SPCL spcl.ethz.ch

@spcl @spcl_eth

ETH zürich

Wei Qiu, Marcin Copik, Yun Wang, Alexandru Calotoiu, Torsten Hoefler

User-guided Page Merging for Memory Deduplication in Serverless Systems

IEEE BIG DATA 2023

15–18 December • Sorrento, Italy

How does Function-as-a-Service (FaaS) work?

How does Function-as-a-Service (FaaS) work?

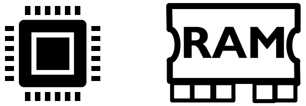
```
def handler_function(req: dict, context: dict):

   model = cloud_storage.download_model()

   input = parse_input(req['payload'])

   output = model.inference(input)

   return output
```

How does Function-as-a-Service (FaaS) work?

```
def handler_function(req: dict, context: dict):

    model = cloud_storage.download_model()

    input = parse_input(req['payload'])

    output = model.inference(input)

    return output
```

+

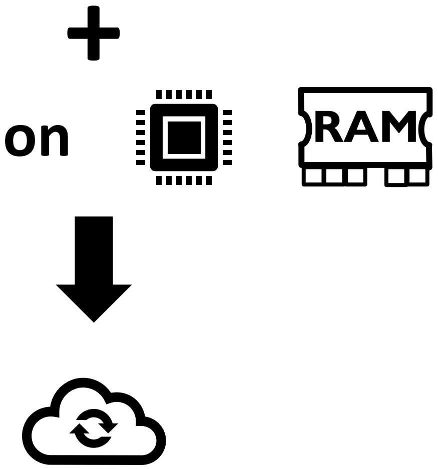
Configuration



How does Function-as-a-Service (FaaS) work?

```
def handler_function(req: dict, context: dict):

    model = cloud_storage.download_model()

    input = parse_input(req['payload'])

    output = model.inference(input)

    return output
```

Configuration



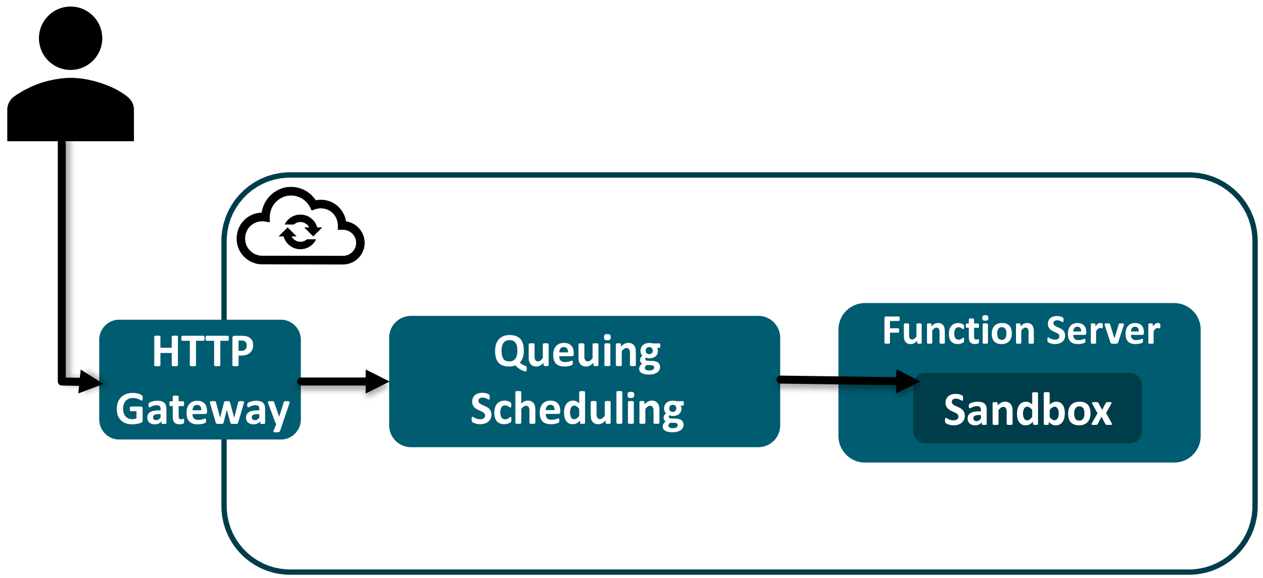
How does Function-as-a-Service (FaaS) work?

```
def handler_function(req: dict, context: dict):

    model = cloud_storage.download_model()

    input = parse_input(req['payload'])

    output = model.inference(input)

    return output
```

+

Configuration

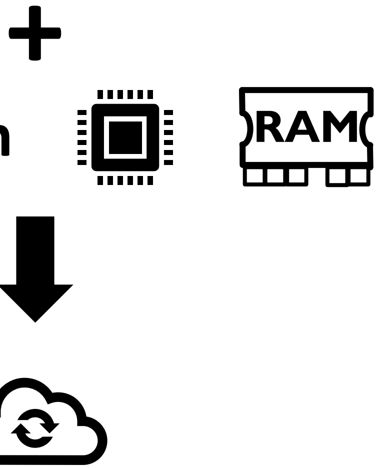
RAM

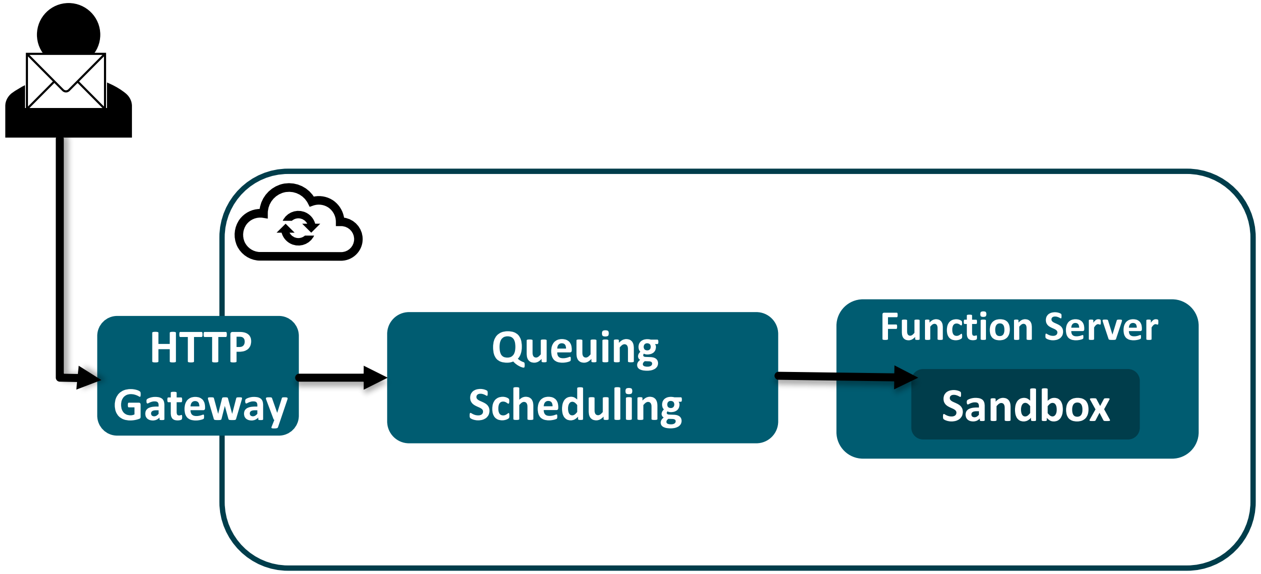


How does Function-as-a-Service (FaaS) work?

```python
def handler_function(req: dict, context: dict):

    model = cloud_storage.download_model()

    input = parse_input(req['payload'])

    output = model.inference(input)

    return output
```

Configuration

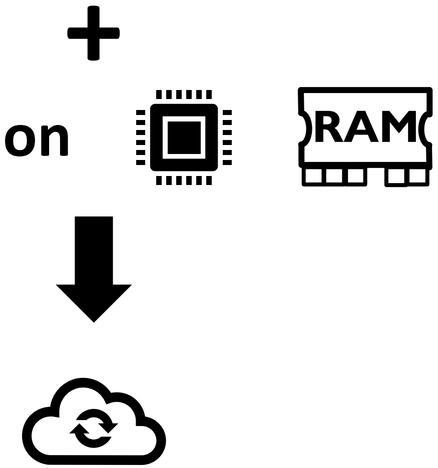


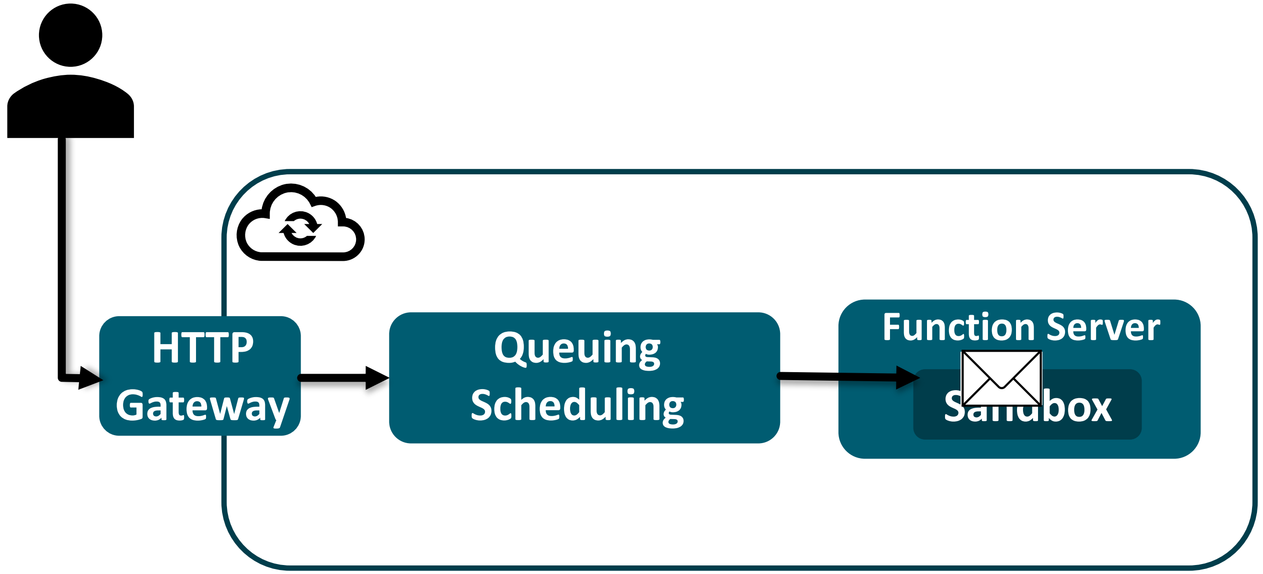


How does Function-as-a-Service (FaaS) work?

```python
def handler_function(req: dict, context: dict):

    model = cloud_storage.download_model()

    input = parse_input(req['payload'])

    output = model.inference(input)

    return output
```

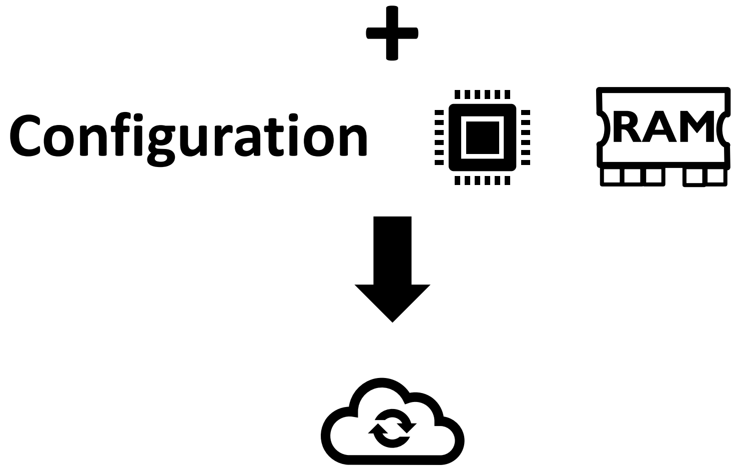
Configuration

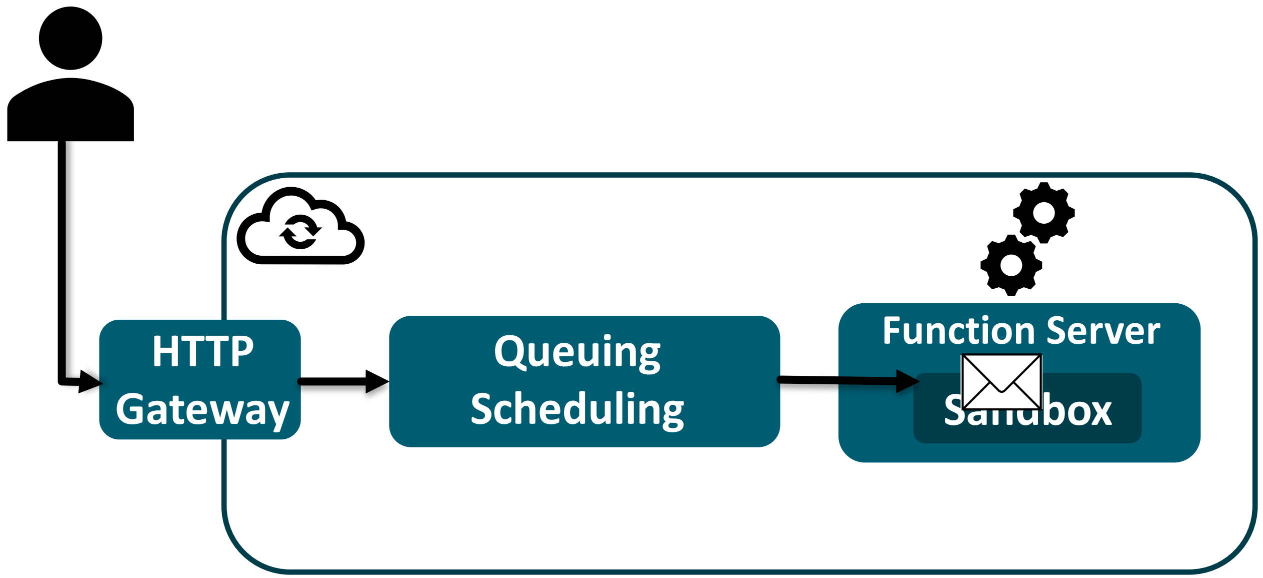




How does Function-as-a-Service (FaaS) work?

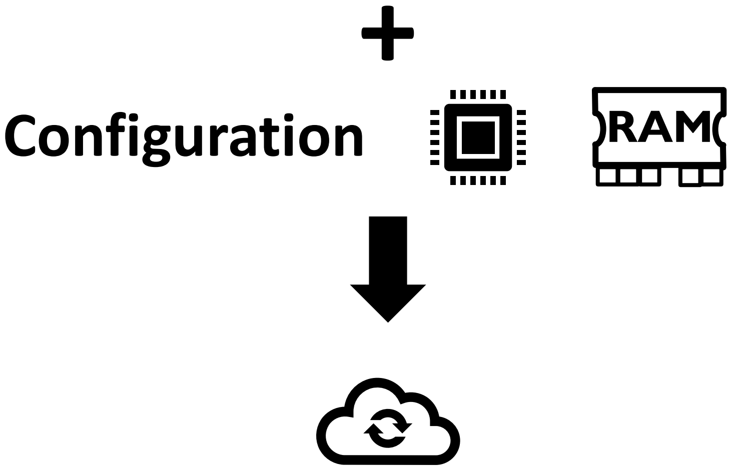
```python
def handler_function(req: dict, context: dict):

    model = cloud_storage.download_model()

    input = parse_input(req['payload'])

    output = model.inference(input)

    return output
```

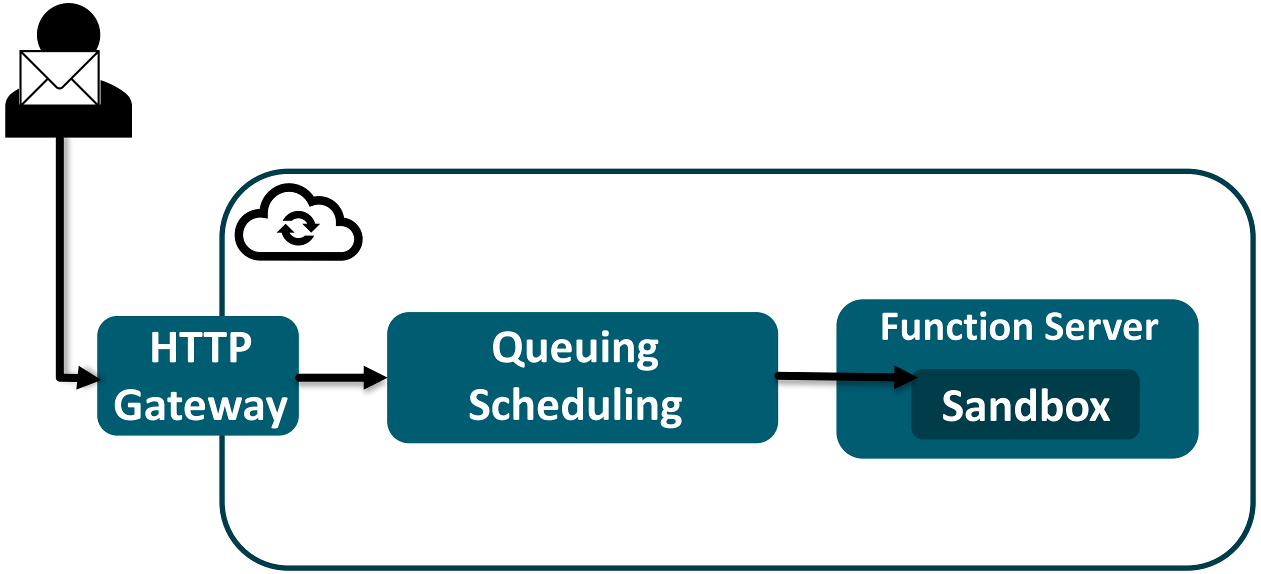




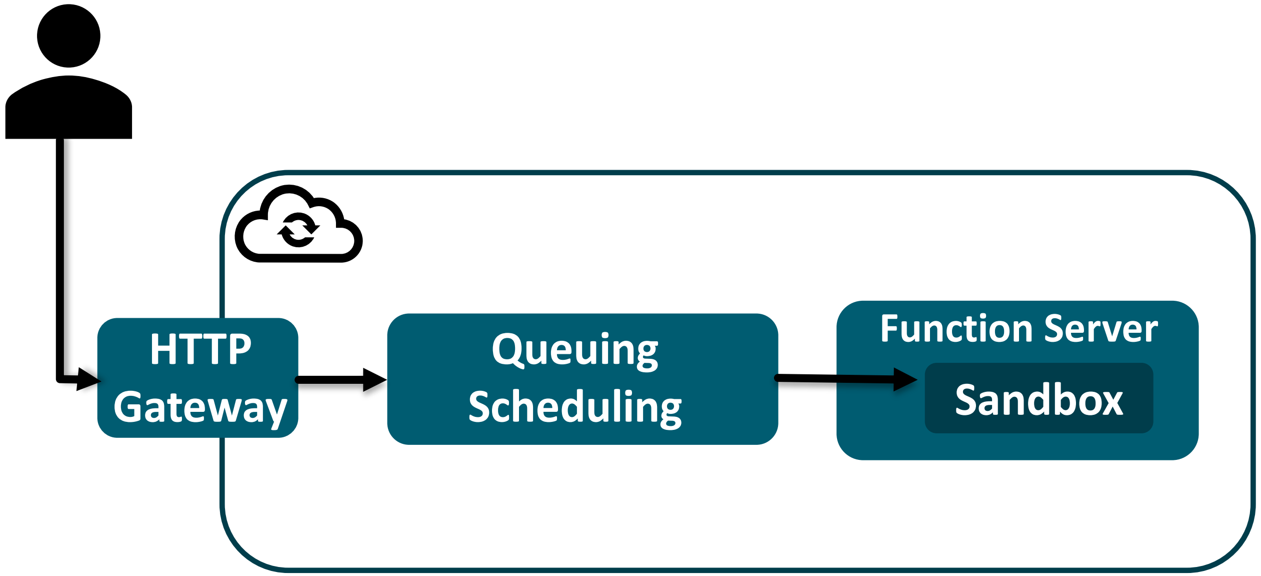
How does Function-as-a-Service (FaaS) work?

```python
def handler_function(req: dict, context: dict):
    model = cloud_storage.download_model()
    input = parse_input(req['payload'])
    output = model.inference(input)
    return output
```

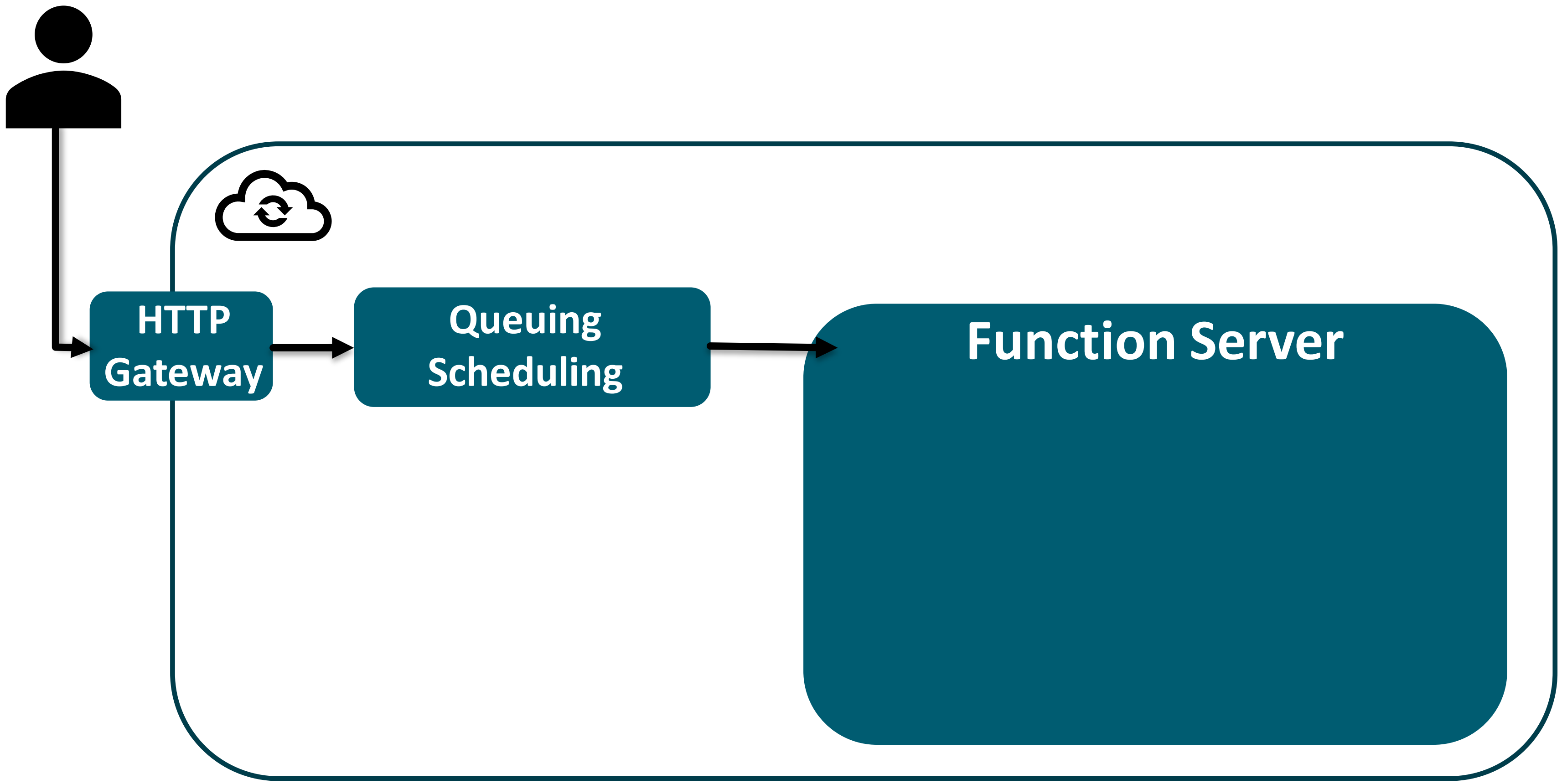




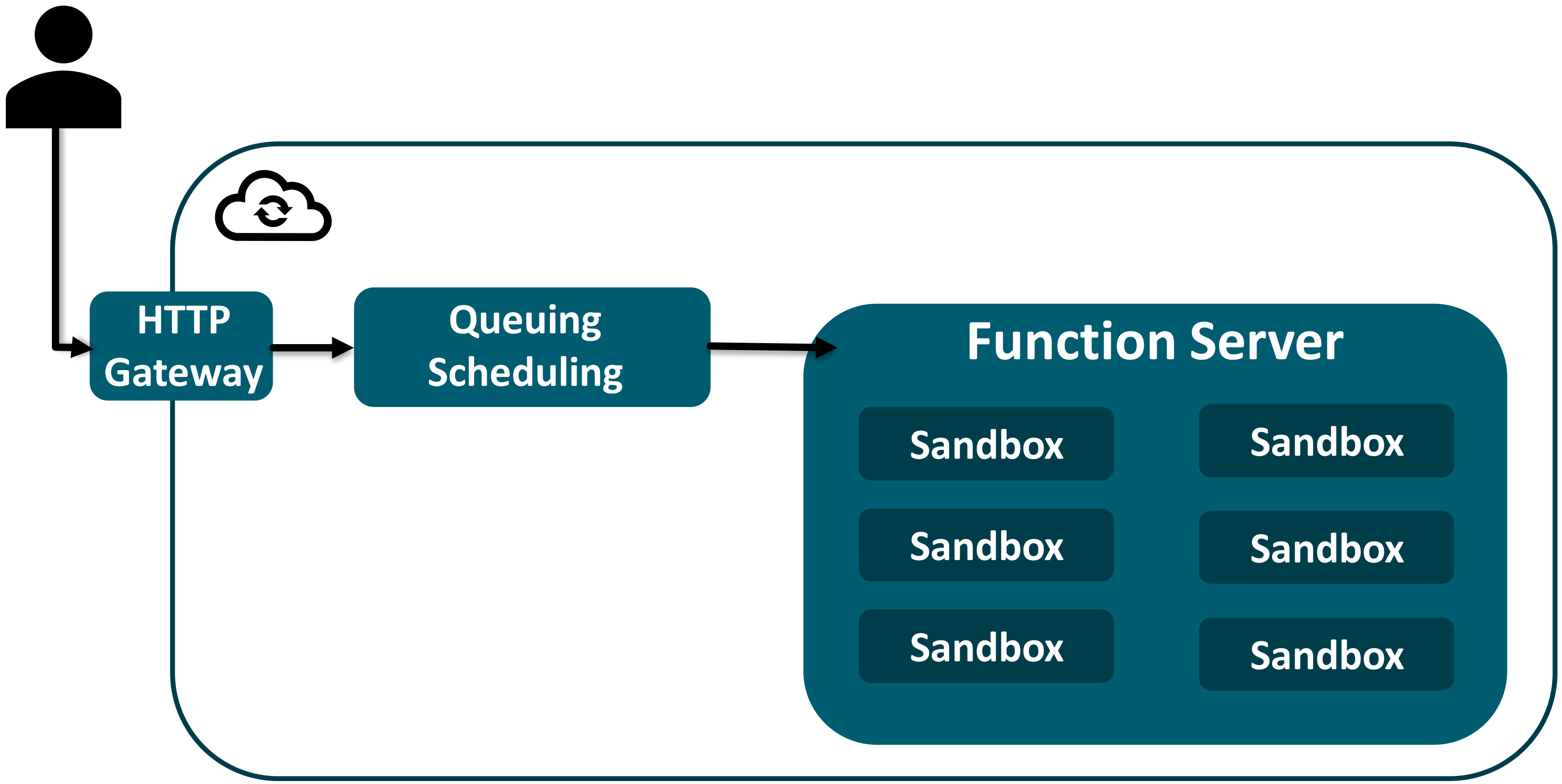
How does serverless systems really work?



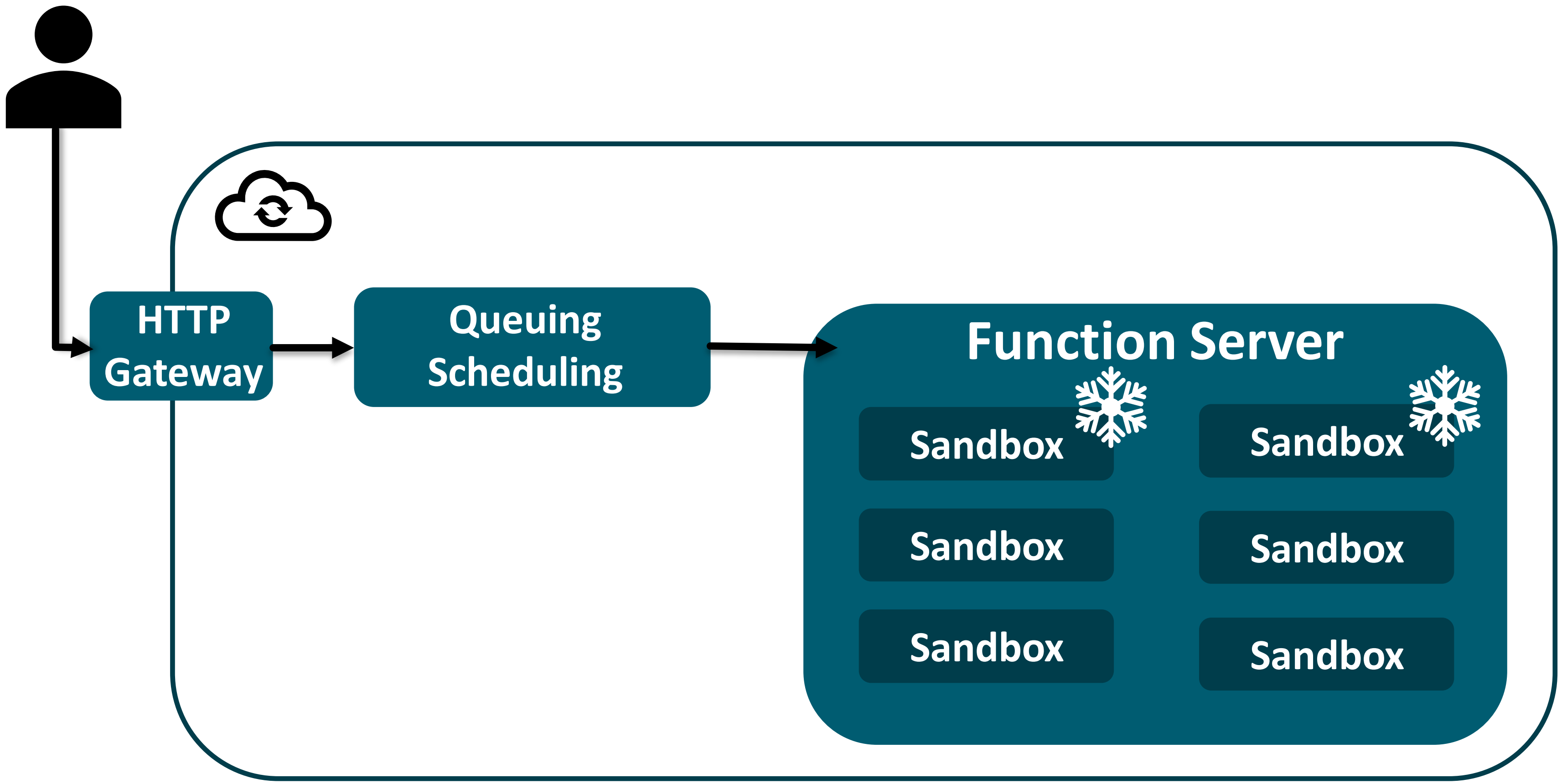
How does serverless systems really work?



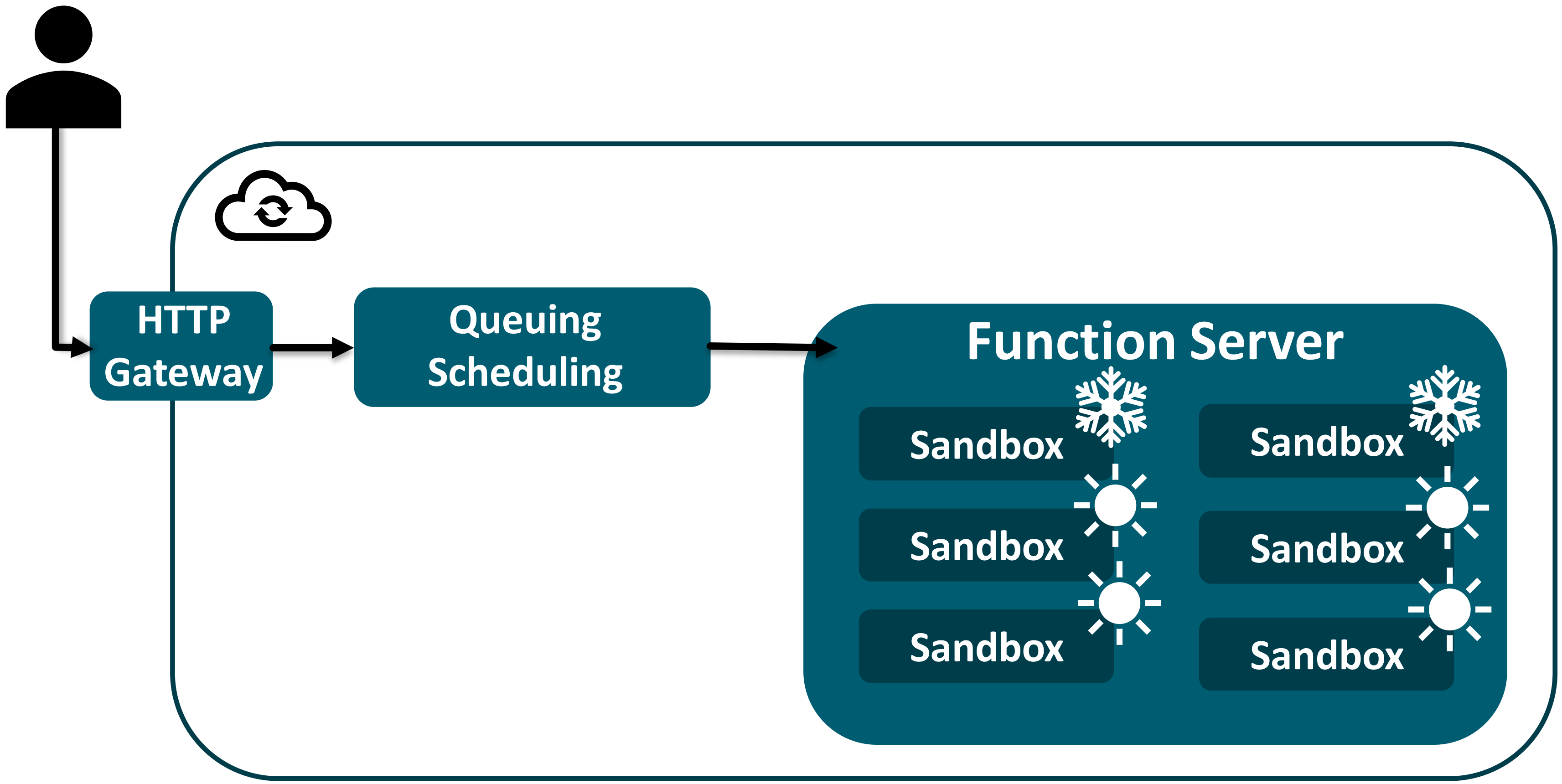
How does serverless systems really work?



