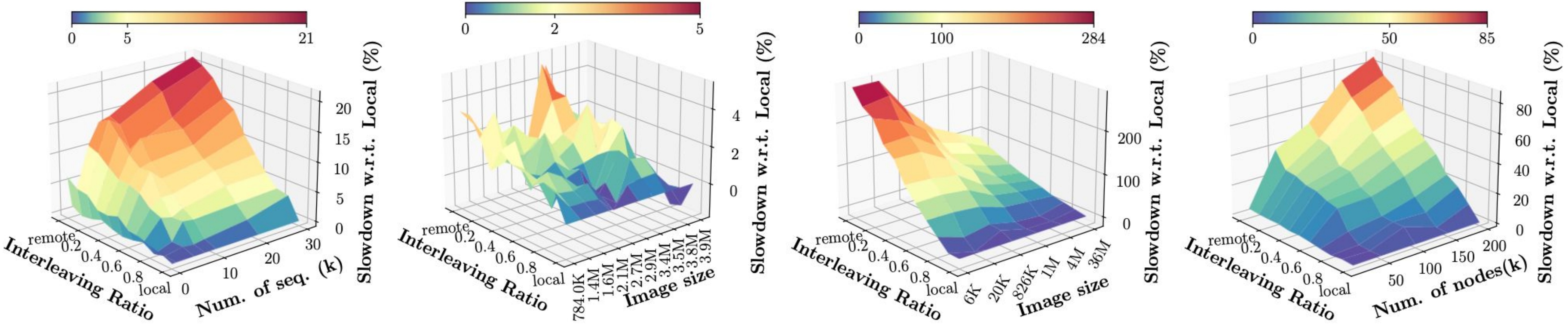
(a) dna visualization (b) thumbnailer (c) image recognition (d) graph pagerank

[1] Copik, Marcin, et al. "Sebs: A serverless benchmark suite for function-as-a-service computing." Proceedings of the 22nd International Middleware Conference. 2021.

HCDS 2025

Impact on latency varies per benchmark

6%-21% increase | noisy/unaffected | up to 284% | 4%-85% increase



(a) dna visualization

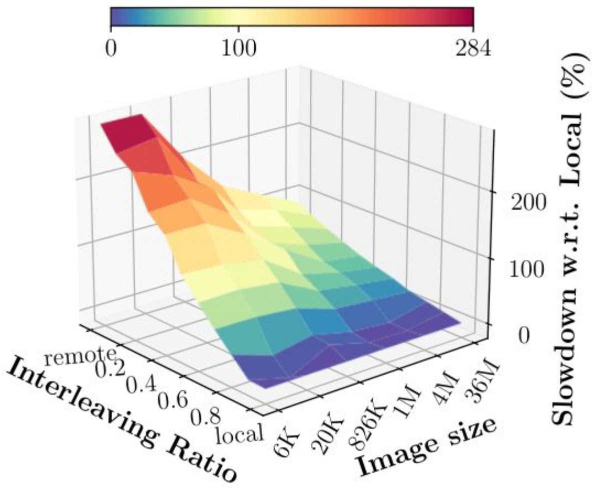
(b) thumbnailer

(c) image recognition

(d) graph pagerank

[1] Copik, Marcin, et al. "Sebs: A serverless benchmark suite for function-as-a-service computing." Proceedings of the 22nd International Middleware Conference. 2021.

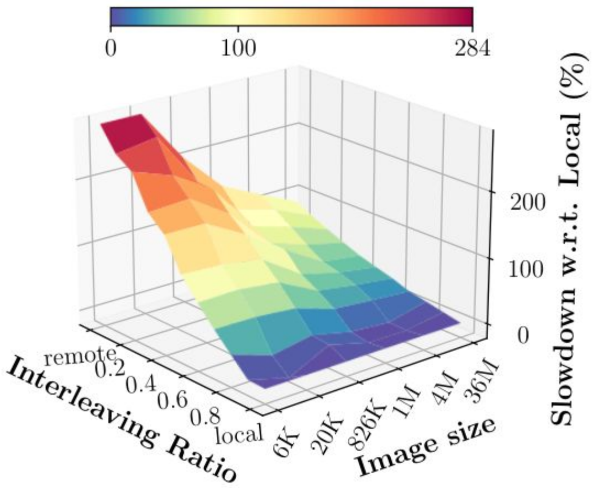
HCDS 2025



(c) image recognition

Impact on latency is **counterintuitive**.
For example in vision model inference serving (ResNet50):

- **Execution phase stages:** Image preprocessing -> Inference
- **Stages contribution to execution latency:** Larger images spent more time in preprocessing, while the inference time remains the same.
- **Sensitivity to remote memory:** Different sensitivity to remote memory across stages, e.g., 100% for preprocessing, 385% for inference.