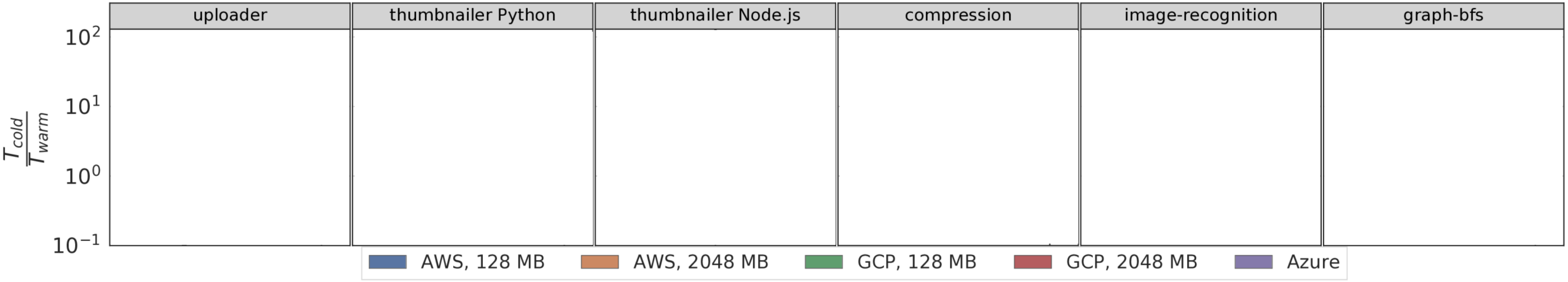
How does serverless systems really work?



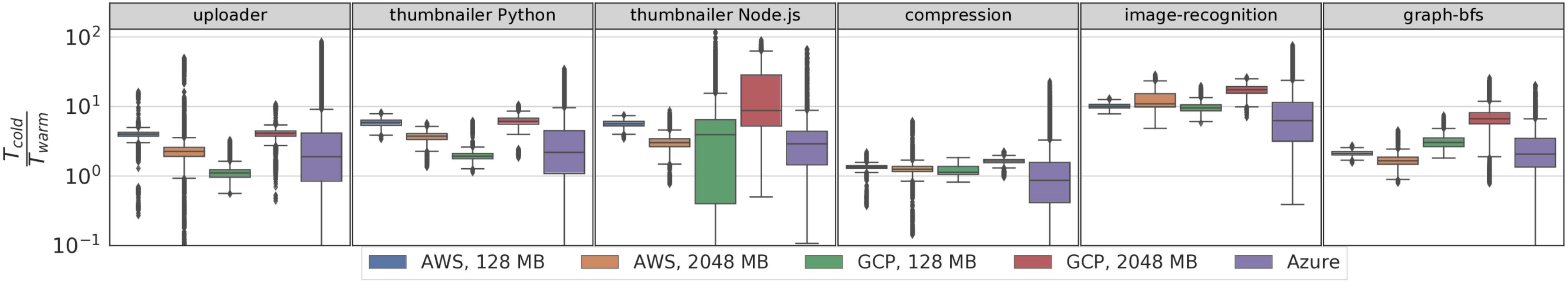
How does serverless systems really work?



Cold Startup Overheads

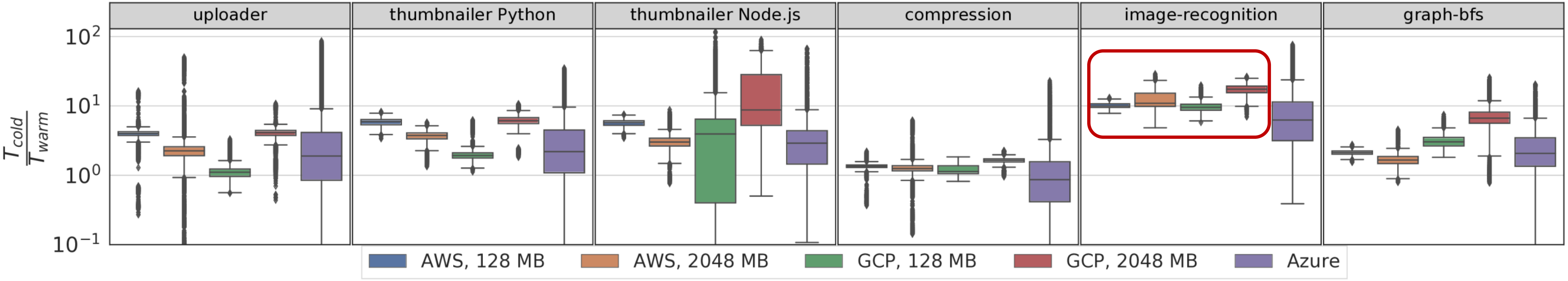


Cold Startup Overheads



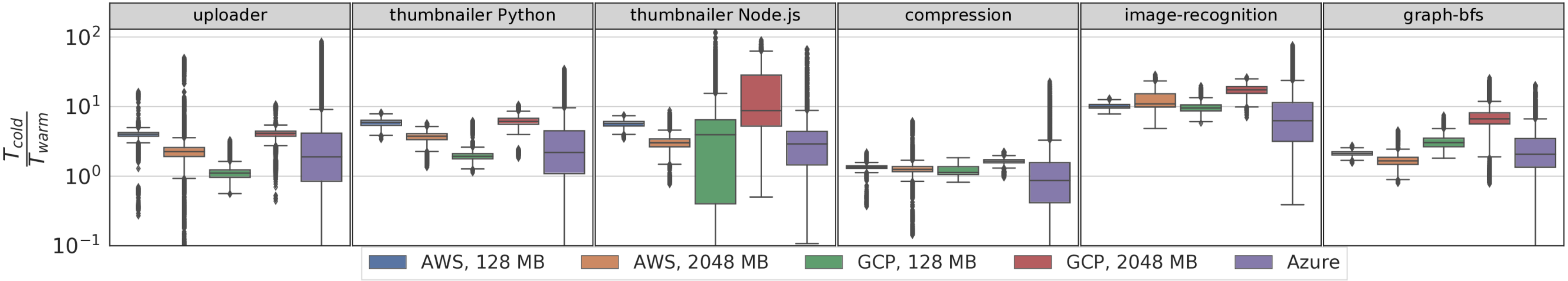
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", ACM/IFIP Middleware 2021

Cold Startup Overheads



"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", ACM/IFIP Middleware 2021

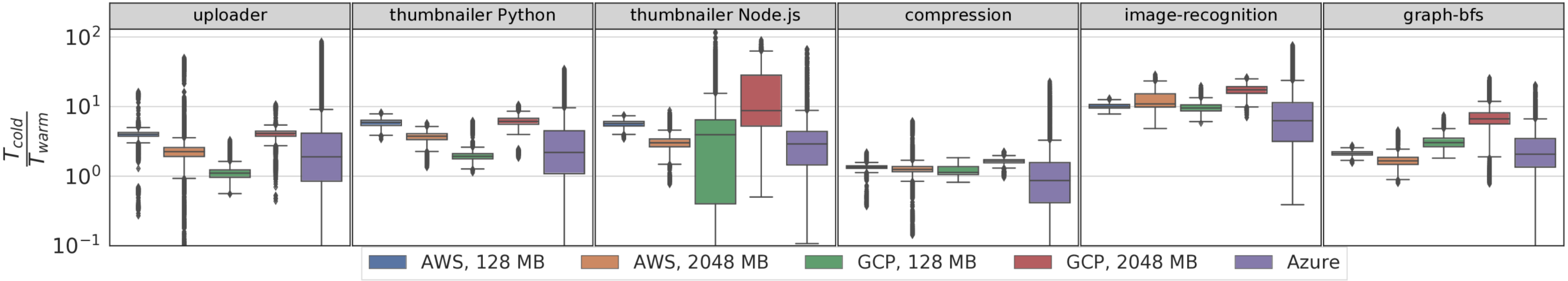
Cold Startup Overheads



- Faster invocations.
- Predictable invocation latency.
- Cacheable state.

"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", ACM/IFIP Middleware 2021

Cold Startup Overheads

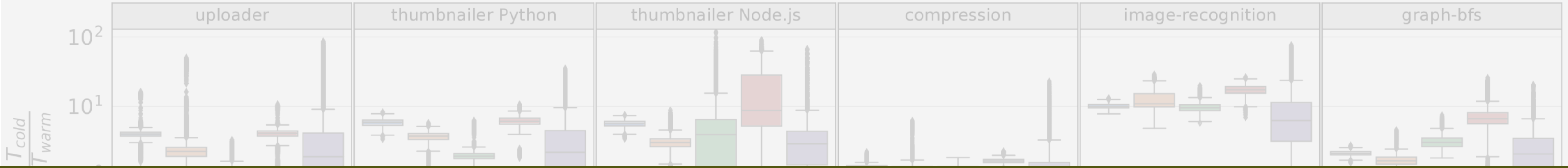


- **Faster invocations.**
- **Predictable invocation latency.**
- **Cacheable state.**

- **Increased memory consumption.**

“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, ACM/IFIP Middleware 2021

Cold Startup Overheads



Memory contributes 10% of capital and operational expenditures (MareNostrum, 2013) and 18% of peak power consumption in data center (2018).

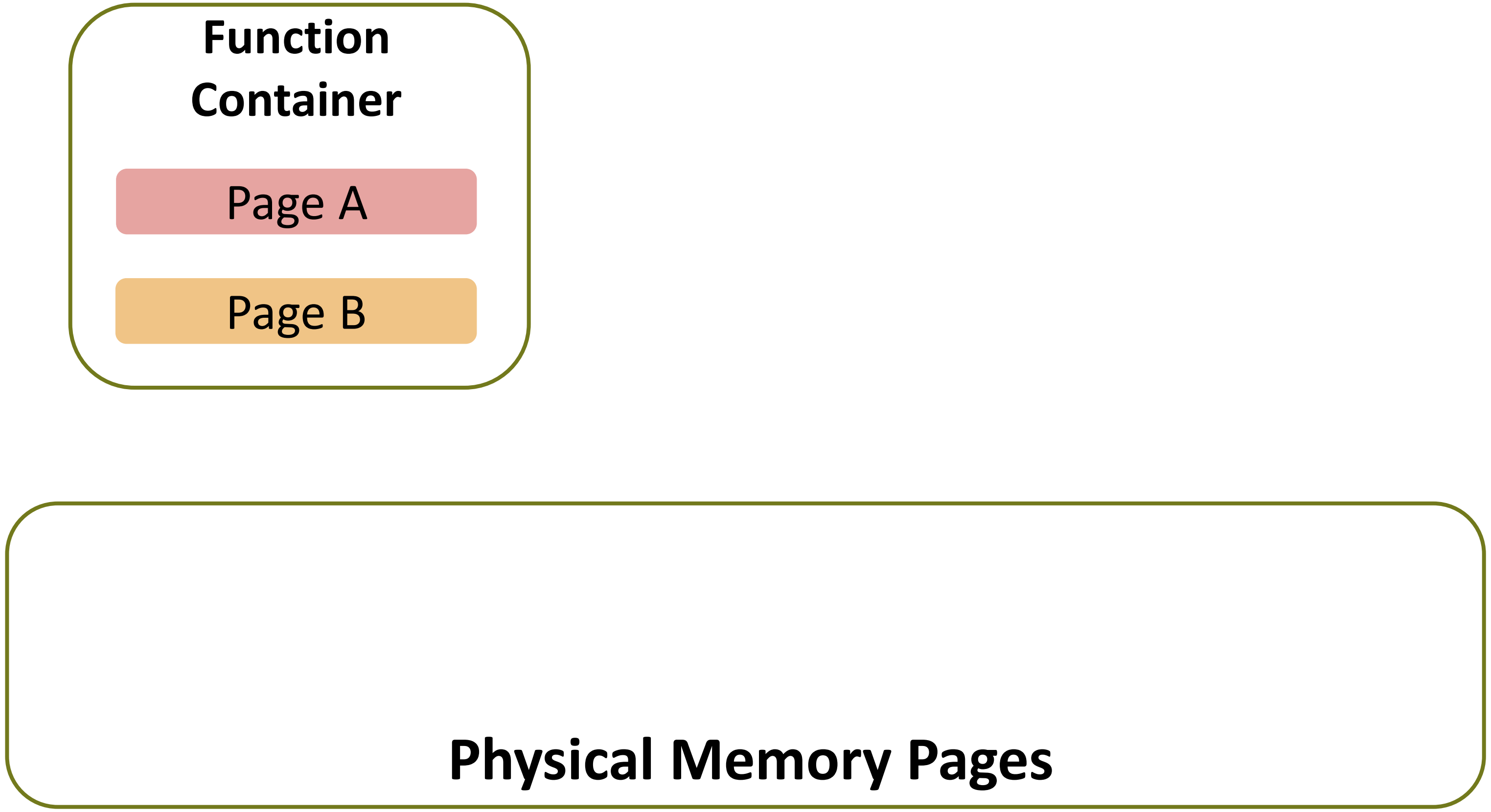
- Faster invocations.
- Predictable invocation latency.
- Cacheable state.

- Increased memory consumption.

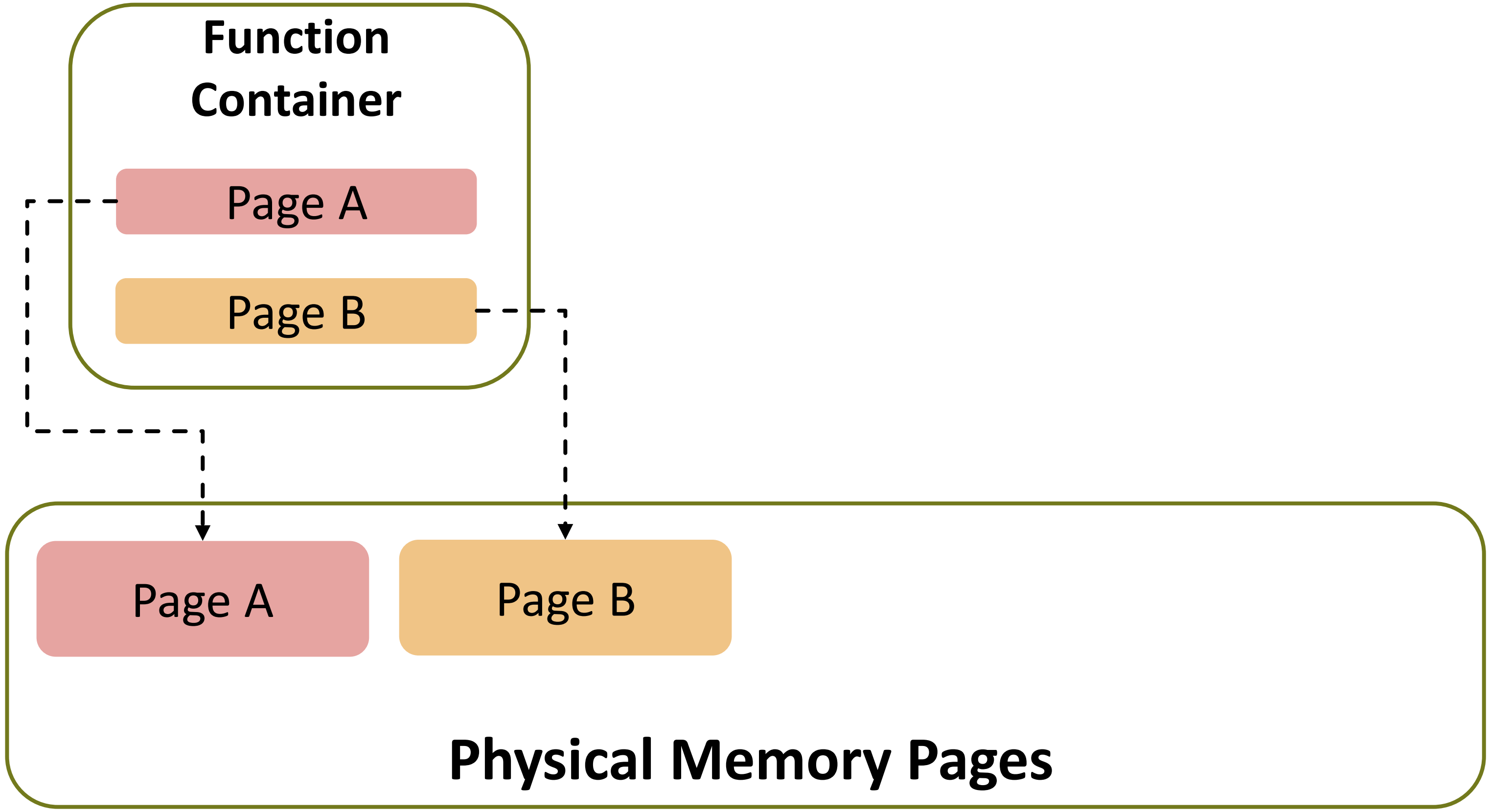
Memory Duplication in Serverless

Physical Memory Pages

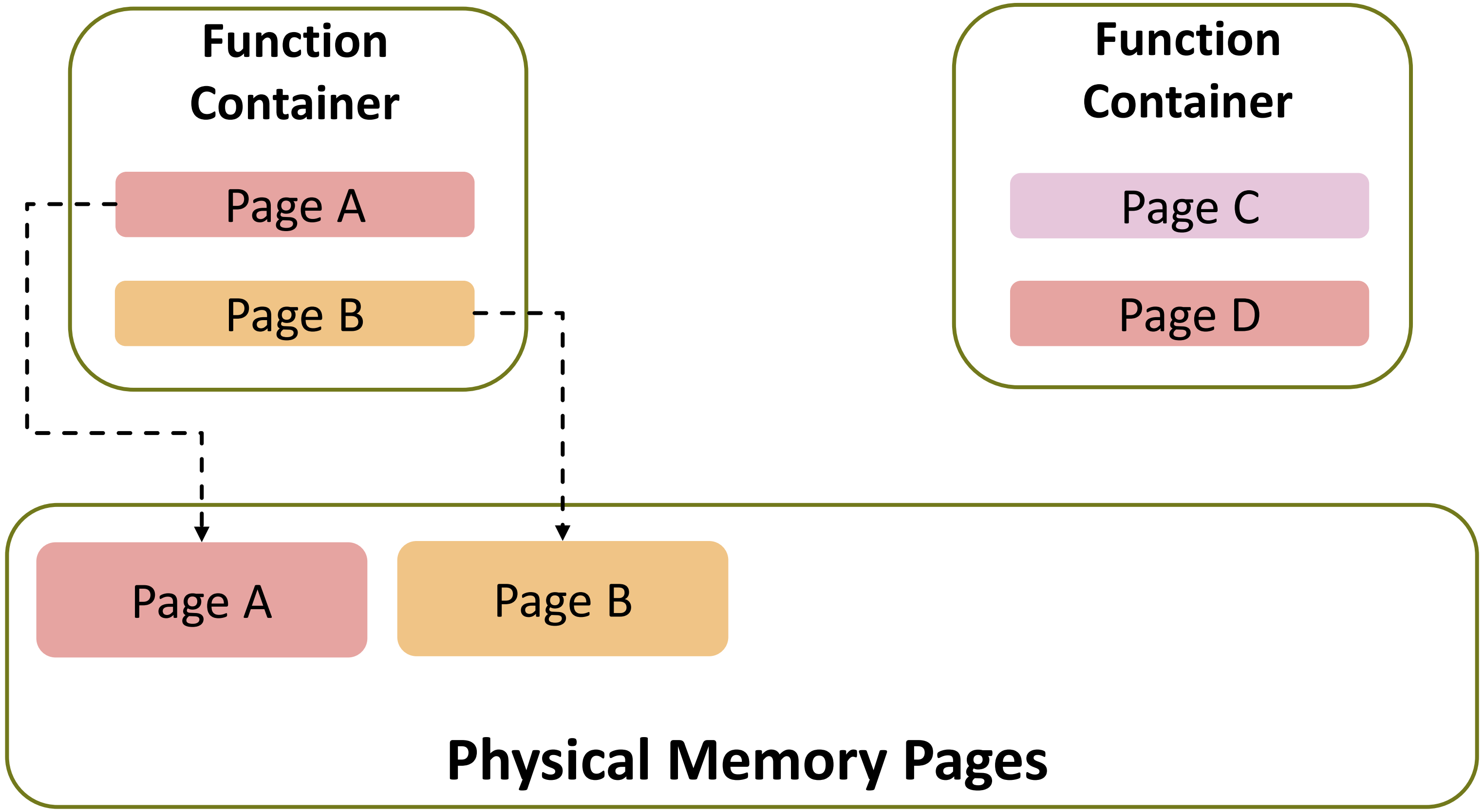
Memory Duplication in Serverless



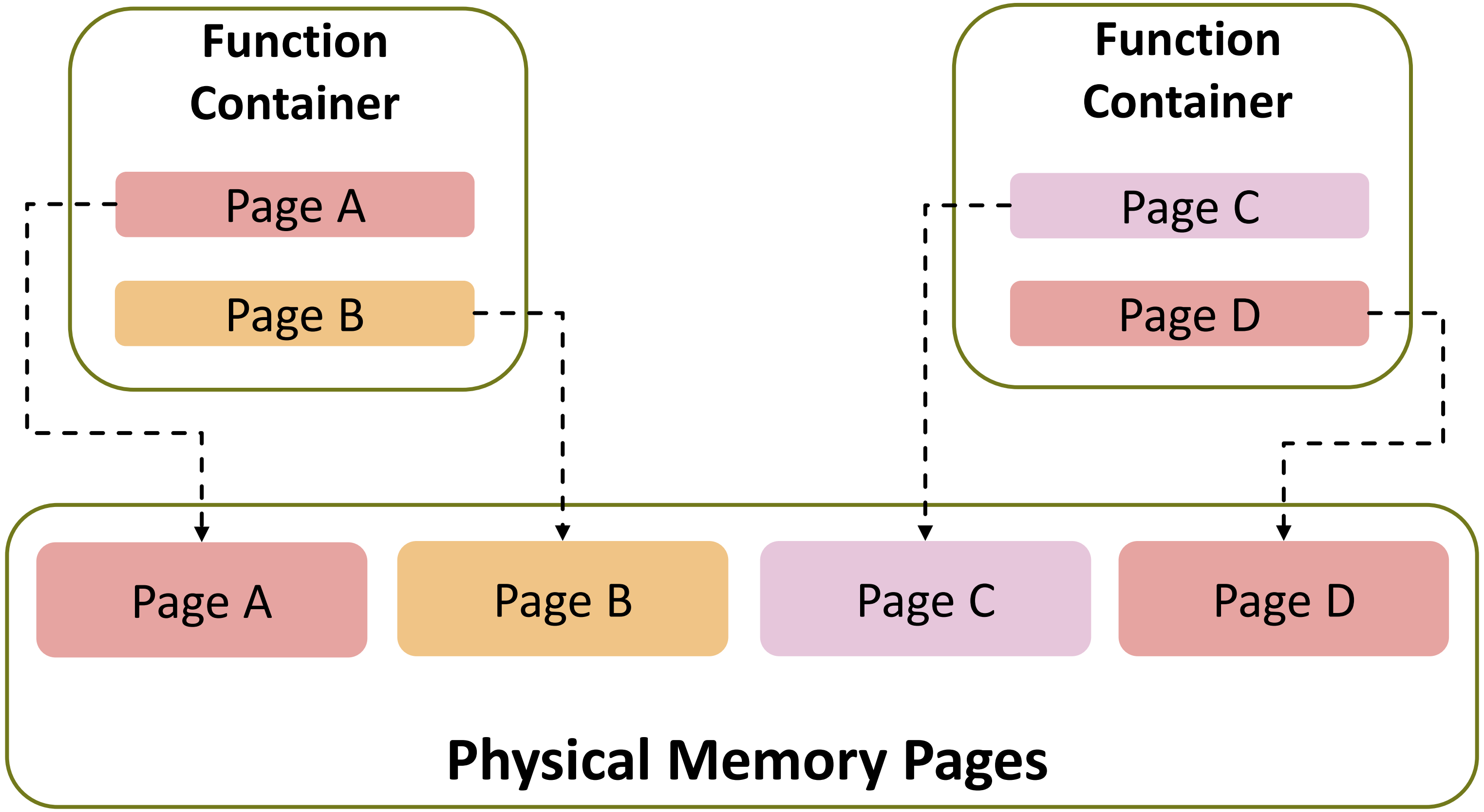
Memory Duplication in Serverless



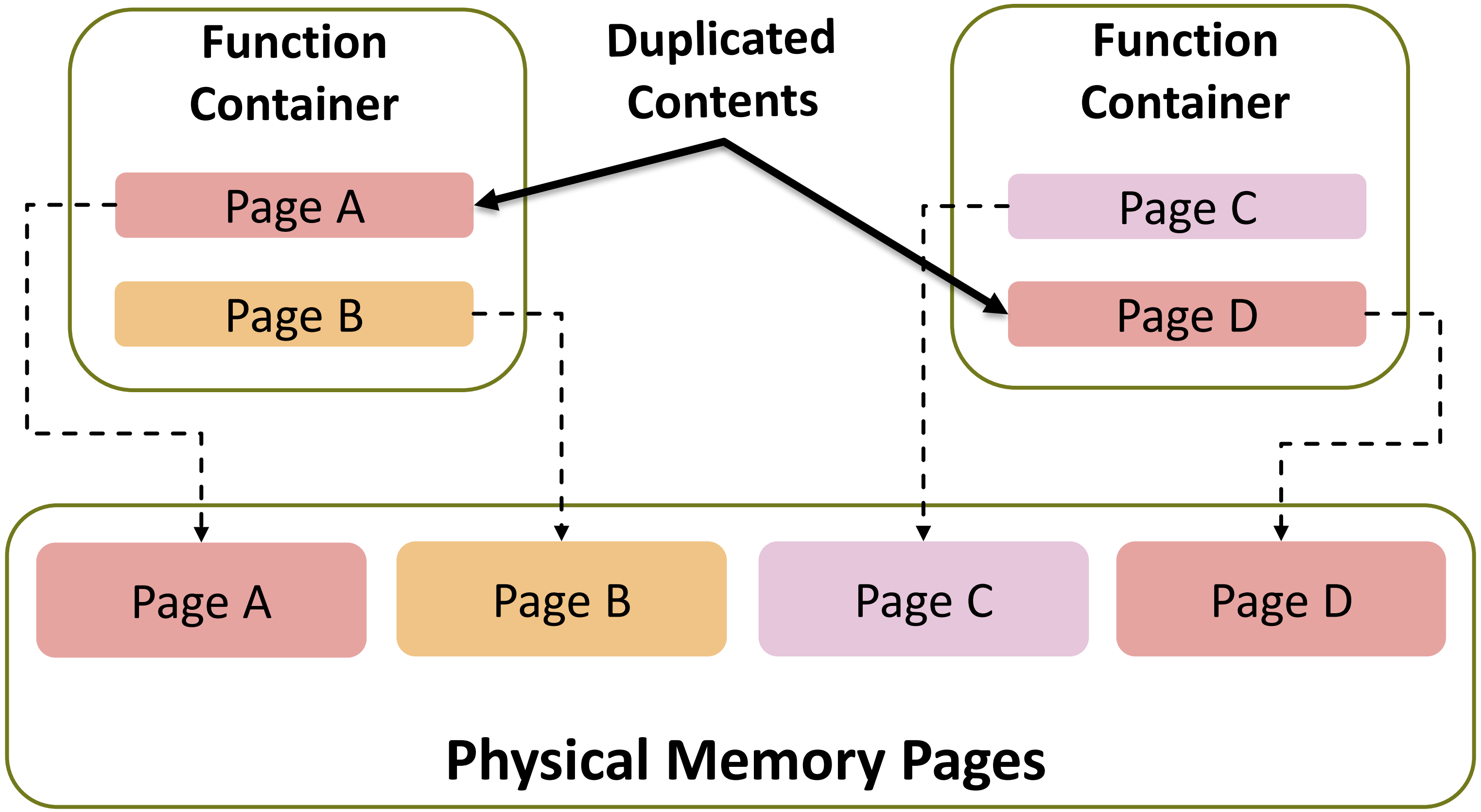
Memory Duplication in Serverless



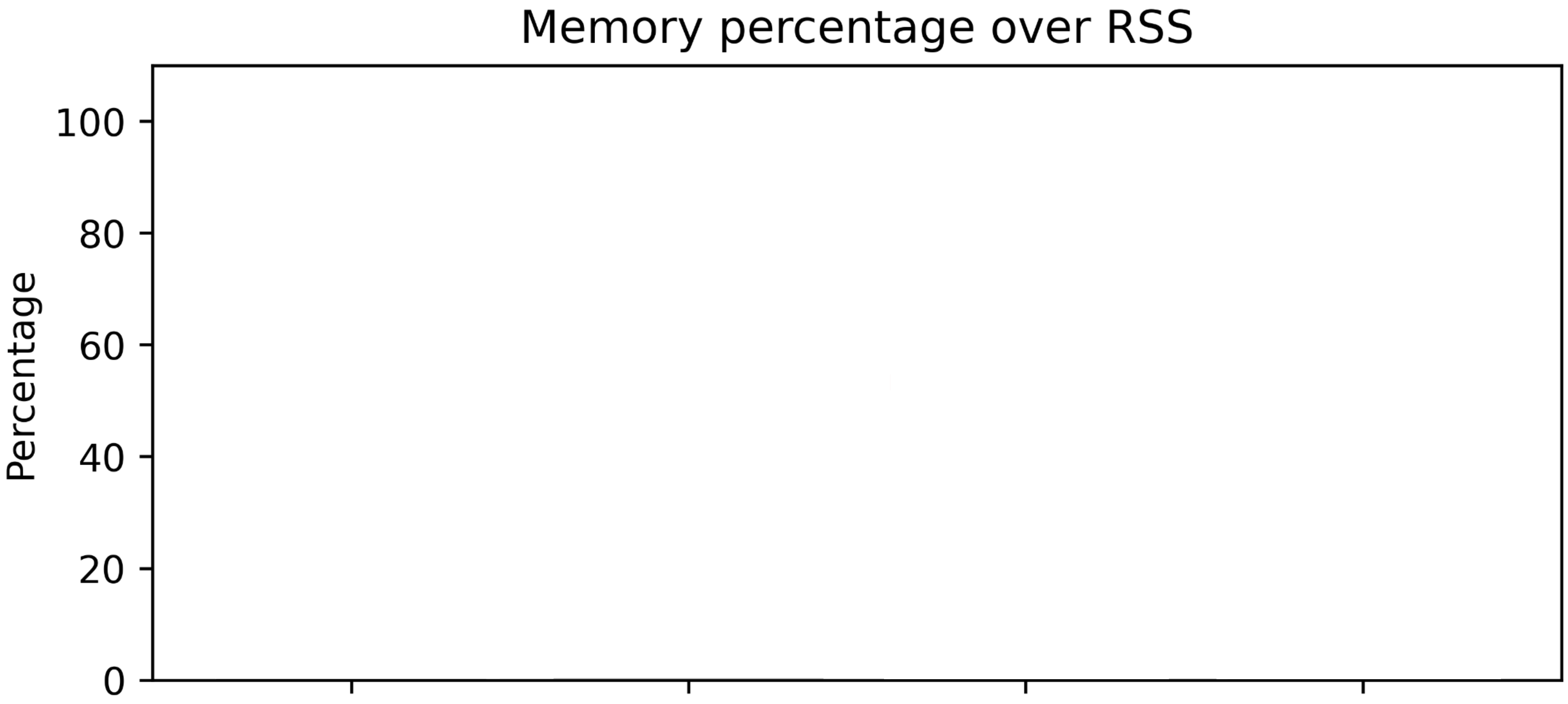
Memory Duplication in Serverless



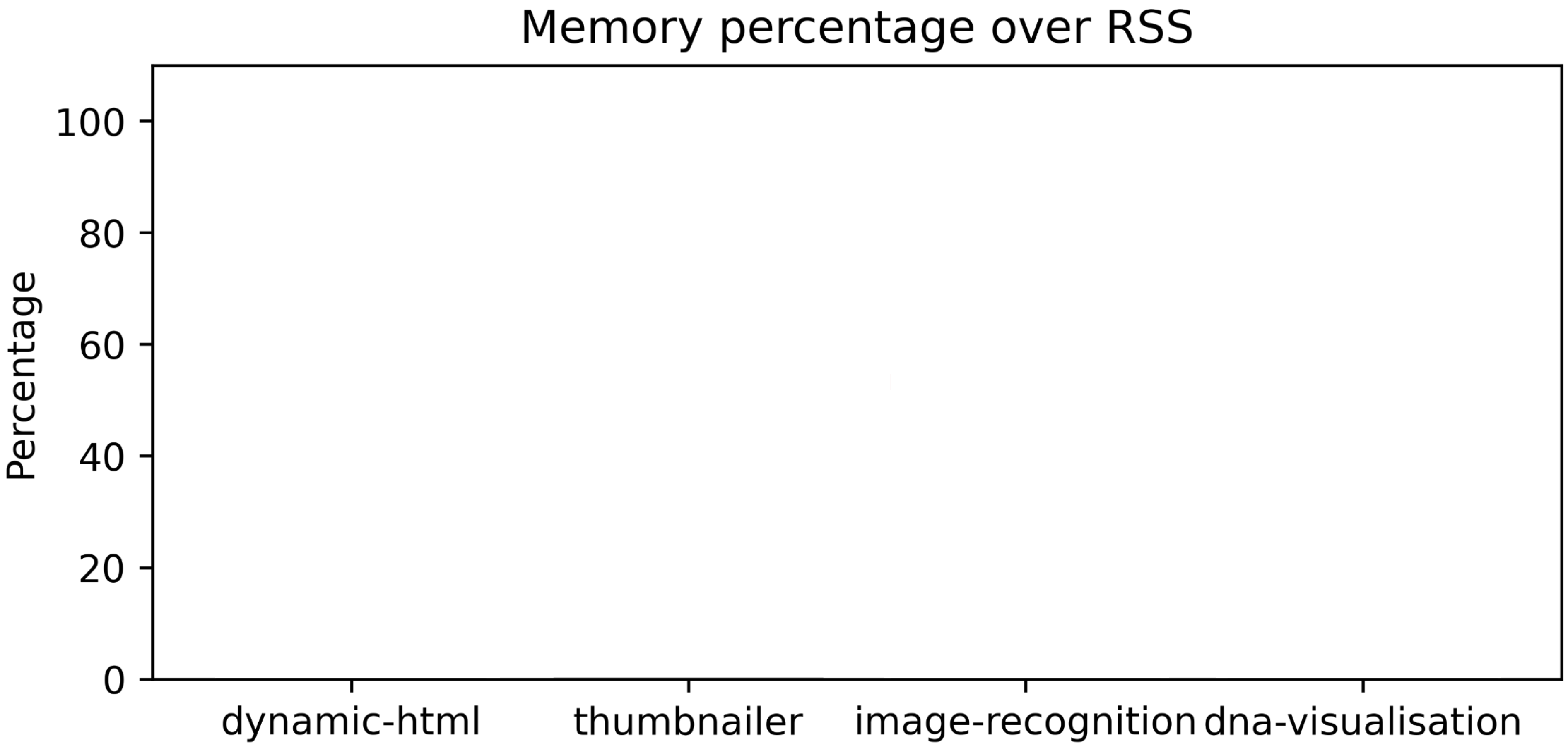
Memory Duplication in Serverless



Sharing opportunities

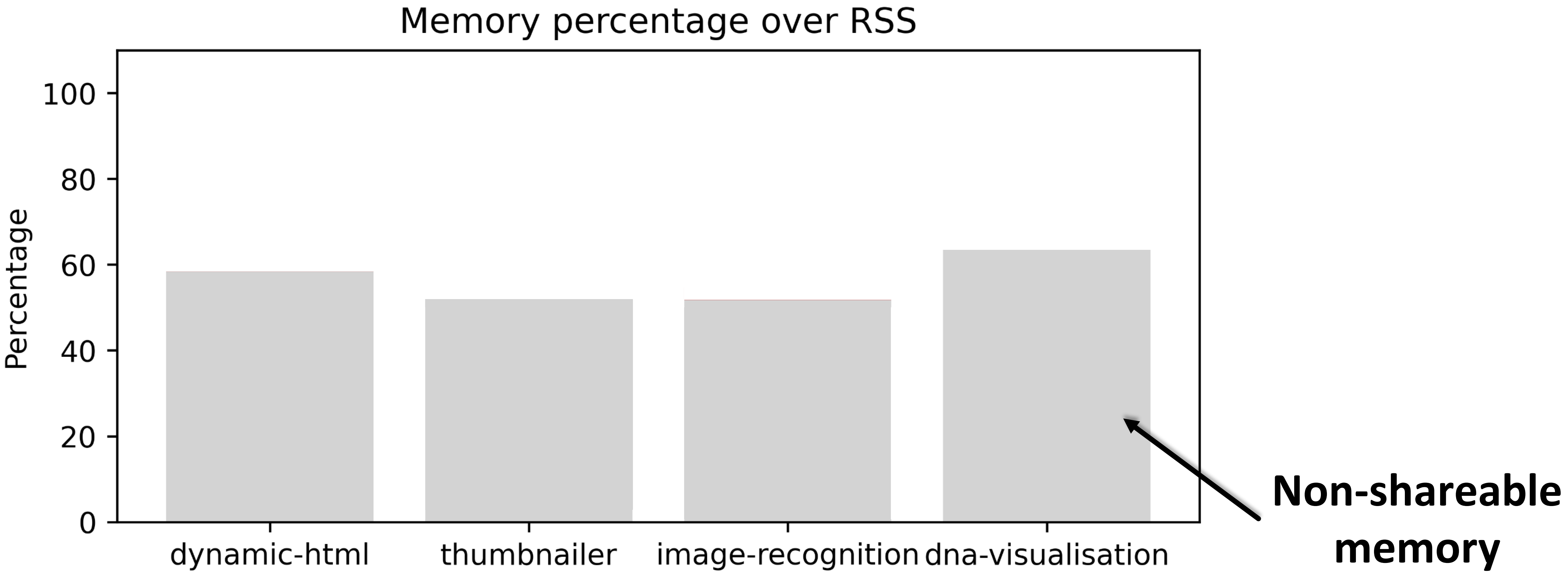


Sharing opportunities



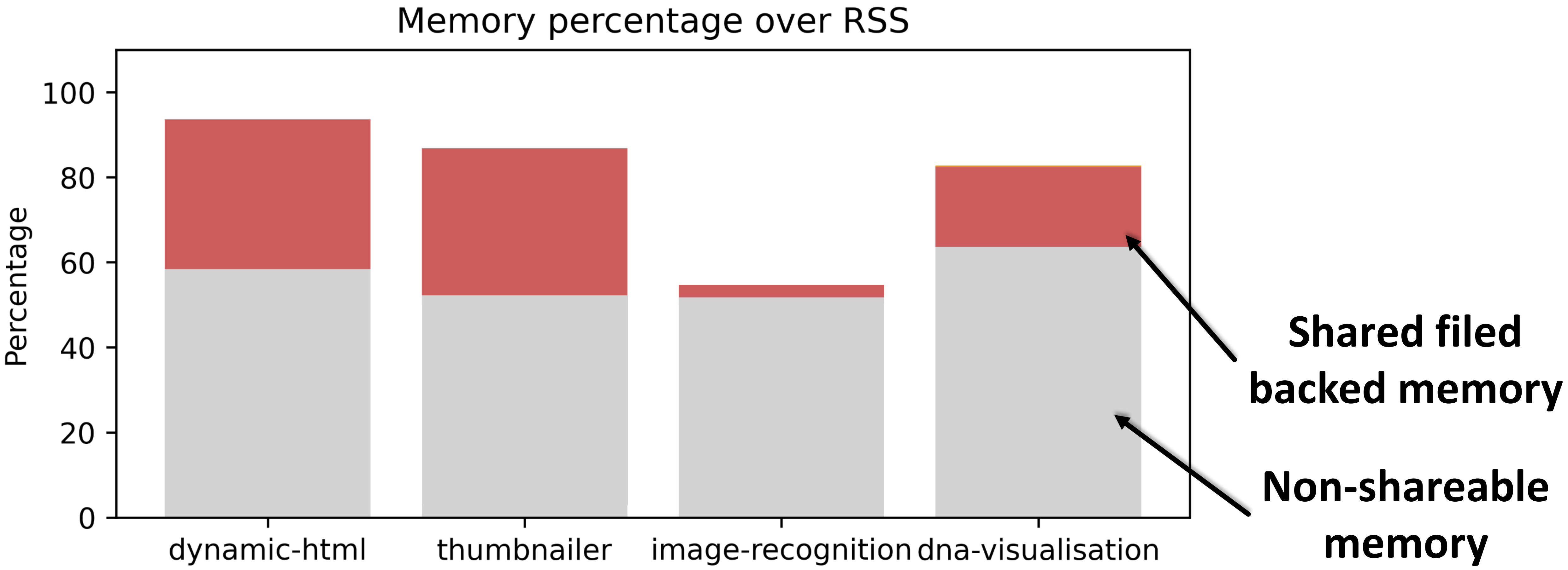
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", ACM/IFIP Middleware 2021

Sharing opportunities



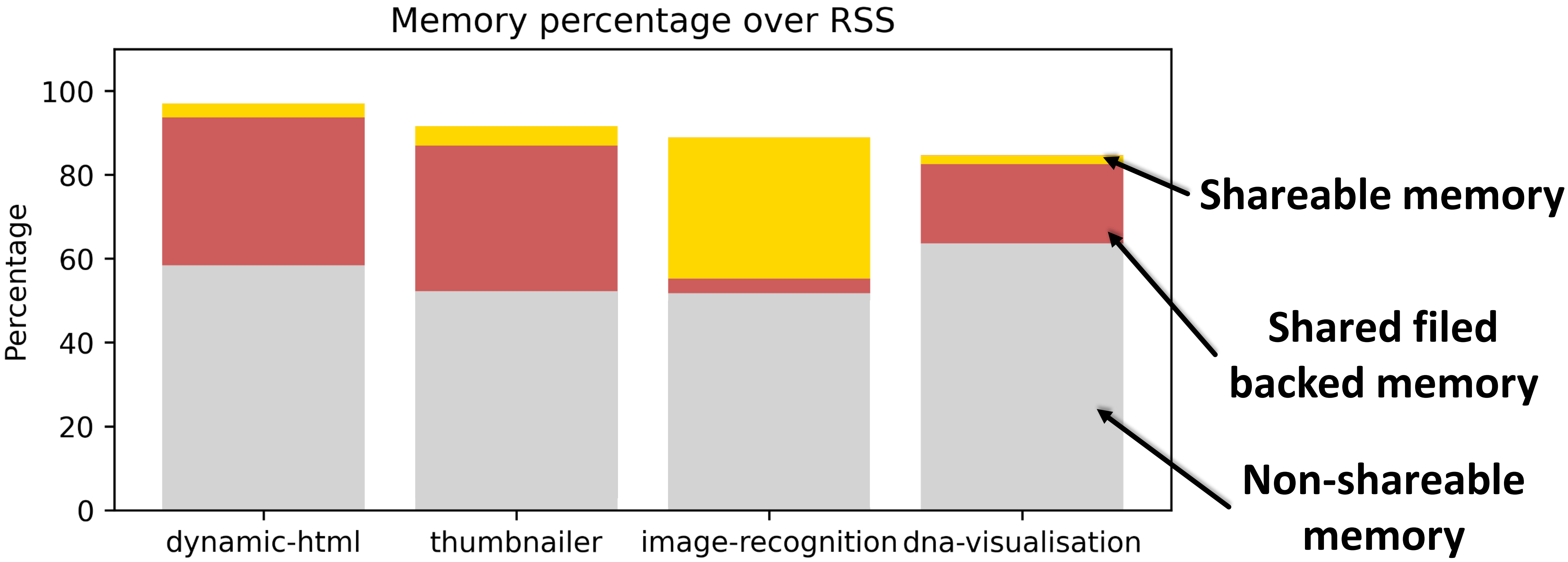
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", ACM/IFIP Middleware 2021

Sharing opportunities



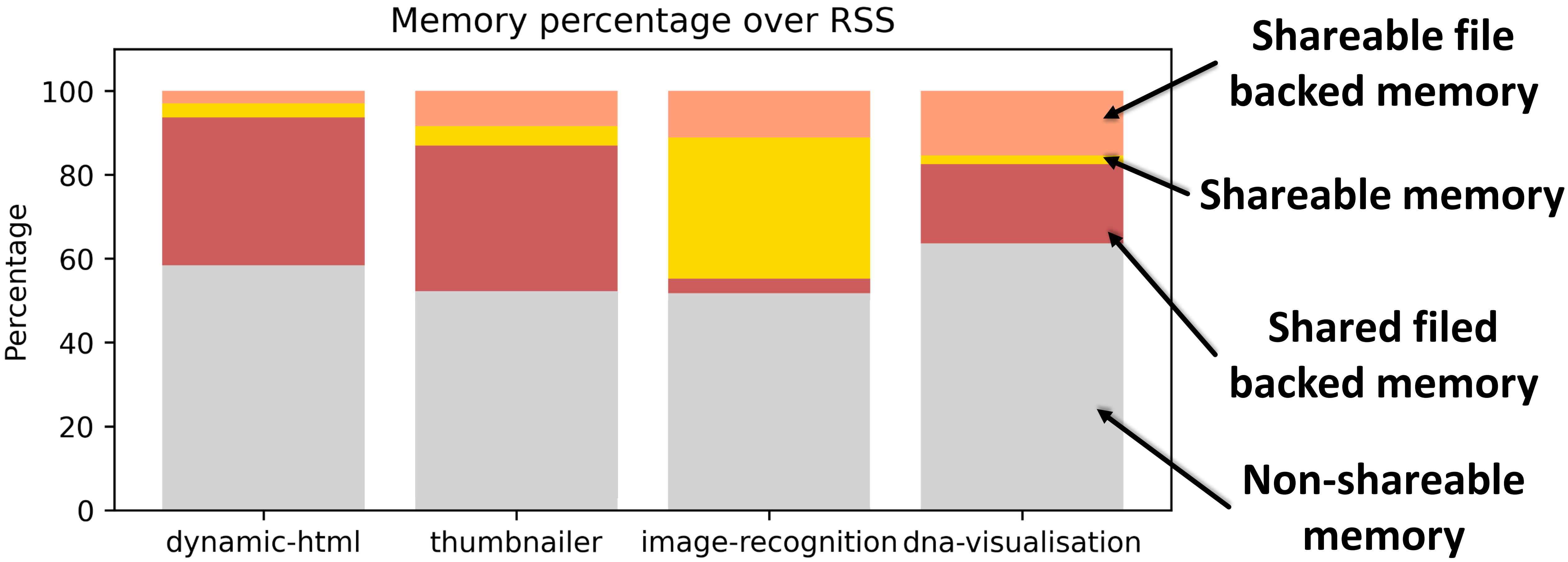
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, ACM/IFIP Middleware 2021

Sharing opportunities



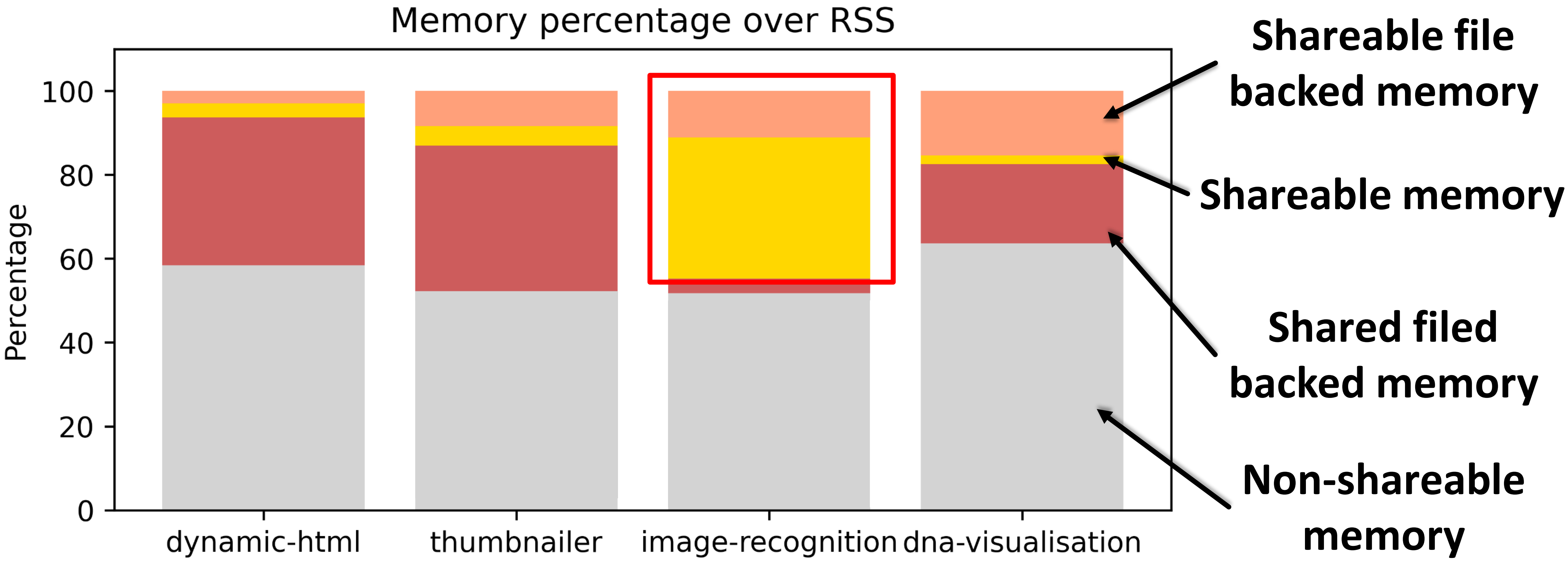
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, ACM/IFIP Middleware 2021

Sharing opportunities



“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, ACM/IFIP Middleware 2021

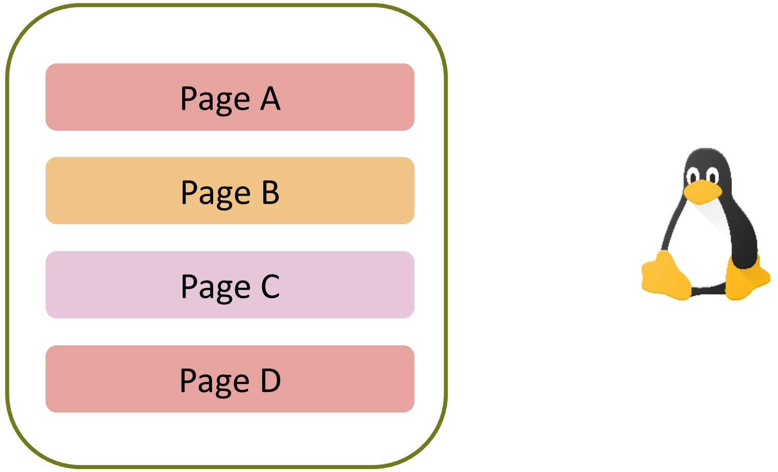
Sharing opportunities



"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", ACM/IFIP Middleware 2021

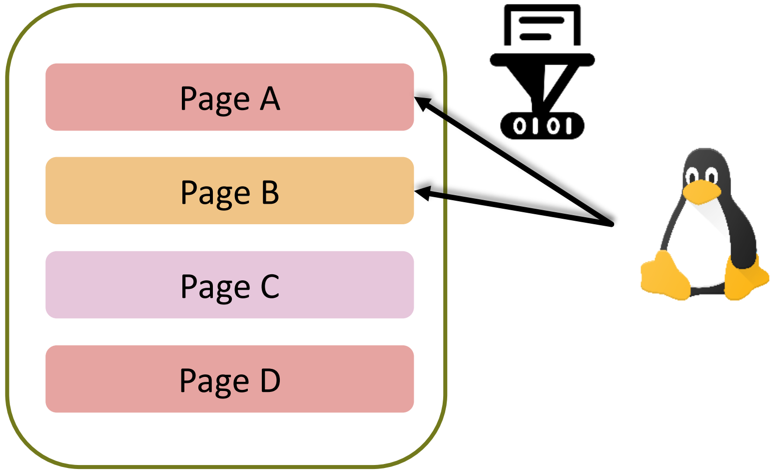
Existing Memory Deduplication Techniques

Example: Kernel Samepage Merging (KSM)



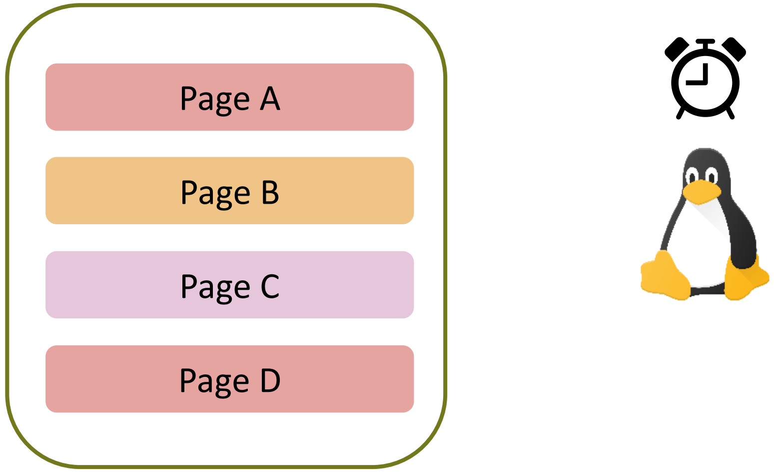
Existing Memory Deduplication Techniques

Example: Kernel Samepage Merging (KSM)



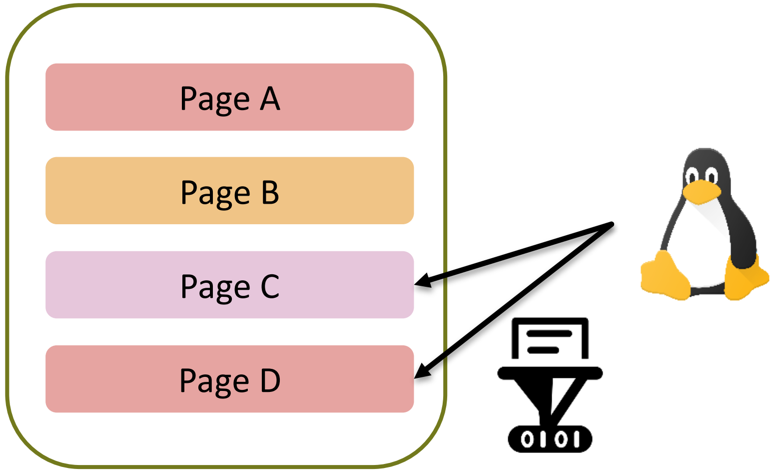
Existing Memory Deduplication Techniques

Example: Kernel Samepage Merging (KSM)



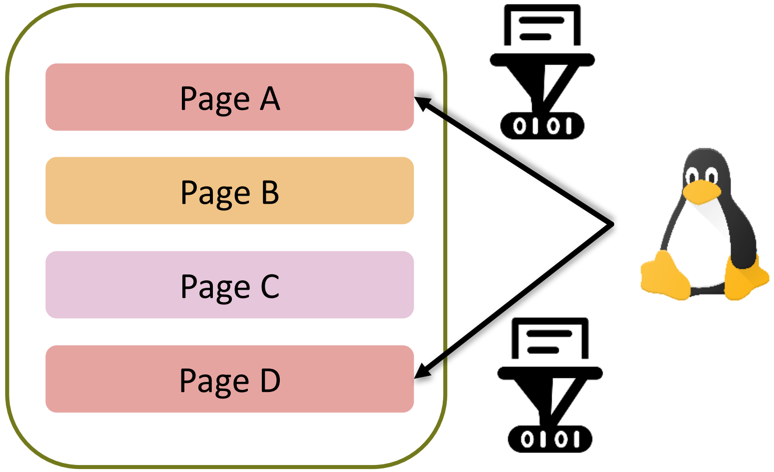
Existing Memory Deduplication Techniques

Example: Kernel Samepage Merging (KSM)



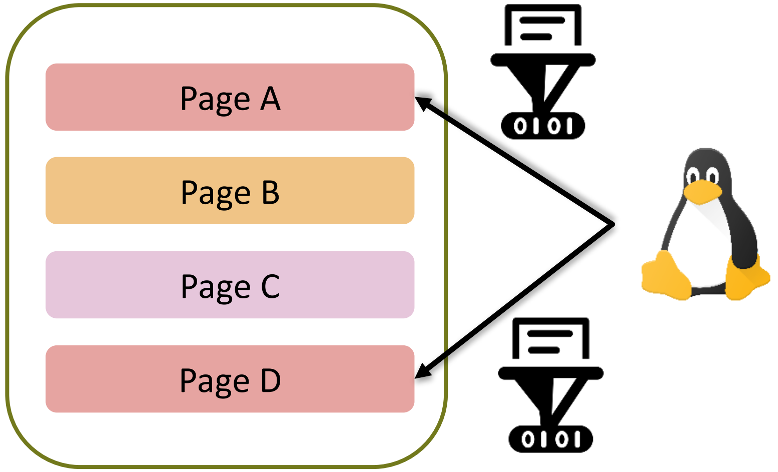
Existing Memory Deduplication Techniques

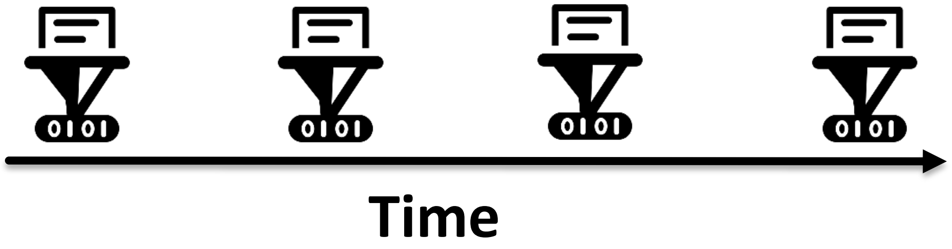
Example: Kernel Samepage Merging (KSM)



Existing Memory Deduplication Techniques

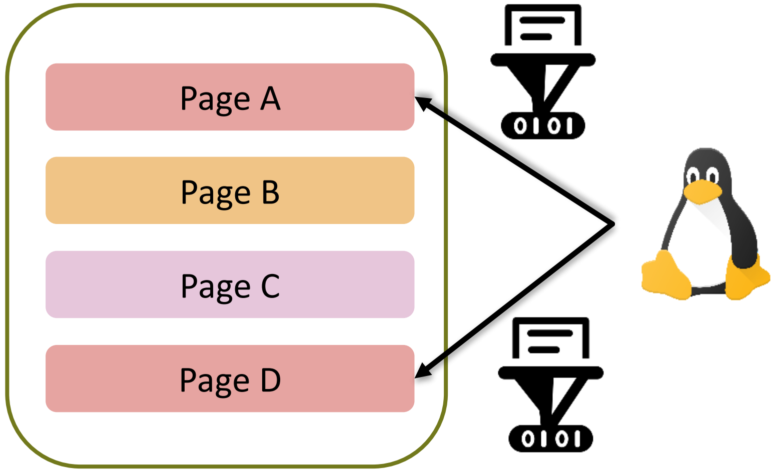
Example: Kernel Samepage Merging (KSM)

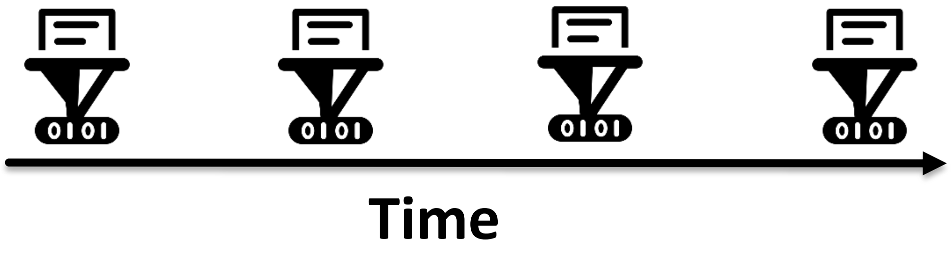




Existing Memory Deduplication Techniques

Example: Kernel Samepage Merging (KSM)





Satori: Enlightened page sharing

Grzegorz Miłoś, Derek G. Murray, Steven Hand
University of Cambridge Computer Laboratory
Cambridge, United Kingdom
First.Last@cl.cam.ac.uk

Michael A. Fetterman
NVIDIA Corporation
Bedford, Massachusetts, USA
mafetter@nvidia.com

*"For example in VMware ESX Server the default memory scan frequency is set to once an hour, with a maximum of six times per hour. Therefore, the theoretical **mean duplicate discovery time for the default setting is 40min**, which means that short-lived sharing opportunities will be missed."*

Existing Memory Deduplication Techniques

Satori: Enlightened page sharing

Grzegorz Miłoś, Derek G. Murray, Steven Hand
University of Cambridge Computer Laboratory
Cambridge, United Kingdom
First.Last@cl.cam.ac.uk

Michael A. Fetterman
NVIDIA Corporation
Bedford, Massachusetts, USA
mafetter@nvidia.com

"For example in VMware ESX Server the default memory scan frequency is set to once an hour, with a maximum of six times per hour. Therefore, the theoretical ***mean duplicate discovery time for the default setting is 40min,*** *which means that short-lived sharing opportunities will be missed."*

Existing Memory Deduplication Techniques

Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider

Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini, *Microsoft Azure and Microsoft Research*

*"We observe that 50% of the functions execute for less than 1s on average, and 50% of the functions have maximum execution time shorter than ~3s; 90% of the functions take at most 60s, and **96% of functions take less than 60s on average.**"*

Satori: Enlightened page sharing

Grzegorz Miłoś, Derek G. Murray, Steven Hand
University of Cambridge Computer Laboratory
Cambridge, United Kingdom
First.Last@cl.cam.ac.uk

Michael A. Fetterman
NVIDIA Corporation
Bedford, Massachusetts, USA
mafetter@nvidia.com

*"For example in VMware ESX Server the default memory scan frequency is set to once an hour, with a maximum of six times per hour. Therefore, the theoretical **mean duplicate discovery time for the default setting is 40min**, which means that short-lived sharing opportunities will be missed."*

Existing Memory Deduplication Techniques

Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider

Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini, *Microsoft Azure and Microsoft Research*

"We observe that 50% of the functions execute for less than 1s on average, and 50% of the functions have maximum execution time shorter than ~3s; 90% of the functions take at most 60s, and ***96% of functions take less than 60s on average.****"*

SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing

Marcin Copik
marcin.copik@inf.ethz.ch
ETH Zürich
Switzerland

Grzegorz Kwaśniewski
ETH Zürich
Switzerland

Maciej Besta
ETH Zürich
Switzerland

Michał Podstawski
Future Processing SA
Poland

Torsten Hoefler
ETH Zürich
Switzerland

Number of idle function containers is **reduced every 380 seconds** on a major commercial serverless provider.

"For example in VMware ESX Server the default memory scan frequency is set to once an hour, with a maximum of six times per hour. Therefore, the theoretical ***mean duplicate discovery time for the default setting is 40min****, which means that short-lived sharing opportunities will be missed."*

Memory Deduplication in Serverless

Memory Deduplication in Serverless

Speed
Deduplication in seconds,
not minutes.

Memory Deduplication in Serverless

Speed
Deduplication in seconds,
not minutes.

Compatibility
No changes to existing runtimes.

Memory Deduplication in Serverless

Speed
Deduplication in seconds, not minutes.

Compatibility
No changes to existing runtimes.

Opt-in
Zero overhead when unused and no sharing for sensitive data.

Memory Deduplication in Serverless

Speed
Deduplication in seconds, not minutes.

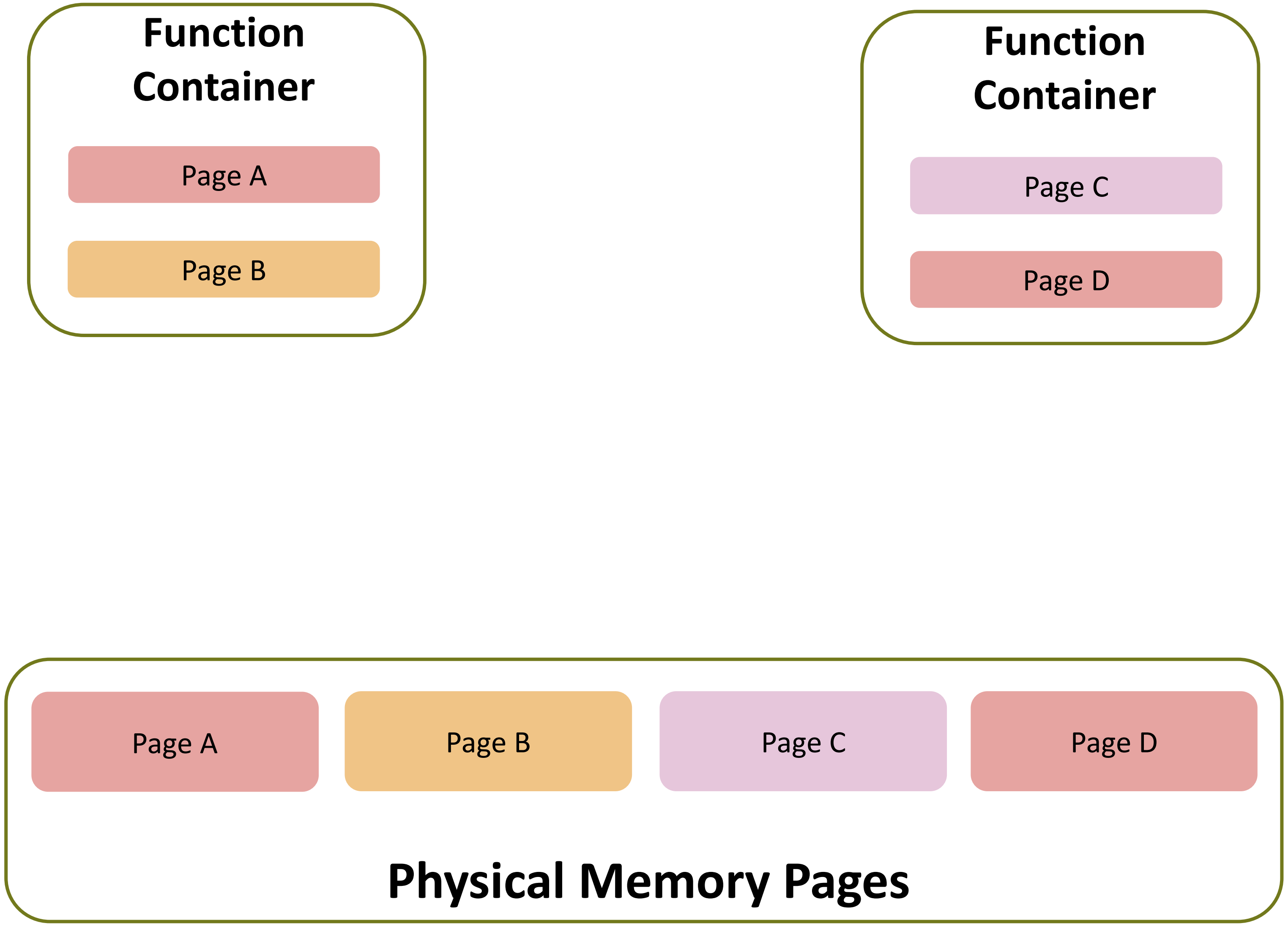
Compatibility
No changes to existing runtimes.

Opt-in
Zero overhead when unused and no sharing for sensitive data.

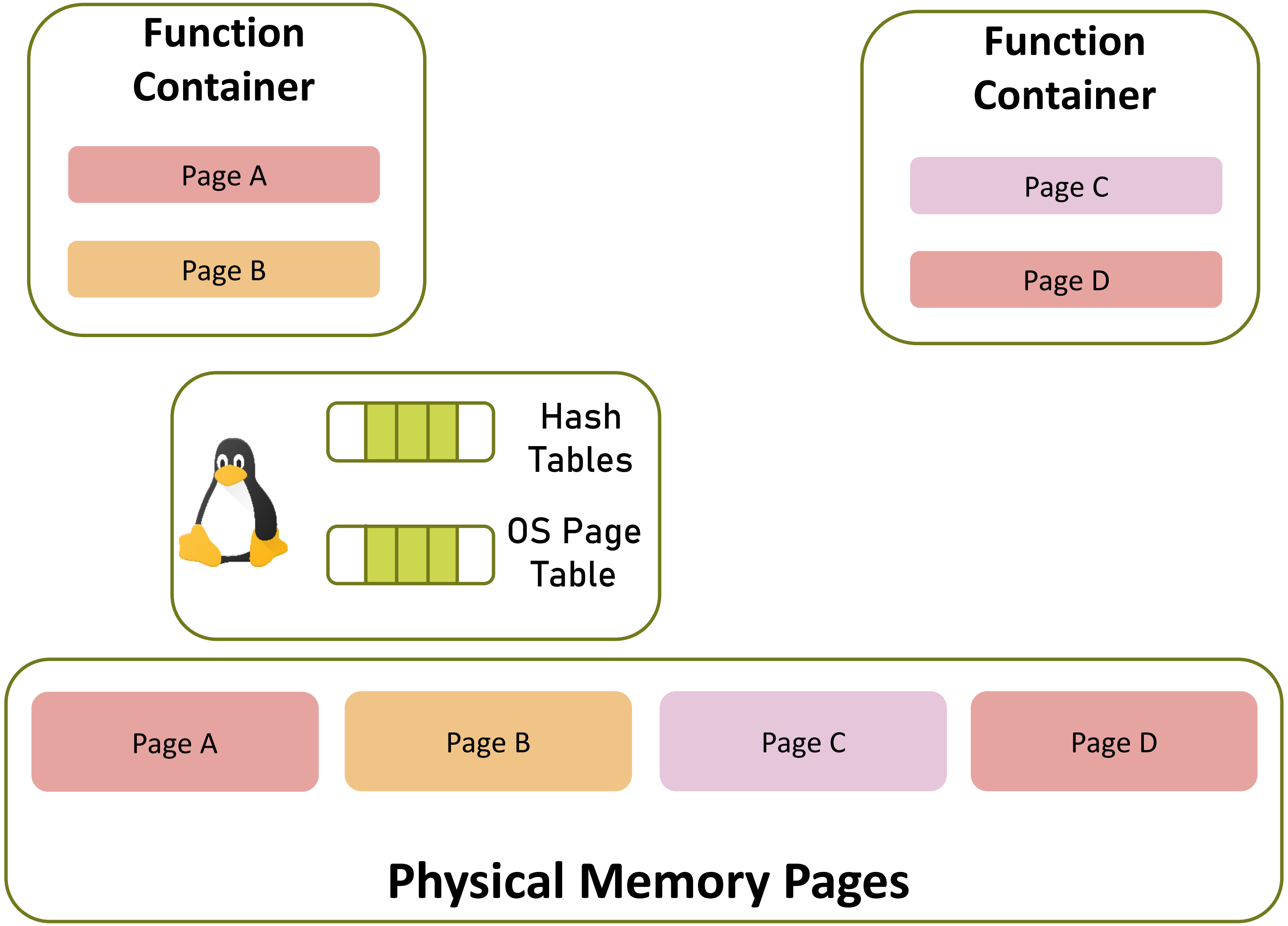
Concurrency
Support many containers operating concurrently.