(c) image recognition

Leveraging remote memory helps **reduce costs** by utilizing otherwise unused memory within the data center.

⇩

This introduces a **trade-off between latency and reduced local memory footprint**.

⇩

In serverless environments, **auto-scaling** maintains performance by spawning new instances when demand exceeds current capacity.

⇩

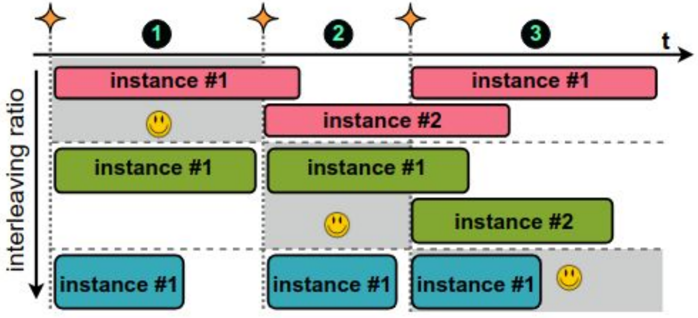
However, **this can increase the number of active instances**, potentially offsetting the benefits of local memory usage reduction.

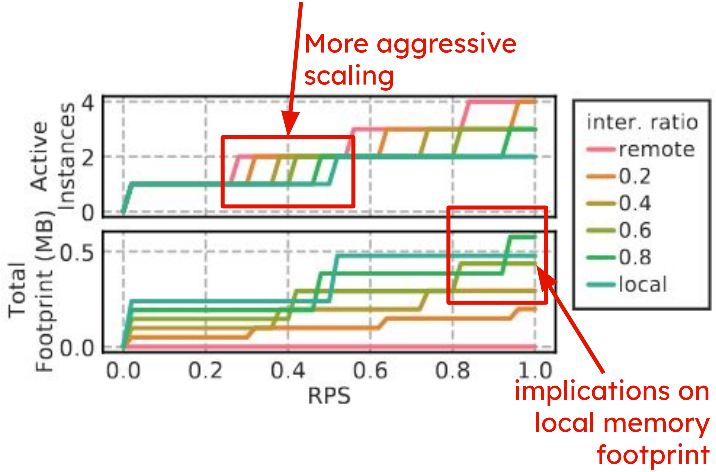
Leveraging remote memory helps **reduce costs** by utilizing otherwise unused memory within the data center.

This introduces a **trade-off between latency and reduced local memory footprint**.

In serverless environments, **auto-scaling** maintains performance by spawning new instances when demand exceeds current capacity.

However, **this can increase the number of active instances**, potentially offsetting the benefits of local memory usage reduction.