UPM: User-Guided Page Merging

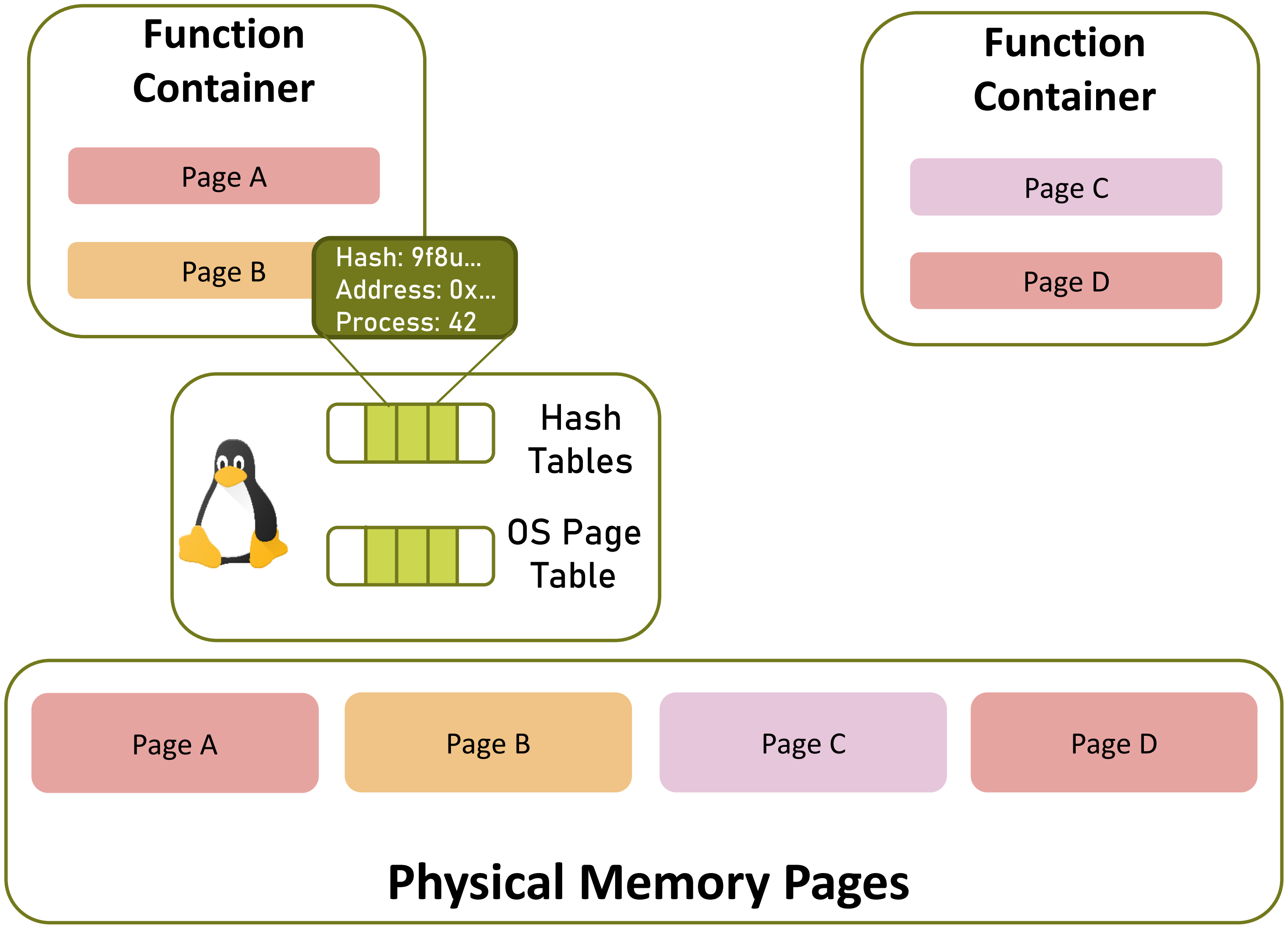
UPM: User-Guided Page Merging



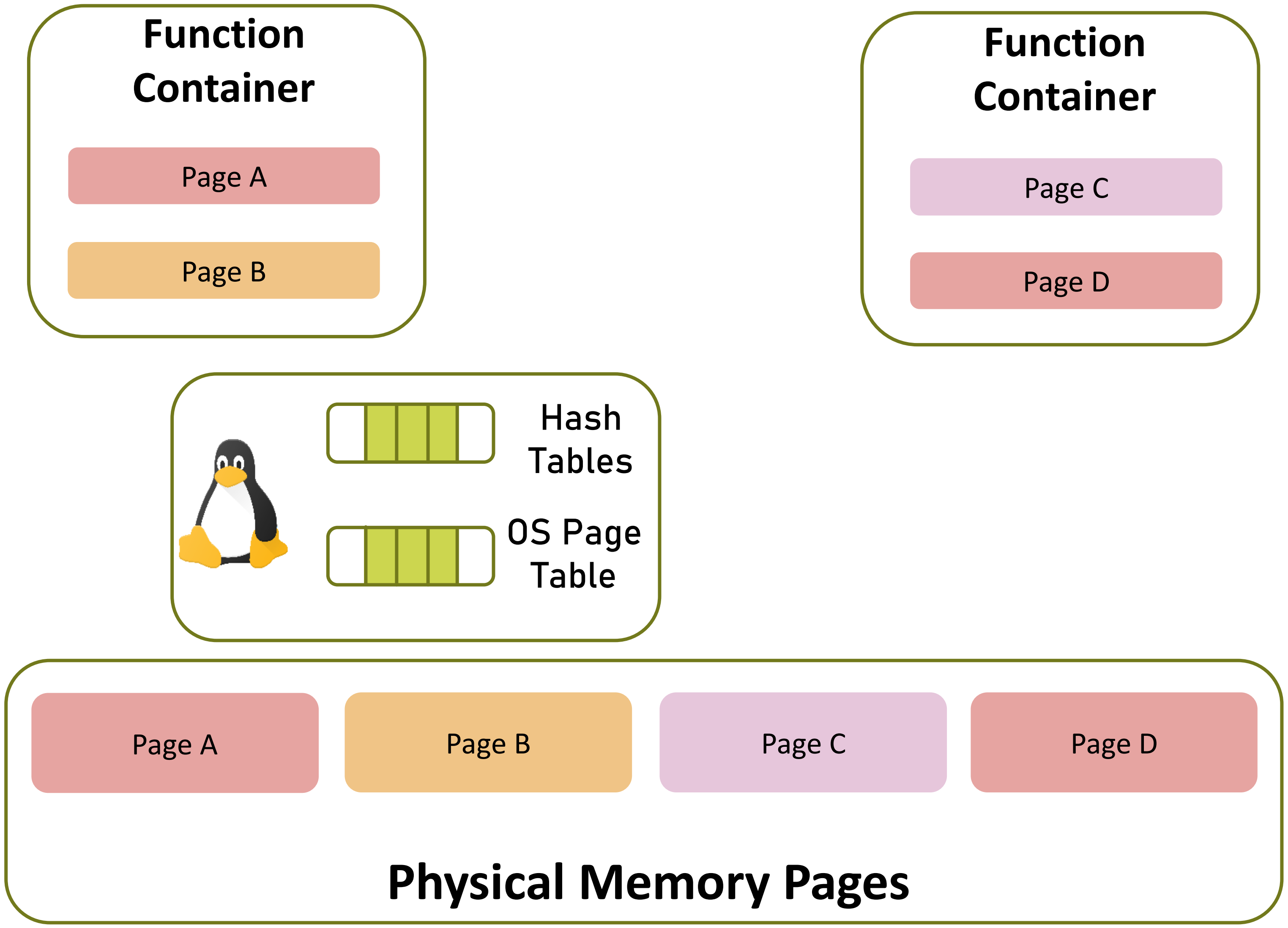
UPM: User-Guided Page Merging



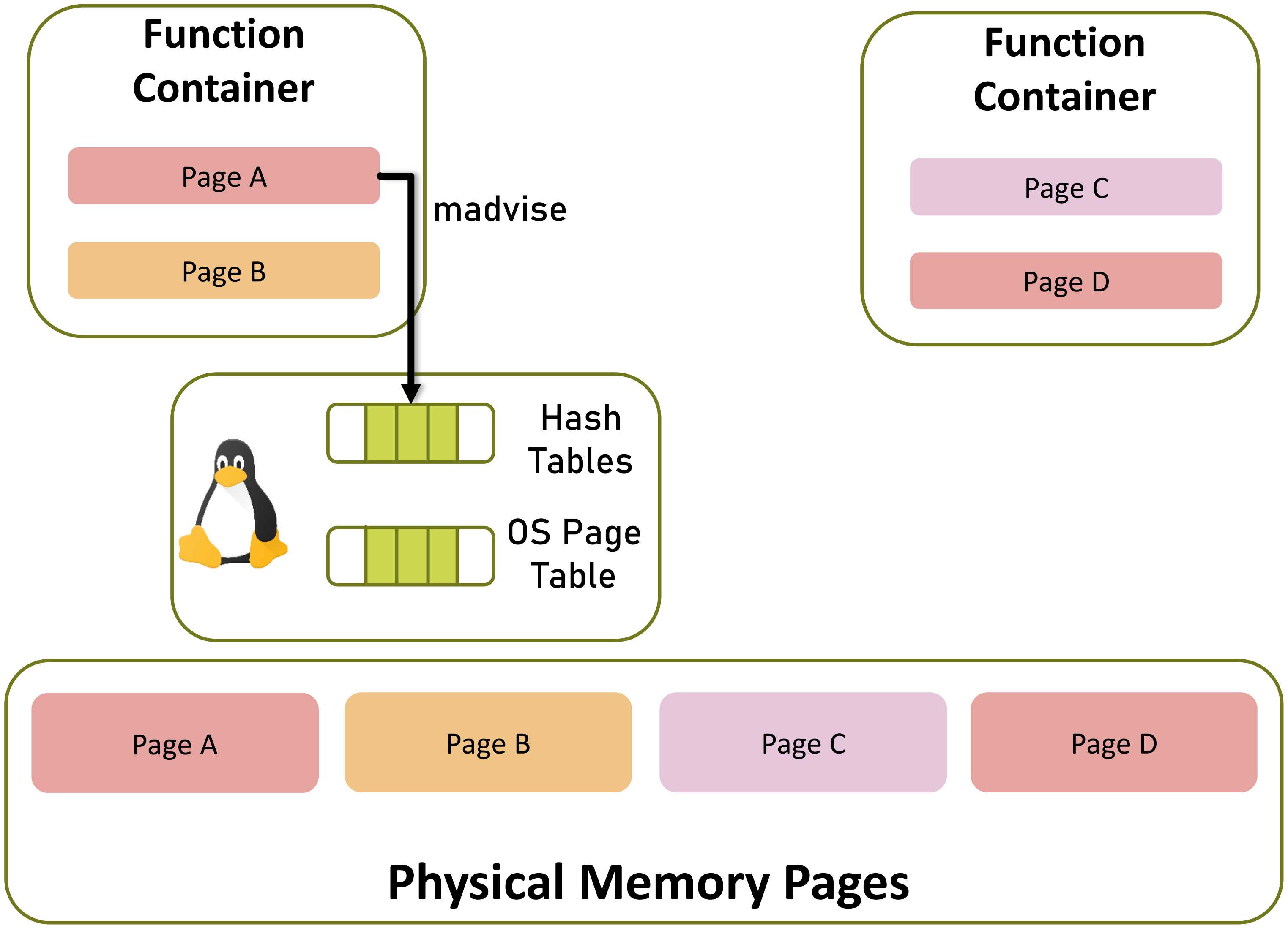
UPM: User-Guided Page Merging



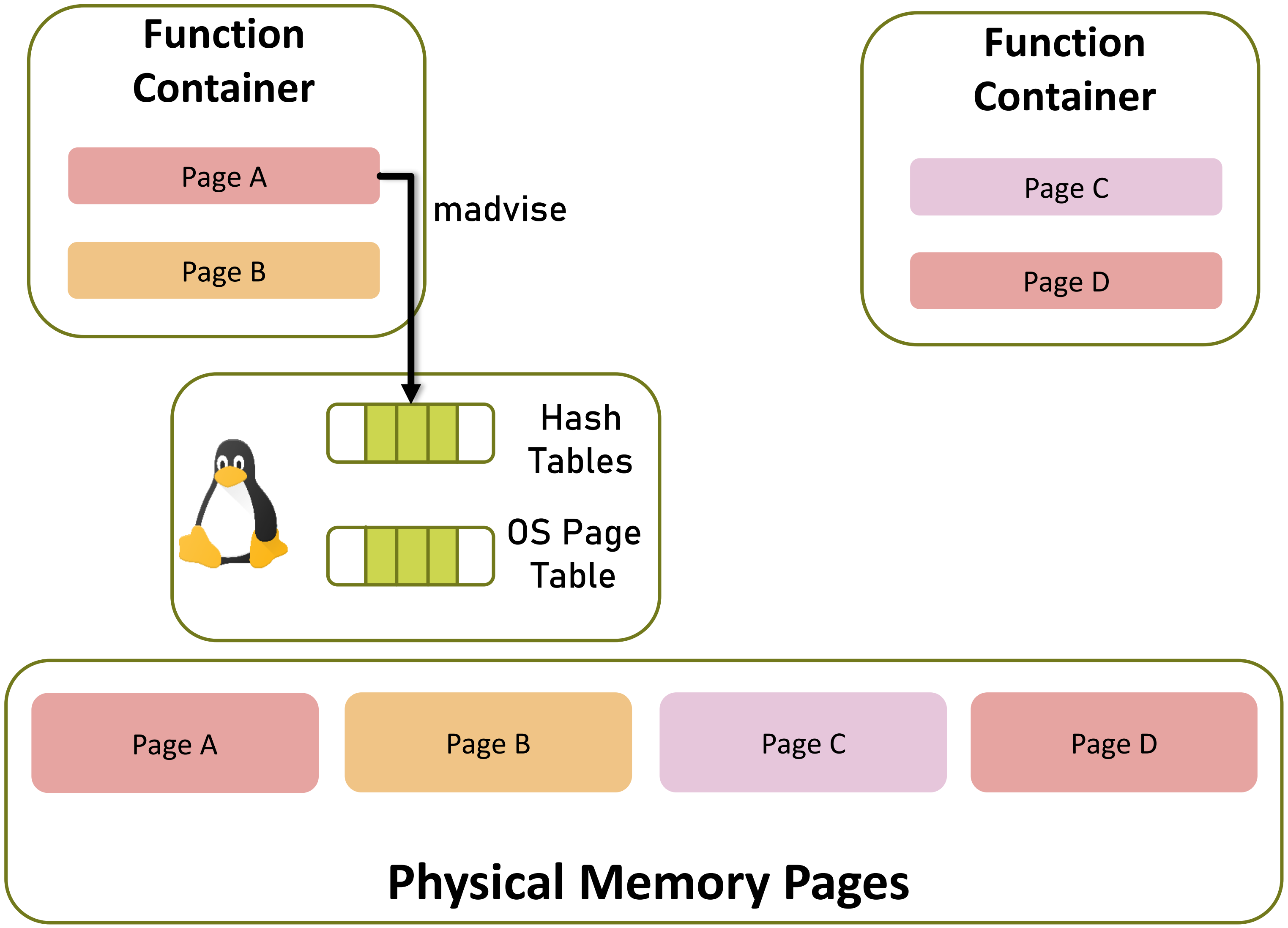
UPM: User-Guided Page Merging



UPM: User-Guided Page Merging

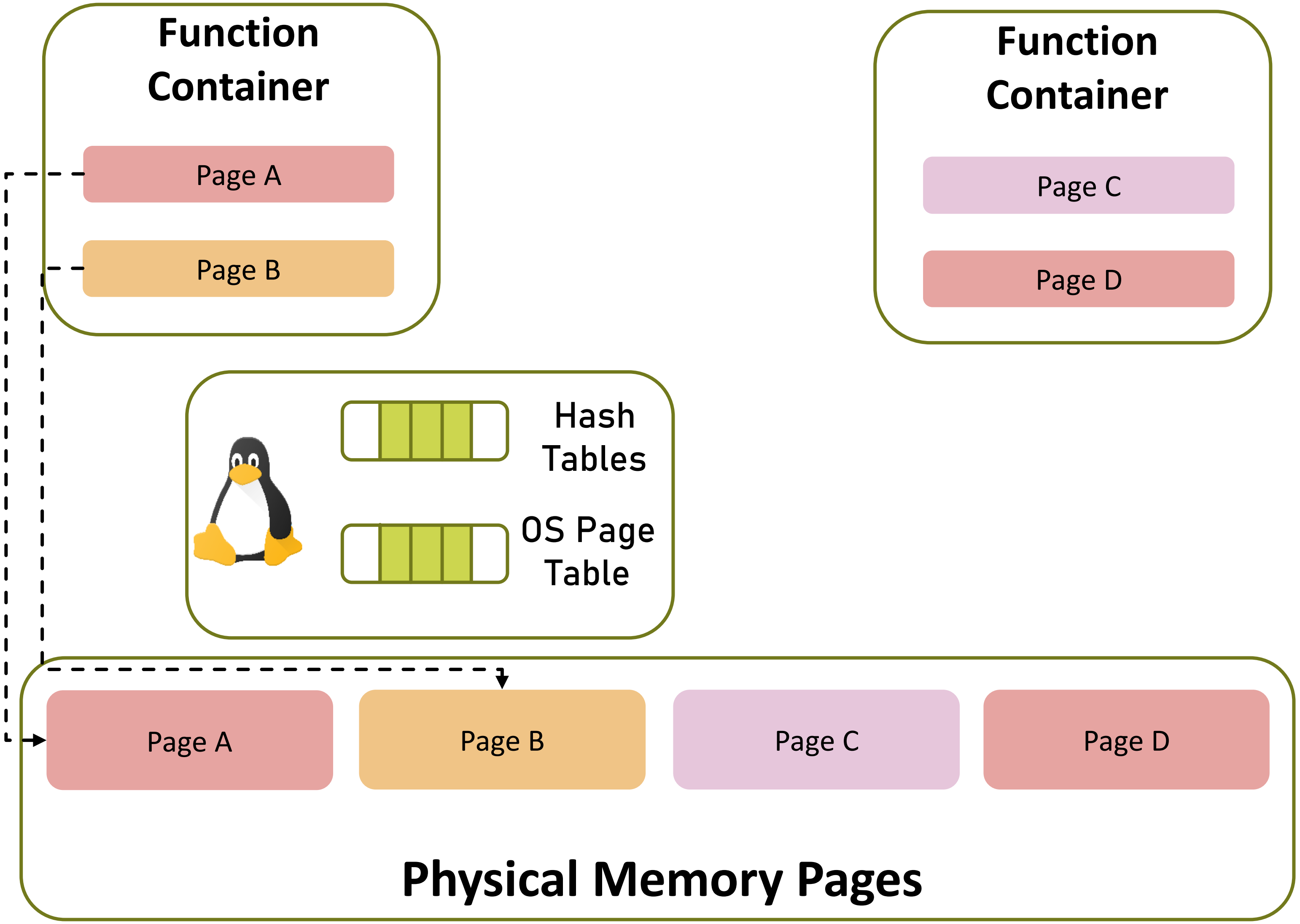


UPM: User-Guided Page Merging



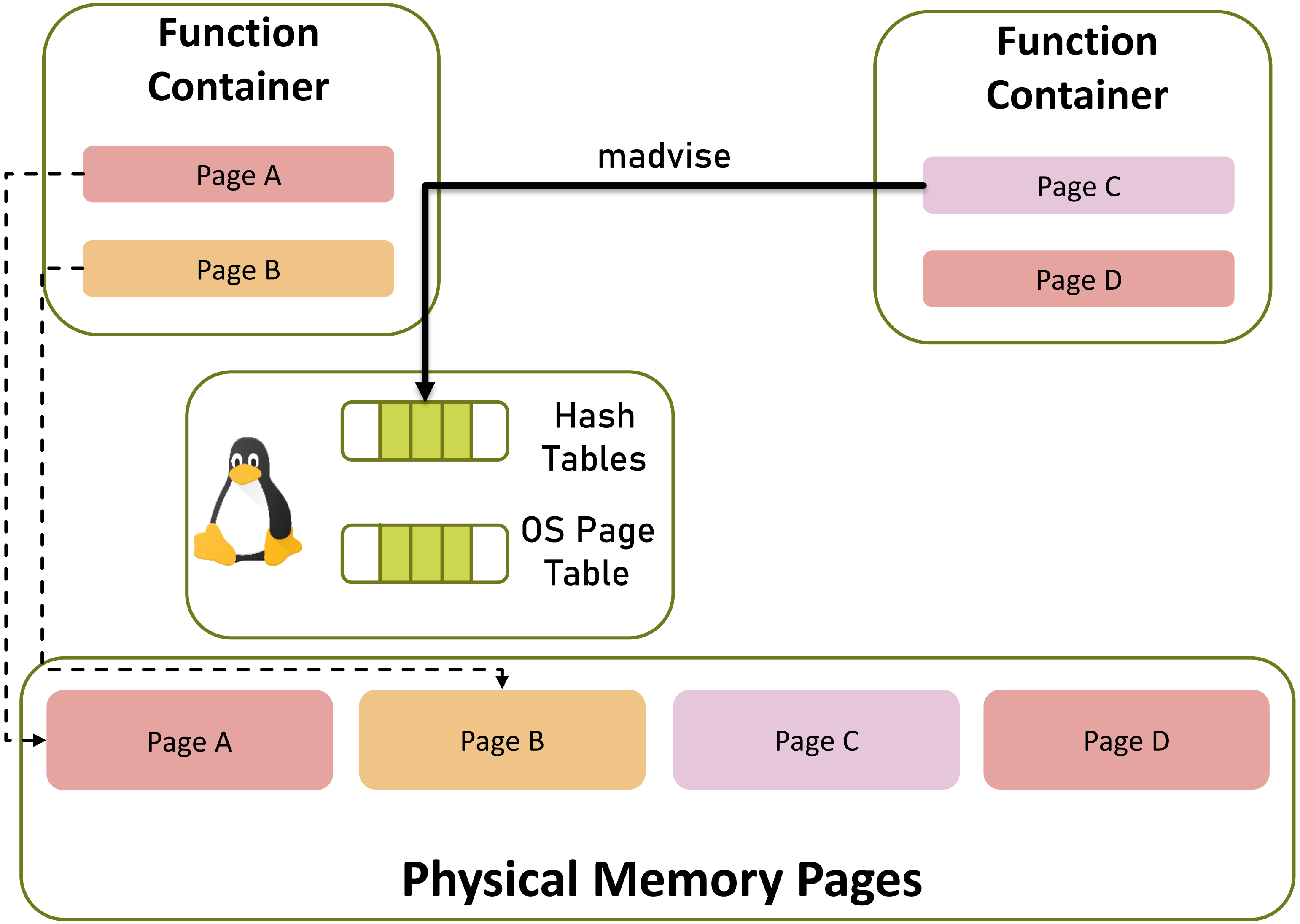

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging



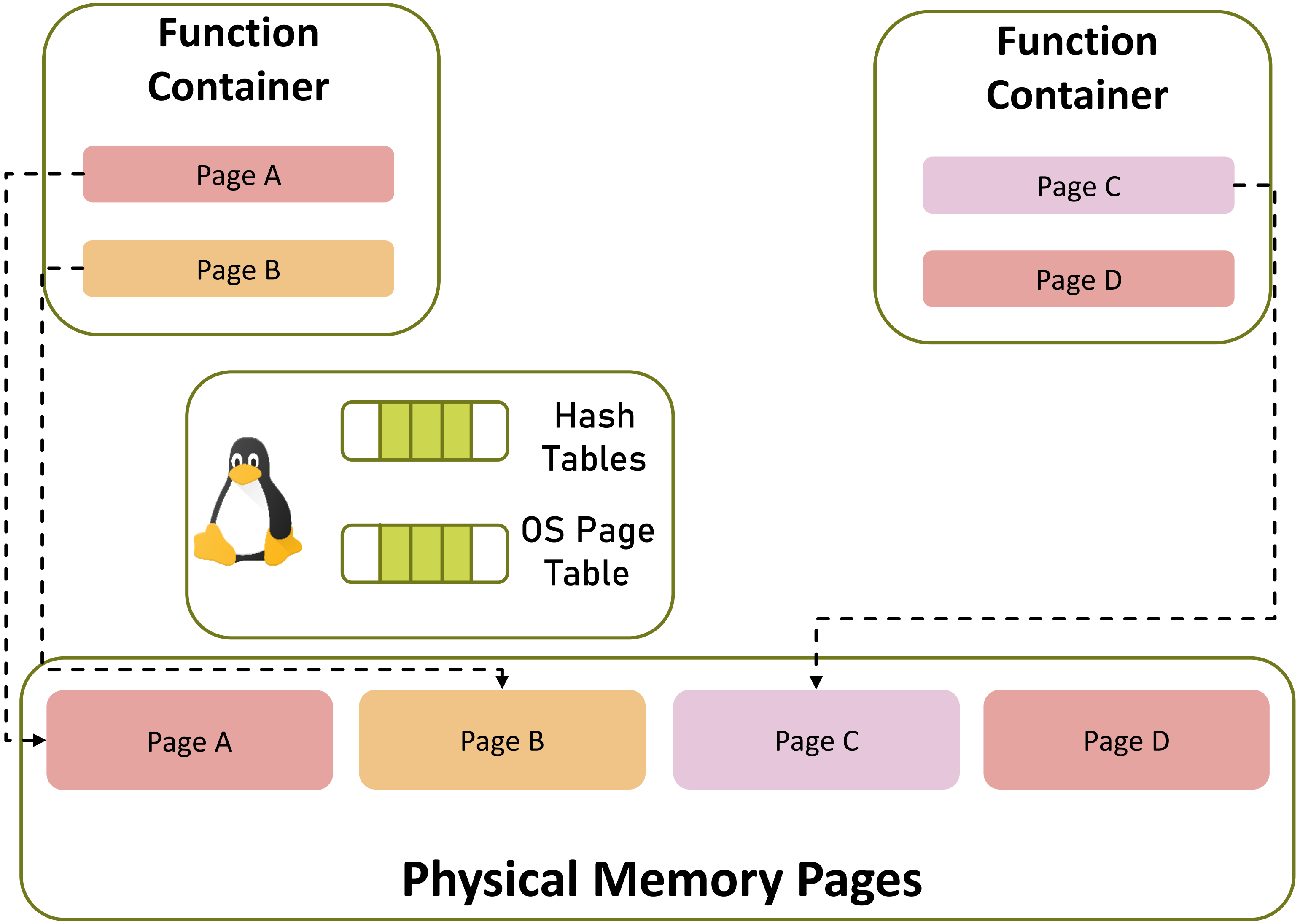

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging



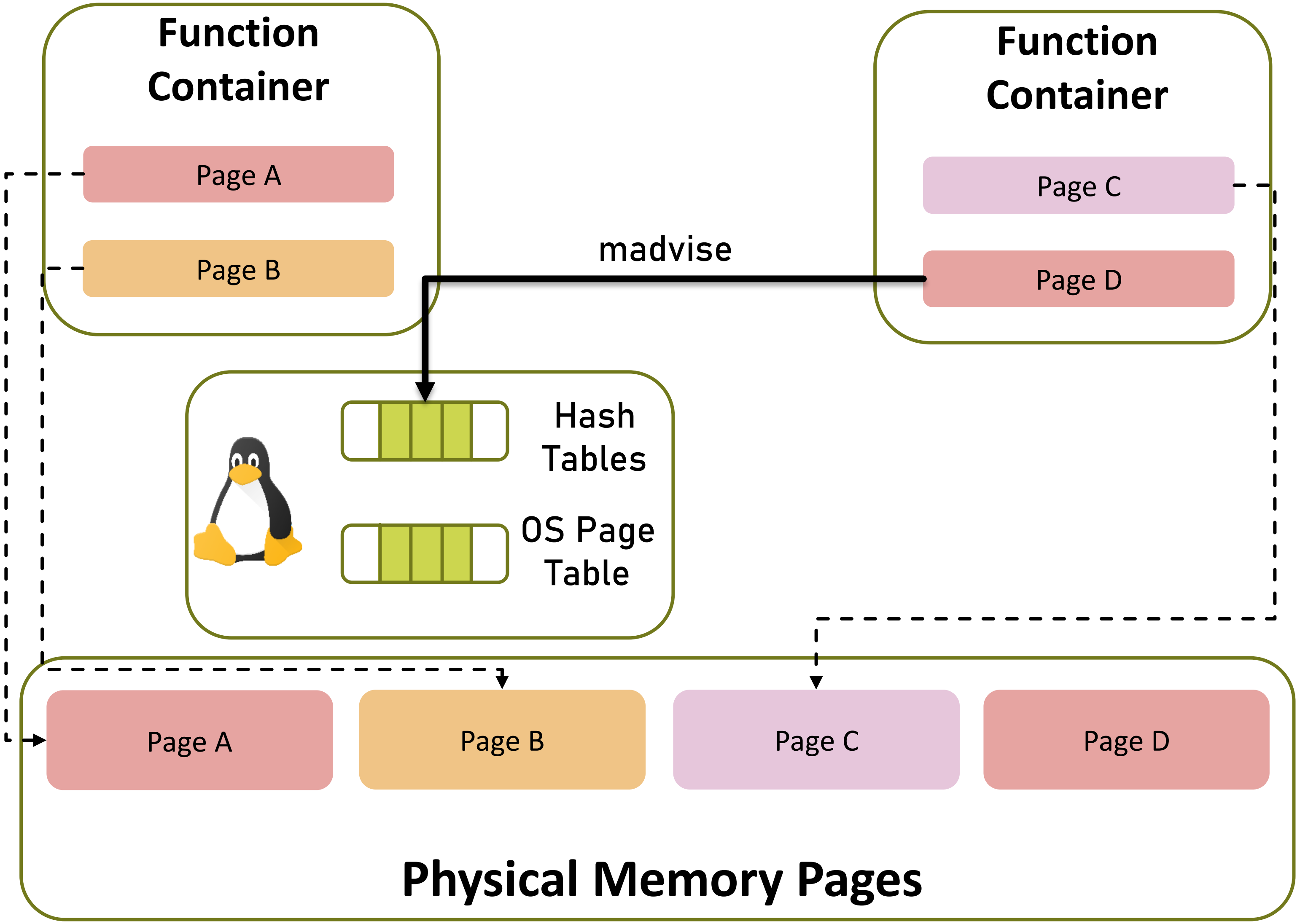

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging



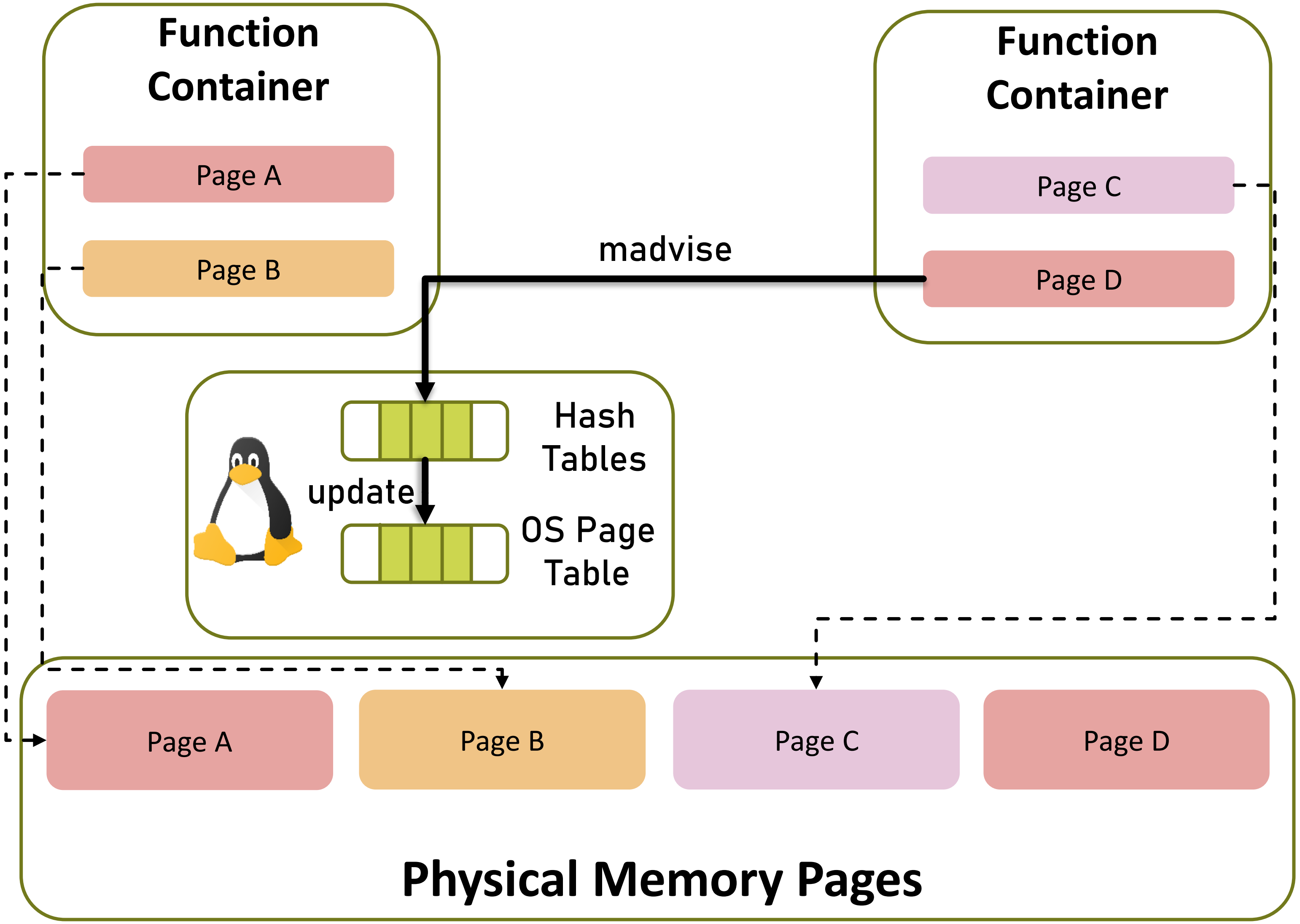

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging



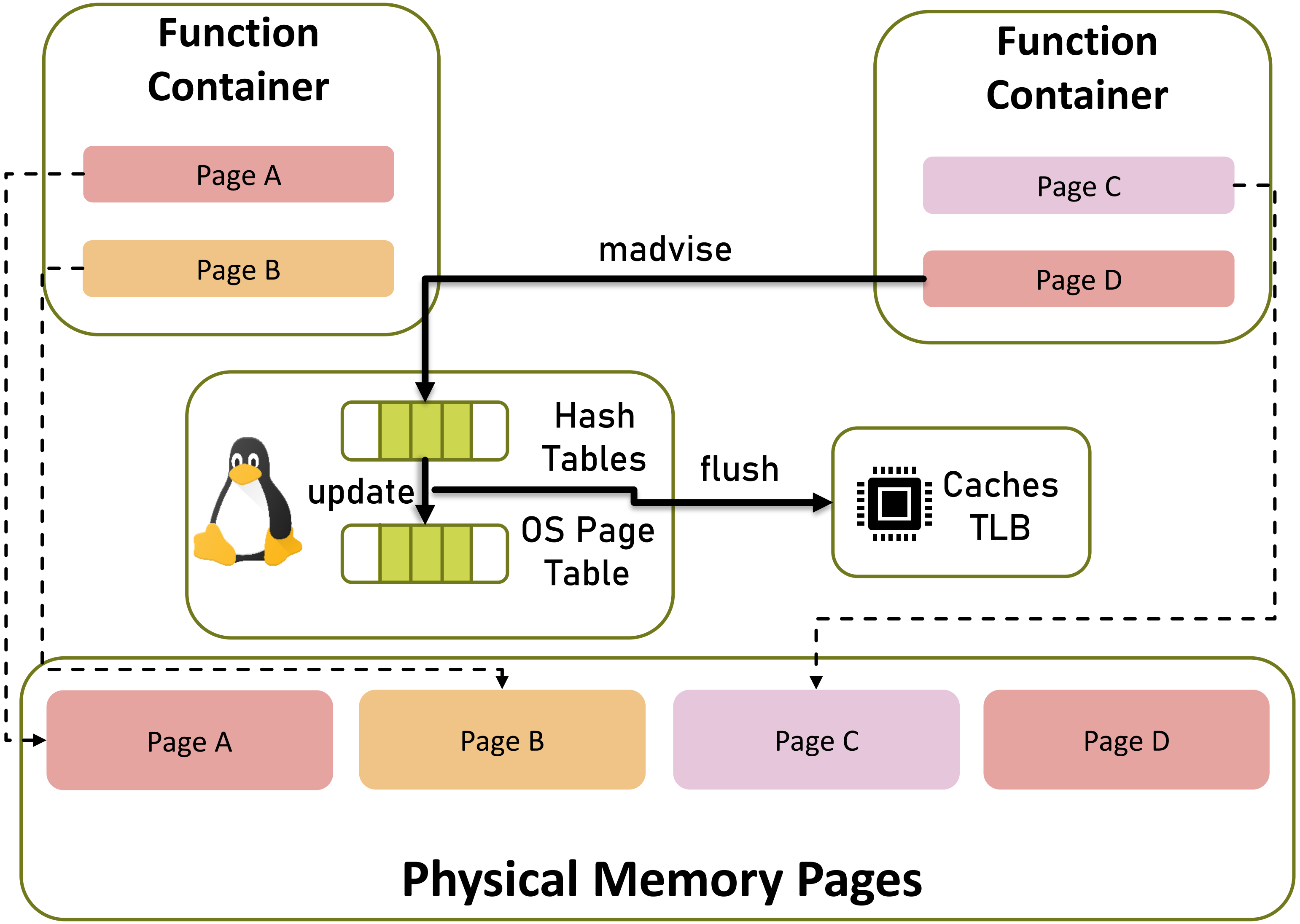

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging



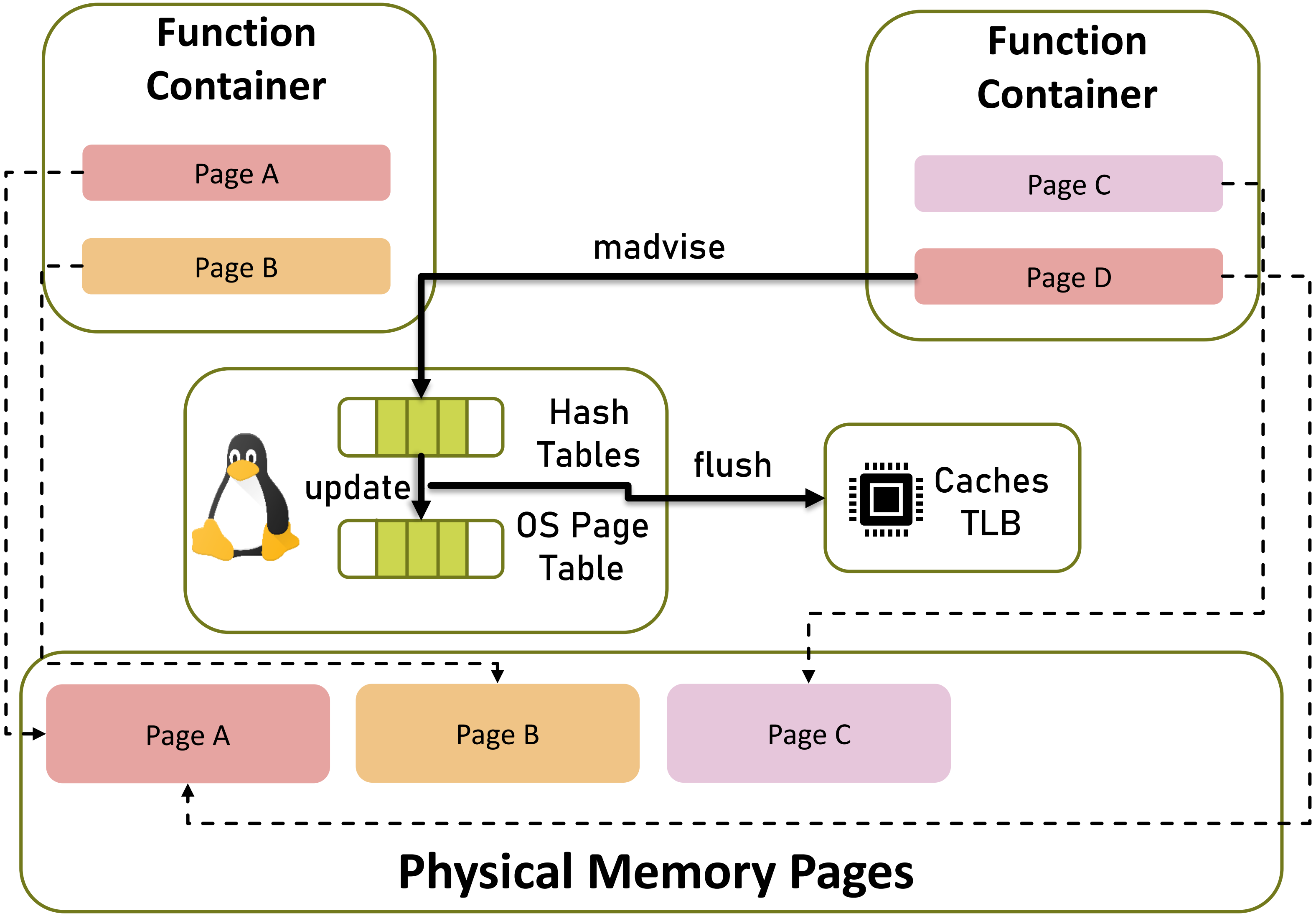

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging




```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM: User-Guided Page Merging



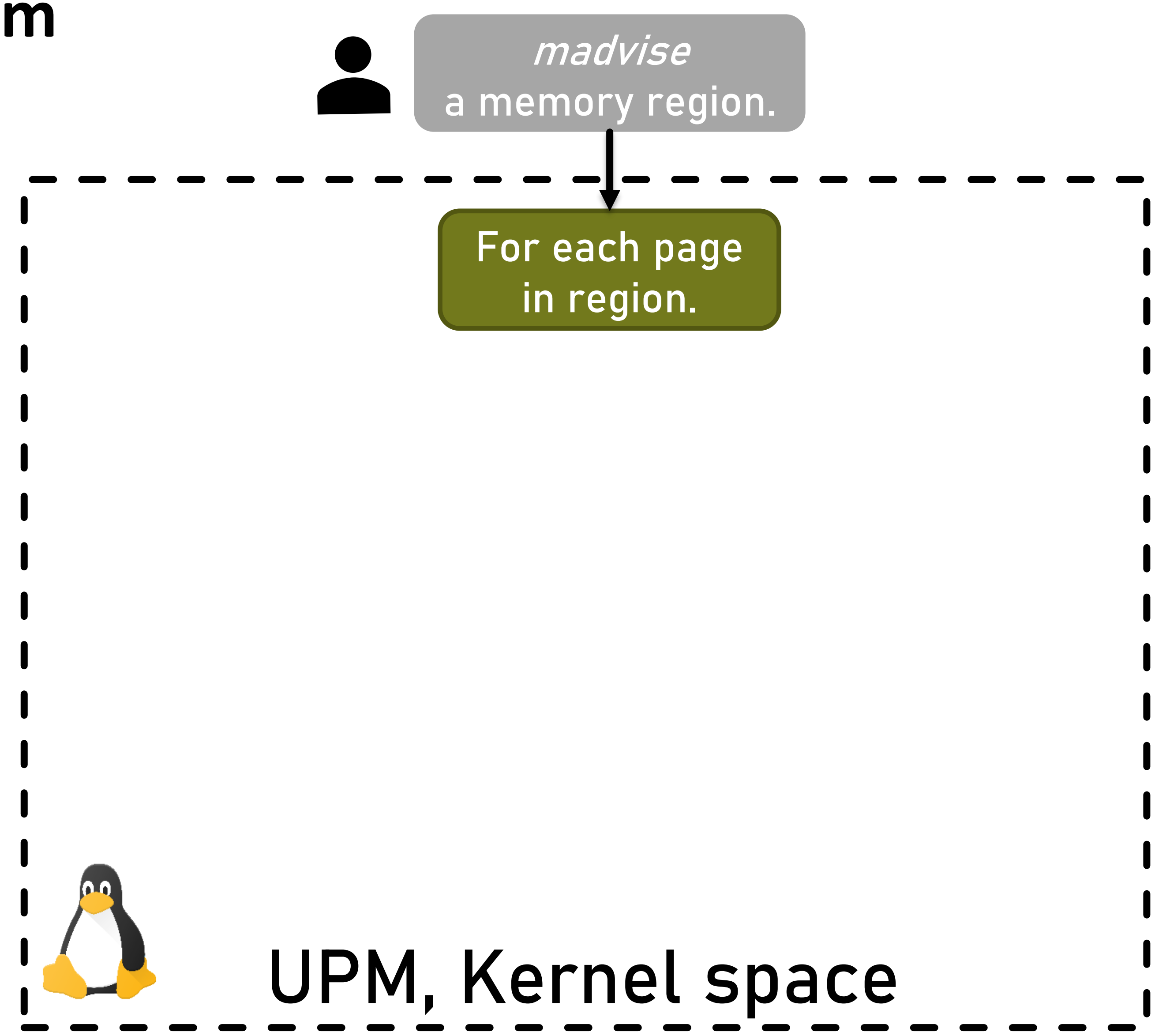

```
int madvise(
    void *addr, size_t length, int advice
);
```

UPM Algorithm

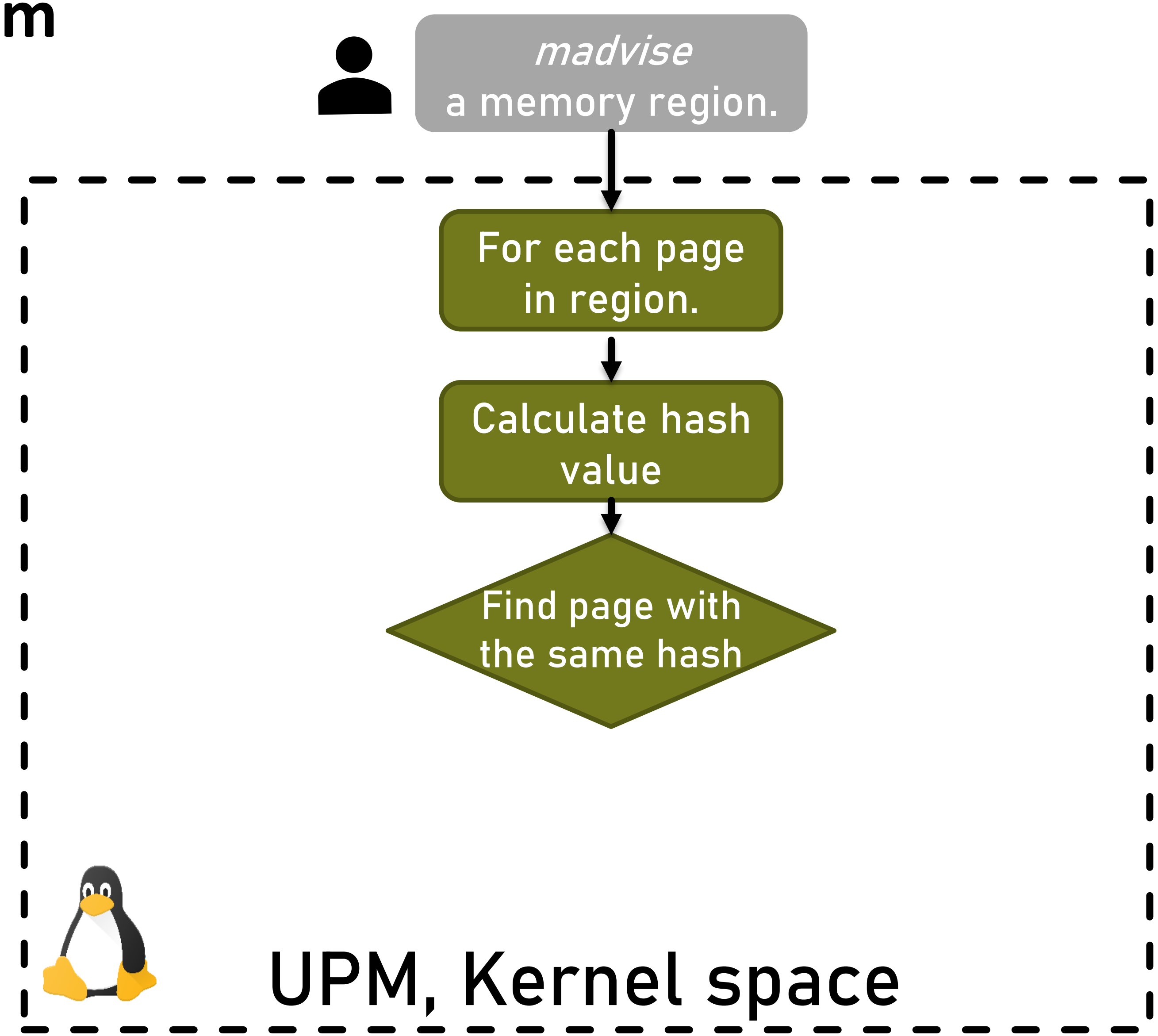
madvise
a memory region.

UPM, Kernel space

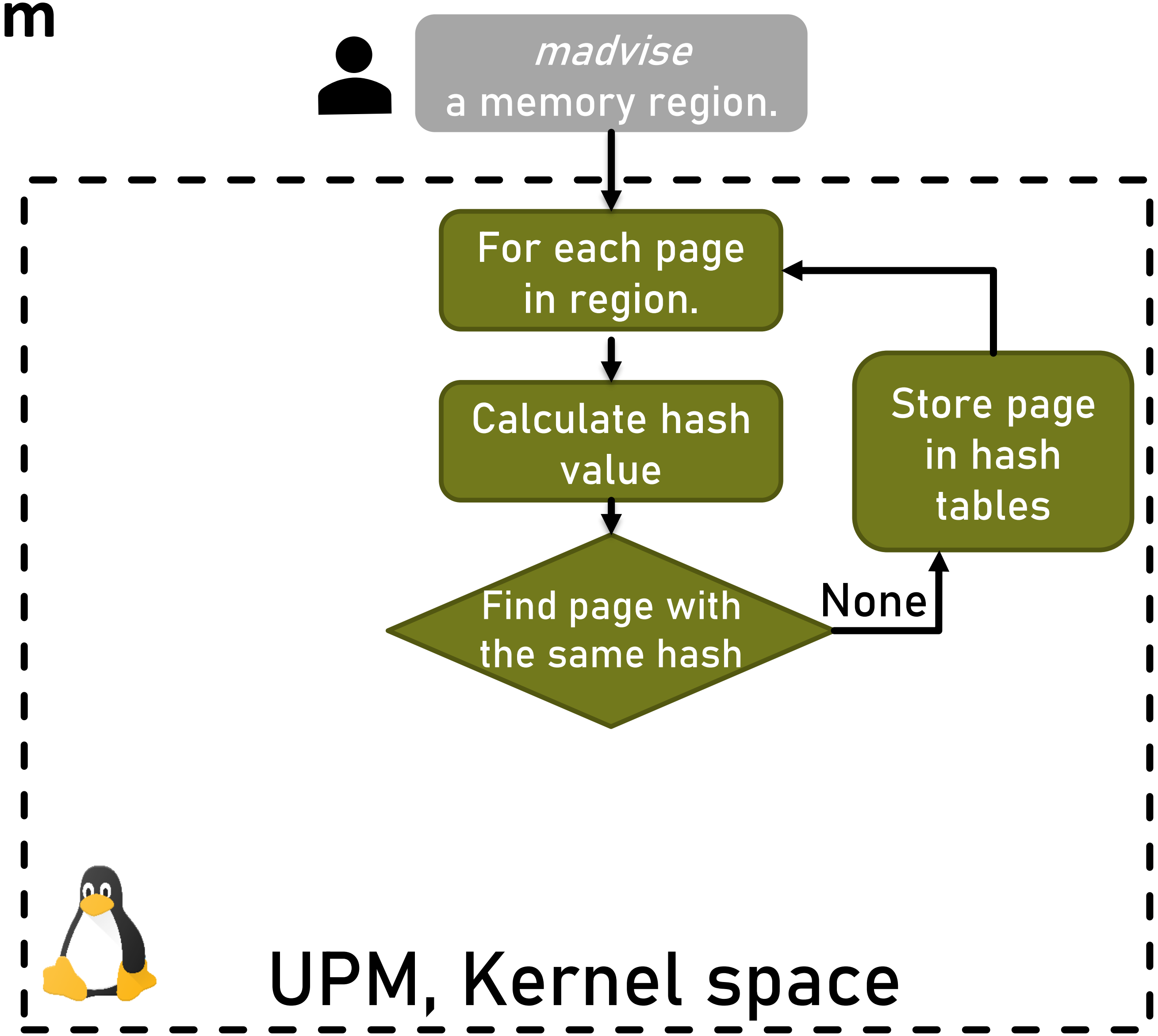
UPM Algorithm



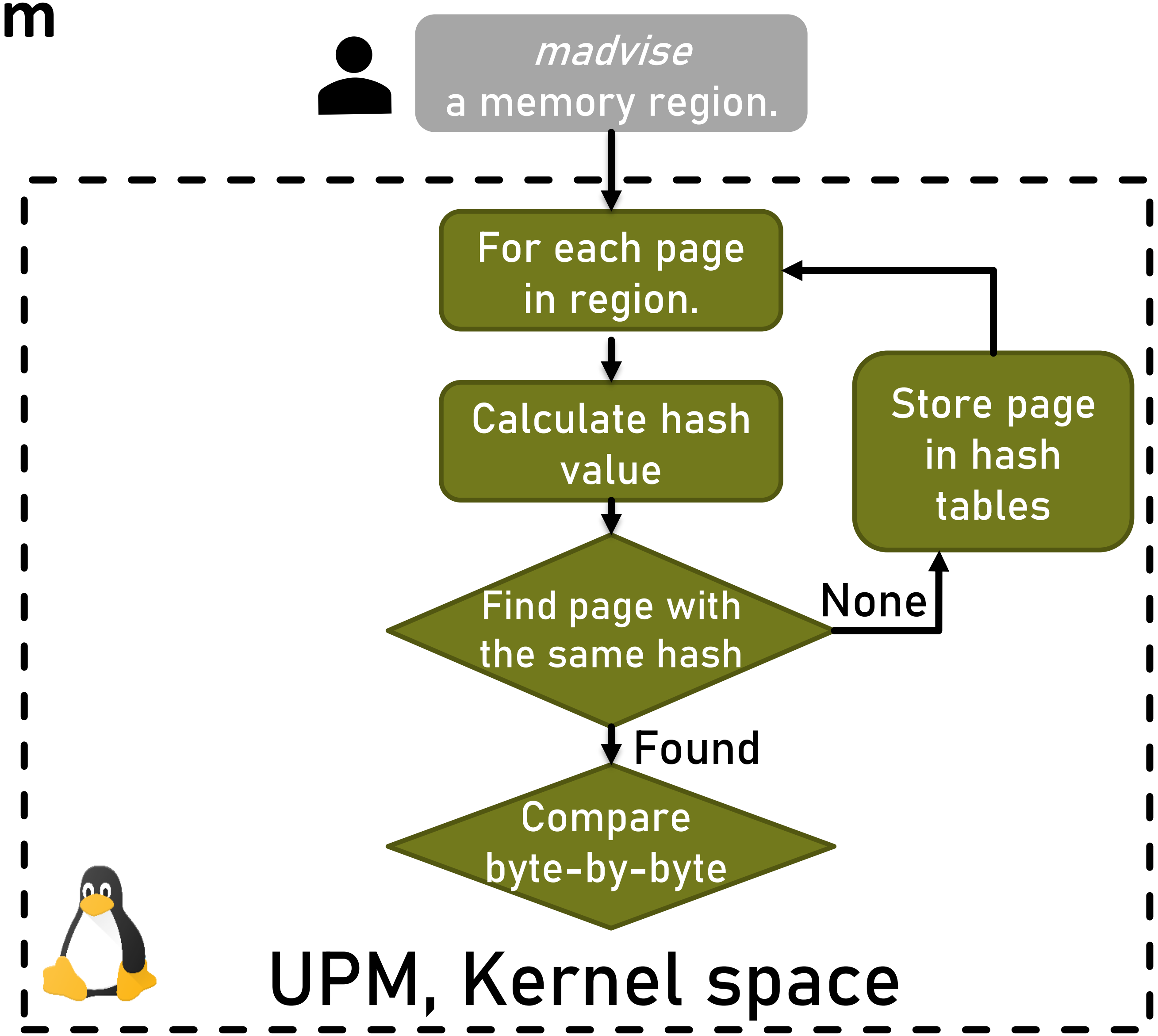
UPM Algorithm



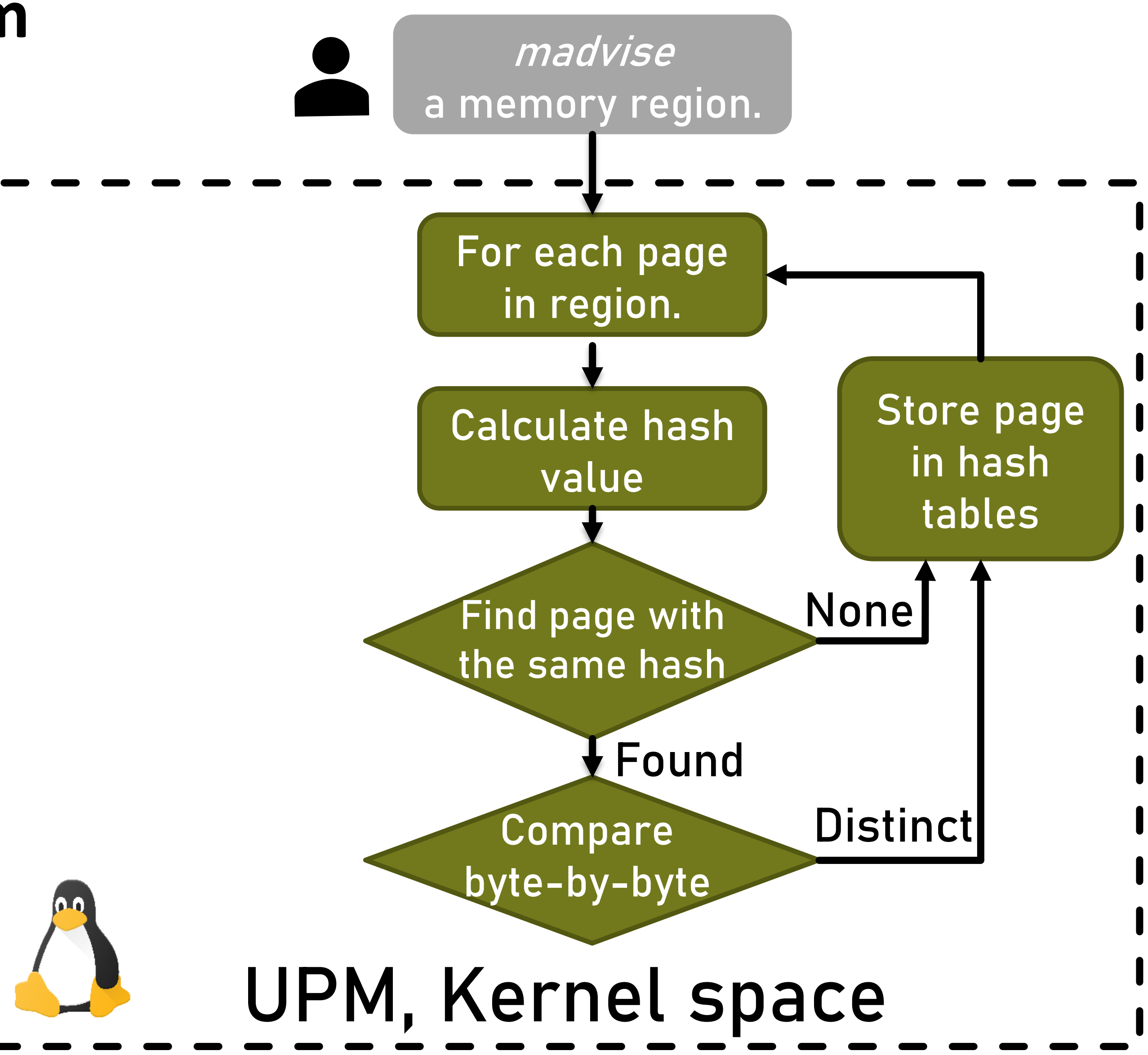
UPM Algorithm



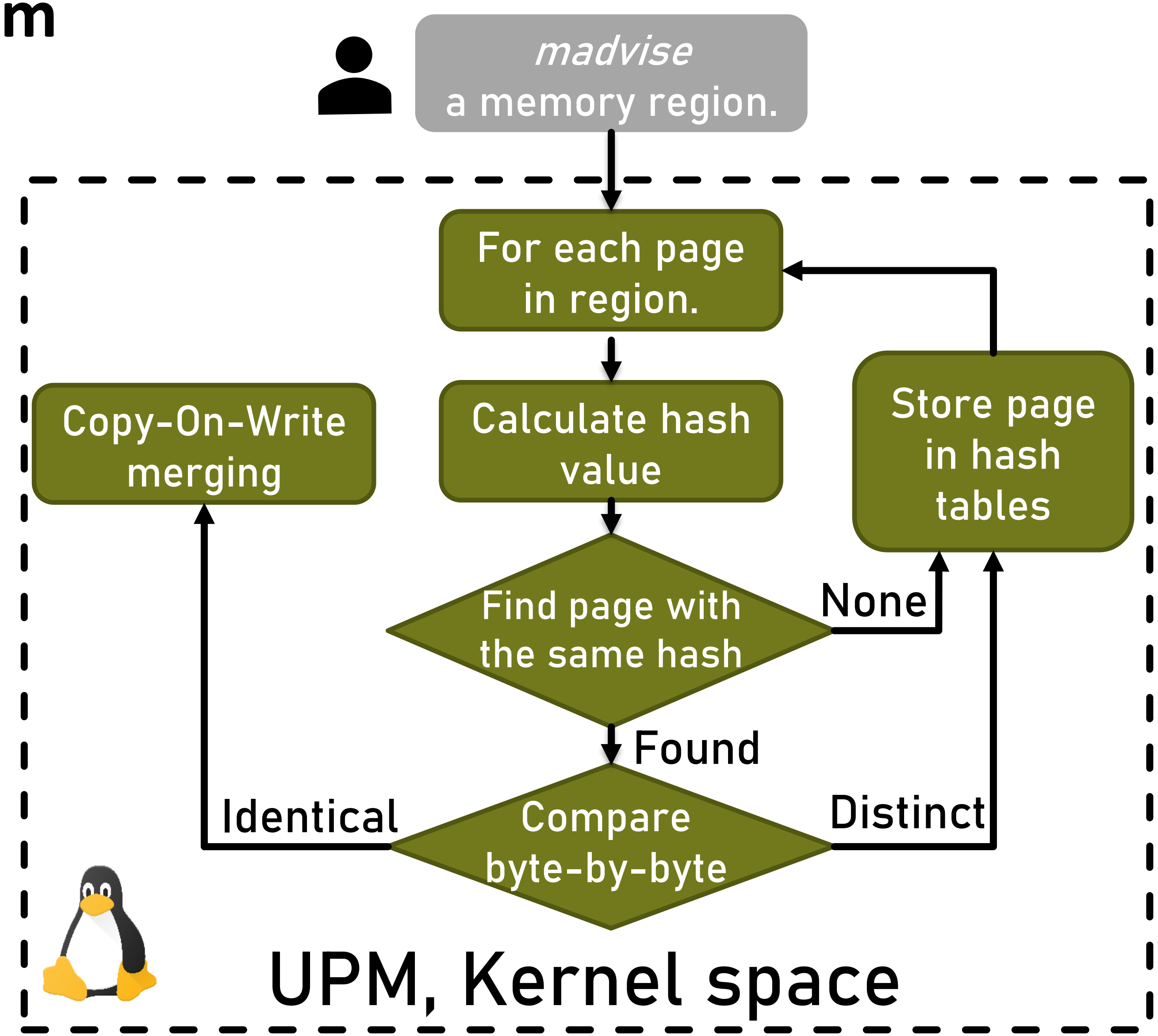
UPM Algorithm



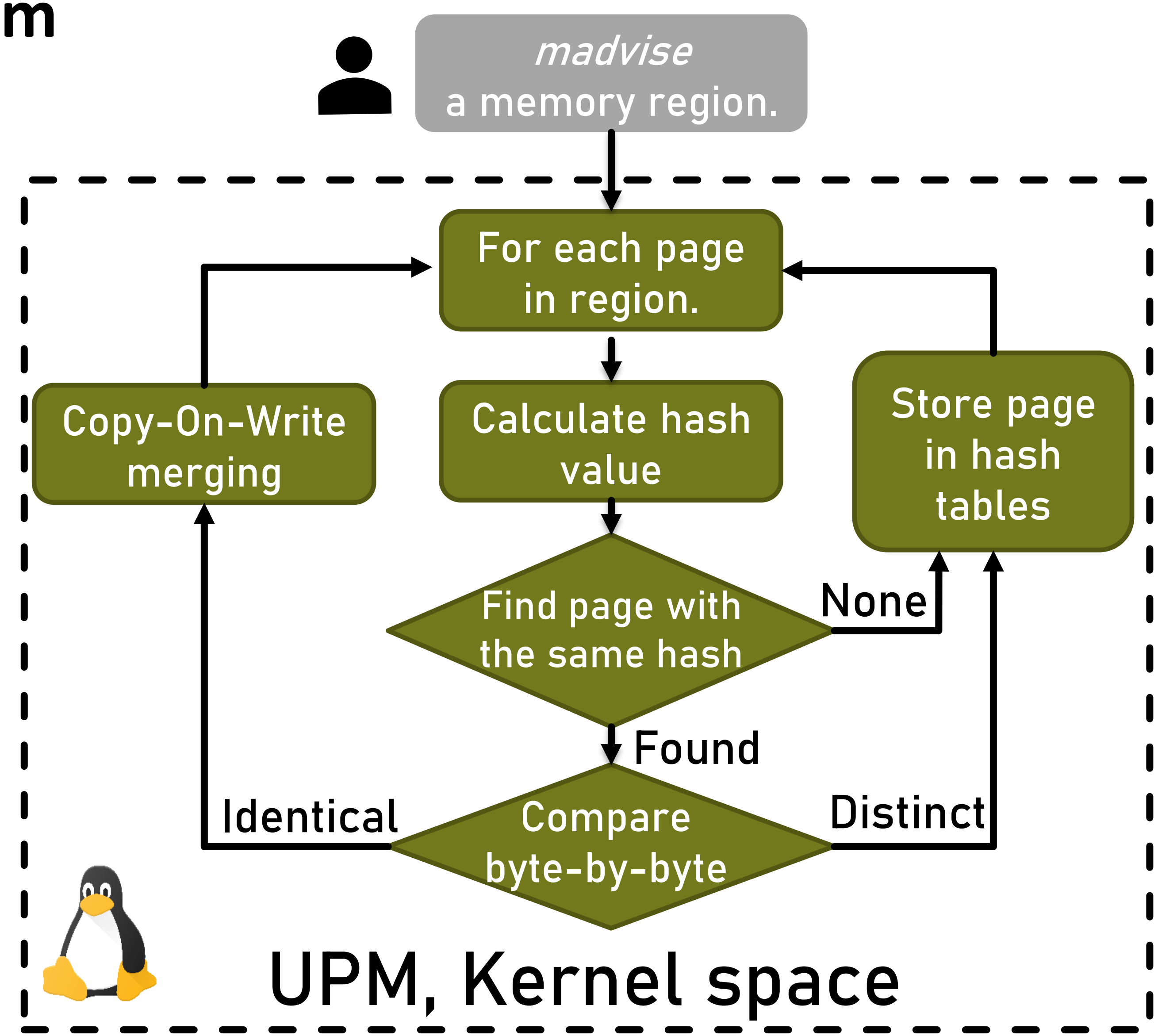
UPM Algorithm



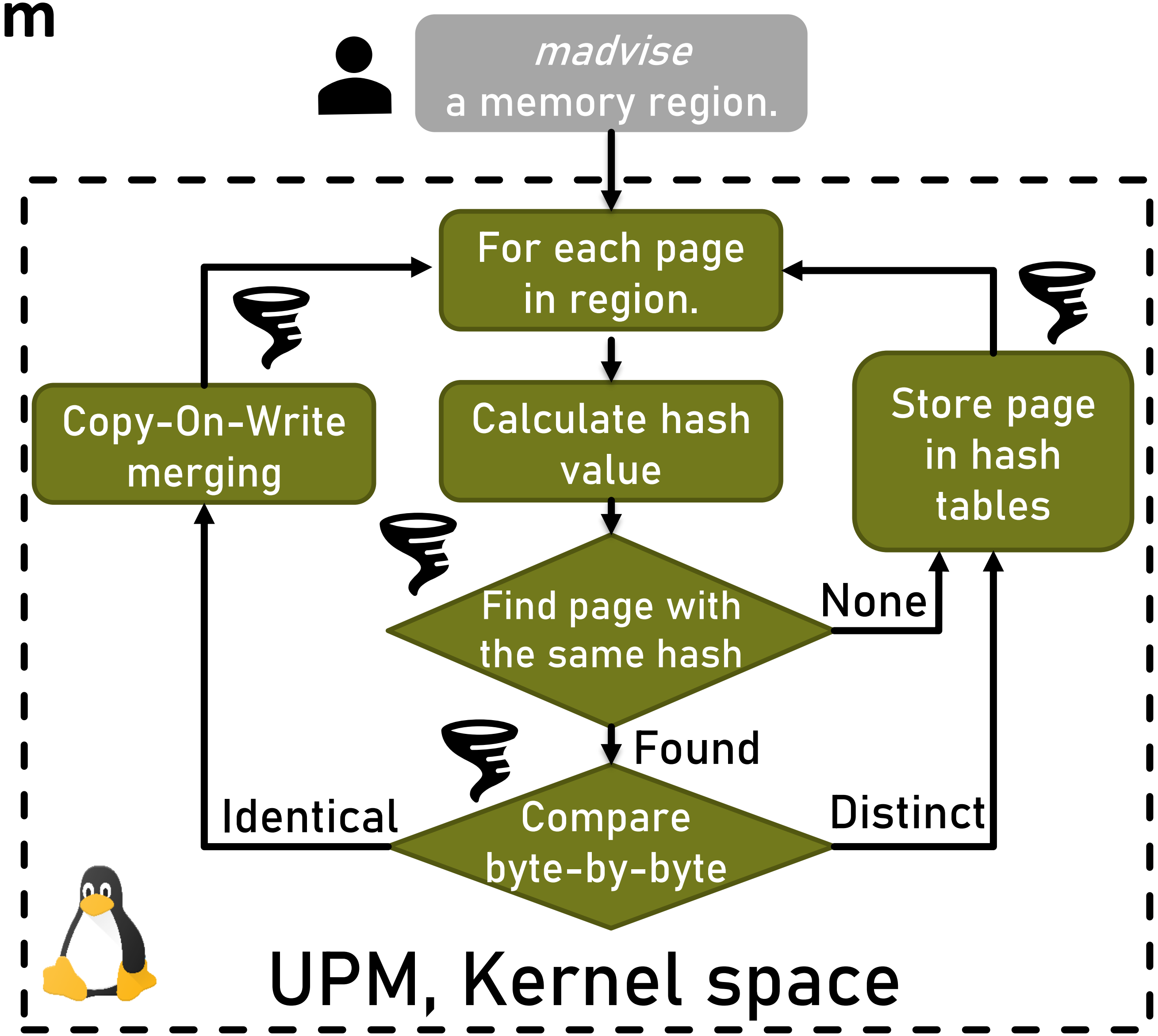
UPM Algorithm



UPM Algorithm



UPM Algorithm



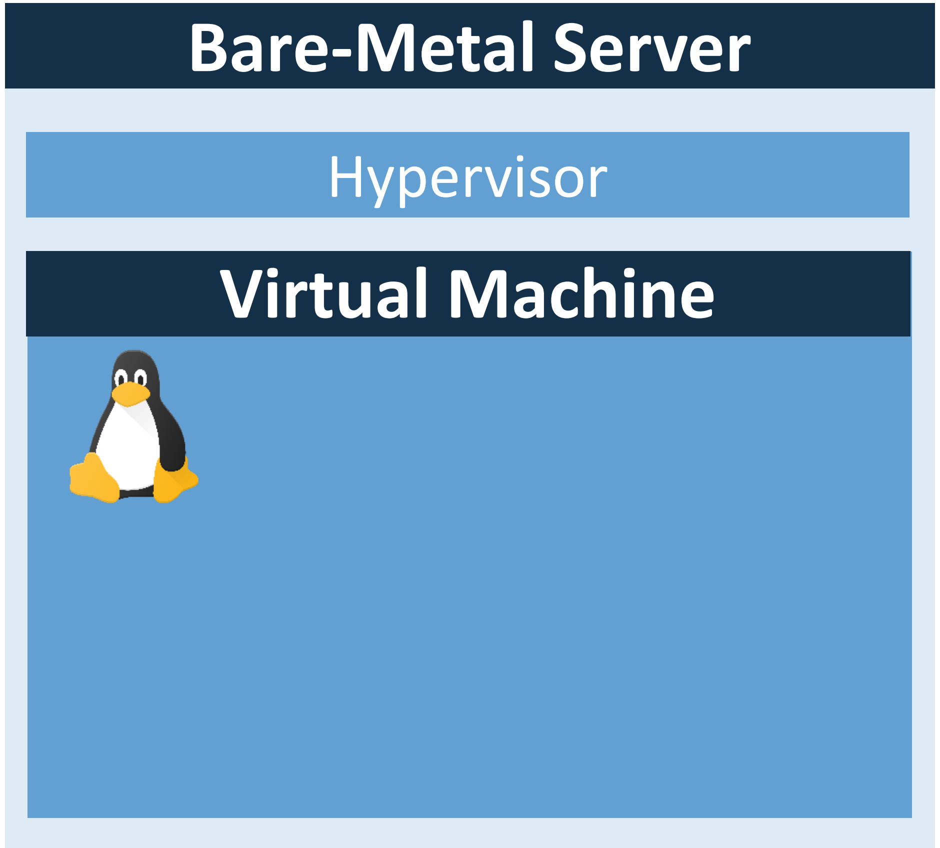
Implementation

- Modified Linux kernel 4.15.18
- UPM is a new built-in kernel module
- Reuse concepts from Kernel Samepage Merging (KSM)

Evaluation

Bare-Metal Server

Evaluation



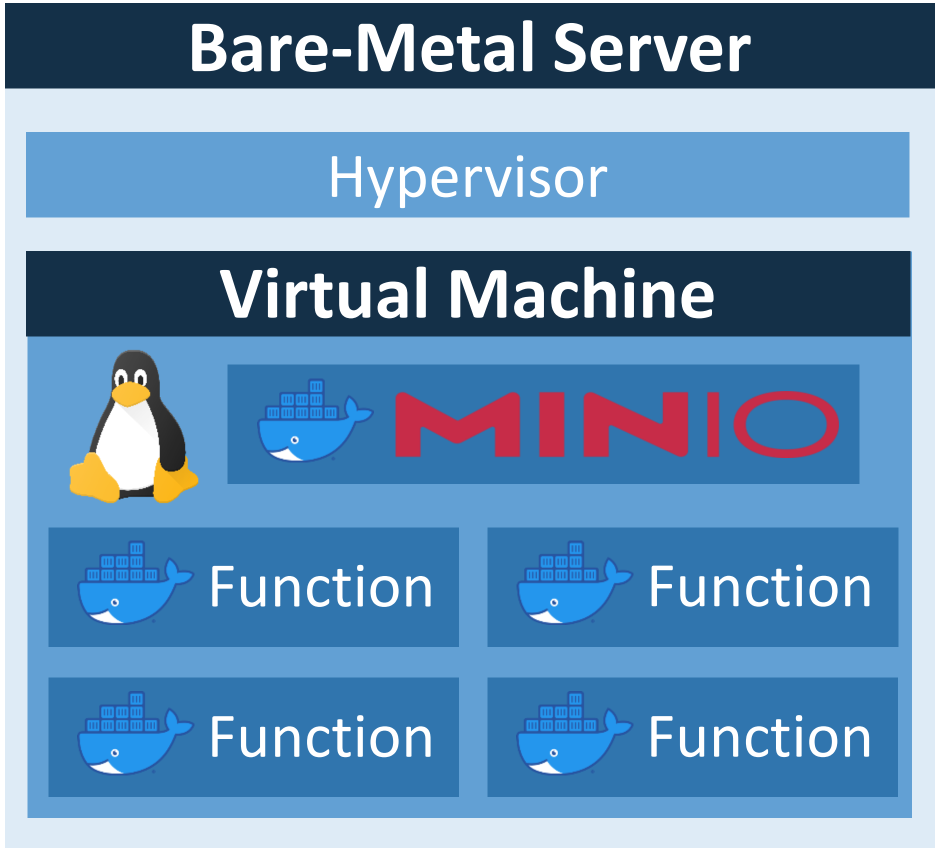
Evaluation

Bare-Metal Server

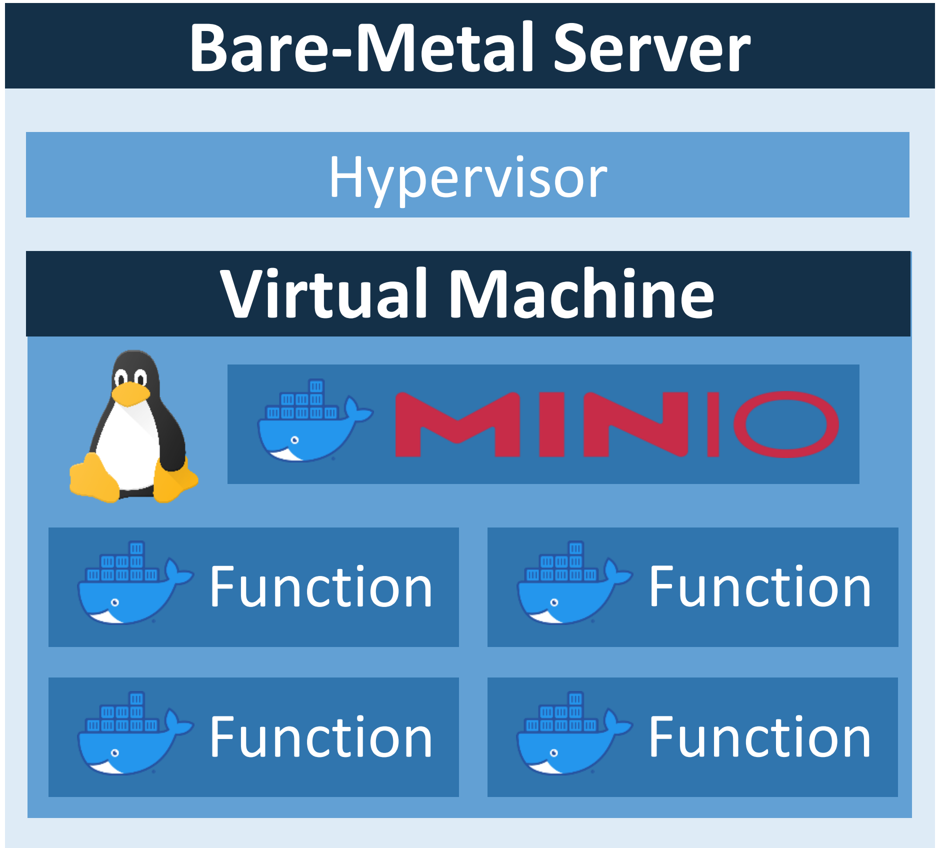
Hypervisor

Virtual Machine

Evaluation


Bare-Metal Server
Hypervisor
Virtual Machine
MINIO
Function
Function
Function
Function

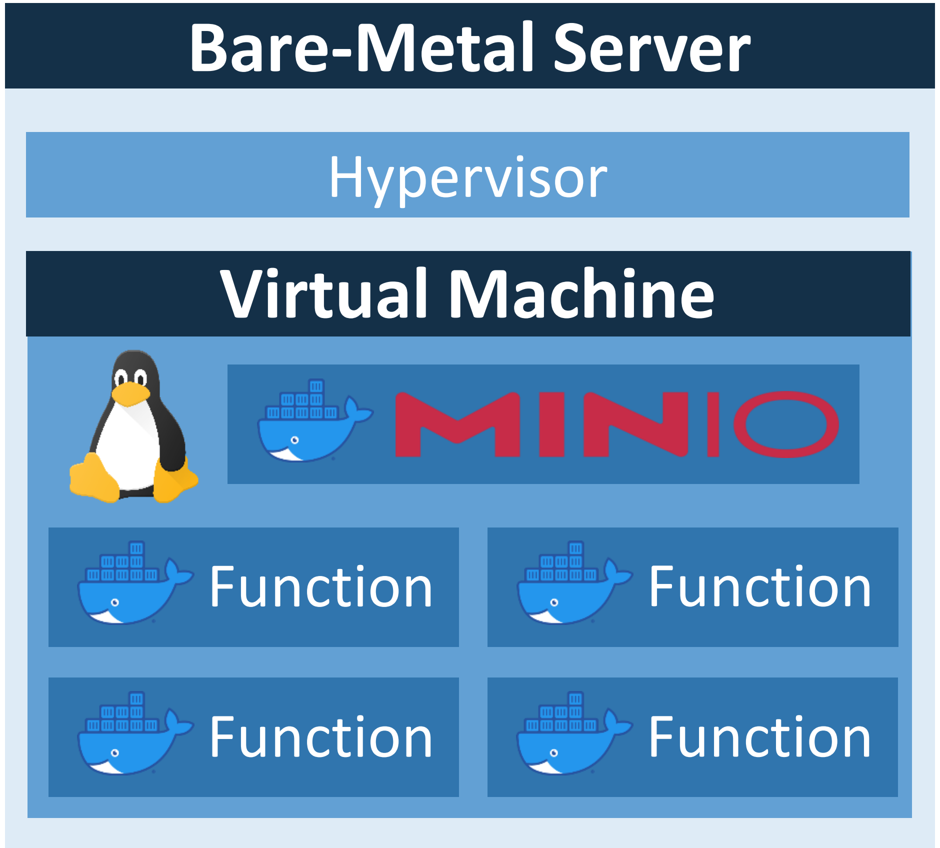
Evaluation



Does UPM decrease memory footprint of functions?

?

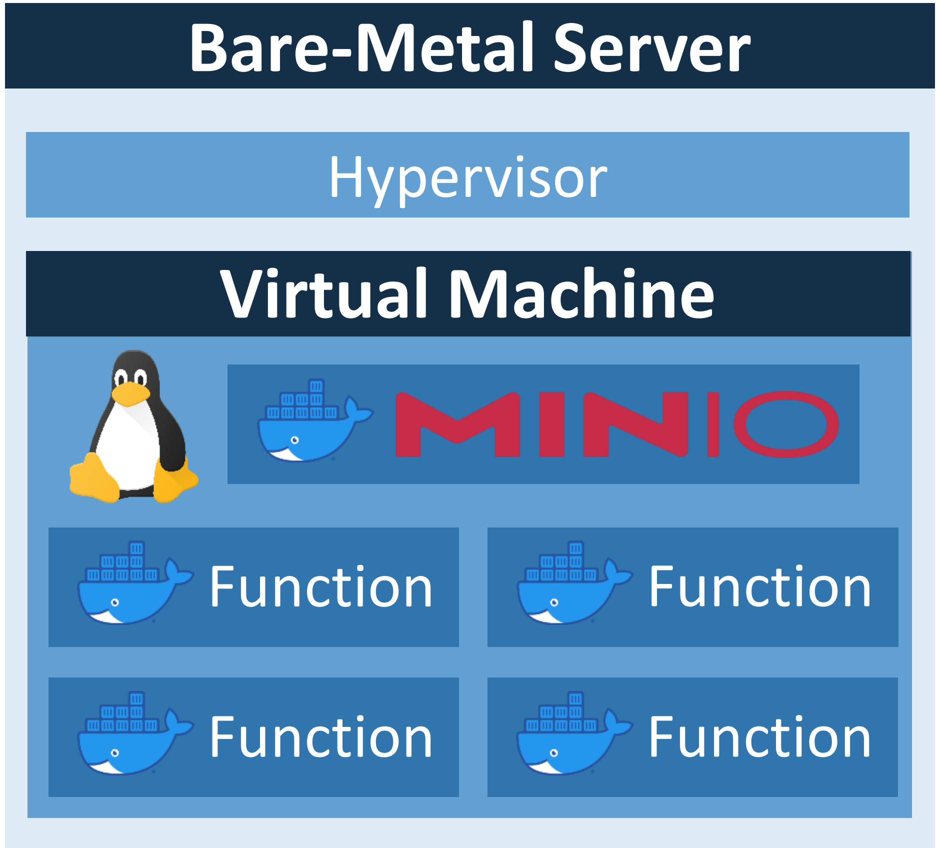
Evaluation



Does UPM decrease memory footprint of functions? ?

How much system memory can be saved? ?

Evaluation



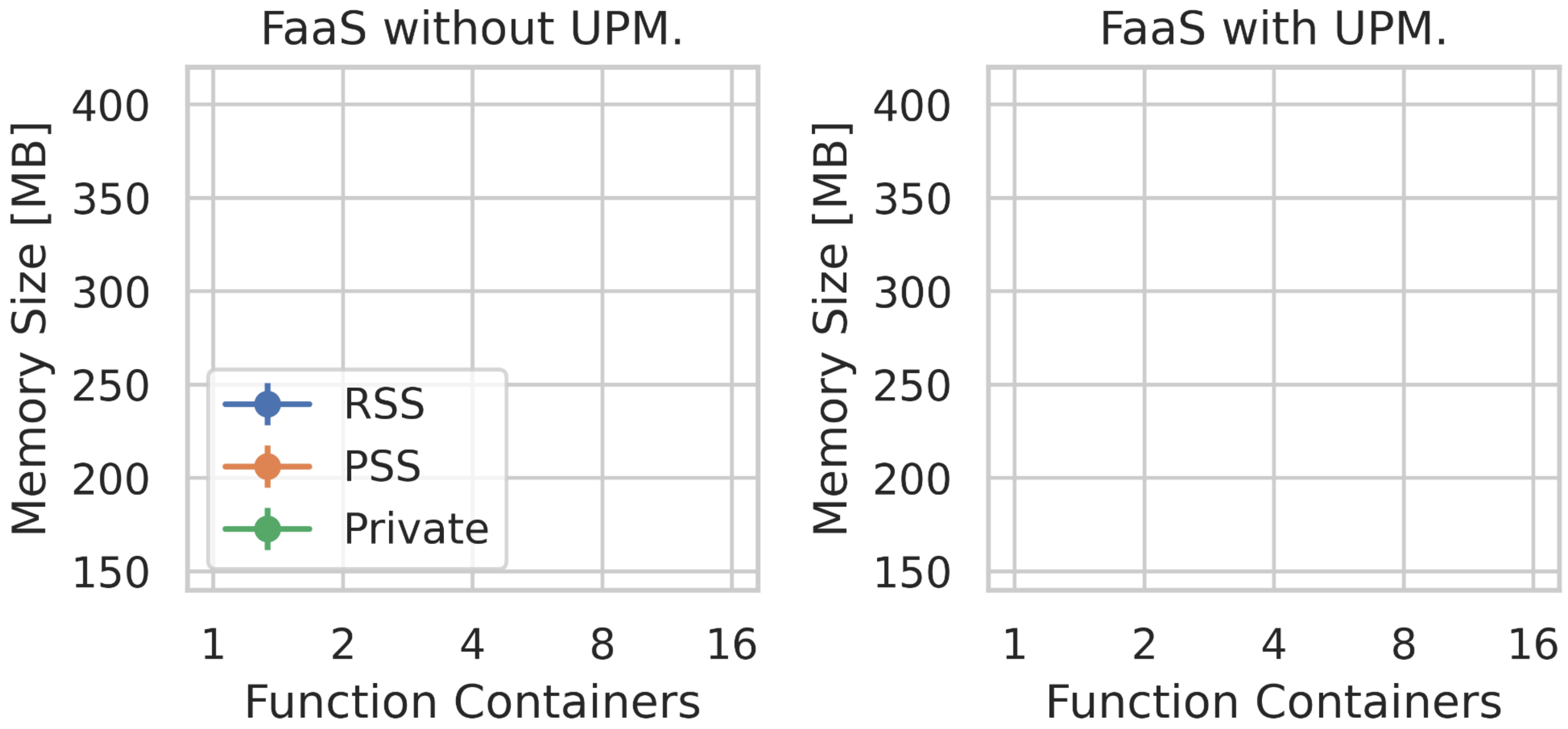
Does UPM decrease memory footprint of functions? ?

How much system memory can be saved? ?

How much overhead does UPM add to serverless? ?

Function Memory Footprint – ResNet 50

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

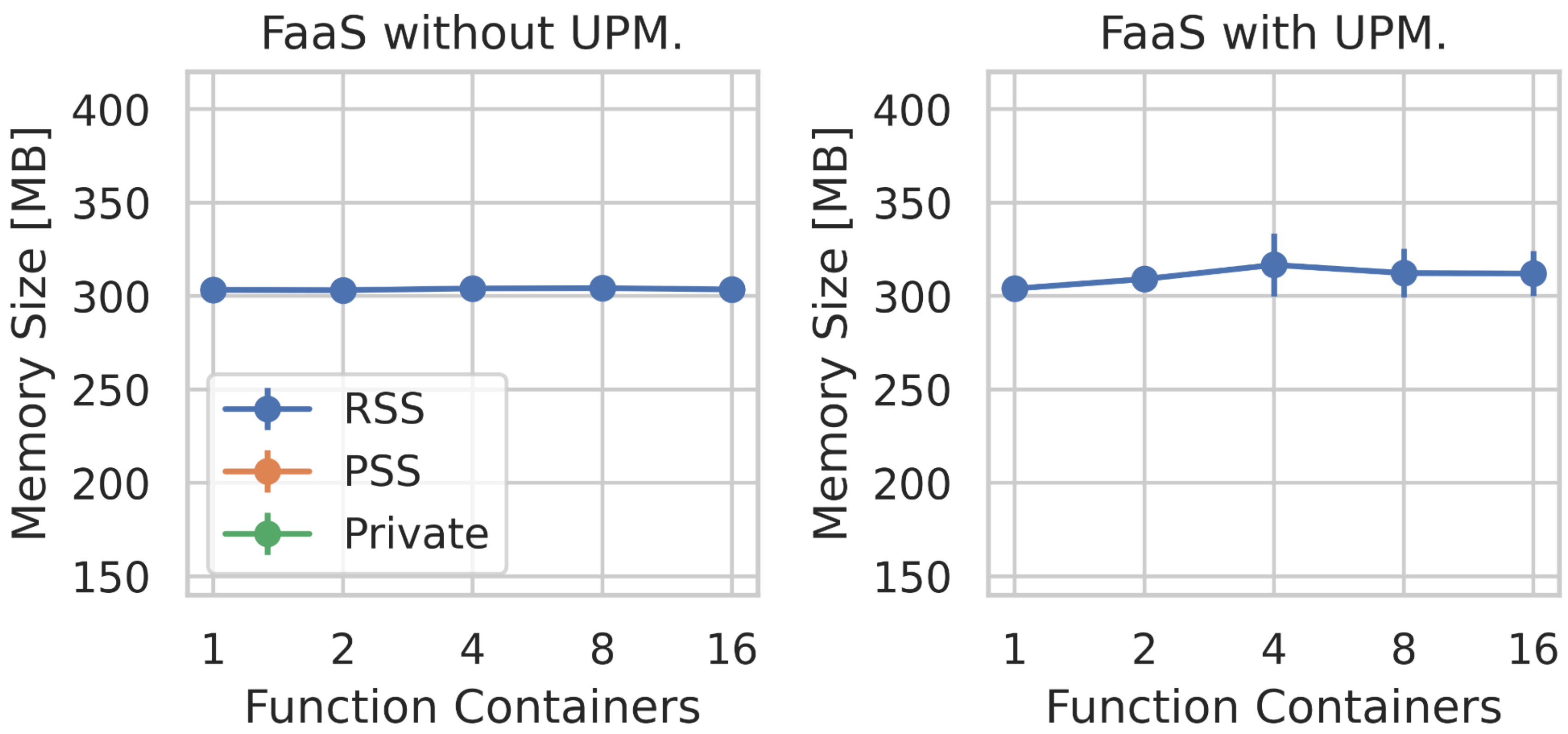


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – ResNet 50

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

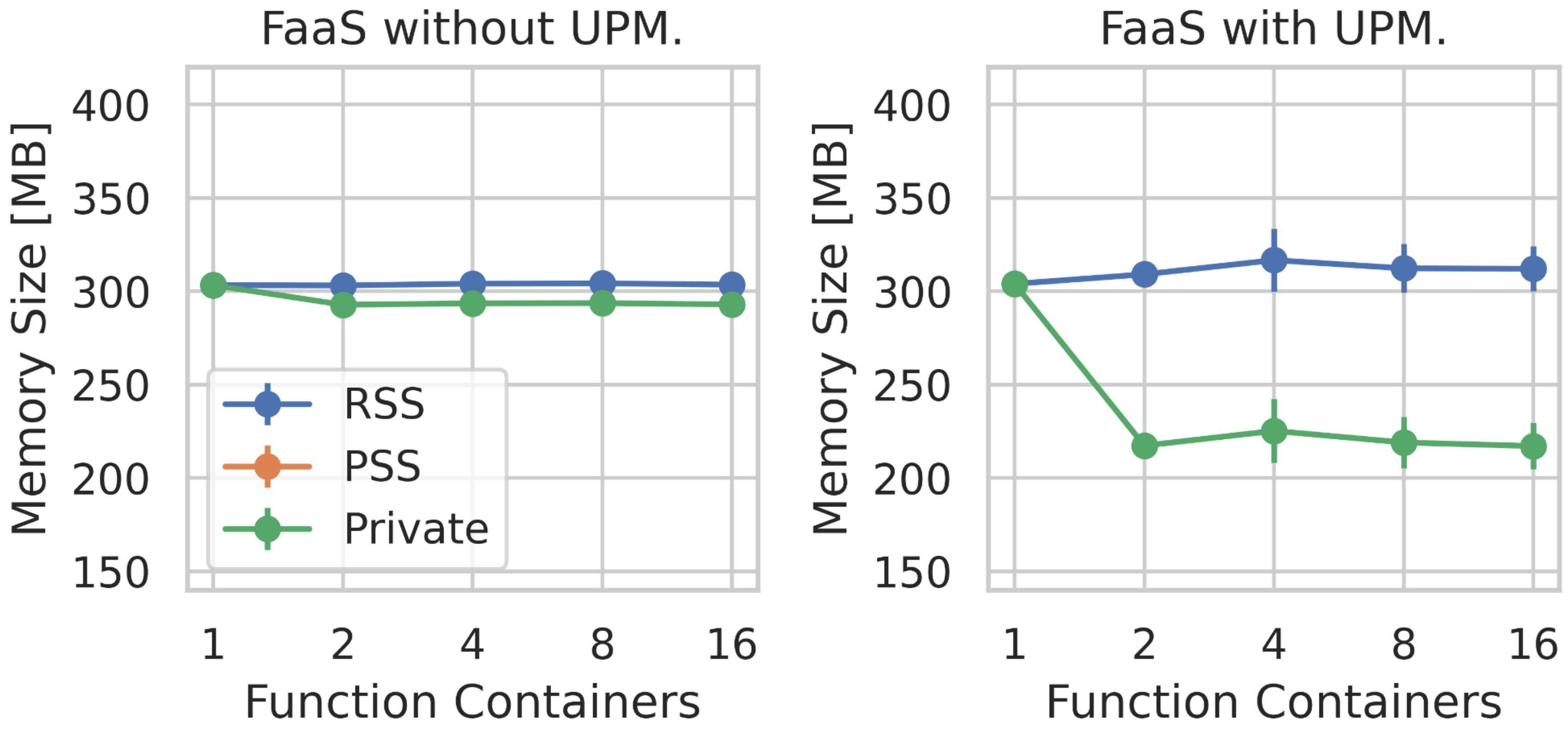


$$RSS = Private + Shared$$

$$PSS = Private + \frac{Shared}{\#Processes}$$

Function Memory Footprint – ResNet 50

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

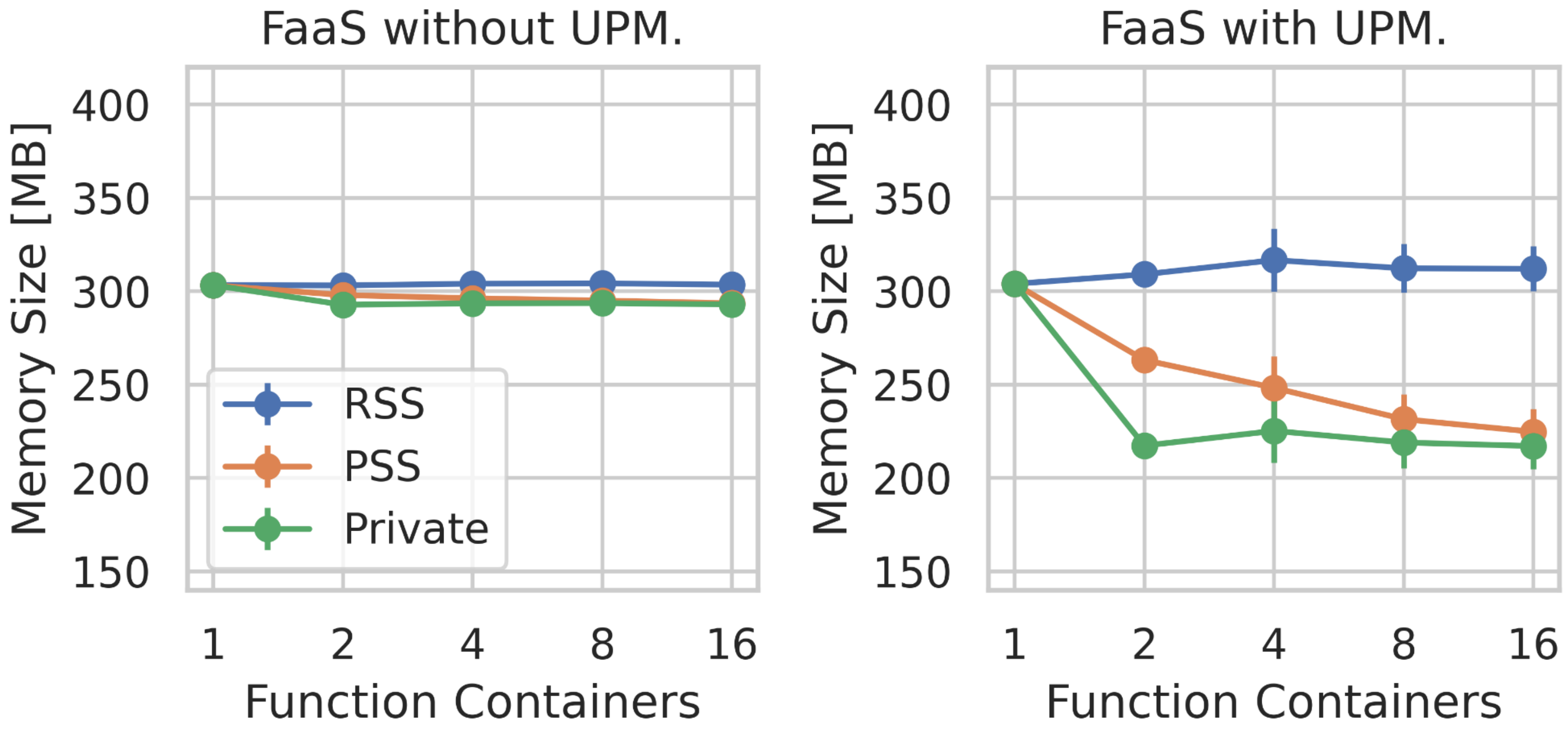


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – ResNet 50