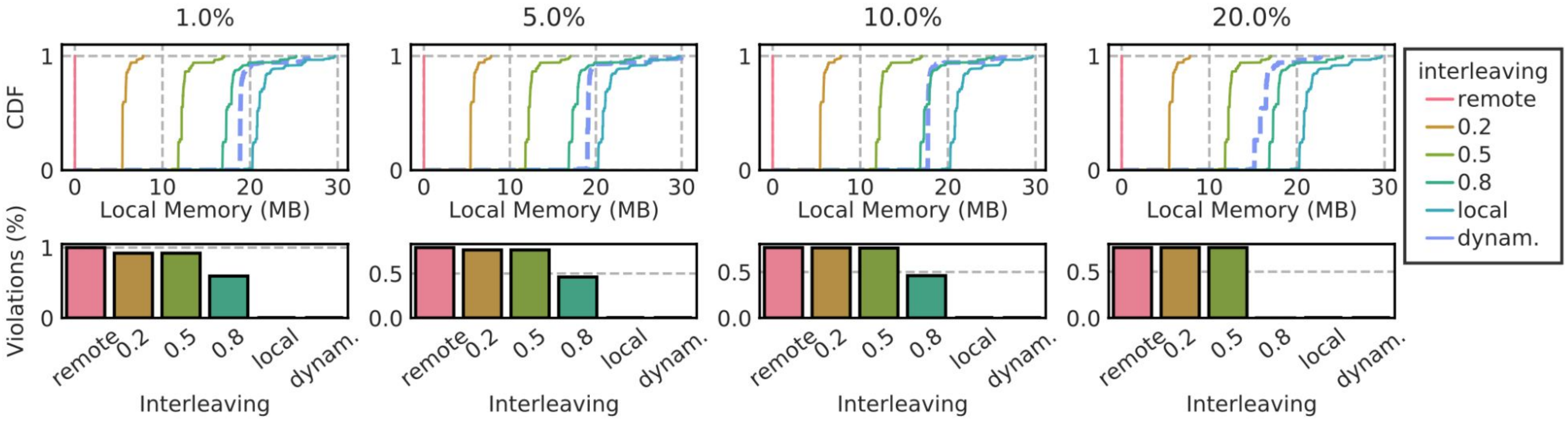




HCDS 2025

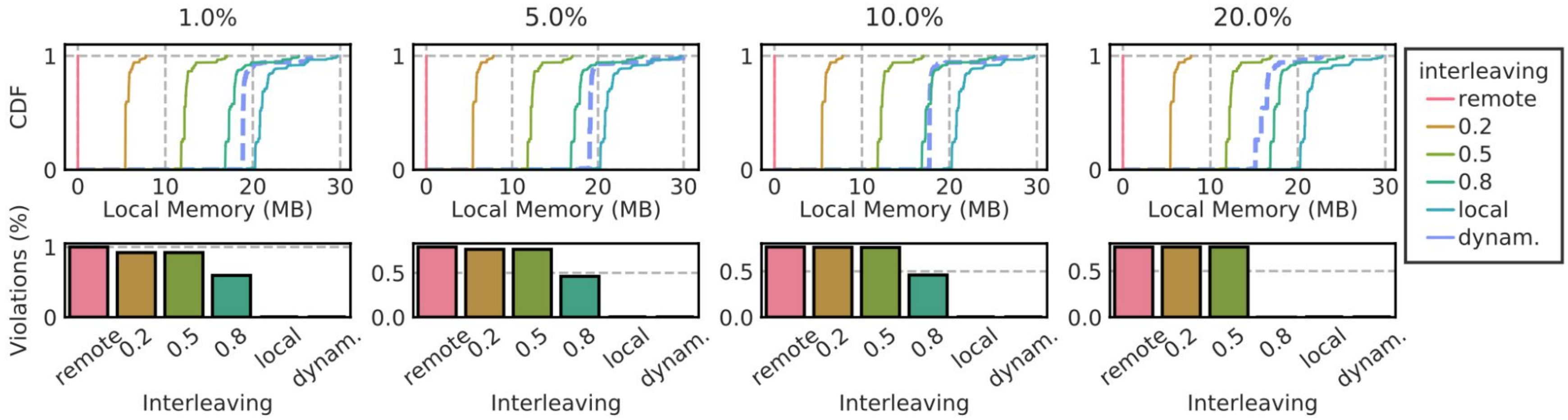
Evaluate the overall impact on **local memory footprint** when setting **different SLO targets.**

1. **Static interleaving policies**
2. **Adaptive:** Selects from the Pareto optimal set (minimum local memory footprint that satisfies the latency target)



Preliminary Results

- **Static interleaving** policies, **fail** to satisfy the **SLO** for most cases.
- **Adaptive policy** reduces local memory footprint **by 6-25% (median)**, without violations.



Conclusion

HCDS 2025

- Impact analysis and insights of leveraging remote memory for different serverless functions, and input sizes.
- **Used weighted interleaving** to allocate **"just-enough"** local memory for serverless functions.
- Preliminary results show that **local memory reductions** can be achieved.

Future Work

Limitations of weighted interleaving:

- Round-robin page allocation
- Frequently used pages may be allocated in the remote memory.

Further optimization strategies:

- Page Access Frequency-aware page placement/migration
- Predictive strategies, e.g., memory prefetching

Thank you for your attention