4x Intel Xeon X7550 @ 2.00GHz,
64 cores total. 1 TB memory.

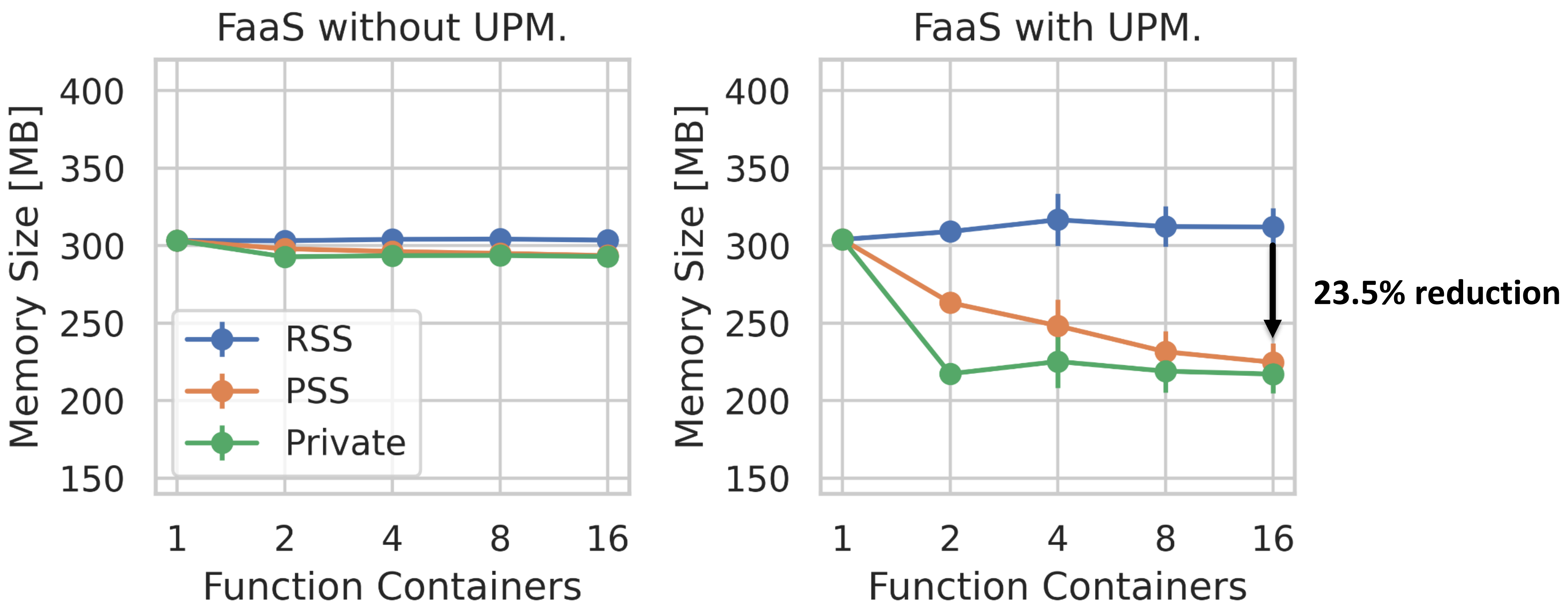


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – ResNet 50

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

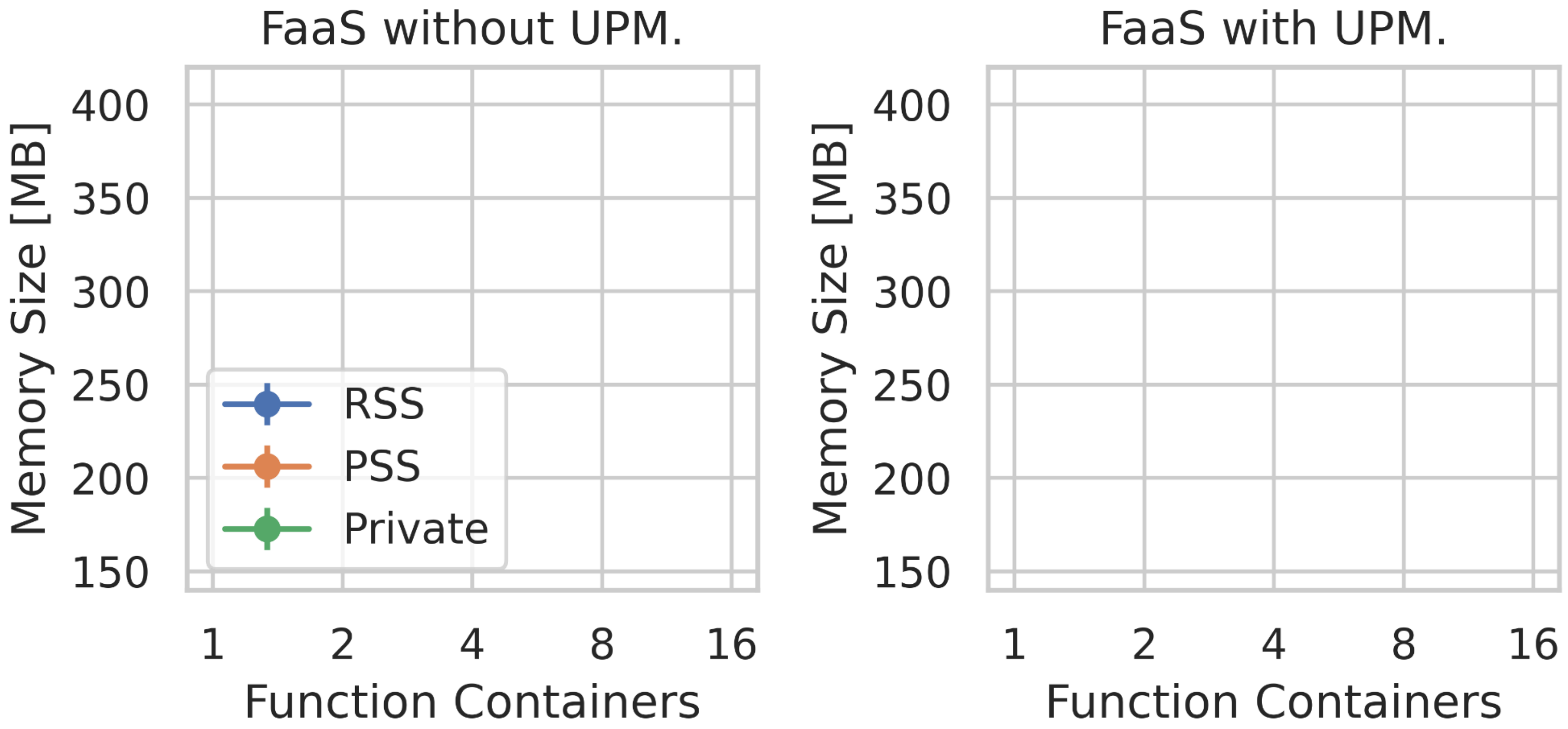


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – AlexNet

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

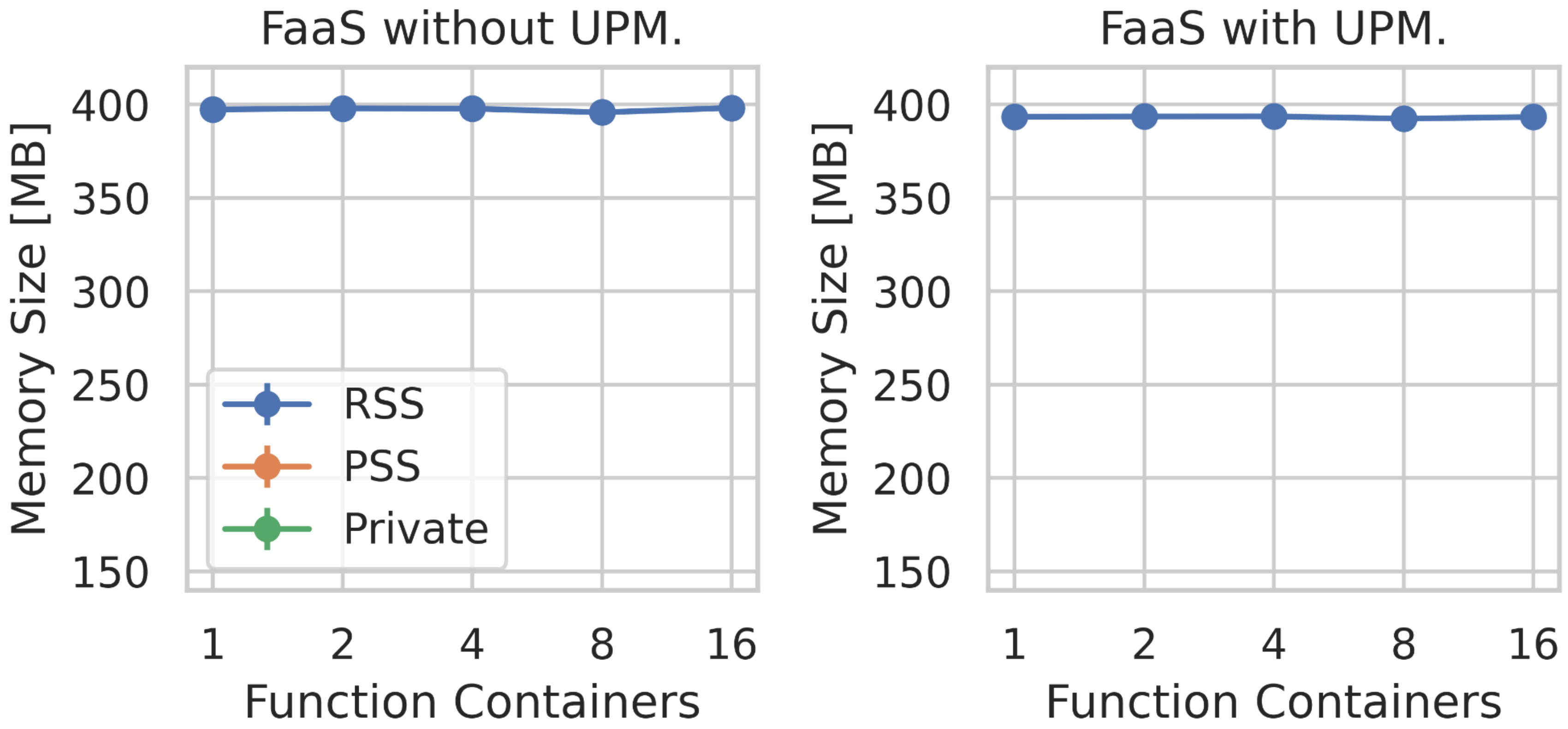


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – AlexNet

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

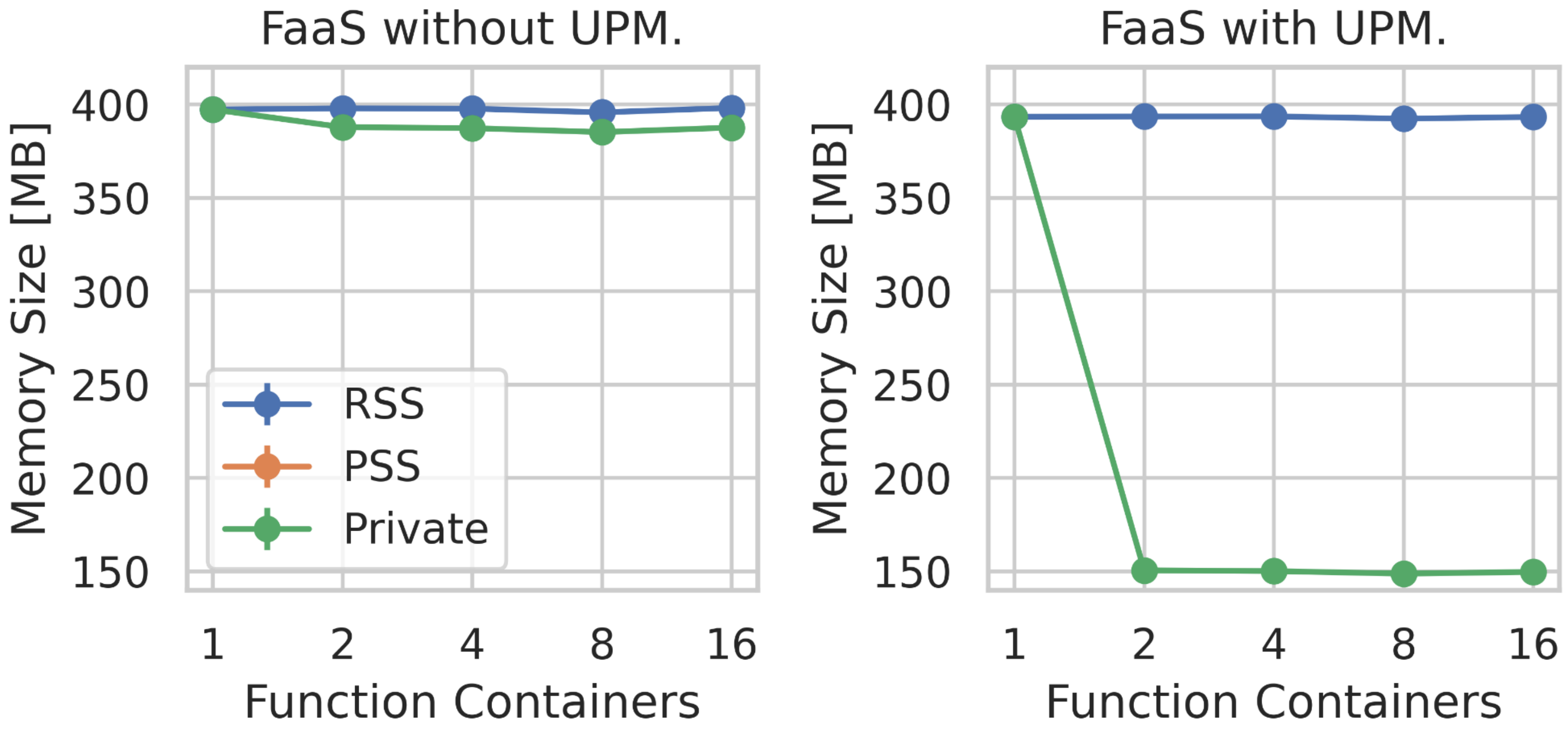


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – AlexNet

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

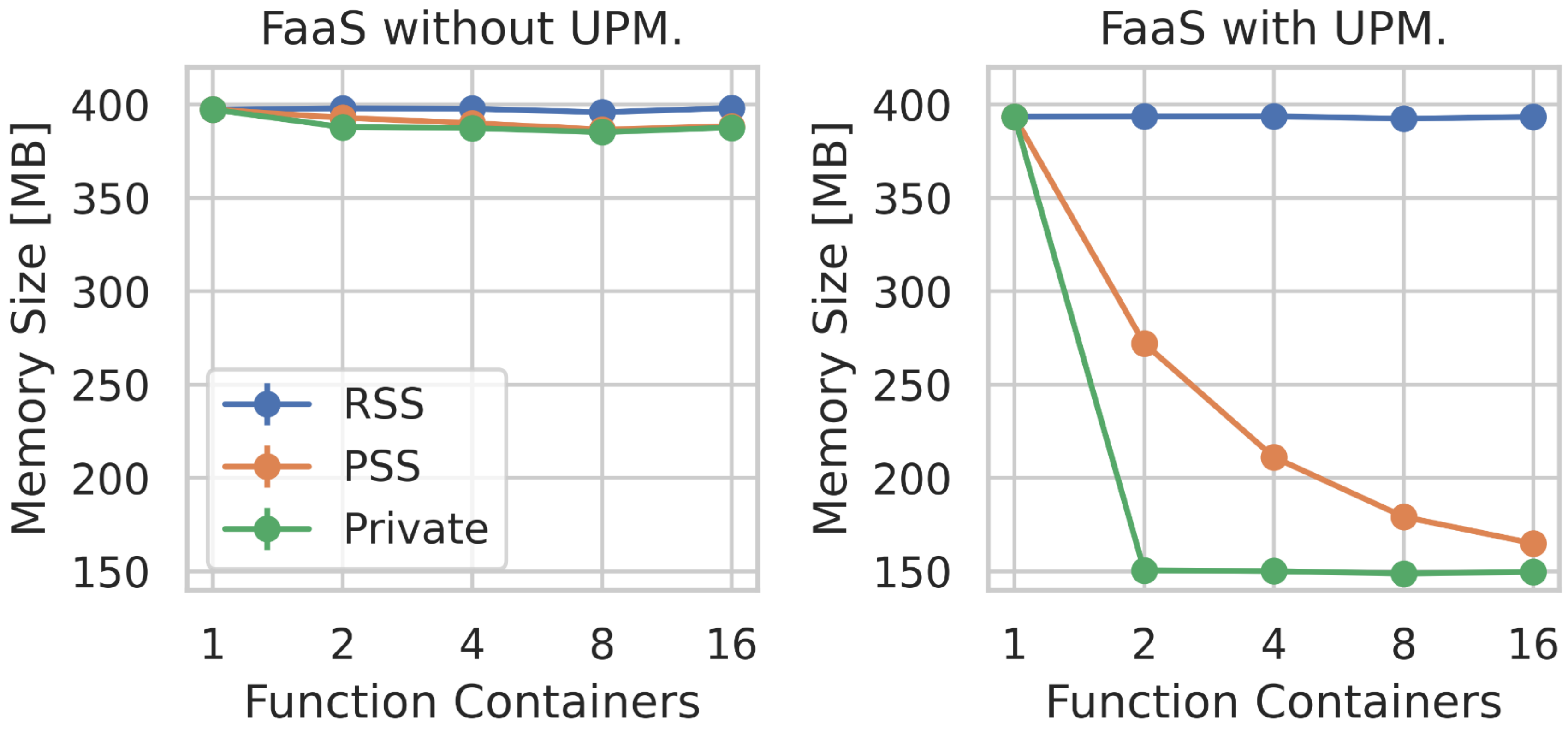


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – AlexNet

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

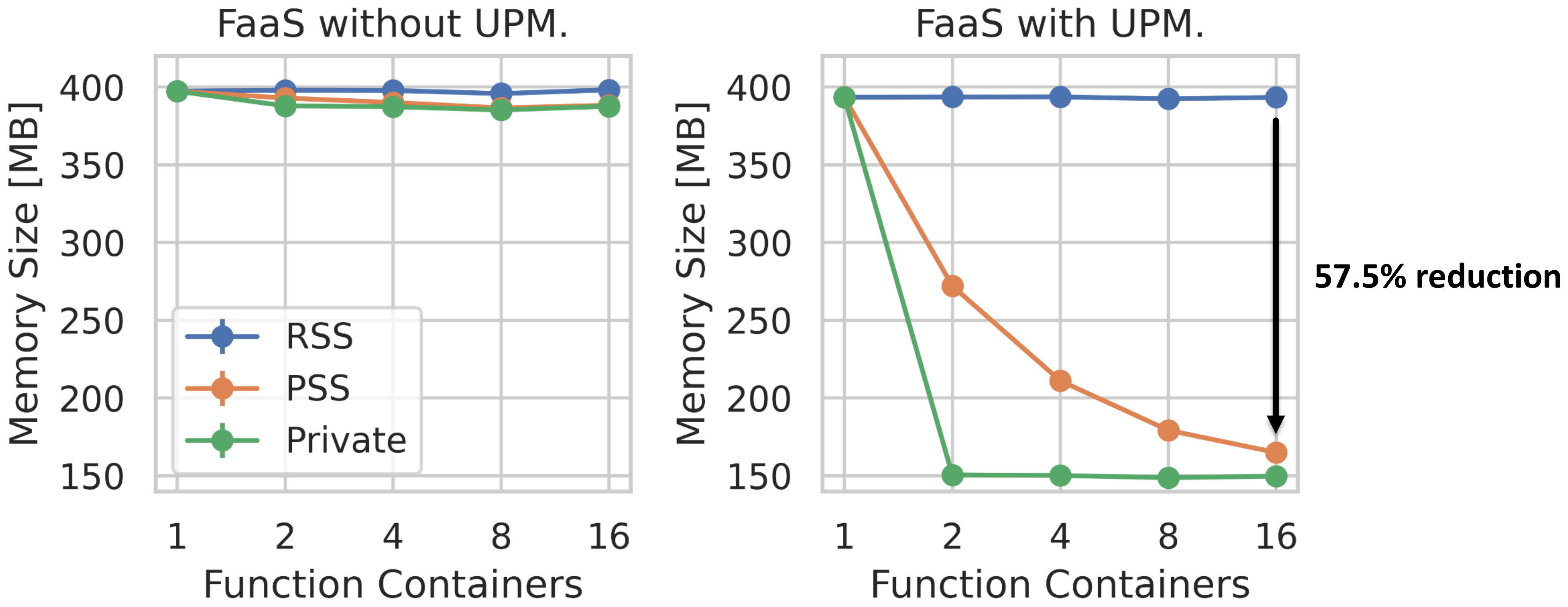


$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

Function Memory Footprint – AlexNet

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.

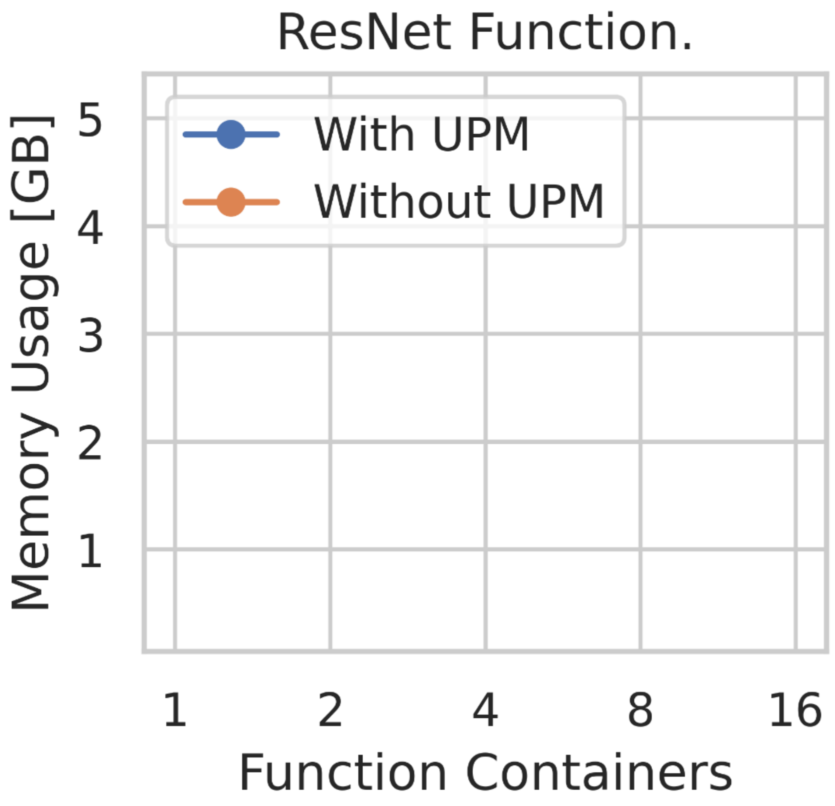


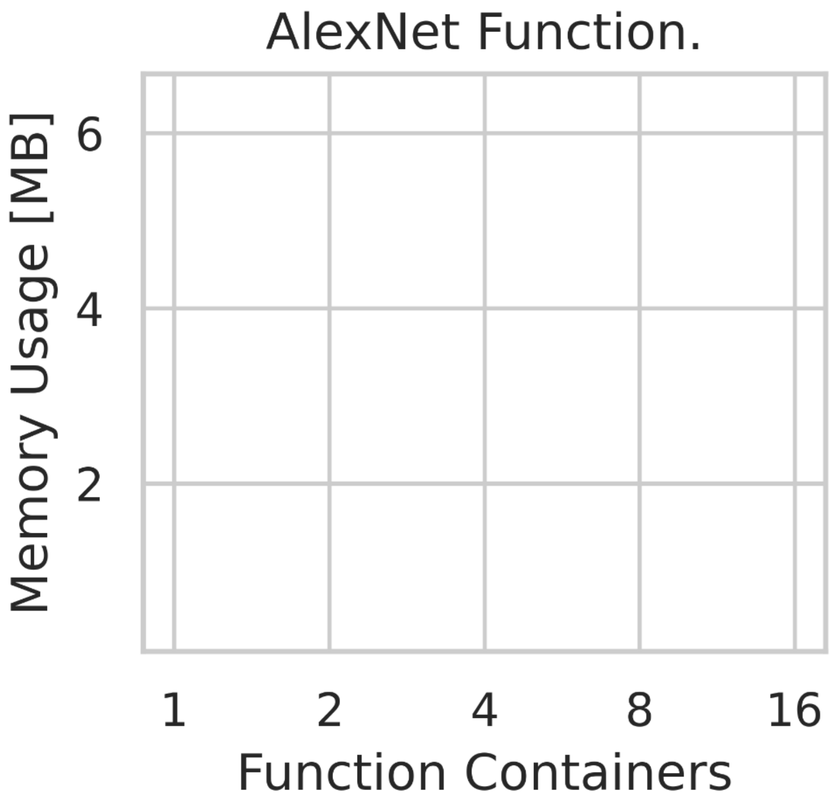
$$\text{RSS} = \text{Private} + \text{Shared}$$

$$\text{PSS} = \text{Private} + \frac{\text{Shared}}{\#\text{Processes}}$$

System Memory Consumption

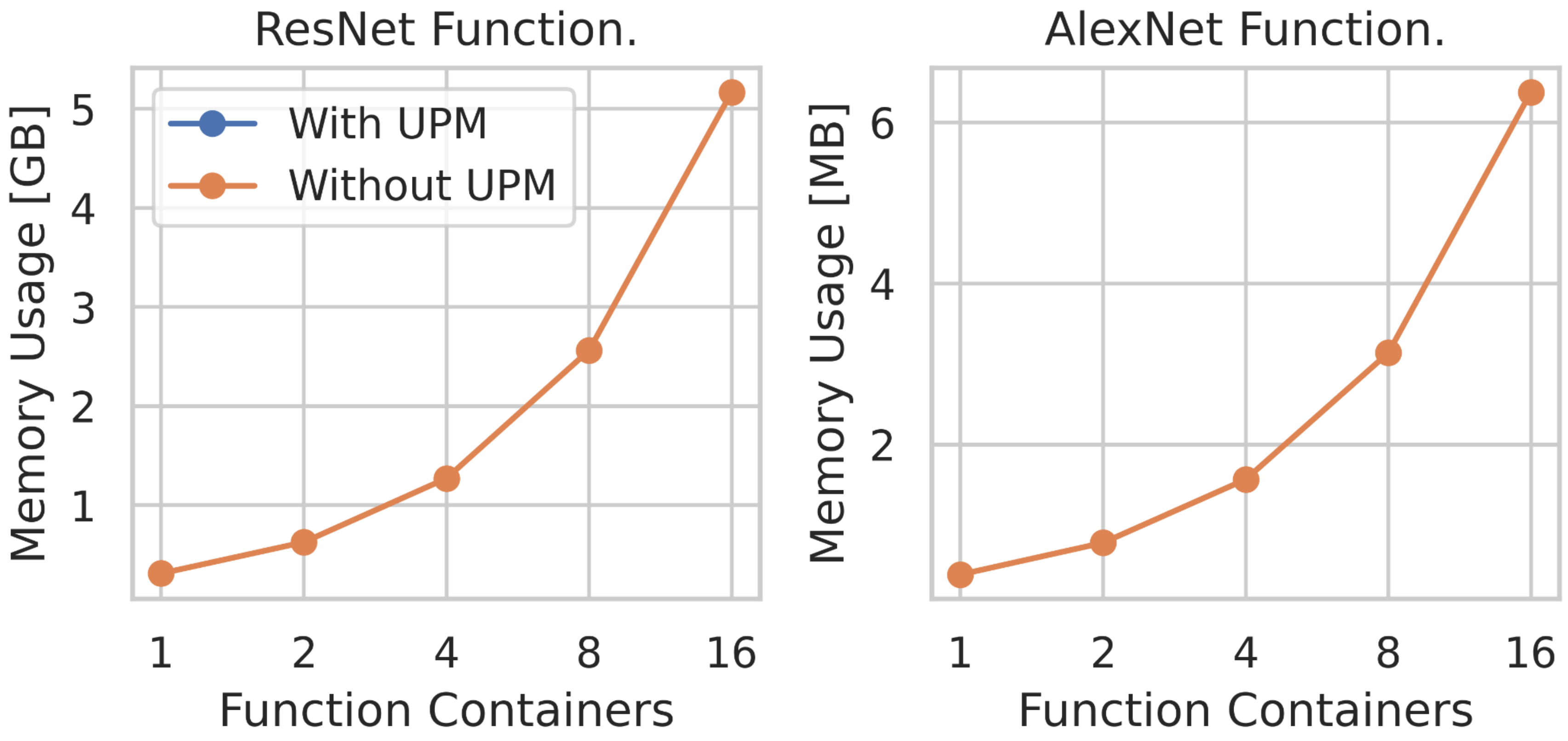
4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.





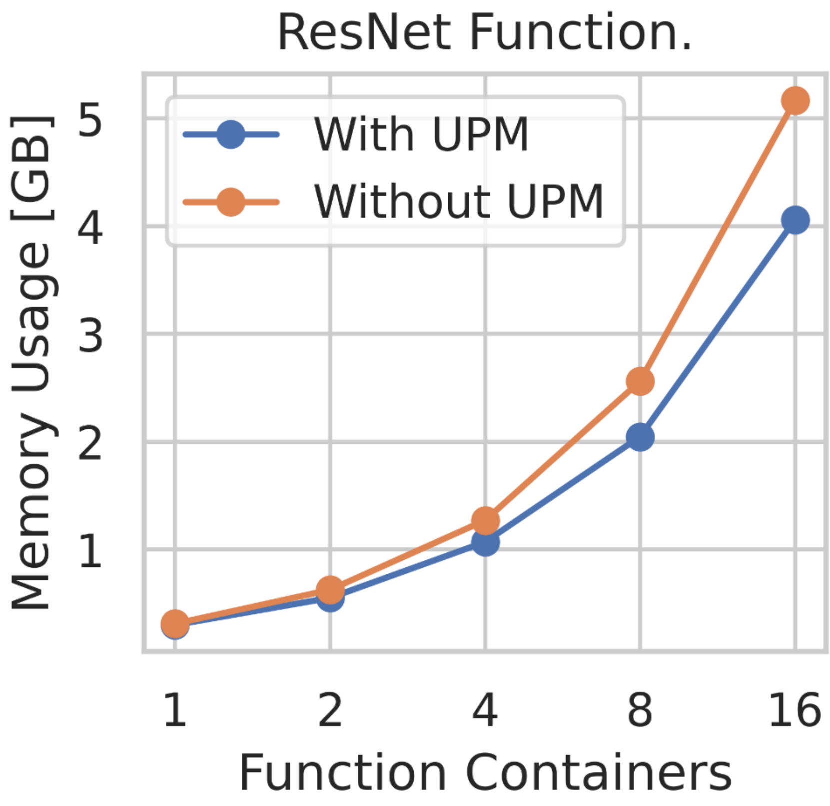
System Memory Consumption

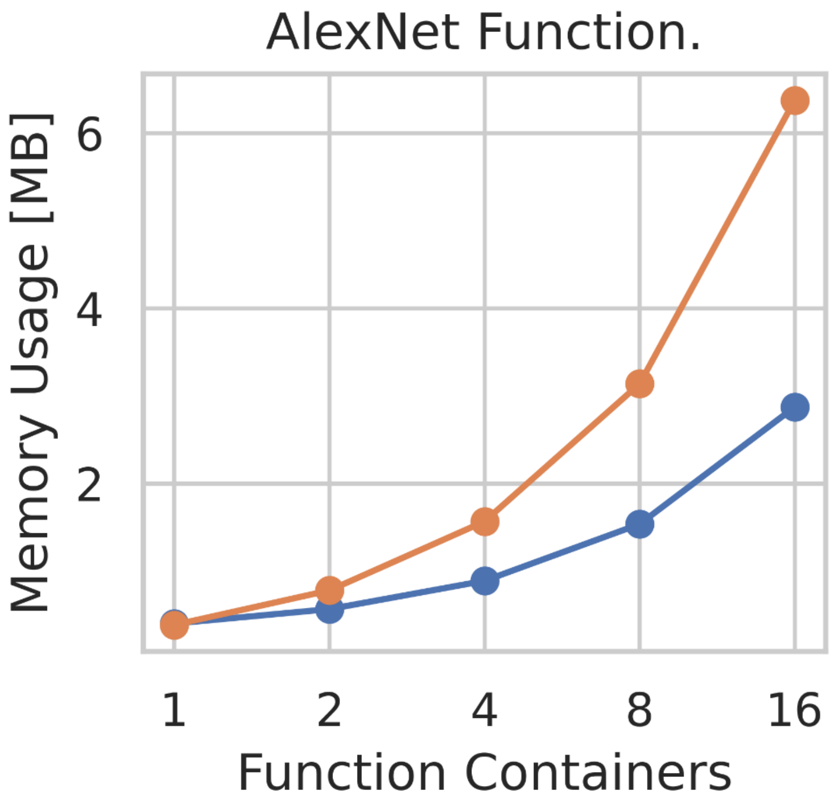
4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.



System Memory Consumption

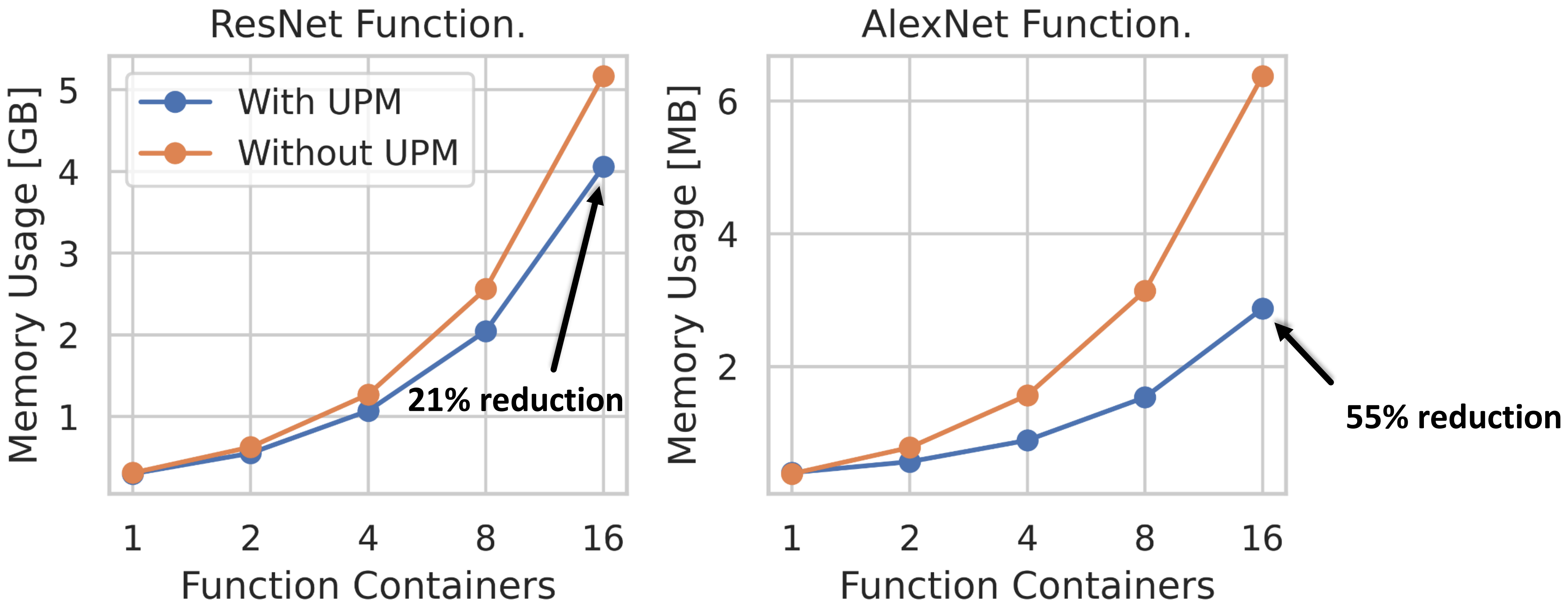
4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.





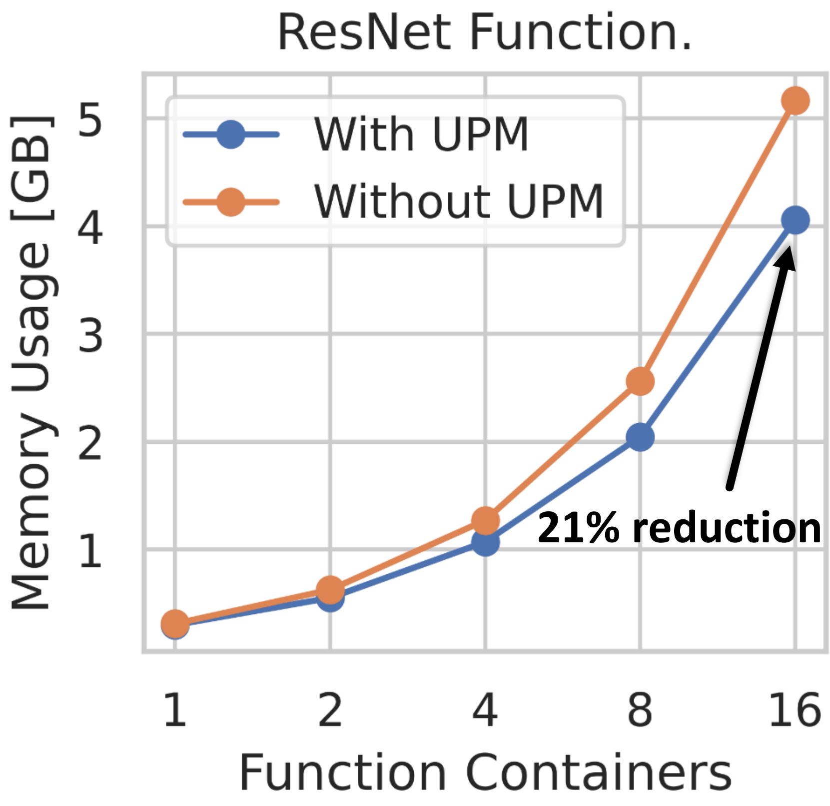
System Memory Consumption

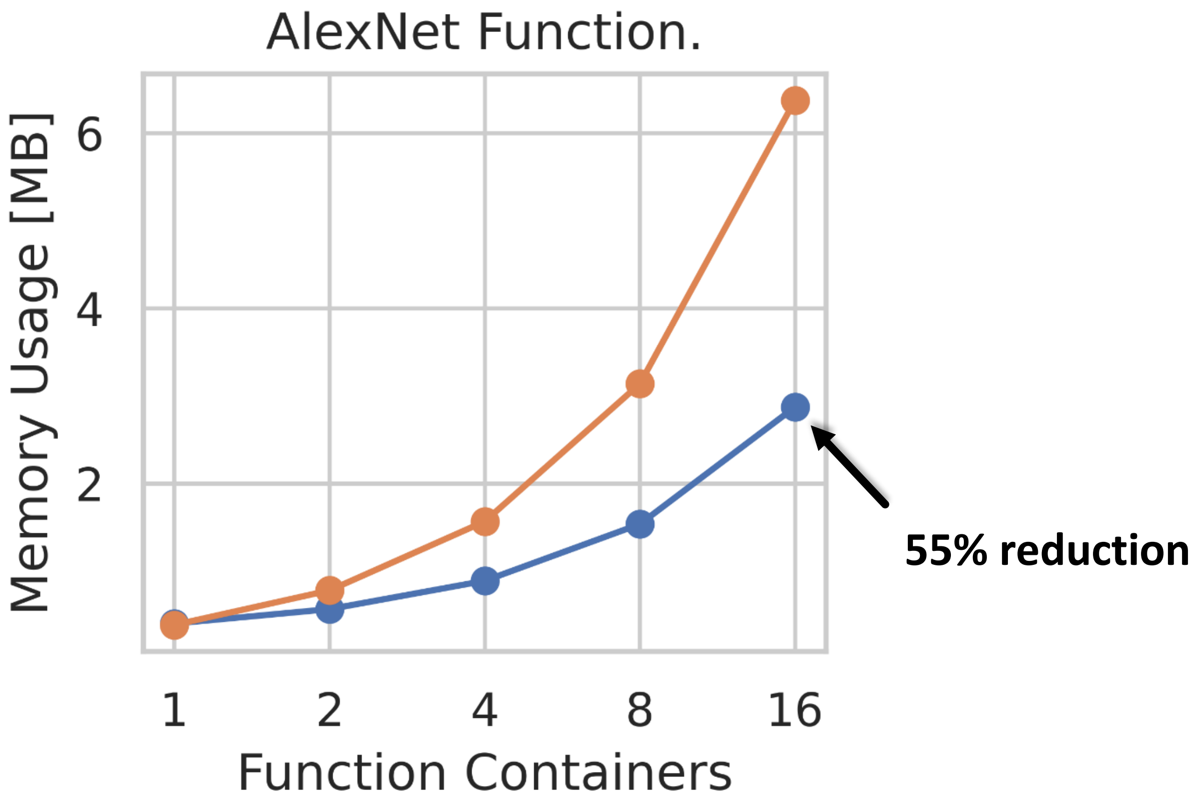
4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.



System Memory Consumption

4x Intel Xeon X7550 @ 2.00GHz, 64 cores total. 1 TB memory.



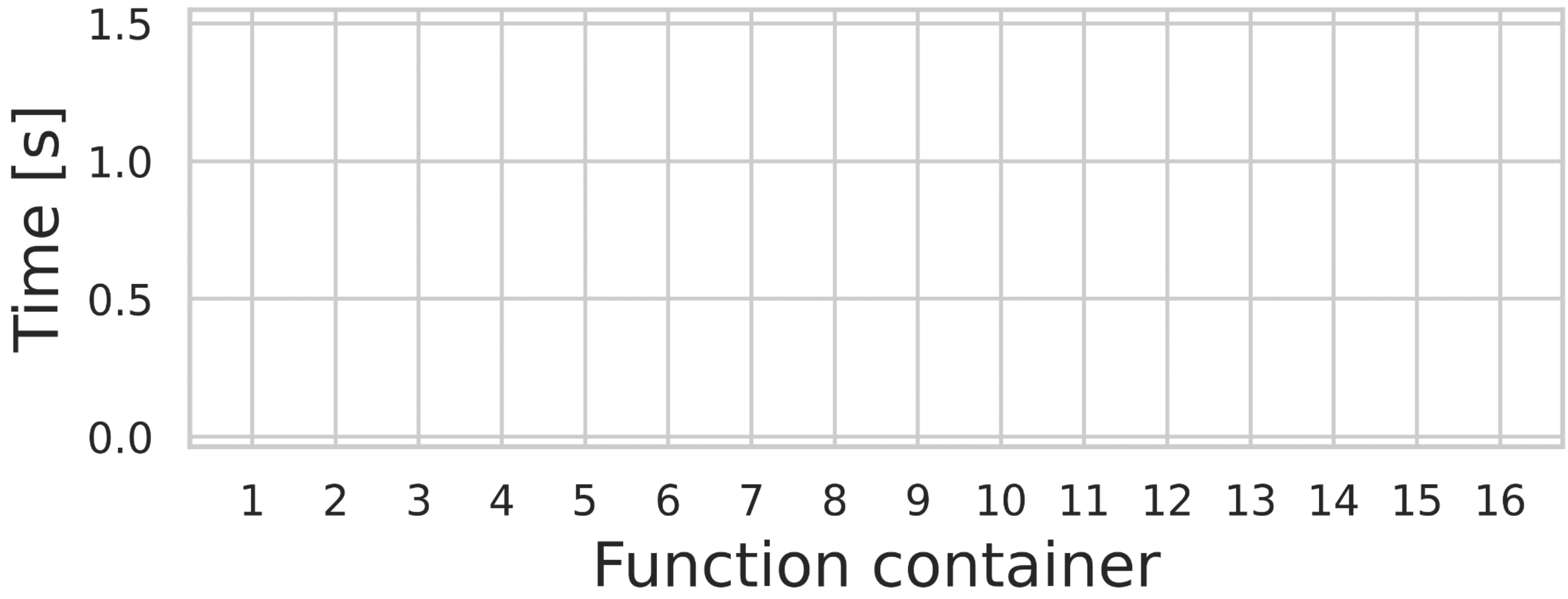


5 more containers

21 more containers

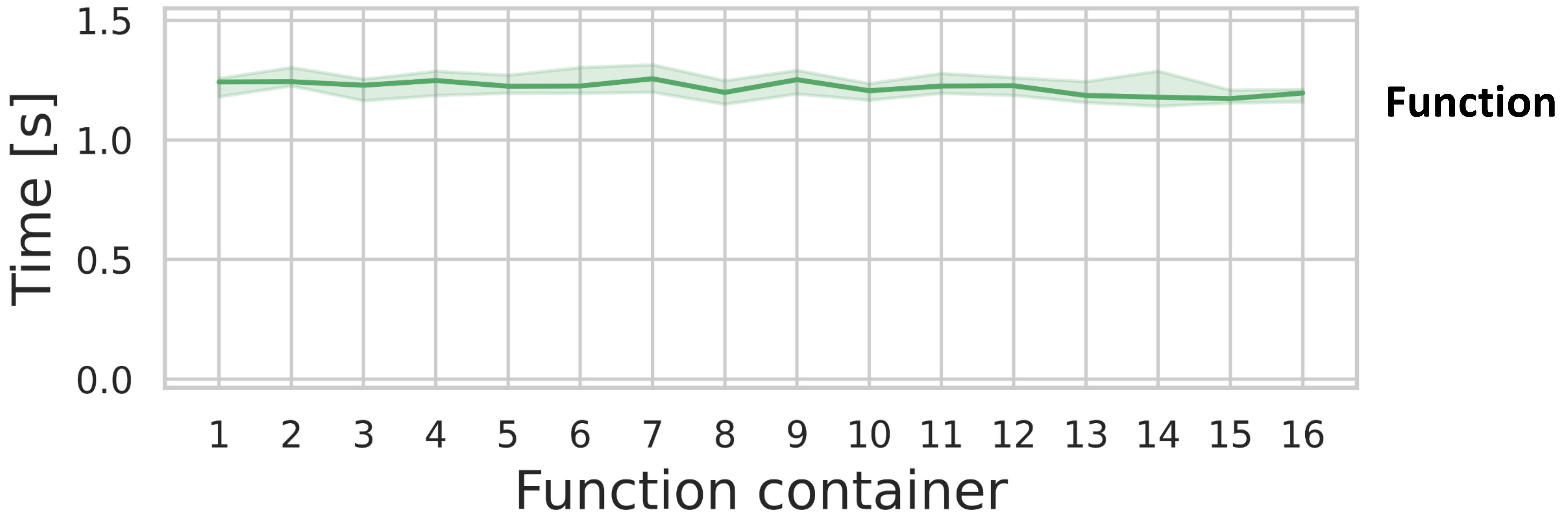
Cold Startup Overhead – ResNet 50

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



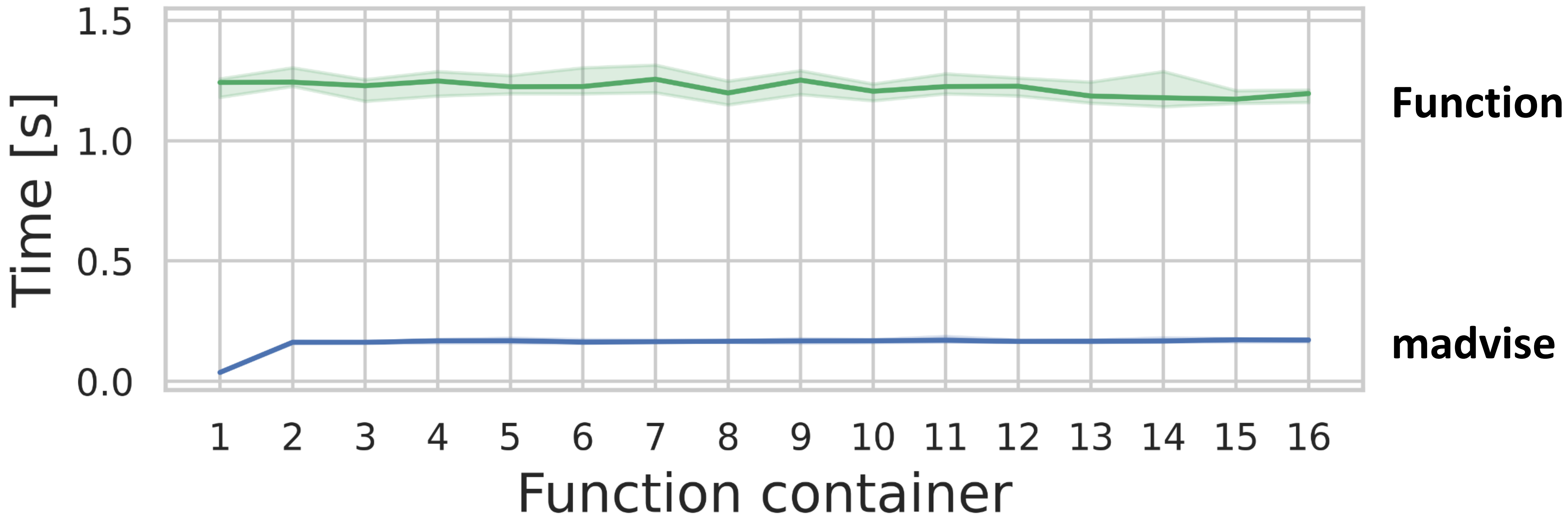
Cold Startup Overhead – ResNet 50

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



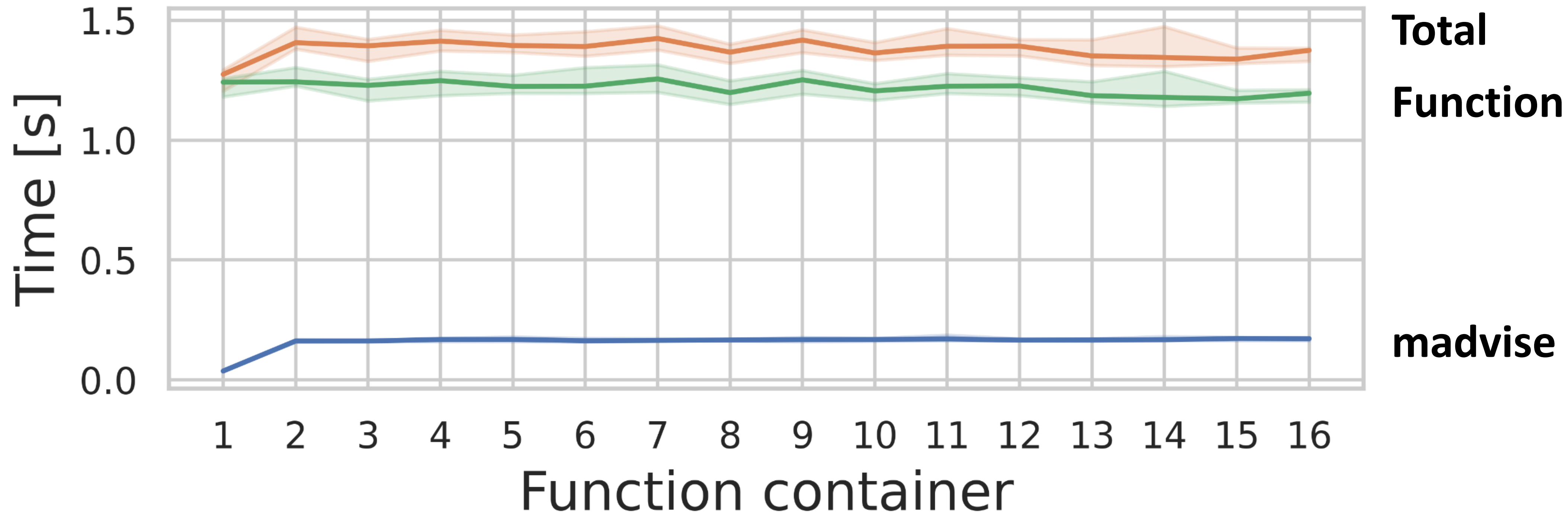
Cold Startup Overhead – ResNet 50

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



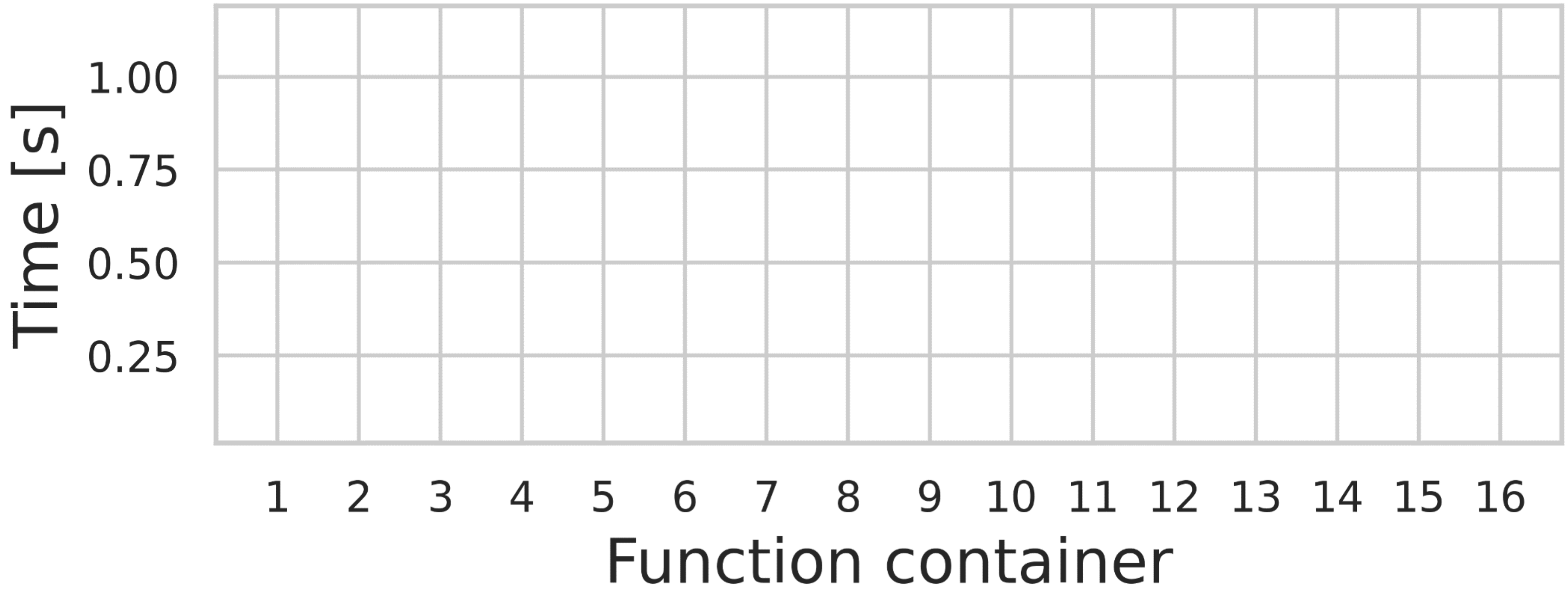
Cold Startup Overhead – ResNet 50

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



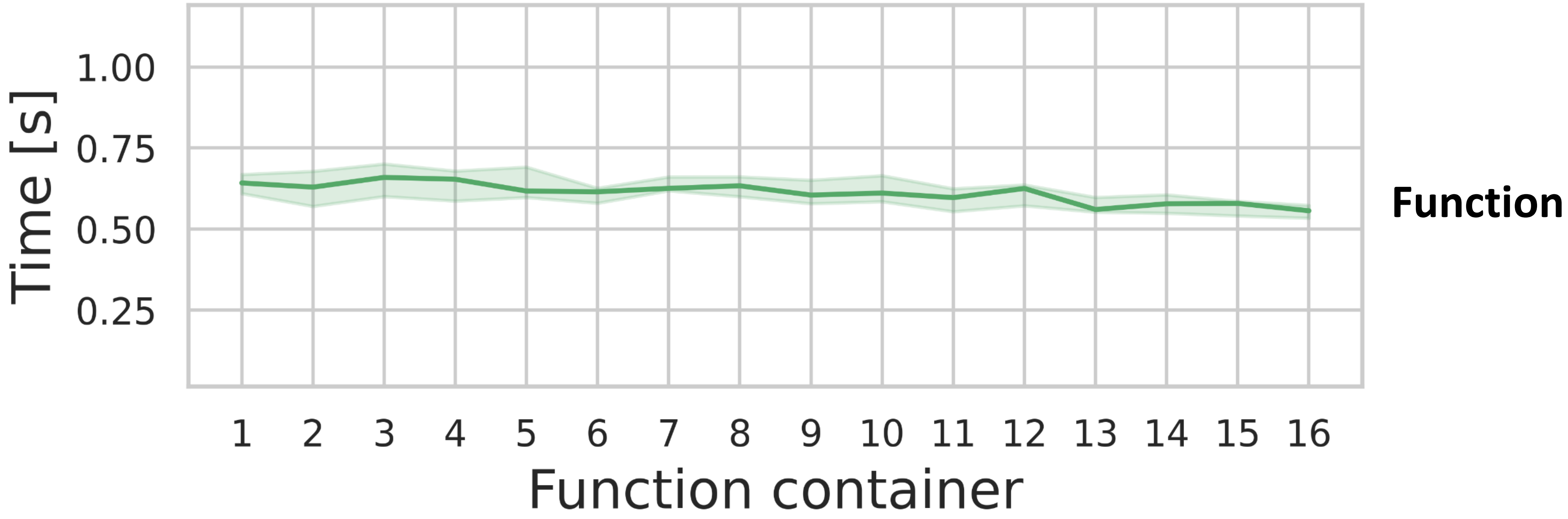
Cold Startup Overhead – AlexNet

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



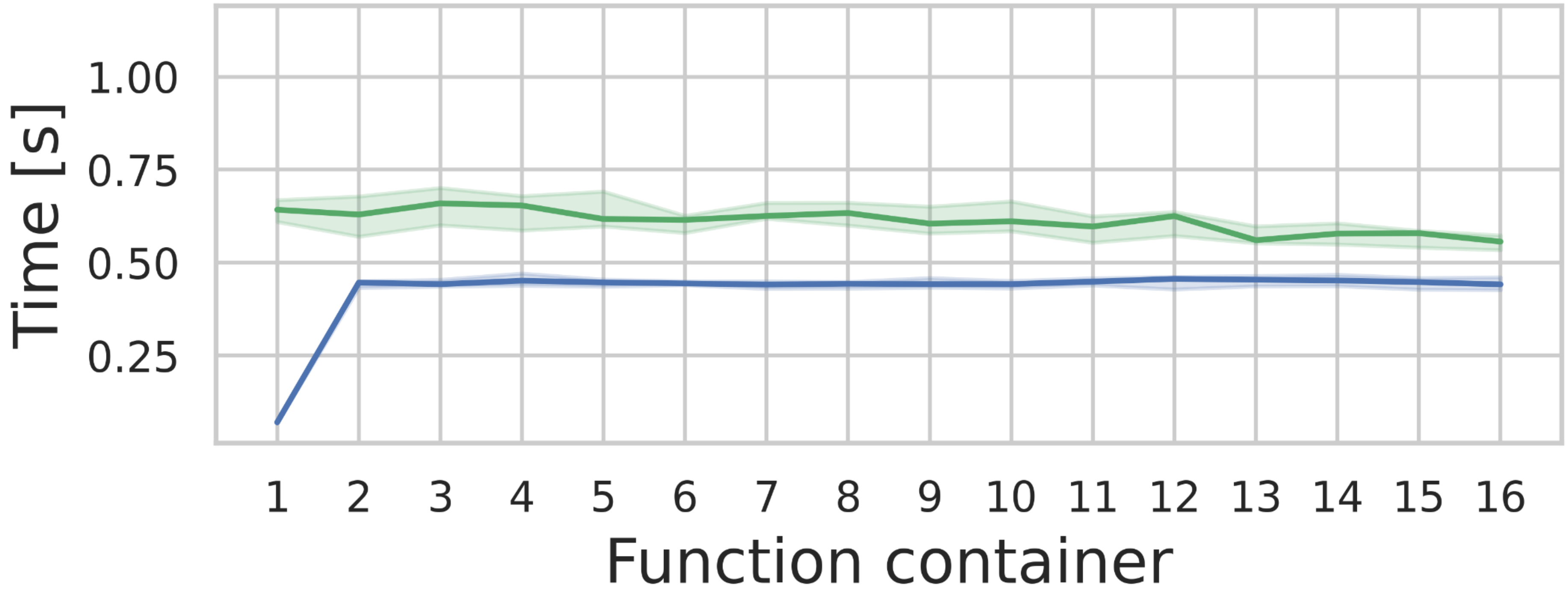
Cold Startup Overhead – AlexNet

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



Cold Startup Overhead – AlexNet

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.

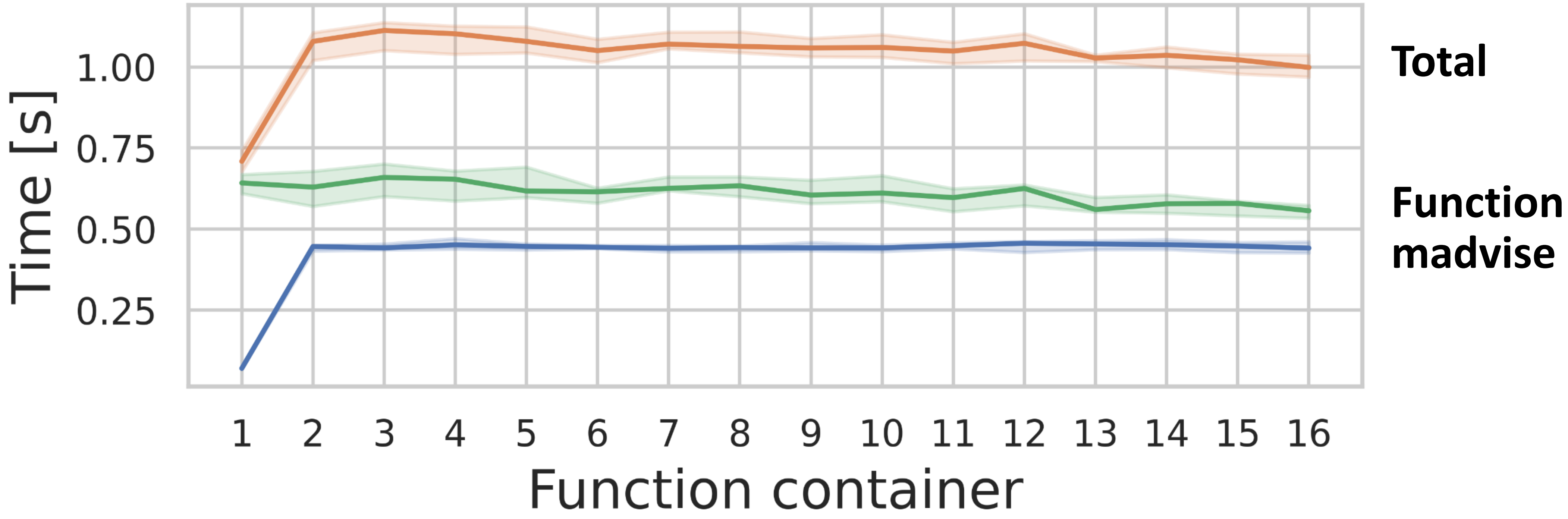


Function

madvise

Cold Startup Overhead – AlexNet

2x Intel Xeon 4110 @ 2.10GHz, 16 cores total. 125 GB memory.



UPM at GitHub

UPM at GitHub

spcl/UPM

spcl/UPM-artifact-data

High-Performance Serverless Solutions

High-Performance Serverless Solutions

spcl/serverless-benchmarks

spcl/rFaaS

spcl/FMI

spcl/PraaS

spcl/FaaSKeeper

Conclusions

More of SPCL's research:

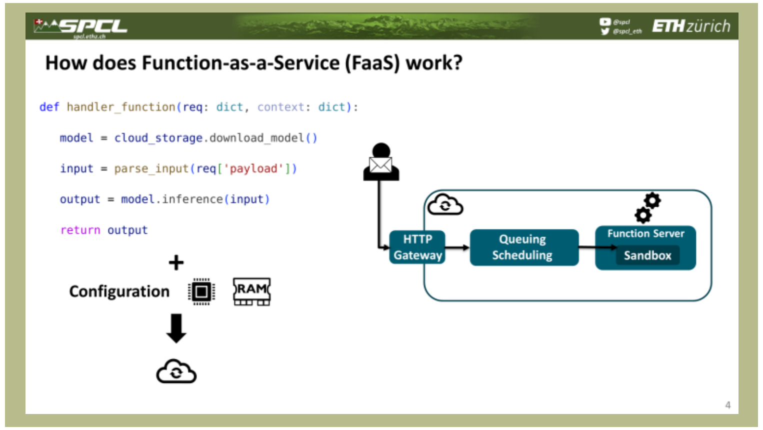
youtube.com/@spcl	180+ Talks
twitter.com/spcl_eth	1.4K+ Followers
github.com/spcl	3.8K+ Stars

… or spcl.ethz.ch

This work has received funding from the European Research Council (ERC).
We acknowledge support from the Swiss National Supercomputing Centre (CSCS).

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Conclusions



More of SPCL's research:

youtube.com/@spcl — 180+ Talks

twitter.com/spcl_eth — 1.4K+ Followers

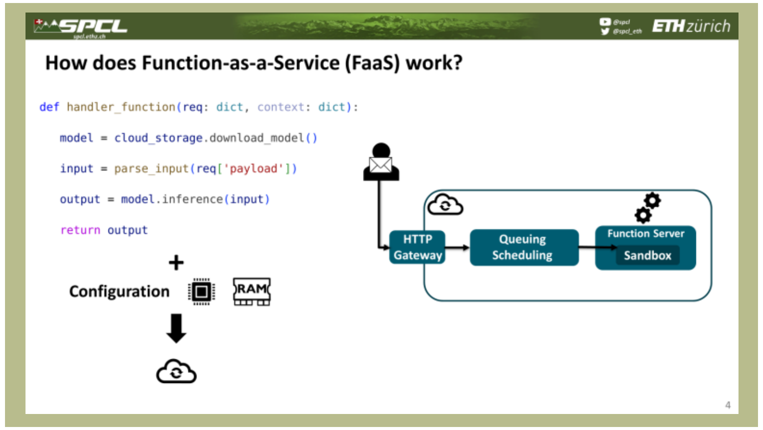
github.com/spcl — 3.8K+ Stars

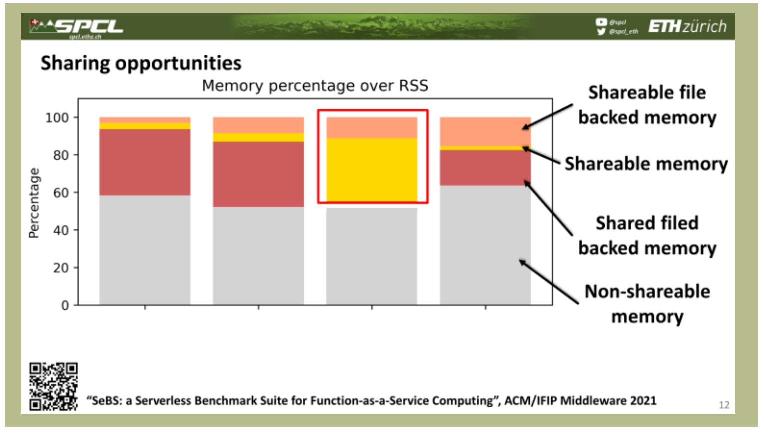
… or spcl.ethz.ch

This work has received funding from the European Research Council (ERC).
We acknowledge support from the Swiss National Supercomputing Centre (CSCS).

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Conclusions





More of SPCL's research:

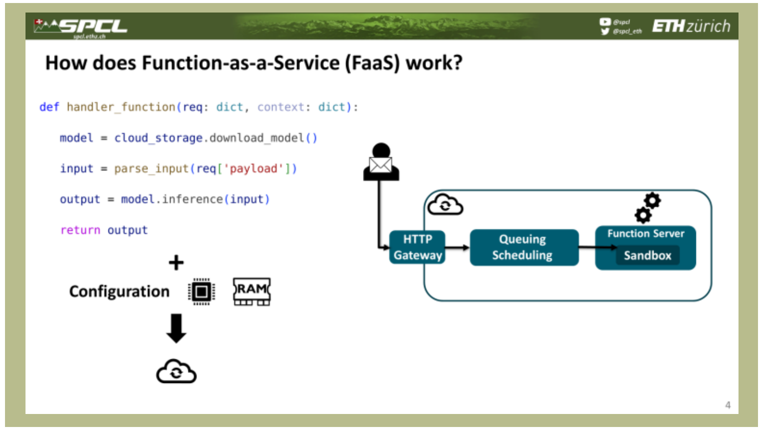
- youtube.com/@spcl — 180+ Talks
- twitter.com/spcl_eth — 1.4K+ Followers
- github.com/spcl — 3.8K+ Stars

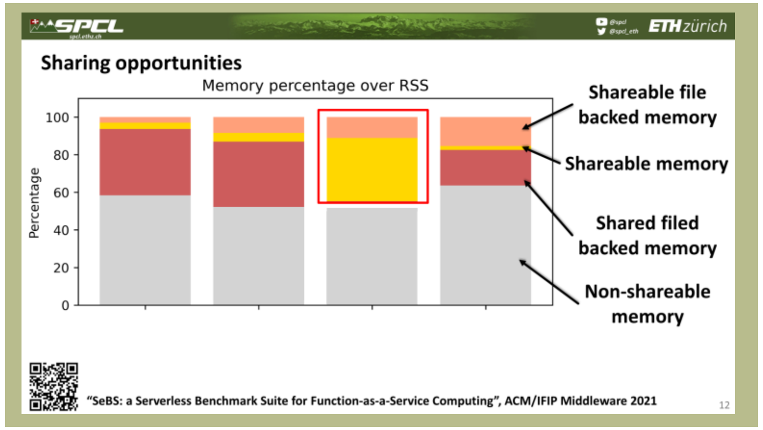
… or spcl.ethz.ch

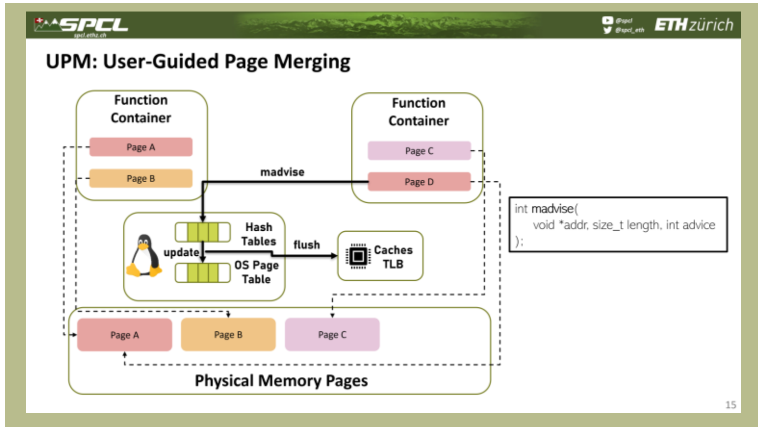
This work has received funding from the European Research Council (ERC).
We acknowledge support from the Swiss National Supercomputing Centre (CSCS).

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Conclusions







More of SPCL's research:

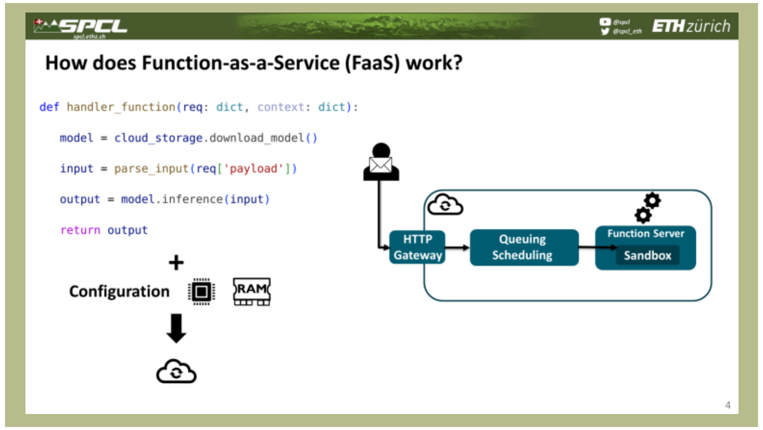
- youtube.com/@spcl — 180+ Talks
- twitter.com/spcl_eth — 1.4K+ Followers
- github.com/spcl — 3.8K+ Stars

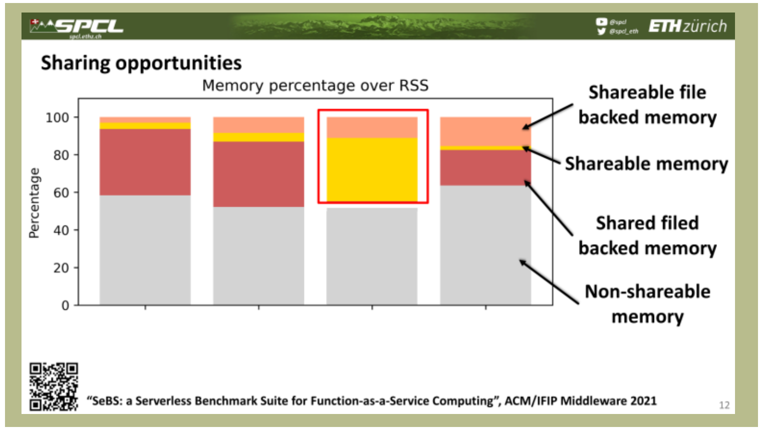
… or spcl.ethz.ch

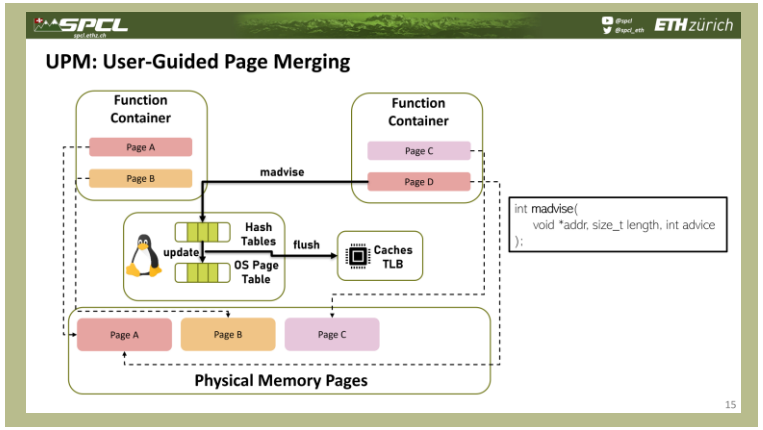
This work has received funding from the European Research Council (ERC).
We acknowledge support from the Swiss National Supercomputing Centre (CSCS).

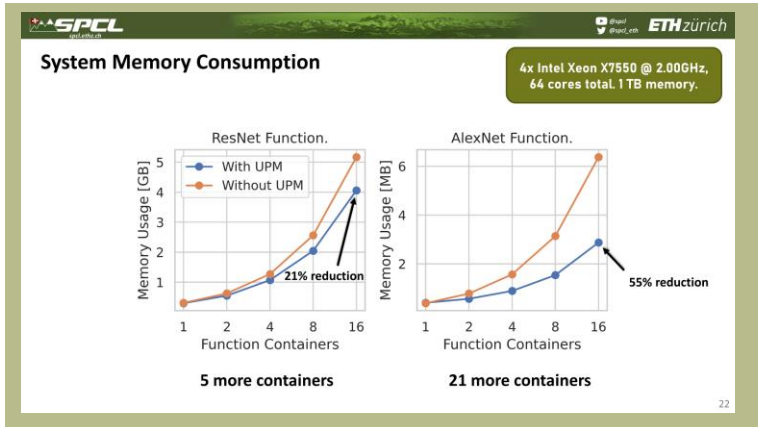
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Conclusions









More of SPCL's research:

youtube.com/@spcl — 180+ Talks

twitter.com/spcl_eth — 1.4K+ Followers

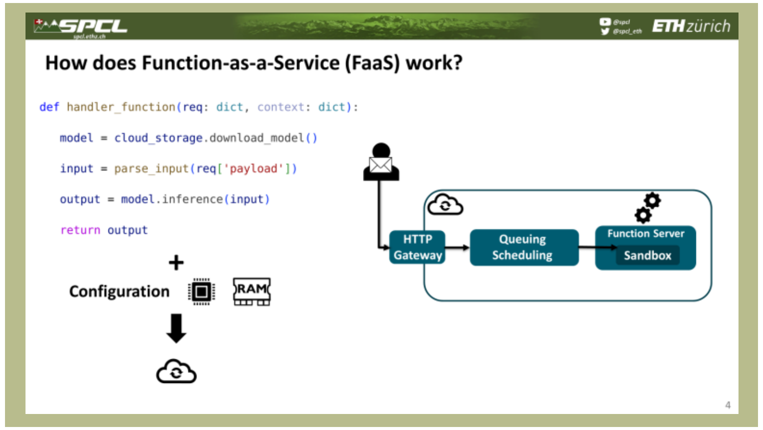
github.com/spcl — 3.8K+ Stars

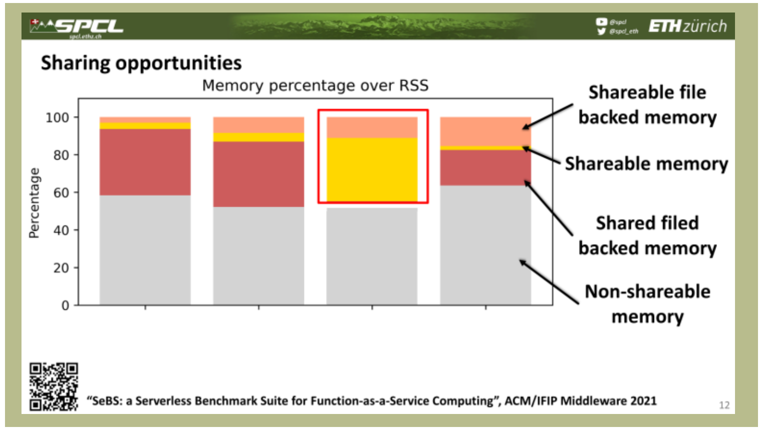
… or spcl.ethz.ch

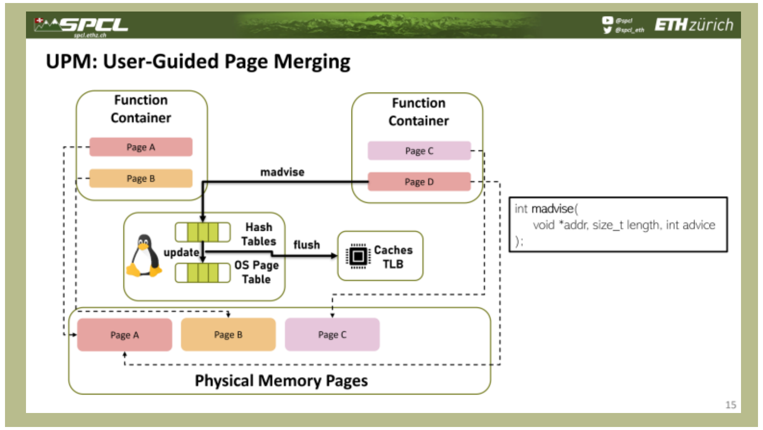
This work has received funding from the European Research Council (ERC).
We acknowledge support from the Swiss National Supercomputing Centre (CSCS).

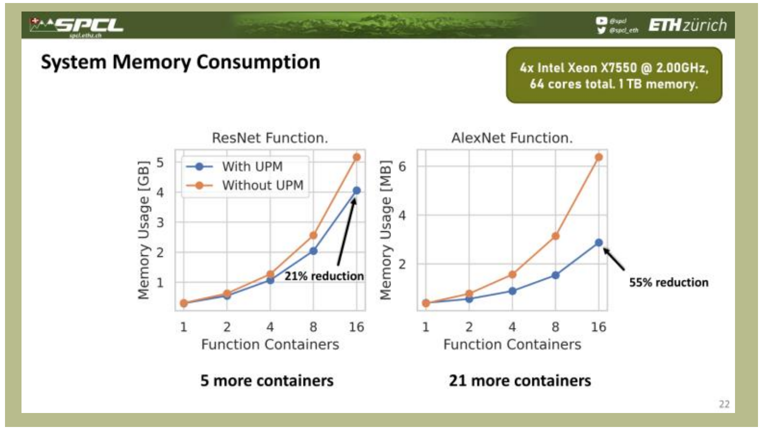
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Conclusions









More of SPCL's research:

- youtube.com/@spcl — 180+ Talks
- twitter.com/spcl_eth — 1.4K+ Followers
- github.com/spcl — 3.8K+ Stars

... or spcl.ethz.ch

Paper preprint **Other projects**

This work has received funding from the European Research Council (ERC).
We acknowledge support from the Swiss National Supercomputing Centre (CSCS).

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre