Marcin Copik

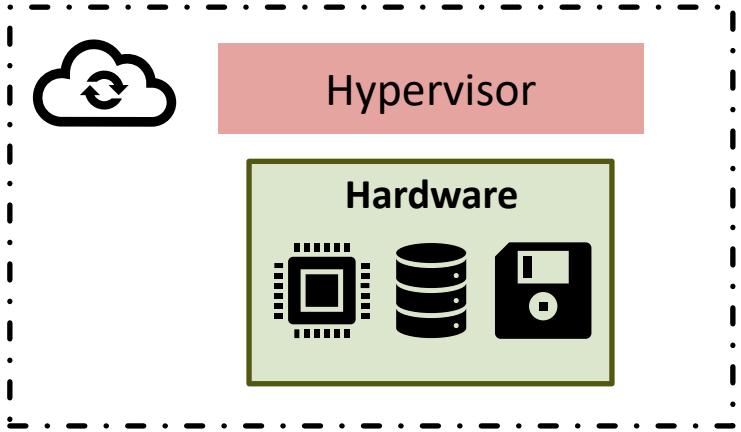
AXeleration as a Service (XaaS): From Batch Jobs to Elastic Services in Scientific Computing

HPC-AI ADVISORY COUNCIL NETWORK OF EXPERTISE

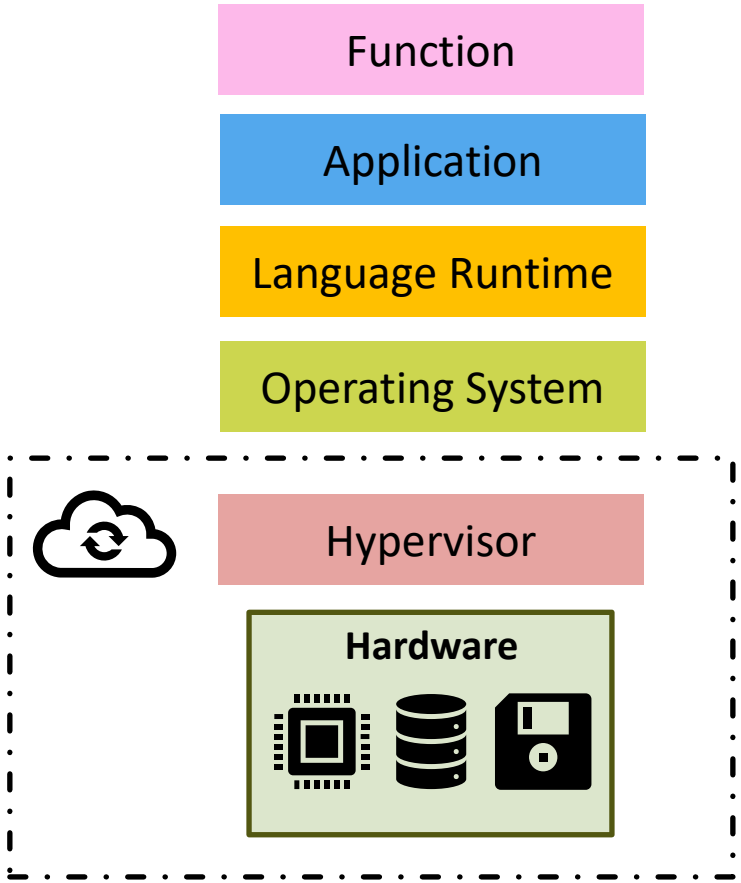
HPC-AI Swiss Conference

Cloud Evolution

Cloud Evolution

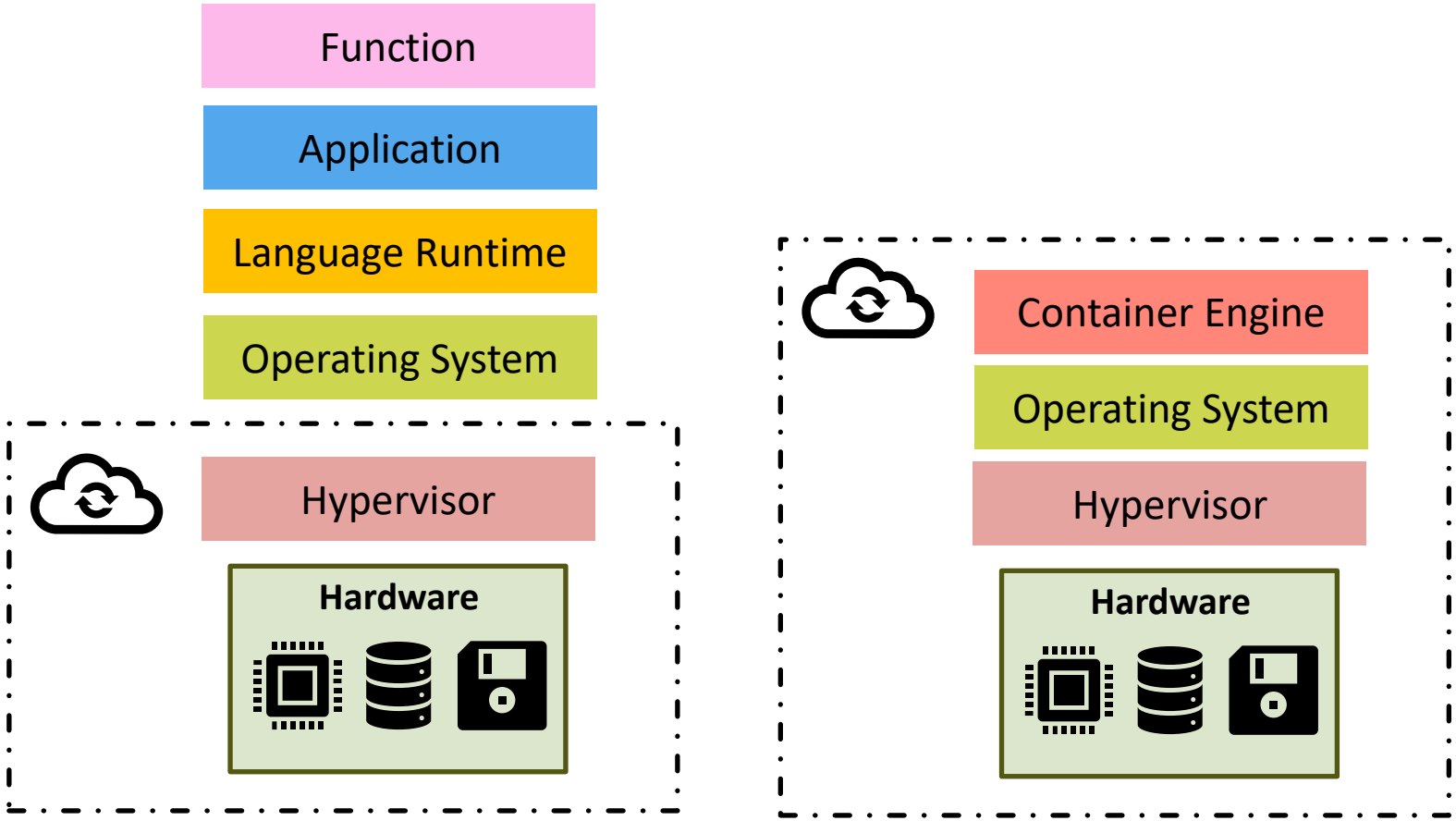


Cloud Evolution



Virtual Machine (Infrastructure-as-a-Service)

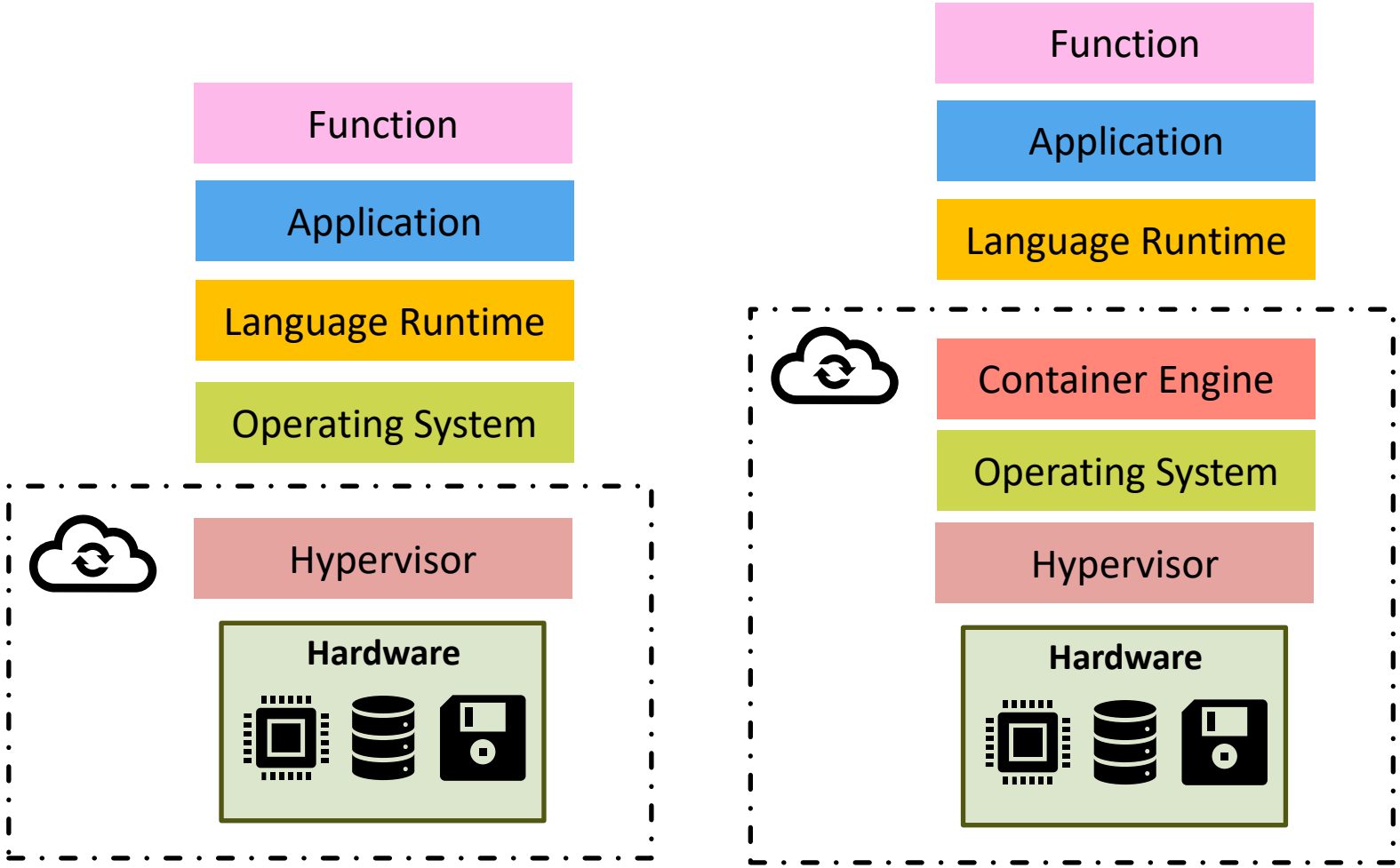
Cloud Evolution



Virtual Machine (Infrastructure-as-a-Service)

Containers (Container-as-a-Service)

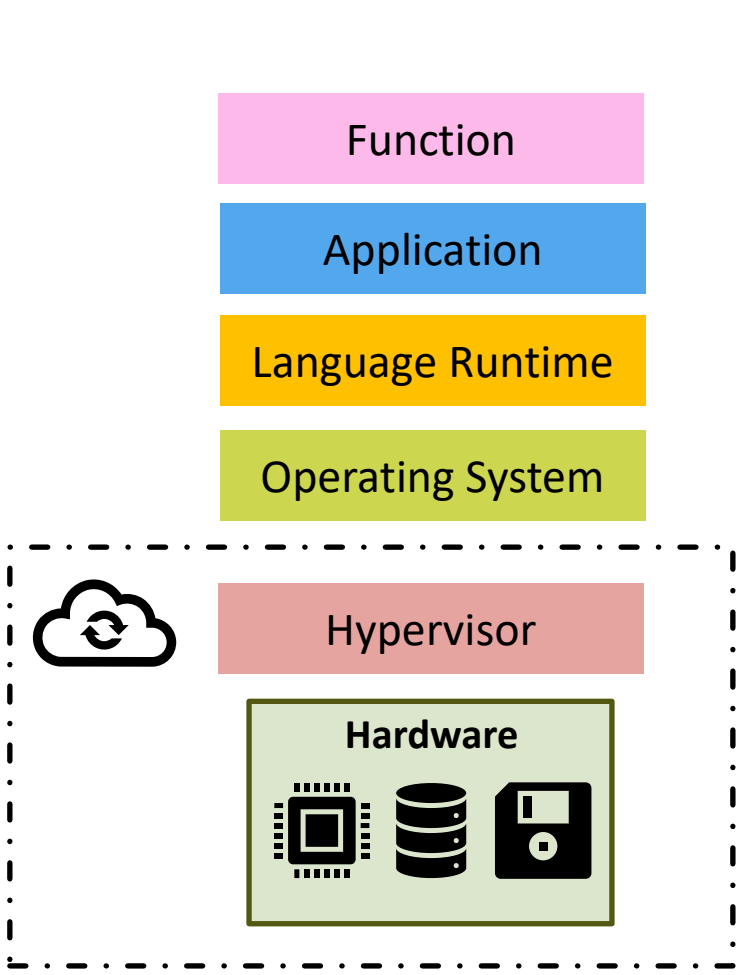
Cloud Evolution



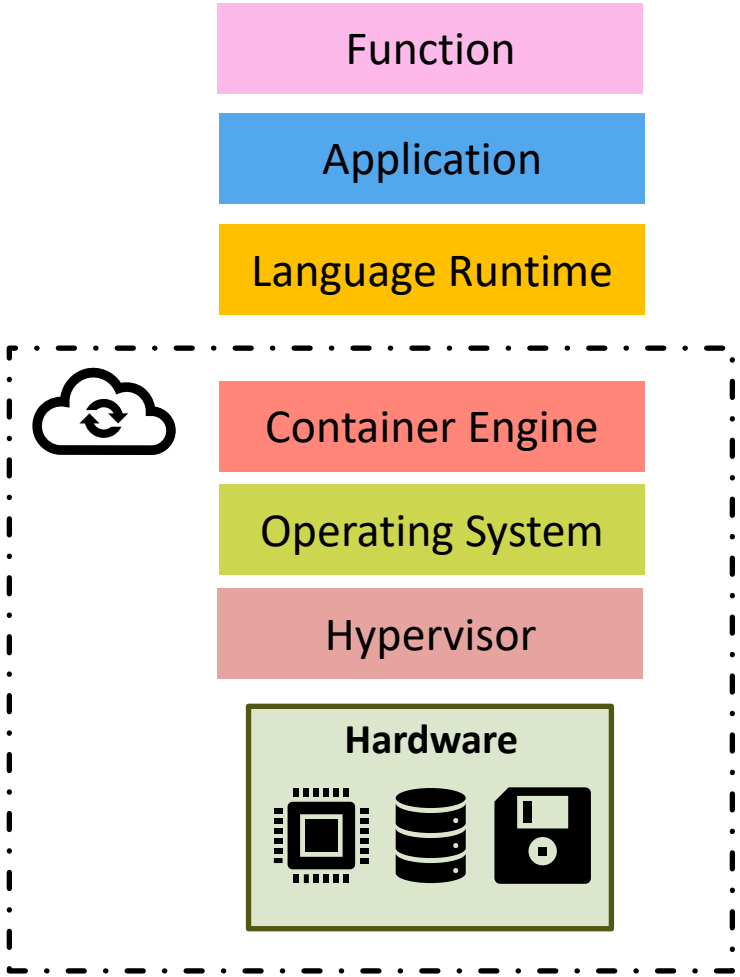
Virtual Machine (Infrastructure-as-a-Service)

Containers (Container-as-a-Service)

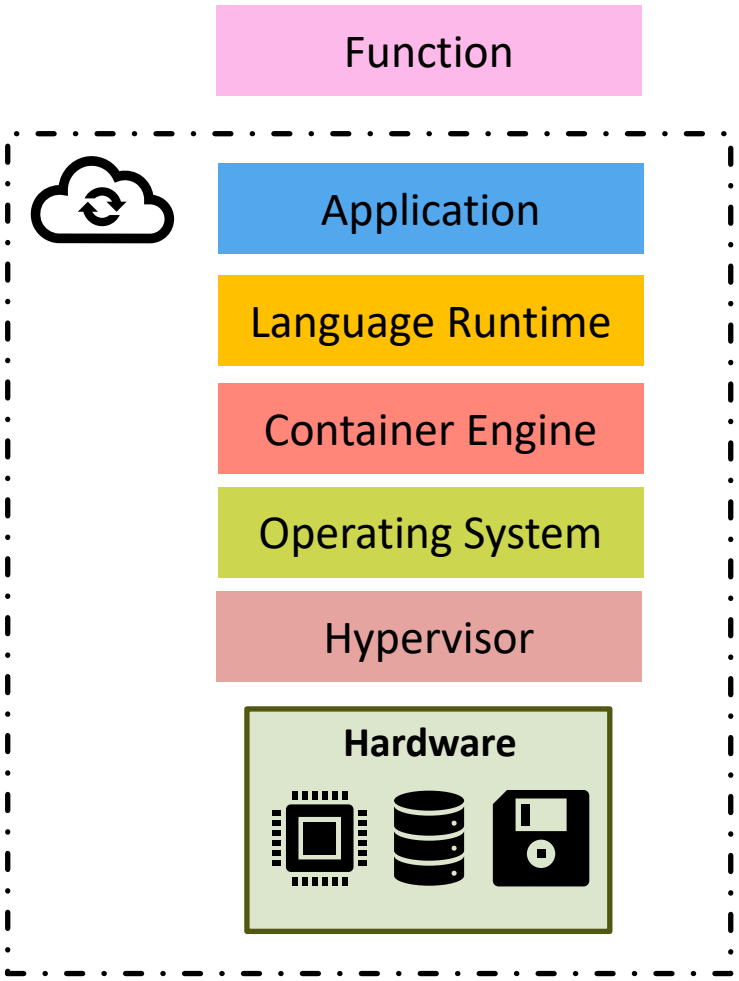
Cloud Evolution



Virtual Machine
(Infrastructure-as-a-Service)



Containers
(Container-as-a-Service)



Functions
(Function-as-a-Service)

Function-as-a-Service (FaaS): Good and Bad

```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    result = inference(request['op'], data)

    return result
```

Function-as-a-Service (FaaS): Good and Bad

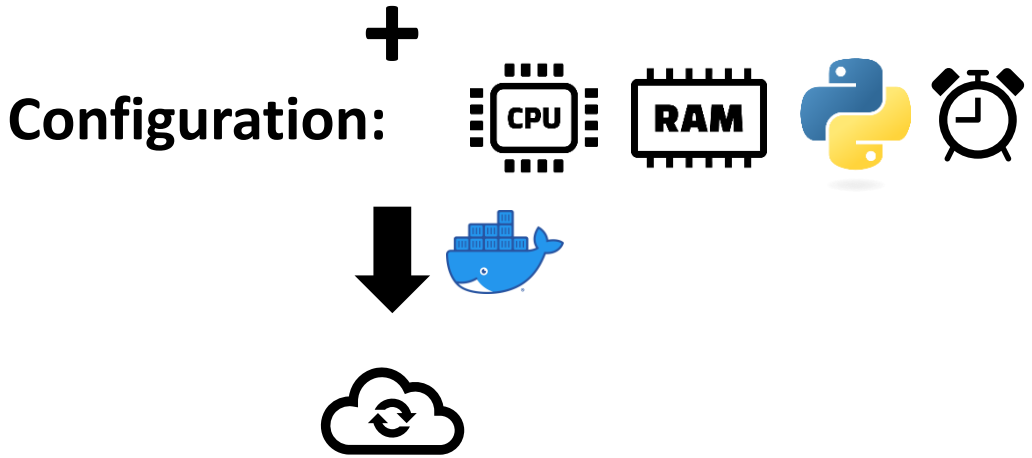
```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    result = inference(request['op'], data)

    return result
```

+

Configuration: 

Function-as-a-Service (FaaS): Good and Bad

```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    result = inference(request['op'], data)

    return result
```



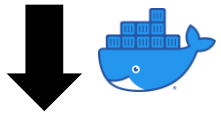
Function-as-a-Service (FaaS): Good and Bad

- No Infrastructure Management
- Elastic Jobs
- Improved Utilization
- Fast Scalability

```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    result = inference(request['op'], data)

    return result
```

+

Configuration:

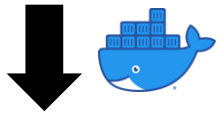


Function-as-a-Service (FaaS): Good and Bad

```
model = model_init(cloud_storage, 'resnet50')

def handler_function(request: dict, context: dict):

    data = cloud_storage.read(request['id'])

    result = inference(request['op'], data)

    return result
```

+

Configuration:



- No Infrastructure Management
- Elastic Jobs
- Improved Utilization
- Fast Scalability

- Disaggregated Compute from Storage
- “Stateless” Functions
- No Infrastructure Management
- Black Box System

Serverless Was Not Designed for HPC…

AWS Lambda turns 10: A rare look at the doc that started it

November 14, 2024 • 5460 words

Serverless Was Not Designed for HPC…

31. How does Lambda support parallel processing?

Developers can run multiple applications and/or ***multiple copies of the same application simultaneously****. They can also access Lambda APIs programmatically from within applications, using the AWS client SDK, which allows them to delegate and orchestrate work by running other applications.*

Serverless Was Not Designed for HPC…

31. How does Lambda support parallel processing?

Developers can run multiple applications and/or ***multiple copies of the same application simultaneously****. They can also access Lambda APIs programmatically from within applications, using the AWS client SDK, which allows them to delegate and orchestrate work by running other applications.*

Burst Launches, Colocation Policies
Bulk Synchronous Parallel Model
Communicators, Message Passing, Collectives
…

Serverless Was Not Designed for HPC…

31. How does Lambda support parallel processing?

Developers can run multiple applications and/or ***multiple copies of the same application simultaneously****. They can also access Lambda APIs programmatically from within applications, using the AWS client SDK, which allows them to delegate and orchestrate work by running other applications.*

Burst Launches, Colocation Policies
Bulk Synchronous Parallel Model
Communicators, Message Passing, Collectives
…

Embarrassingly Parallel

…But Science Adapted It Anyway

FAASTLOOP: **Optimizing Loop-Based Applications for Serverless Computing**

Occupy the Cloud: Distributed Computing for the 99%

Eric Jonas, Qifan Pu, Shivaram Venkataraman, Ion Stoica, Benjamin Recht

Burst Computing: Quick, Sudden, Massively Parallel Processing on Serverless Resources

Daniel Barcelona-Por
Center; Aitor Arjo
Step

***funcX*: A Federated Function Serving Fabric for Science**

Ryan Chard
Argonne National Laboratory

Yadu Babuji
University of Chicago

Zhuozhao Li
University of Chicago

Tyler Skluzacek
University of Chicago

Anna Woodard
University of Chicago

Ben Blaiszik
University of Chicago

Ian Foster
Argonne National Laboratory and University of Chicago

Kyle Chard
University of Chicago and Argonne National Laboratory

...But Science Adapted It Anyway

Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions

Maciej Malawski *, Adam Gajek, Adam Zima, Bartosz Balis, Kamil Figiela

AGH Univer

DayDream: Executing Dynamic Scientific Workflows on Serverless Platforms with Hot Starts

Rohan Basu Roy
Northeastern University
Boston, MA, USA

Tirthak Patel
Northeastern University
Boston, MA, USA

Devesh Tiwari
Northeastern University
Boston, MA, USA

Mashup: Making Serverless Computing Useful for HPC Workflows via Hybrid Execution

Rohan Basu Roy
Northeastern University

Tirthak Patel
Northeastern University

Vijay Gadepally
MIT Lincoln Laboratory

Devesh Tiwari
Northeastern University

Zhuozhao Li
University of Chicago

Ben Blaiszik
University of Chicago

SOFTWARE

Democratising high performance computing for bioinformatics through serverless cloud computing: A case study on CRISPR-Cas9 guide RNA design with Crackling Cloud

Jacob Bradford[1,2]*, Divya Joy[1], Mattias Winsen[1], Nich
Mackenzie Wilkins[1], Laurence O.W. Wilson[3,4], Denis C.

A Serverless Engine for High Energy Physics Distributed Analysis

1st Jacek Kuśnierz
Institute of Computer Science
AGH
Kraków, Poland

2nd Vincenzo E. Padulano
EP-SFT, CERN
Geneva, Switzerland

3rd Maciej Malawski
Institute of Computer Science
AGH
Kraków, Poland

4th Kamil Burkiewicz
Institute of Computer Science
AGH
Kraków, Poland

8th Valentina Avati
EP-UHC
CERN
Geneva, Switzerland

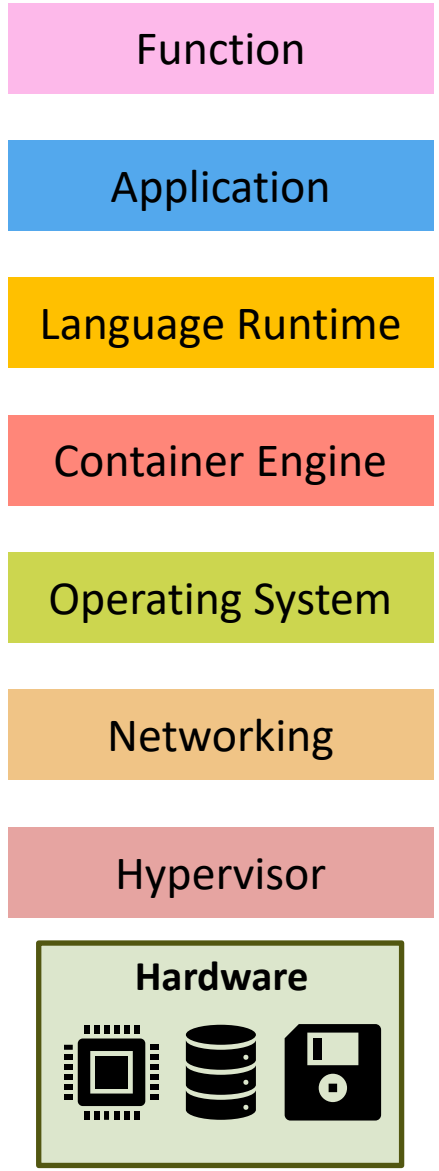
A serverless computing architecture for Martian aurora detection with the Emirates Mars Mission

David Pacios, José Luis Vázquez-Poletti ✉, Dattaraj B. Dhuri, Dimitra Atri, Rafael Moreno-Vozmediano,

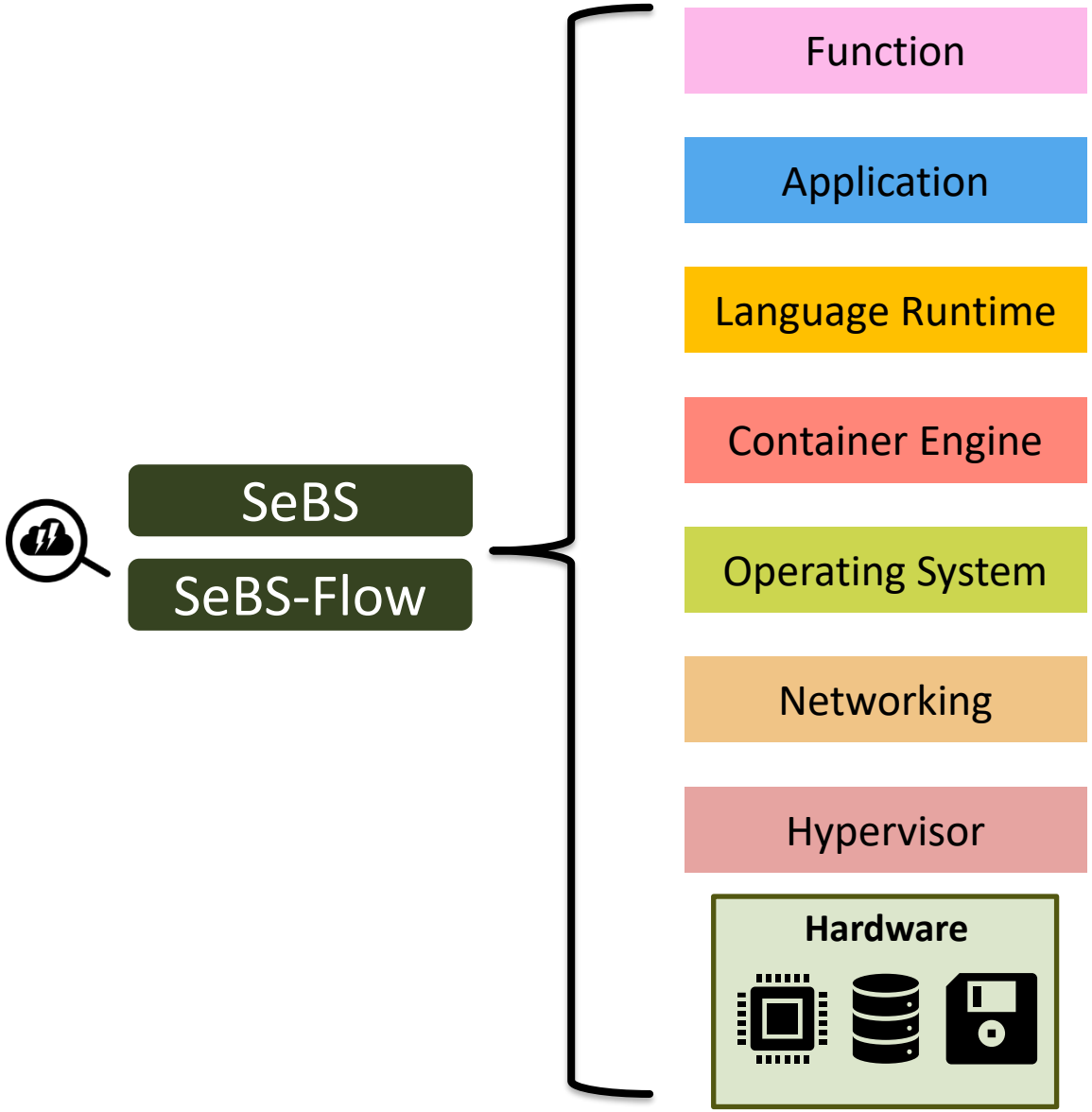
Robert J. Lillis, Nikolaos Schetakis, Jorge Gómez-Sanz, Alessio Di Iorio & Luis Vázquez

Scientific Reports **14**, Article number: 3029 (2024) | Cite this article

AXelerating Serverless Computing: Measure!



AXelerating Serverless Computing: Measure!



SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

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

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

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

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

Representative Benchmarks

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

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

Cloud-Agnostic

Representative Benchmarks

Automatic Experiments

Performance & Cost
Invocation Overhead
Container Eviction
Communication

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

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

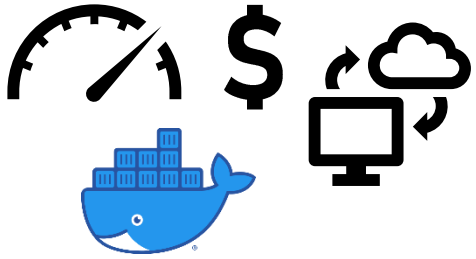
Cloud-Agnostic

Representative Benchmarks

Automatic Experiments

Performance & Cost
Invocation Overhead
Container Eviction
Communication

Insights



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

SeBS: The Serverless Benchmark Suite

spcl/serverless-benchmarks

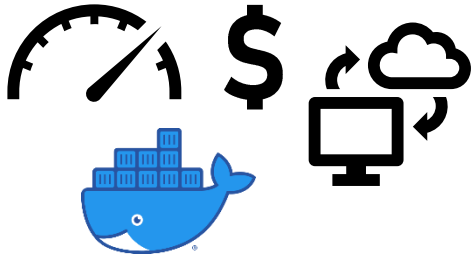
Cloud-Agnostic

Representative Benchmarks

Automatic Experiments

Performance & Cost
Invocation Overhead
Container Eviction
Communication

Insights



Adoption & Community

99 forks
30+ contributors

Google
Summer of Code

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

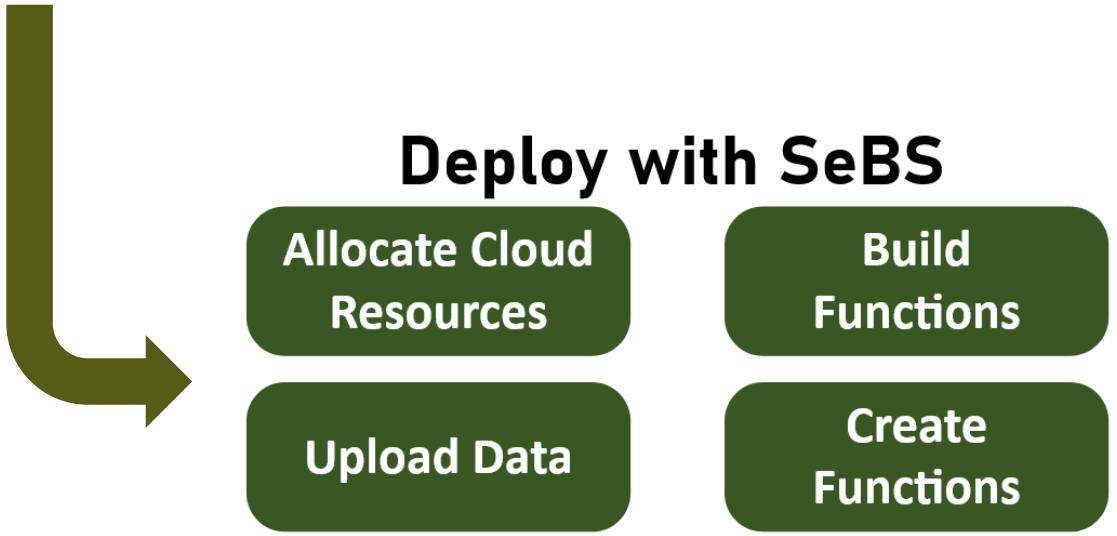
SeBS: The Serverless Benchmark Suite

"Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?"

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

SeBS: The Serverless Benchmark Suite

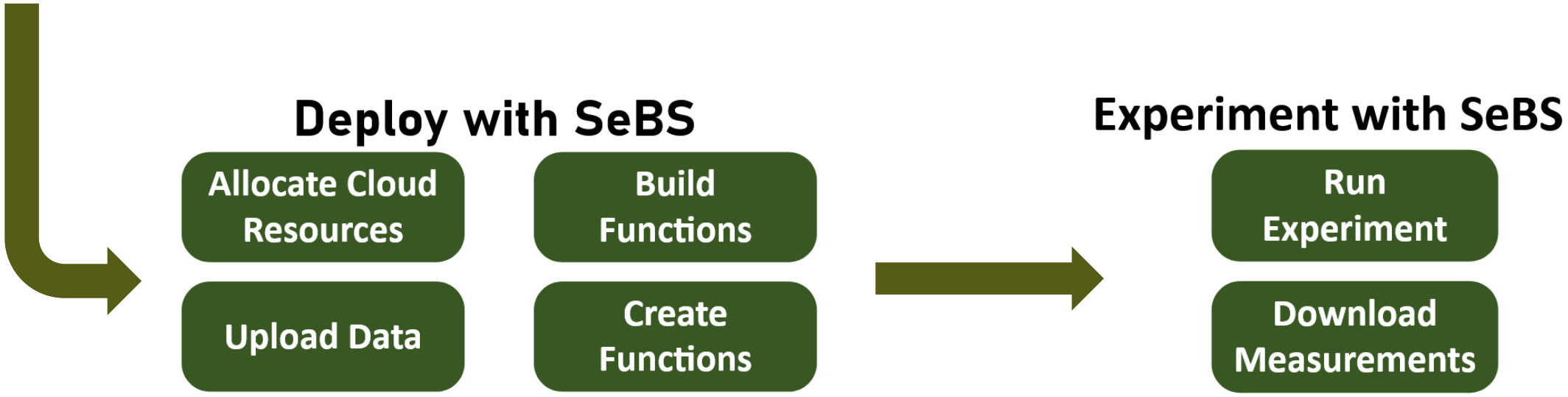
"Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?"



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

SeBS: The Serverless Benchmark Suite

"Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?"

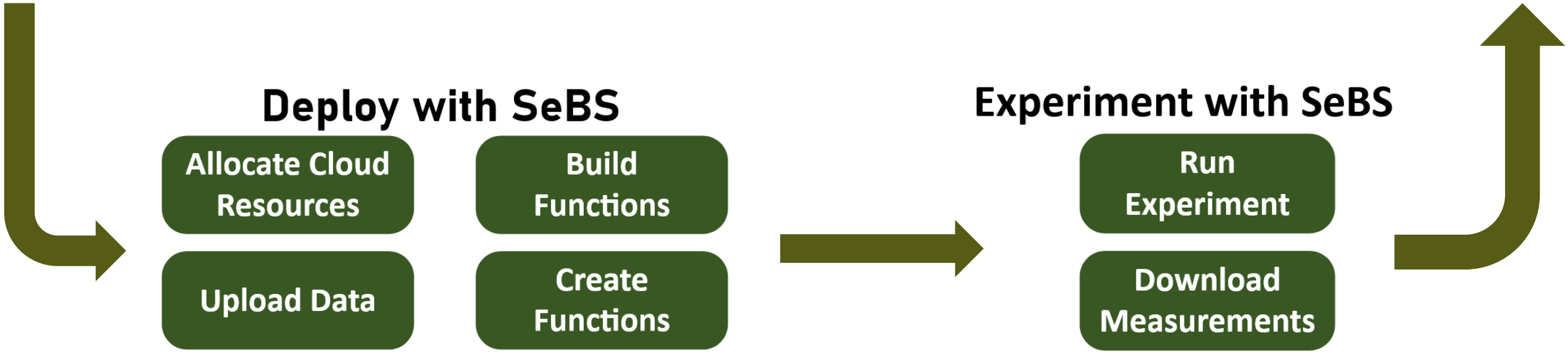


"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: The Serverless Benchmark Suite

"Which platform is best for my workload? Can my application scale? What are the performance limits of each platform?"

"The end-to-end latency is not what I expected because of lower scalability and variable I/O performance."

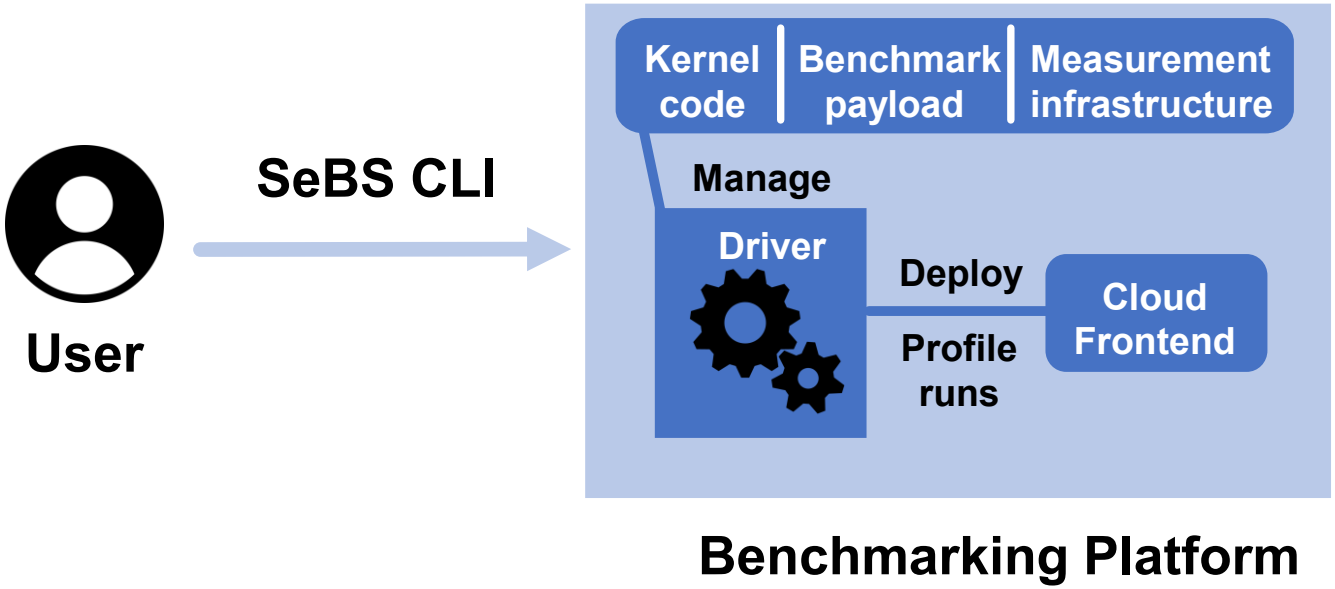


"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: Architecture

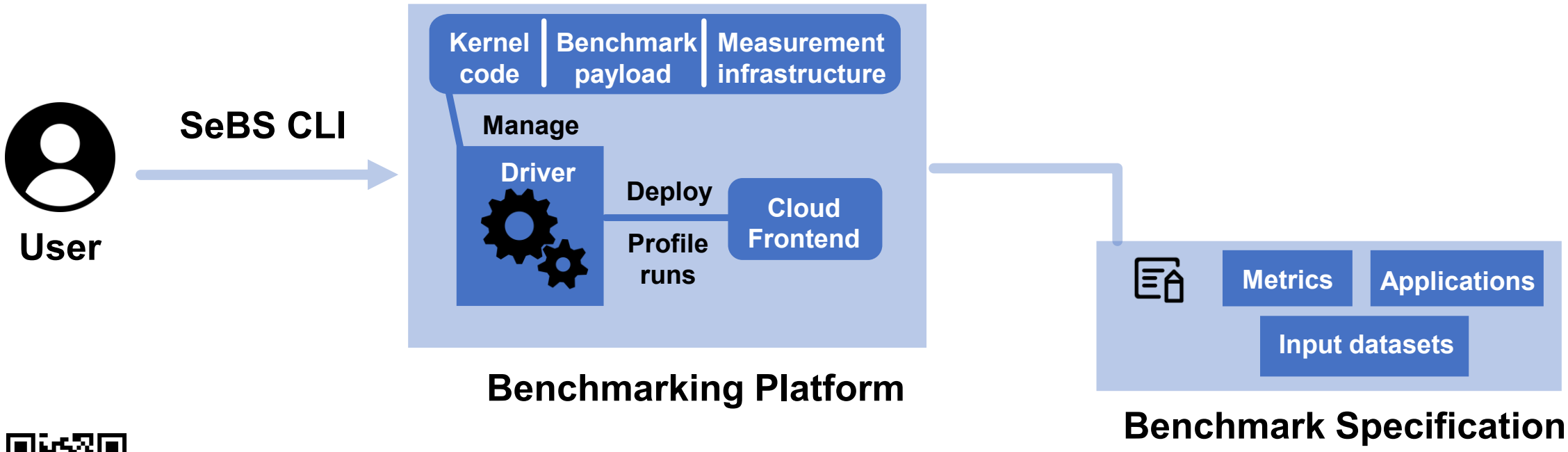
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

SeBS: Architecture



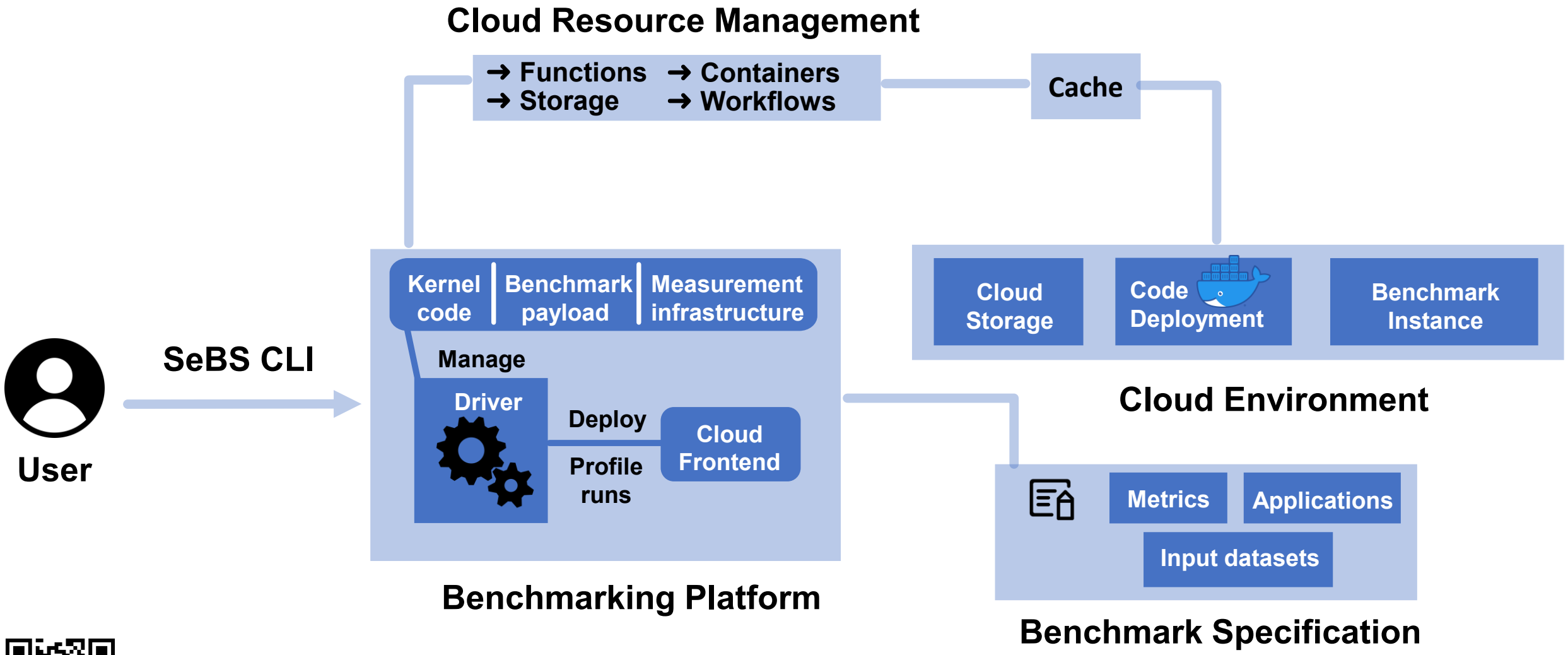
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

SeBS: Architecture



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

SeBS: Architecture



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

SeBS: Cloud Platforms

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

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

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

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

Open-source FaaS, run on any Kubernetes.

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

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

Open-source FaaS, run on any Kubernetes.

Cloudflare

Experimental support: see GitHub PRs for the branch.

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

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

Open-source FaaS, run on any Kubernetes.

Cloudflare

Experimental support: see GitHub PRs for the branch.

Local

Debugging, local measurements, hardware counters.

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

SeBS: Cloud Platforms

Big Three: similar capabilities, different semantics.

Open-source FaaS, run on any Kubernetes.

Cloudflare

Experimental support: see GitHub PRs for the branch.

Local

Debugging, local measurements, hardware counters.

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

SeBS

Introducing Cloudflare Workers: Run JavaScript Service Workers at the Edge

2017-09-29

ilities,
s.

APACHE OpenWhisk

Open-source FaaS, run on any Kubernetes.

fission

Cloudflare

Experimental support: see GitHub PRs for the branch.

Local

Debugging, local measurements, hardware counters.

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

SeBS

Introducing Cloudflare Workers: Run JavaScript Service Workers at the Edge

2017-09-29

Bringing Python to Workers using Pyodide and WebAssembly

2024-04-02

Cloudflare

Local

ilities, s.

Open-source FaaS, run on any

see ch.

Debugging, local measurements, hardware counters.

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

SeBS

Introducing Cloudflare Workers: Run JavaScript Service Workers at the Edge

2017-09-29

Bringing Python to Workers using Pyodide and WebAssembly

2024-04-02

Sandboxing AI agents, 100x faster

2026-03-24

FaaS Analysis: Eviction Modeling

(1) Which factors impact container eviction?

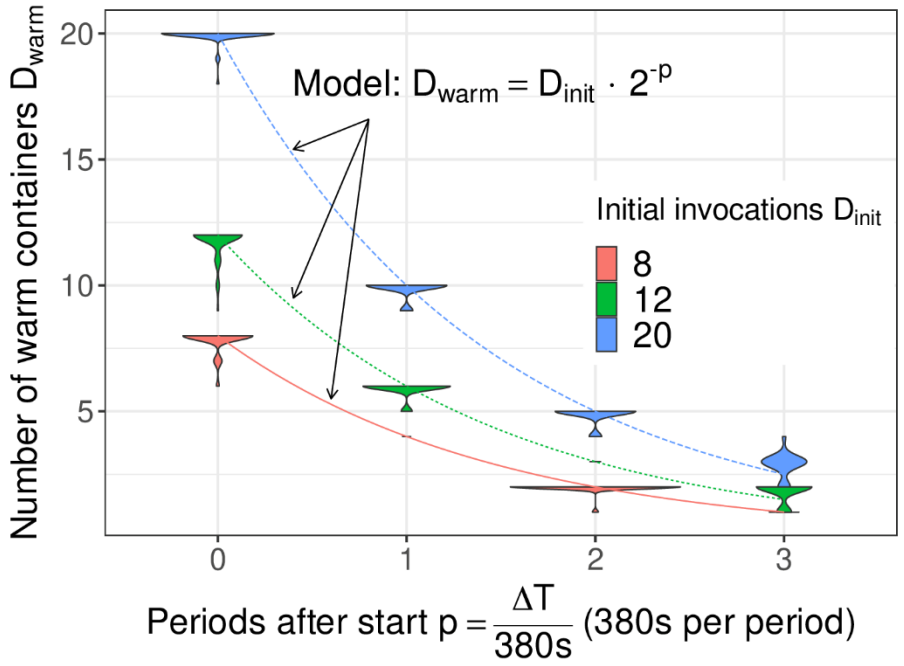
(2) Can we recover the original eviction policy?

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

FaaS Analysis: Eviction Modeling

(1) Which factors impact container eviction?

(2) Can we recover the original eviction policy?

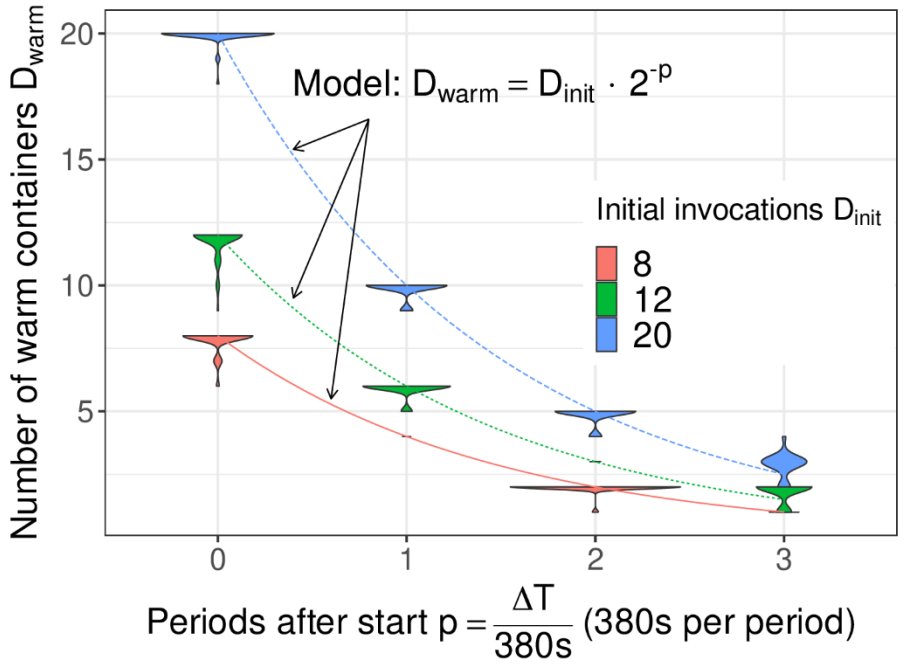


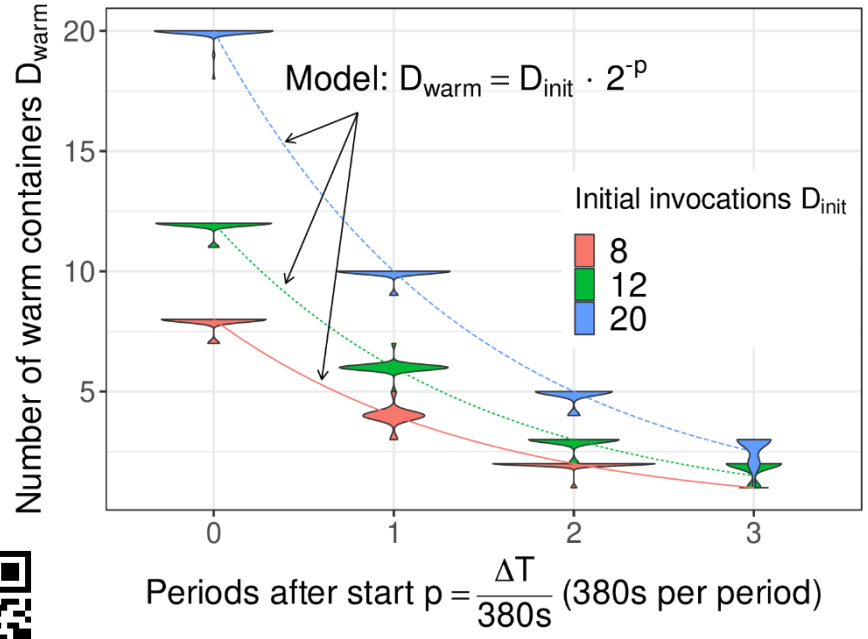
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

FaaS Analysis: Eviction Modeling

(1) Which factors impact container eviction?

(2) Can we recover the original eviction policy?



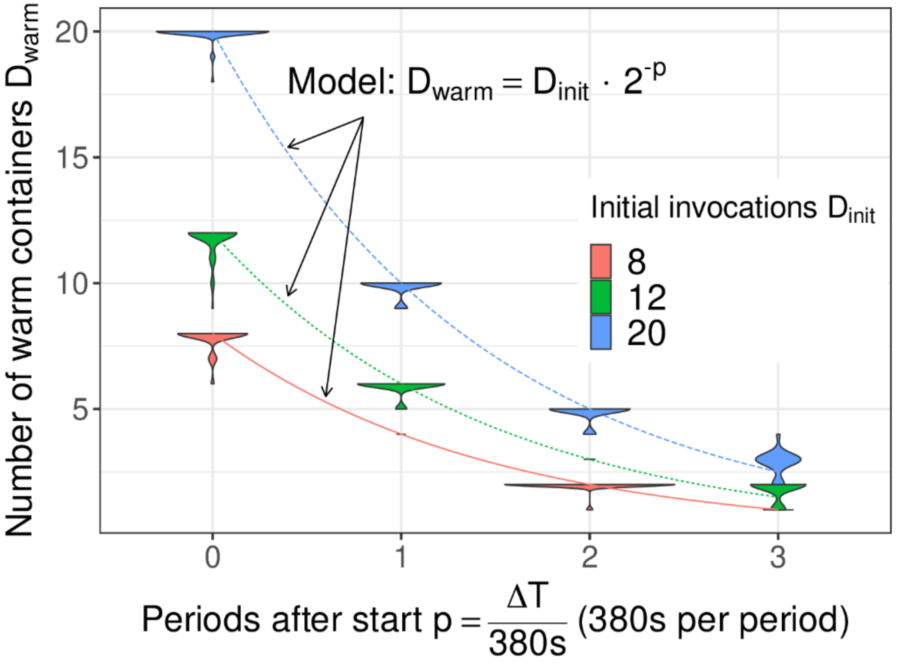


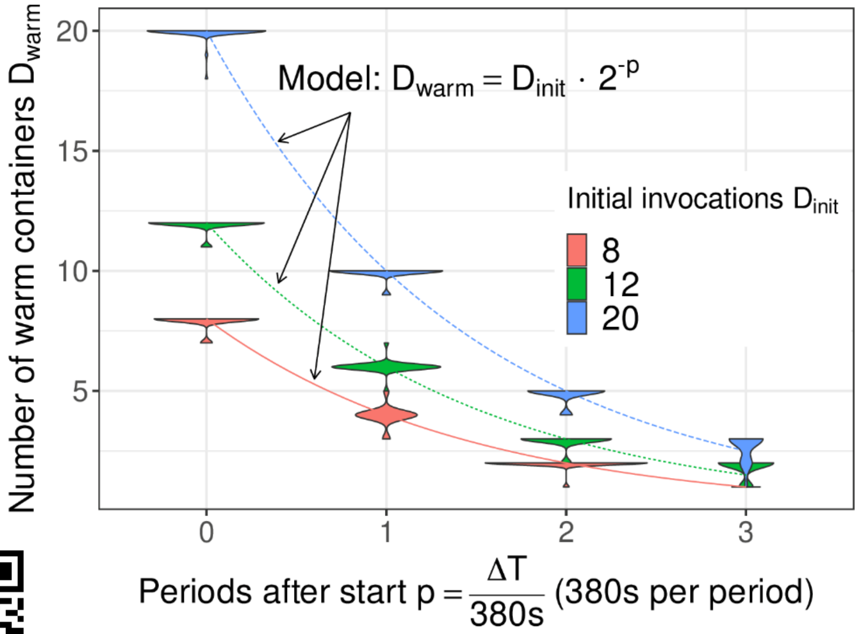
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

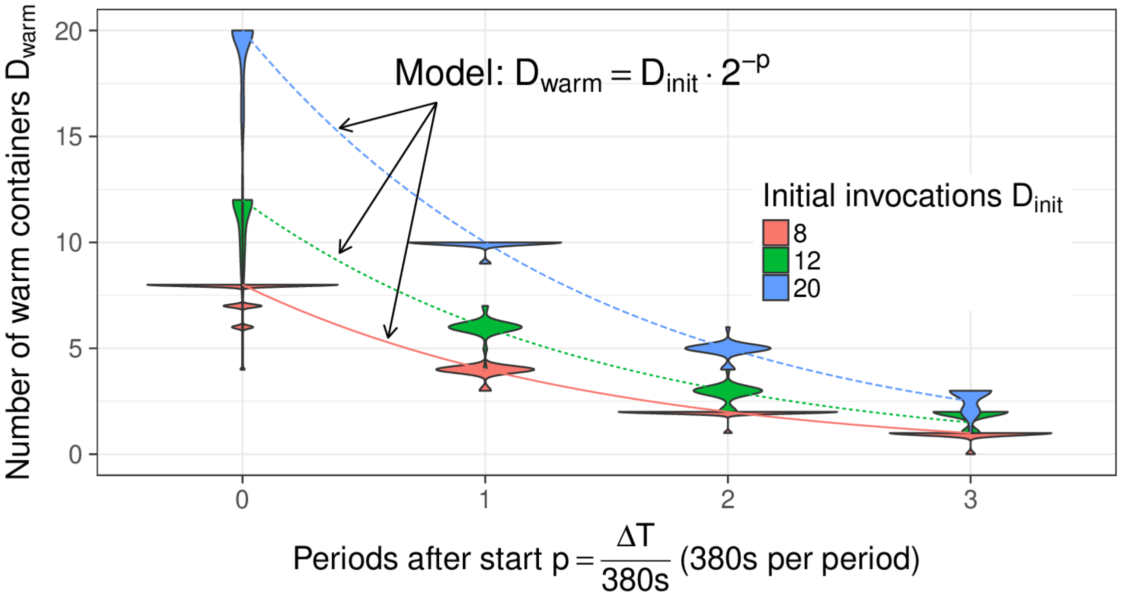
FaaS Analysis: Eviction Modeling

(1) Which factors impact container eviction?

(2) Can we recover the original eviction policy?







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

SeBS-Flow: Let the Work Flow in the Cloud

“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

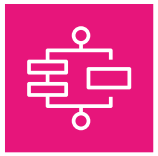
SeBS-Flow: Let the Work Flow in the Cloud

Azure Durable

```
tasks = []
for i in range(4):
  tasks.append(context.call_activity("process", i)
res = yield context.task_all(parallel_tasks)
```

“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud



AWS Step Functions

```
"init": {
  "Type": "Pass",
  "Result": "States.Array(0,
            1, 2, 3)",
  "ResultPath": "$.array",
  "Next": "map"
},
"map": {
  "Type": "Map",
  "ItemsPath": "$.array",
  "Parameters": {
    "payload.$":
      "$$.Map.Item.Value"
  },
  "Iterator": {
    "StartAt": "process",
    "States": {
      "process": {
        "Type": "Task",
        "Resource": "arn:proc",
        "Parameters": {
          "payload.$":
            "$.payload"
        },
        "End": true
      }
    }
  },
  "ResultPath": "$.res",
  "End": true
}
```

Azure Durable

```
tasks = []
for i in range(4):
  tasks.append(context.call_activity("process", i)
res = yield context.task_all(parallel_tasks)
```

“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

AWS Step Functions

```
"init": {
  "Type": "Pass",
  "Result": "States.Array(0,
            1, 2, 3)",
  "ResultPath": "$.array",
  "Next": "map"
},
"map": {
  "Type": "Map",
  "ItemsPath": "$.array",
  "Parameters": {
    "payload.$":
      "$$.Map.Item.Value"
  },
  "Iterator": {
    "StartAt": "process",
    "States": {
      "process": {
        "Type": "Task",
        "Resource": "arn:proc",
        "Parameters": {
          "payload.$":
            "$.payload"
        },
        "End": true
      }
    }
  },
  "ResultPath": "$.res",
  "End": true
}
```

Azure Durable

```
tasks = []
for i in range(4):
  tasks.append(context.call_activity("process", i)
res = yield context.task_all(parallel_tasks)
```

Google Cloud Workflows

```
"assign_array": {
    "assign": [
      {"array": [0, 1, 2, 3]}]
},
"process": {
    "call":"exp.exec.map",
    "args":{
      "workflow_id":"map",
      "arguments":"${array}"
    },
    "result":"res"
}
"separate map-workflow:"
"main": {
"params":[ "elem" ],
"steps":[
  {
  "map":{
    "call":"http.post",
    "args":{
      "url":"google.process",
      "body":{
        "payload":"${elem}"
      }
    },
    "result":"elem" }
  },
  { "ret": {
      "return":
      "${elem.body}" } }
] }
```

“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

Google Cloud Workflows

AWS Step Functions

```
"init": {
  "Type": "Pass",
  "Result": "States.Array(0,
            1, 2, 3)",
  "ResultPath": "$.array",
  "Next": "map"
},
"map": {
  "Type": "Map",
```

```
"assign_array": {
    "assign": [
      {"array": [0, 1, 2, 3]}]
},
"process": {
    "call":"exp.exec.map",
    "args":{
      "workflow_id":"map",
```

#1 Different Workflow Semantics

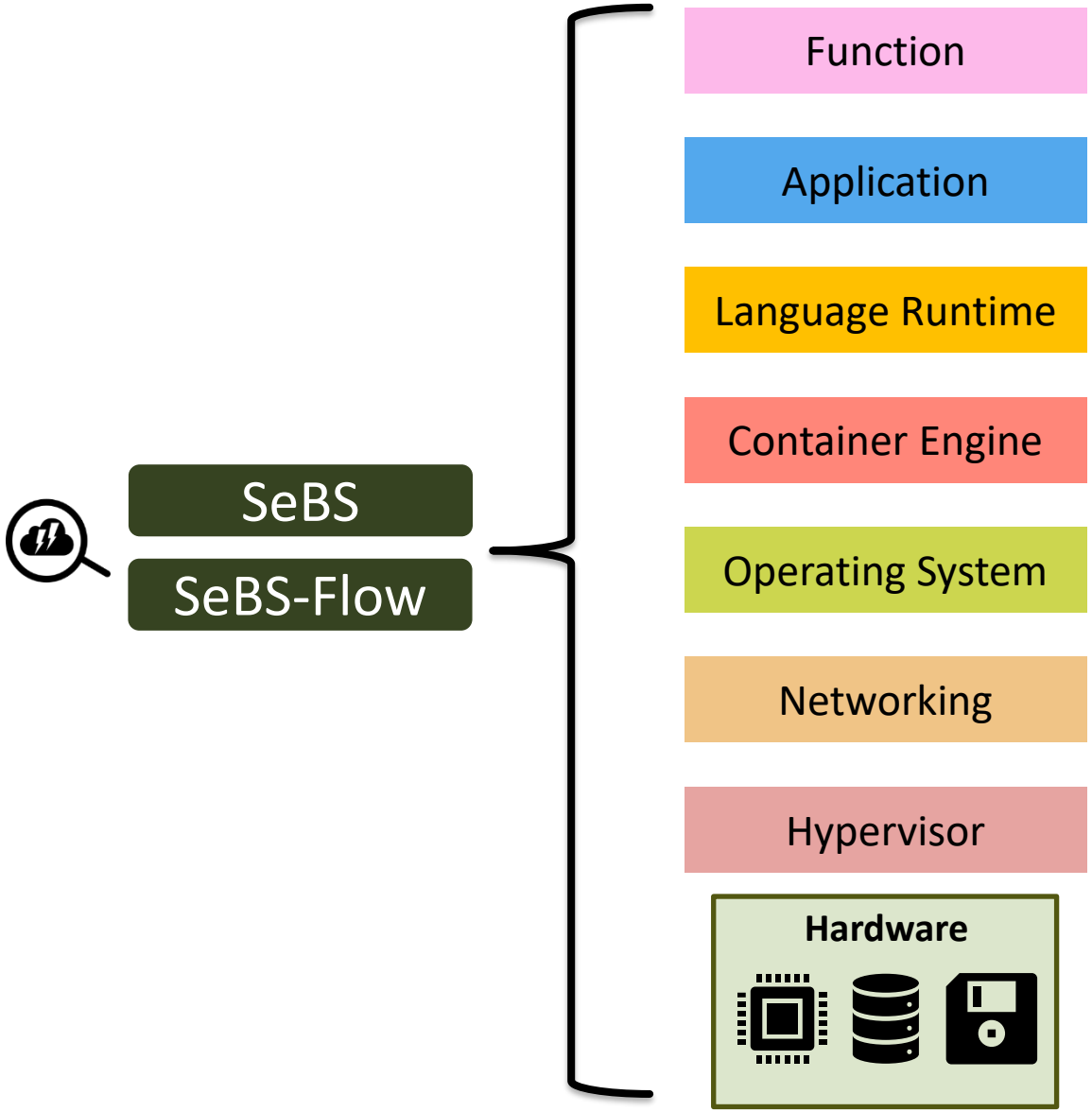
#2 Different Workflow Definitions

```
          "Parameters": {
            "payload.$":
              "$.payload"
          },
          "End": true
        }
      }
    },
    "ResultPath": "$.res",
    "End": true
  }
```

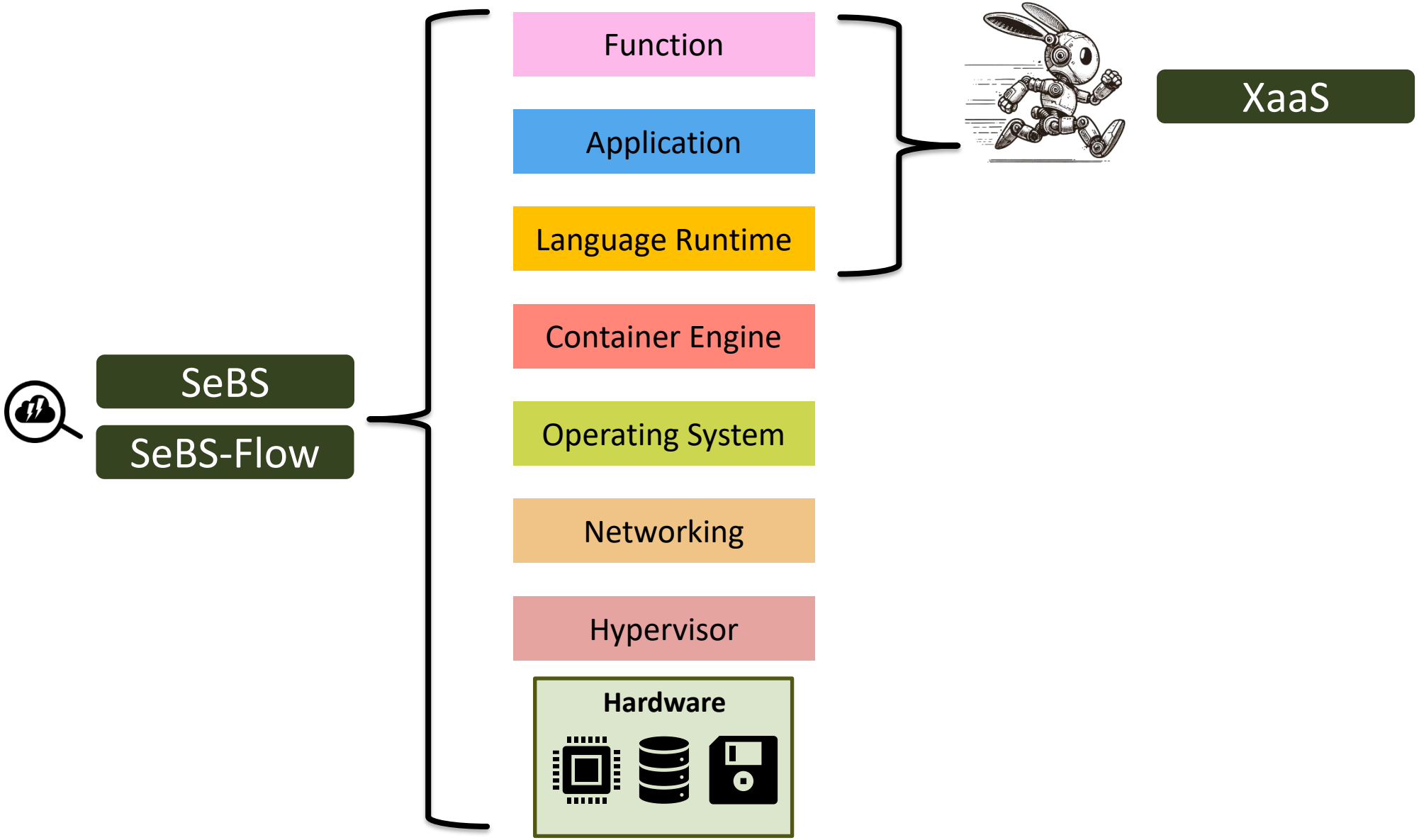
```
      "url":"google.process",
      "body":{
        "payload":"${elem}"
      }
    },
    "result":"elem" }
  },
  { "ret": {
      "return":
      "${elem.body}" } }
] }
```

"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

AXelerating Serverless Computing: Containerize!



AXelerating Serverless Computing: Containerize!



AXelerating Serverless Computing: Containerize!

XaaS: Acceleration as a Service to Enable Productive High-Performance Cloud Computing

Torsten Hoefler and Marcin Copik, *ETH Zürich, Zürich, 8093, Switzerland*

Pete Beckman, *Argonne National Laboratory, Lemont, IL, 60439, USA*

Andrew Jones, *Microsoft, Redmond, WA, 98052, USA*

Ian Foster, *Argonne National Laboratory, Lemont, IL, 60439, USA*

Manish Parashar, *Utah University, Salt Lake City, UT, 84112, USA*

Daniel Reed, *Utah University, Salt Lake City, UT, 84117, USA*

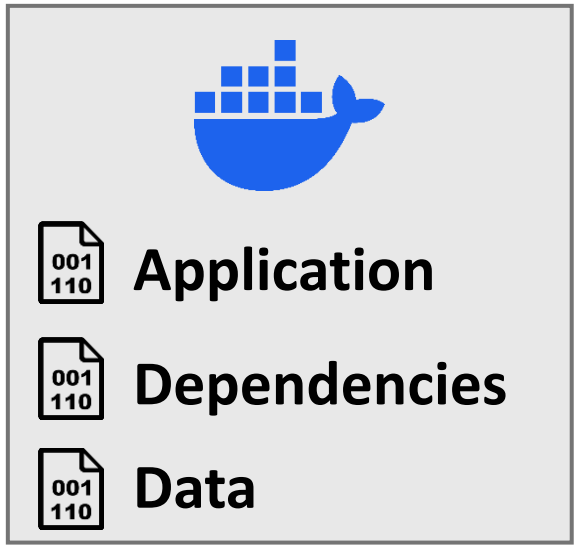
Matthias Troyer, *Microsoft, Redmond, WA, 98052, USA*

Thomas Schulthess, *Swiss National Supercomputing Centre, Lugano, 6900, Switzerland*

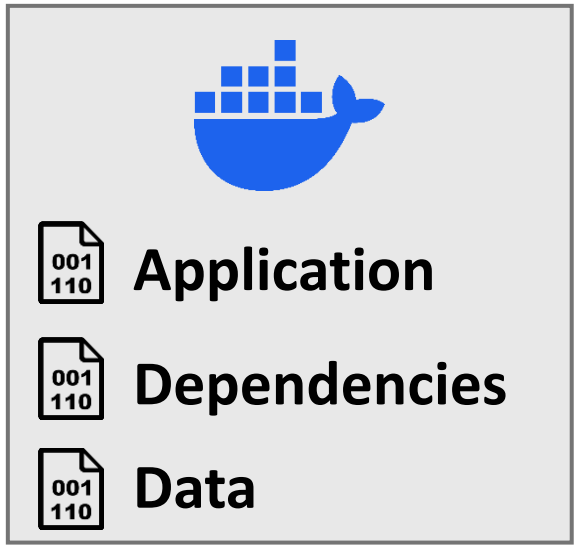
Daniel Ernst, *Nvidia, Santa Clara, CA, 95051, USA*

Jack Dongarra, *University of Tennessee, Knoxville, TN, 37996, USA*

Wonderful World of Containers



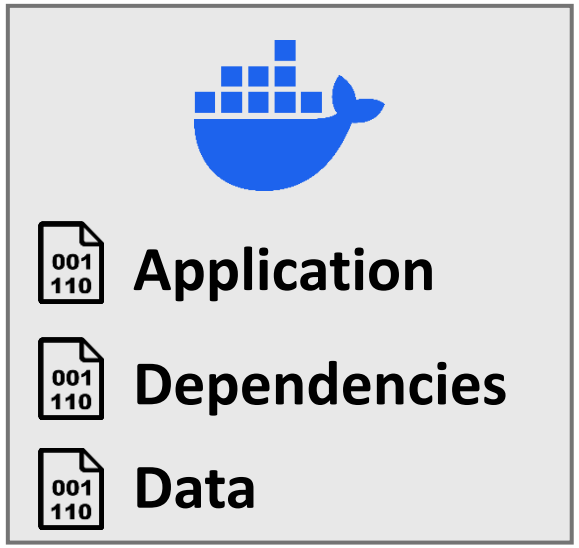
Wonderful World of Containers



docker run gromacs:2025.0

x64 & arm64

Wonderful World of Containers

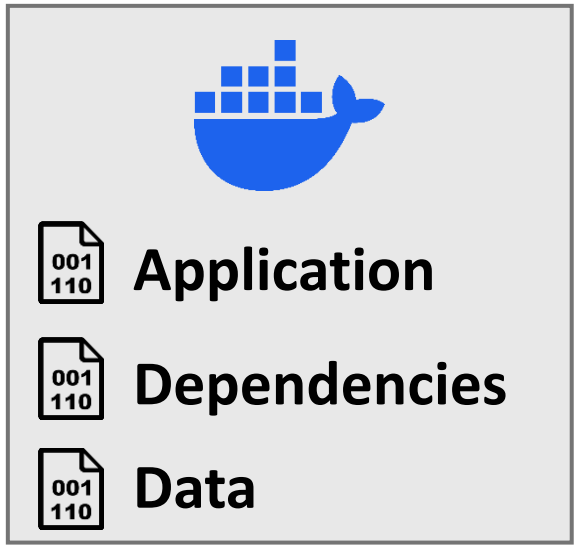


Will it work?

docker run gromacs:2025.0

x64 & arm64

Wonderful World of Containers



Will it work?

Will it be fast?

docker run gromacs:2025.0

x64 & arm64

Containers: Portable, but not Performant

gromacs:2025.0, x64 & arm64

Containers: Portable, but not Performant

gromacs:2025.0, x64 & arm64

x86 Execution Time: Intel Xeon Gold 6130

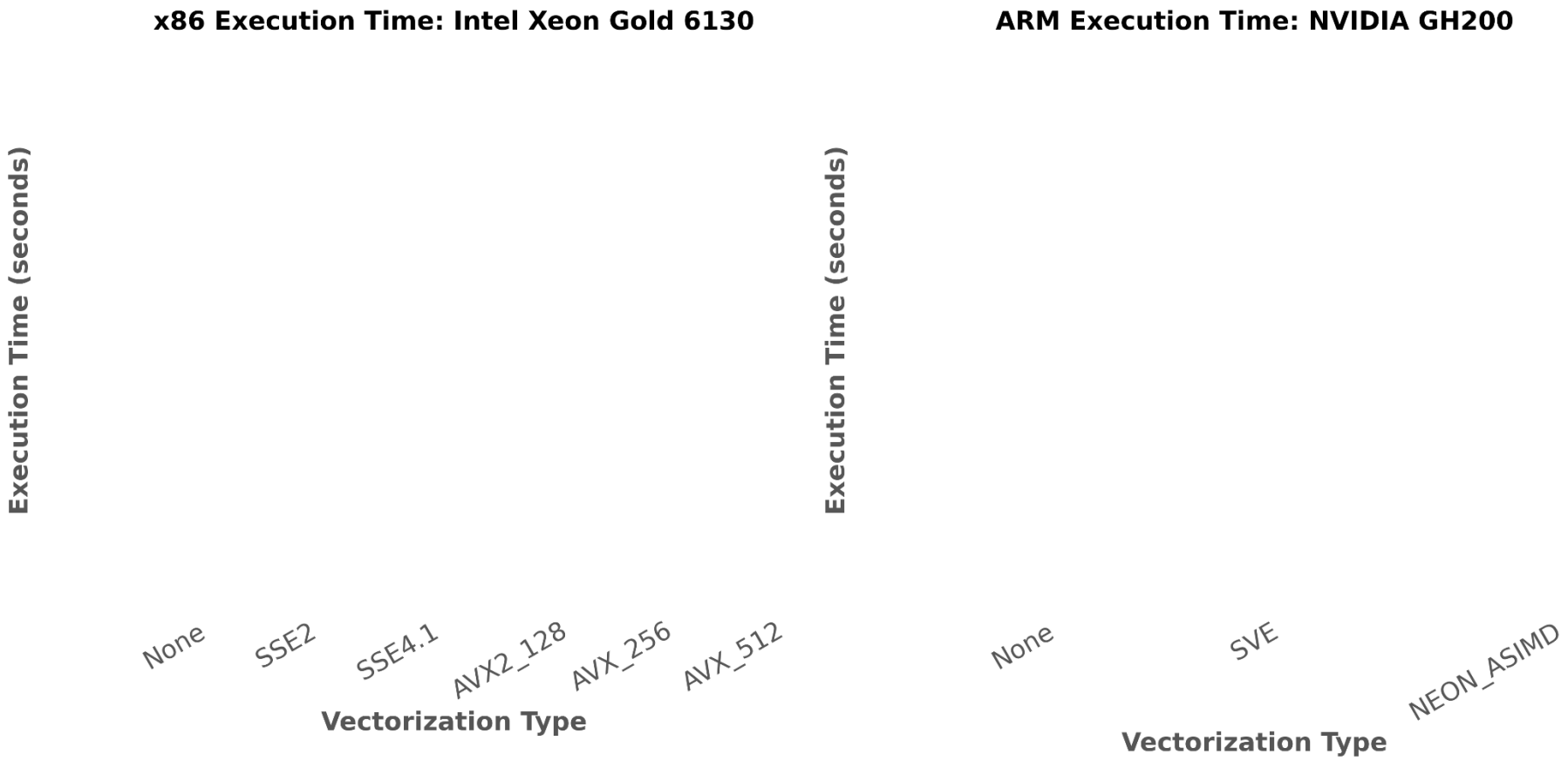
Execution Time (seconds)

ARM Execution Time: NVIDIA GH200

Execution Time (seconds)

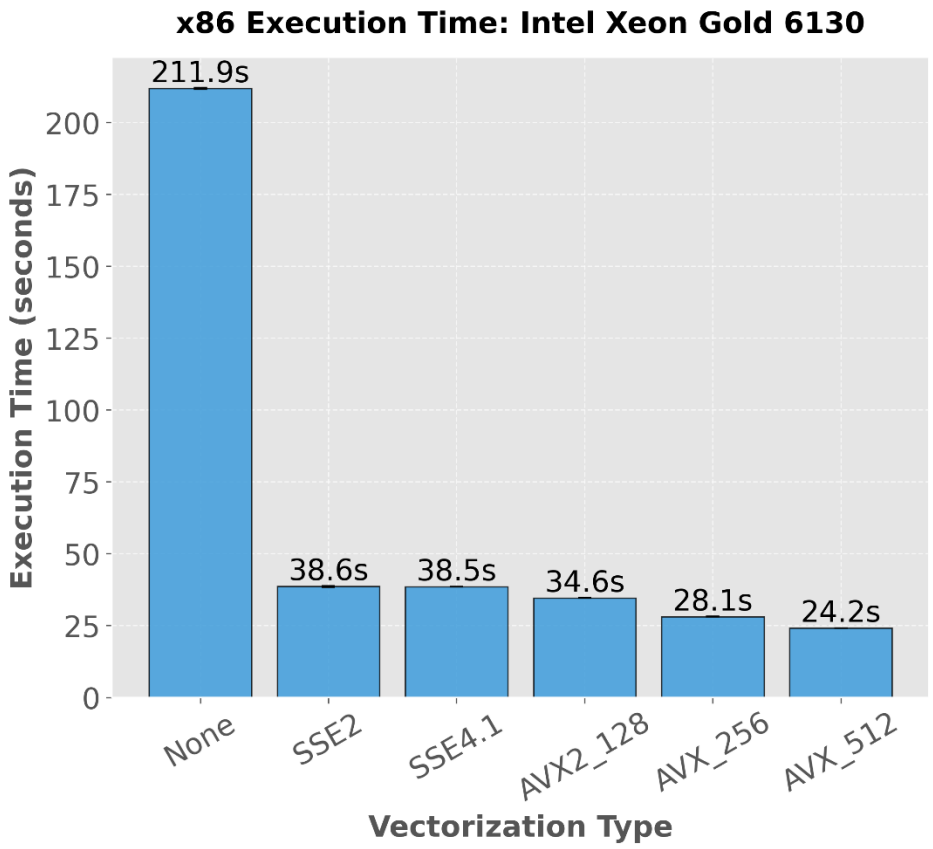
Containers: Portable, but not Performant

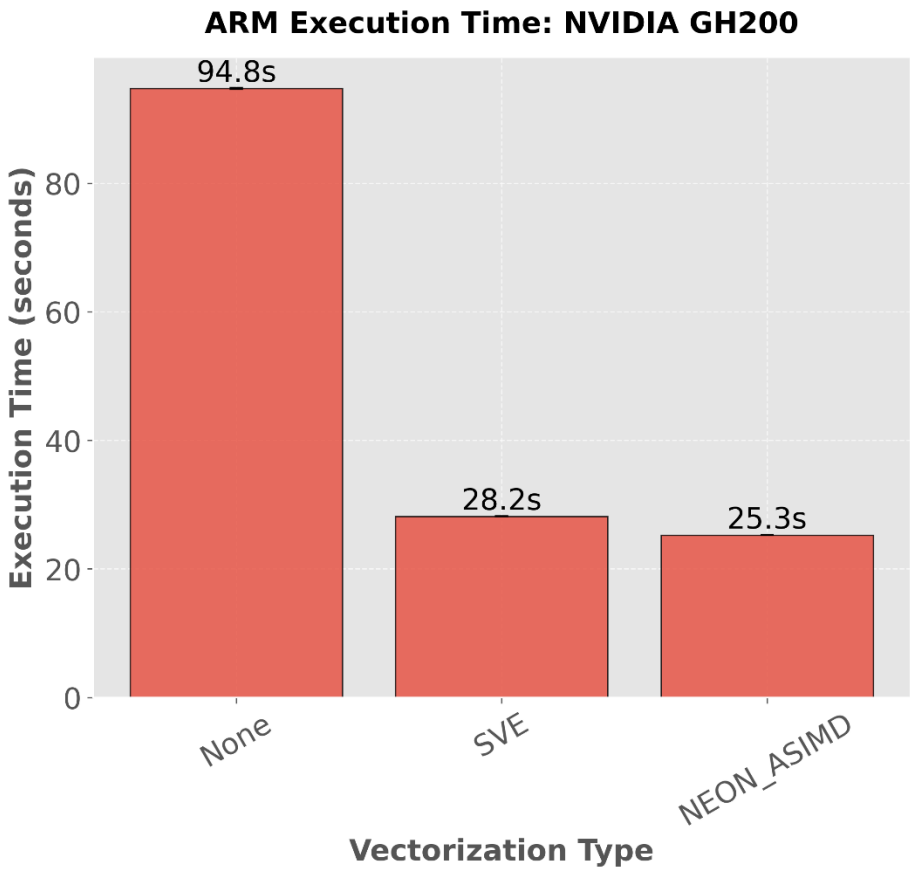
gromacs:2025.0, x64 & arm64



Containers: Portable, but not Performant

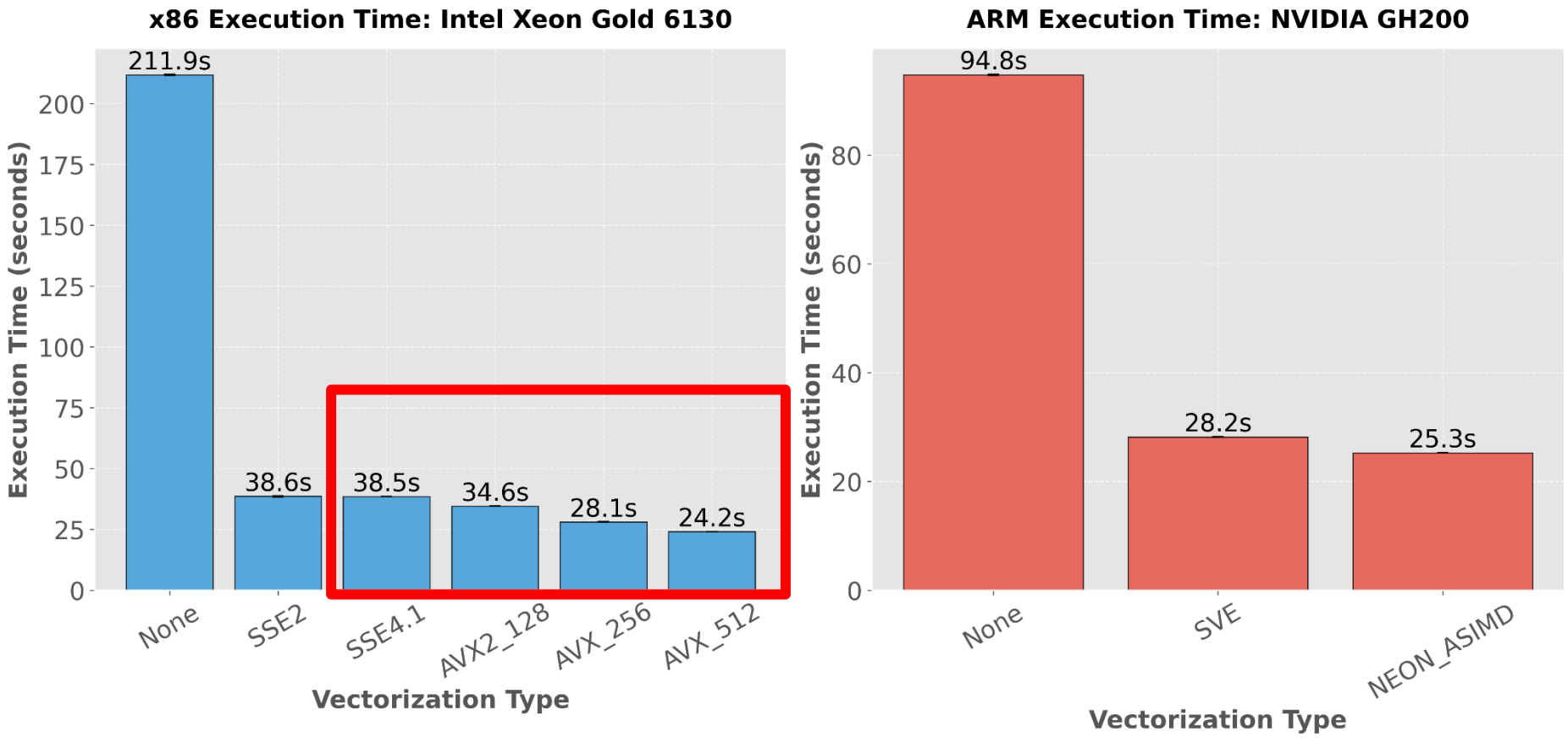
gromacs:2025.0, x64 & arm64





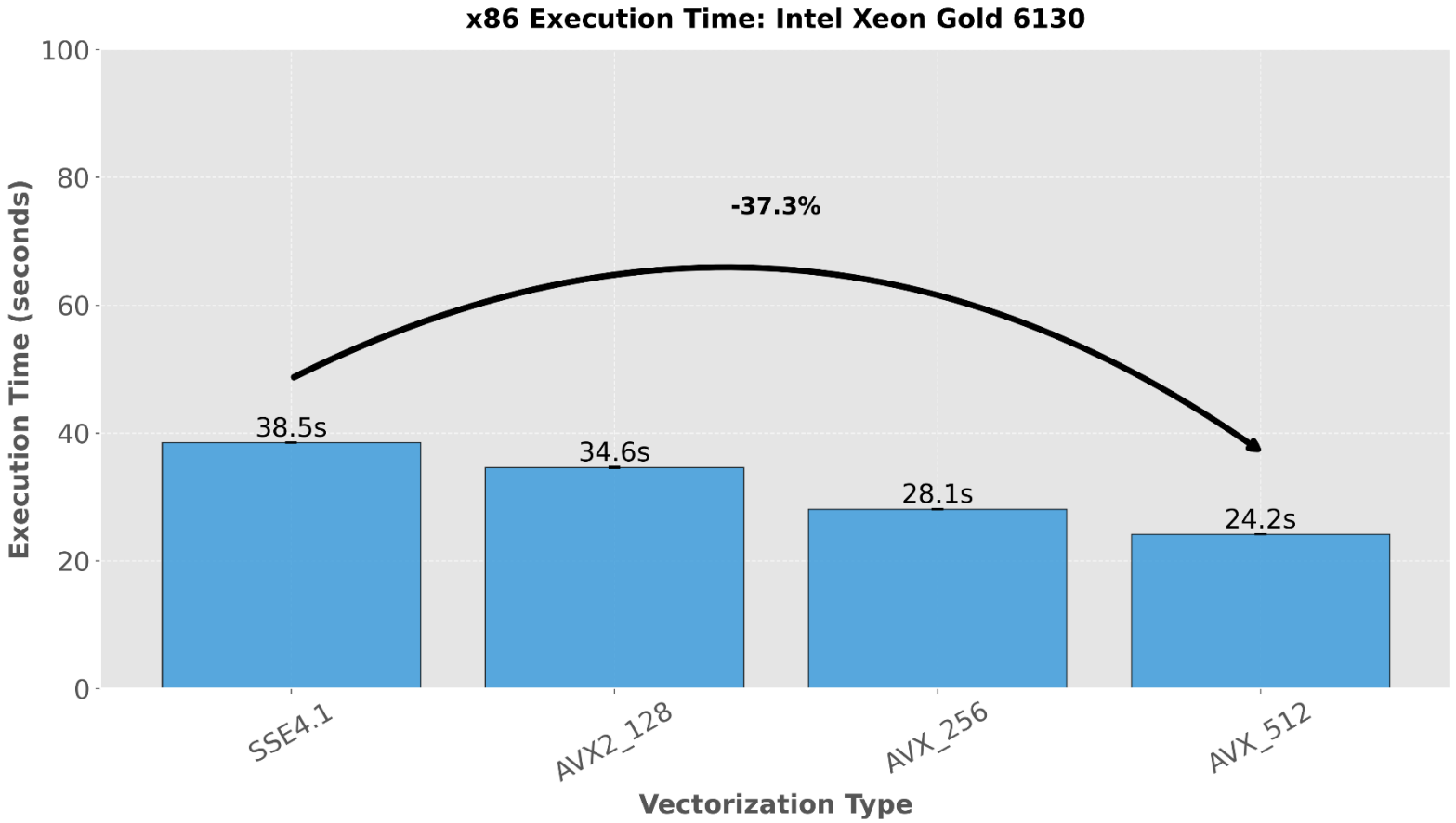
Containers: Portable, but not Performant

gromacs:2025.0, x64 & arm64

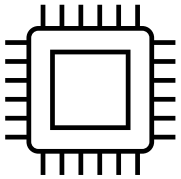


Containers: Portable, but not Performant

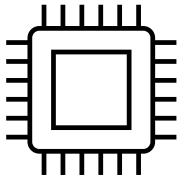
gromacs:2025.0, x64 & arm64

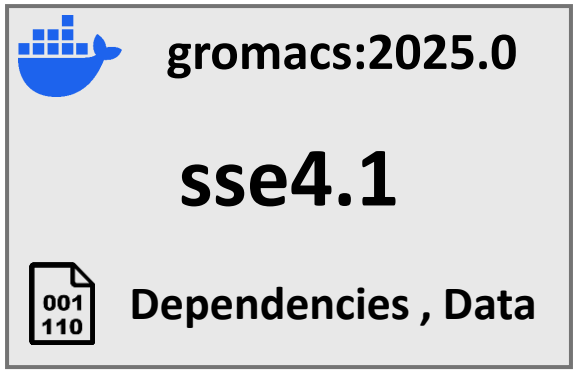


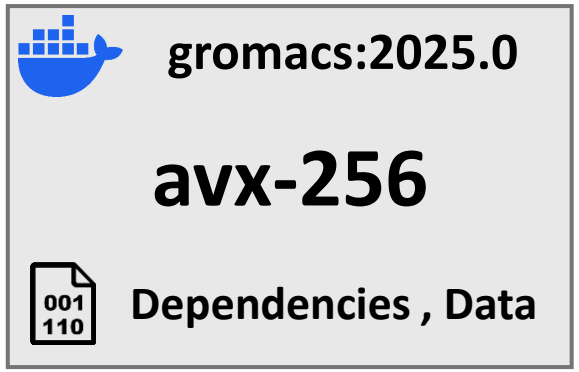
Just One More Container…

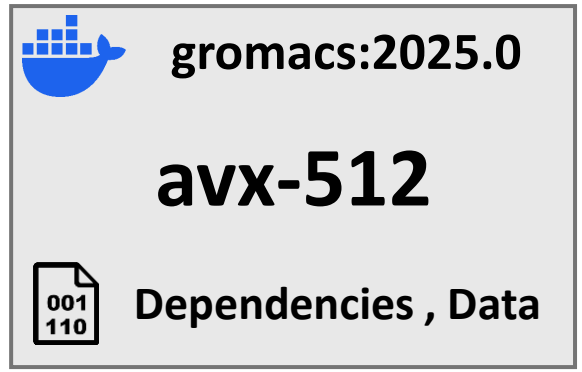


Just One More Container…

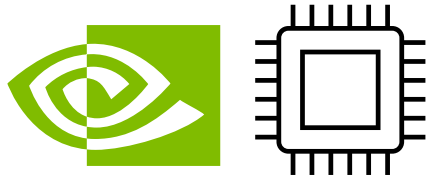


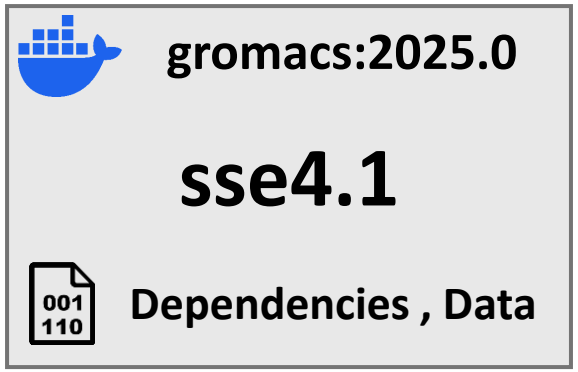


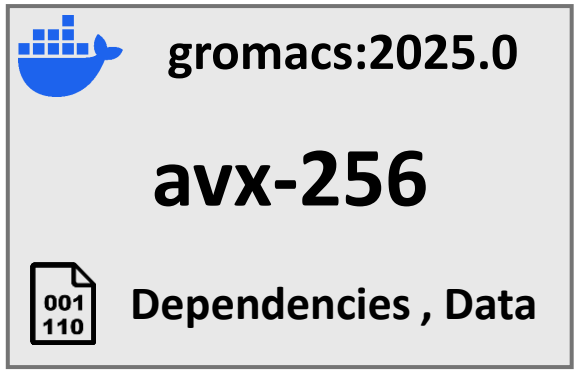


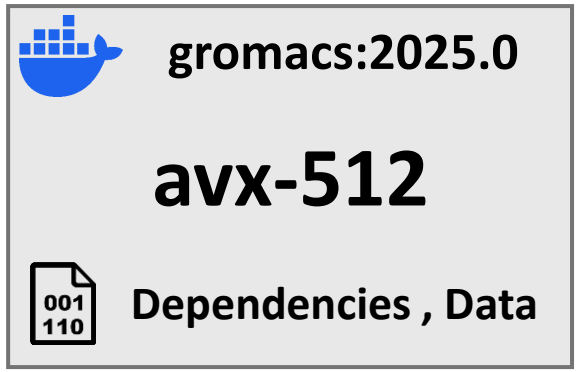


Just One More Container…

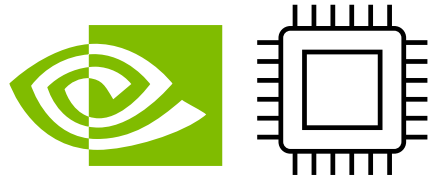








Just One More Container…



gromacs:2025.0

sse4.1

Dependencies , Data

gromacs:2025.0

avx-128

Dependencies , Data

gromacs:2025.0

avx-256

Dependencies , Data

gromacs:2025.0

avx-512

Dependencies , Data

gromacs:2025.0

sse4.1-cuda

Dependencies , Data

gromacs:2025.0

avx-128-cuda

Dependencies , Data

gromacs:2025.0

avx-256-cuda

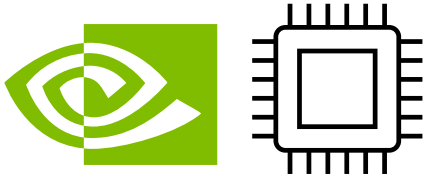
Dependencies , Data

gromacs:2025.0

avx-512-cuda

Dependencies , Data

Just One More Container…



gromacs:2025.0

sse4.1

gromacs:2025.0

sse4.1-cuda

001 110 Dependencies , Data

gromacs:2025.0

avx-128

gromacs:2025.0

avx-128-cuda

001 110 Dependencies , Data

gromacs:2025.0

avx-256

gromacs:2025.0

avx-256-cuda

001 110 Dependencies , Data

gromacs:2025.0

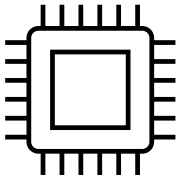
avx-512

gromacs:2025.0

avx-512-cuda

001 110 Dependencies , Data

Just One More Container…

gromacs:2025.0

sse4.1

gromacs:2025.0

sse4.1-cuda

001 110 Dependencies , Data

gromacs:2025.0

avx-128

gromacs:2025.0

avx-128-cuda

001 110 Dependencies , Data

gromacs:2025.0

avx-256

gromacs:2025.0

avx-256-cuda

001 110 Dependencies , Data

gromacs:2025.0

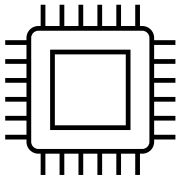
avx-512

gromacs:2025.0

avx-512-cuda

001 110 Dependencies , Data

Just One More Container…

gromacs:2025.0

sse4.1

gromacs:2025.0

sse4.1-cuda

gromacs:2025.0

sse4.1-rocm

001 110 Dependencies , Data

gromacs:2025.0

avx-128

gromacs:2025.0

avx-128-cuda

gromacs:2025.0

avx-128-rocm

001 110 Dependencies , Data

gromacs:2025.0

avx-256

gromacs:2025.0

avx-256-cuda

gromacs:2025.0

avx-256-rocm

001 110 Dependencies , Data

gromacs:2025.0

avx-512

gromacs:2025.0

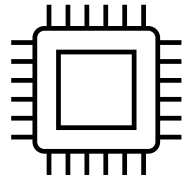
avx-512-cuda

gromacs:2025.0

avx-512-rocm

001 110 Dependencies , Data

Just One More Container…



gromacs:2025.0
sse4.1

gromacs:2025.0
sse4.1-cuda

gromacs:2025.0
sse4.1-rocm

001 110 Dependencies , Data

gromacs:2025.0
avx-128

gromacs:2025.0
avx-128-cuda

gromacs:2025.0
avx-128-rocm

001 110 Dependencies , Data

gromacs:2025.0
avx-256

gromacs:2025.0
avx-256-cuda

gromacs:2025.0
avx-256-rocm

001 110 Dependencies , Data

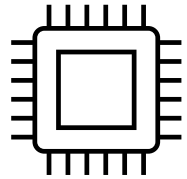
gromacs:2025.0
avx-512

gromacs:2025.0
avx-512-cuda

gromacs:2025.0
avx-512-rocm

001 110 Dependencies , Data

Just One More Container…



gromacs:2025.0 **sse4.1**

gromacs:2025.0 **sse4.1-cuda**

gromacs:2025.0 **sse4.1-rocm**

gromacs:2025.0 **sse4.1-sycl**

001 110 Dependencies , Data

gromacs:2025.0 **avx-128**

gromacs:2025.0 **avx-128-cuda**

gromacs:2025.0 **avx-128-rocm**

gromacs:2025.0 **avx-128-sycl**

001 110 Dependencies , Data

gromacs:2025.0 **avx-256**

gromacs:2025.0 **avx-256-cuda**

gromacs:2025.0 **avx-256-rocm**

gromacs:2025.0 **avx-256-sycl**

001 110 Dependencies , Data

gromacs:2025.0 **avx-512**

gromacs:2025.0 **avx-512-cuda**

gromacs:2025.0 **avx-512-rocm**

gromacs:2025.0 **avx-512-sycl**

001 110 Dependencies , Data

Just One More Container…

Which BLAS library should I use? Which FFT library?

gromacs:2025.0
sse4.1-cuda

gromacs:2025.0
avx-128-cuda

gromacs:2025.0
avx-256-cuda

gromacs:2025.0
avx-512-cuda

gromacs:2025.0
sse4.1-rocm

gromacs:2025.0
avx-128-rocm

gromacs:2025.0
avx-256-rocm

gromacs:2025.0
avx-512-rocm

gromacs:2025.0
sse4.1-sycl

Dependencies , Data

gromacs:2025.0
avx-128-sycl

Dependencies , Data

gromacs:2025.0
avx-256-sycl

Dependencies , Data

gromacs:2025.0
avx-512-sycl

Dependencies , Data

Just One More Container…

Which BLAS library should I use? Which FFT library?

Should we use OpenMP? Which MPI?

gromacs:2025.0

sse4.1-rocm

avx-128-rocm

avx-256-rocm

avx-512-rocm

sse4.1-sycl

avx-128-sycl

avx-256-sycl

avx-512-sycl

Dependencies , Data

Just One More Container…

Which BLAS library should I use? Which FFT library?

Should we use OpenMP? Which MPI?

What if container has CUDA 12.9, and my system driver is at 12.1?
For which CUDA architecture should we emit code?

HPC Specialization Points

HPC Specialization Points

Vectorization

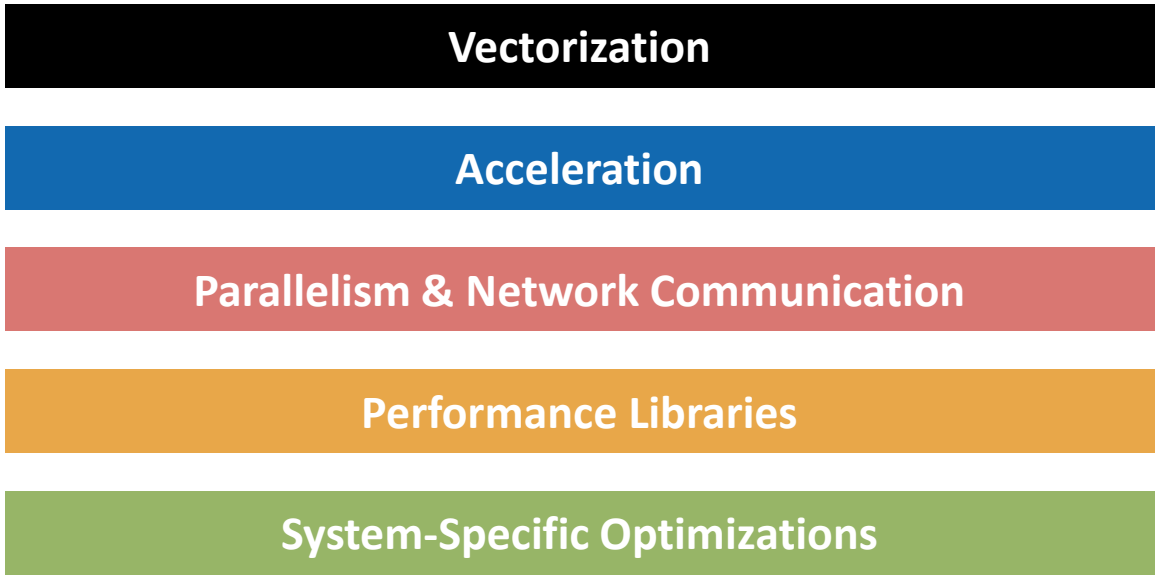
Acceleration

Parallelism & Network Communication

Performance Libraries

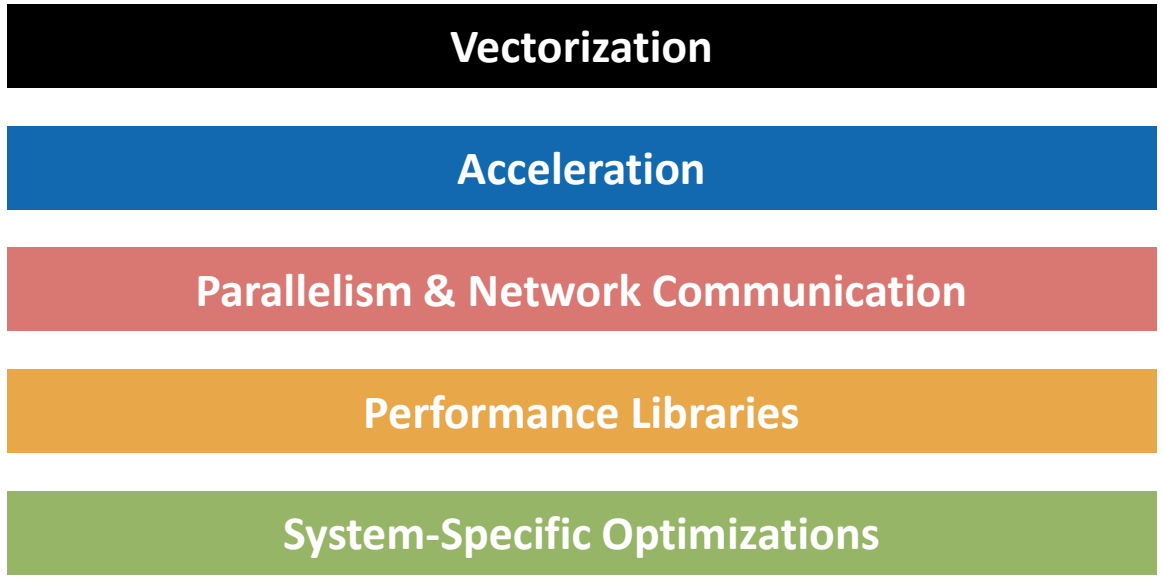
System-Specific Optimizations

HPC Specialization Points



Specialization Points

HPC Specialization Points



Specialization Points

Difficult to build one binary that supports all possible options!

HPC Specialization Points

Vectorization

Acceleration

Parallelism & Network Communication

Performance Libraries

System-Specific Optimizations

Specialization Points

WheelNext

Difficult to build one binary that supports all possible options!

HPC Specialization Points

Vectorization

Acceleration

Parallelism & Network Communication

Performance Libraries

System-Specific Optimizations

Specialization Points

Difficult to build one binary that supports all possible options!

WheelNext

blas-lapack-variant-provider

This is a plugin for the proposed wheel variants implementation that provides BLAS/LAPACK library selection.

This plugin always returns a fixed list of supported variants, and so can be used as a build-time plugin.

HPC Specialization Points

Vectorization

Acceleration

Parallelism & Network Communication

Performance Libraries

System-Specific Optimizations

Specialization Points

Difficult to build one binary that supports all possible options!

WheelNext

blas-lapack-variant-provider

This is a plugin for the proposed wheel variants implementation that provides BLAS/LAPACK library selection.

This plugin always returns a fixed list of supported variants, and so can be used as a build-time plugin.

HPC is not only Python!
We need a **general-purpose** solution.

How to Provide Portability for HPC Software?

How to Provide Portability for HPC Software?

Building Source

How to Provide Portability for HPC Software?

Building Source

Relinking Dependencies

How to Provide Portability for HPC Software?

Spack

EasyBuild

Building Source

NVIDIA PTX

(Re)lowering Application

SARUS

podman

Relinking Dependencies

How to Provide Portability for HPC Software?

Spack

EasyBuild

NVIDIA
PTX

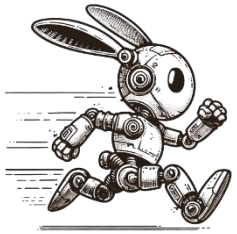
SARUS

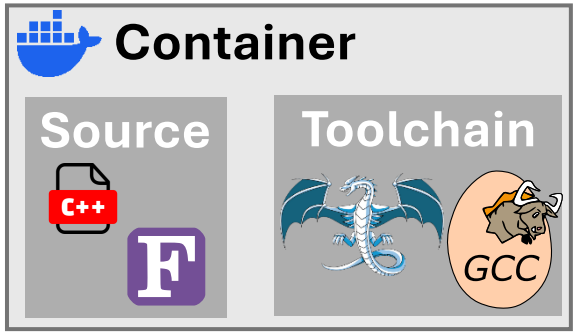
podman

Building
Source

(Re)lowering
Application

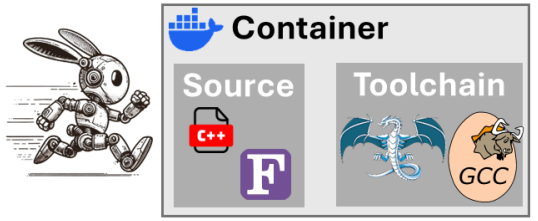
Relinking
Dependencies





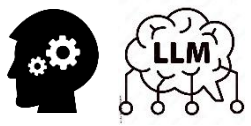
XaaS Source Container

Configuring Containers for HPC Specializations



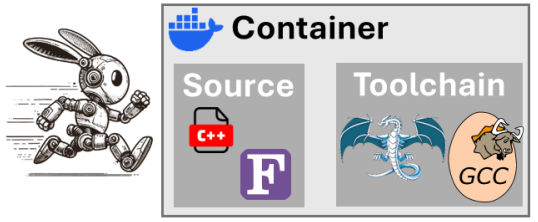
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



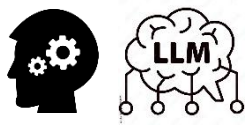
Specialization Discovery

Configuring Containers for HPC Specializations



XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



Specialization Discovery

GROMACS

In-Context Learning

Gemini 1.5 & 2.0

F1 Scores: 0.86 – 0.99

llama.cpp

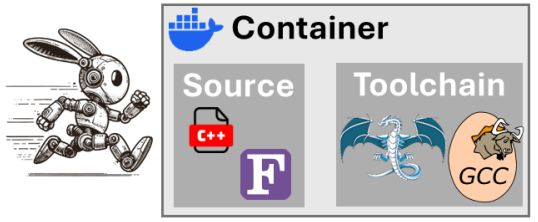
Generalization

GPT-o3

F1 Scores: 0.73 - 0.79

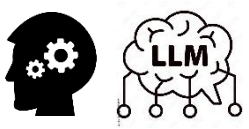
F1 Score: harmonic mean between **precision** (minimize false positives) and **recall** (minimize false negatives).

Configuring Containers for HPC Specializations



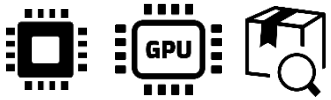
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



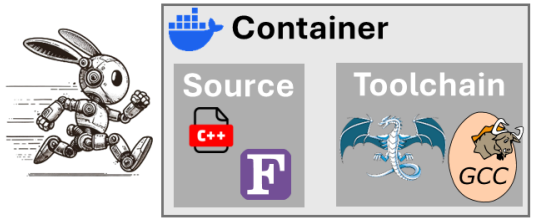
Specialization Discovery

```
{
 "CPU Info": {
   "Architecture": "x86_64",
   "Vectorization": [
    "avx512f", "avx", "avx2", "sse4_1"
   ]
 },
 "GPU Backends": {
   "CUDA": { "version": "12.1",
     "lib": ["/lib/libcuda.so.1"],
   }
 }
}
```



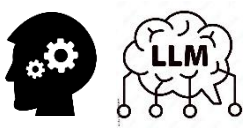
System Discovery

Configuring Containers for HPC Specializations



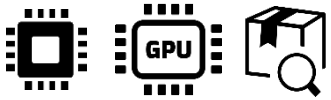
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



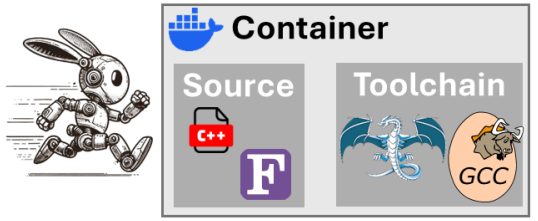
Specialization Discovery

```
{
 "CPU Info": {
   "Architecture": "x86_64",
   "Vectorization": [
    "avx512f", "avx", "avx2", "sse4_1"
   ]
 },
 "GPU Backends": {
   "CUDA": { "version": "12.1",
     "lib": ["/lib/libcuda.so.1"],
   }
 }
}
```



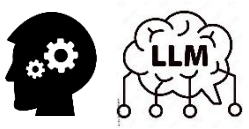
System Discovery

Configuring Containers for HPC Specializations



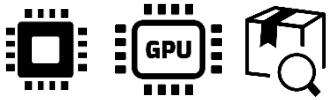
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



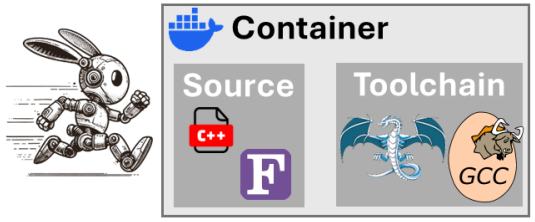
Specialization Discovery

```
{
 "CPU Info": {
   "Architecture": "x86_64",
   "Vectorization": [
    "avx512f", "avx", "avx2", "sse4_1"
   ]
 },
 "GPU Backends": {
   "CUDA": { "version": "12.1",
     "lib": ["/lib/libcuda.so.1"],
   }
 }
}
```



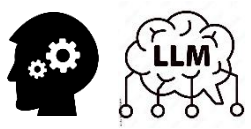
System Discovery

Configuring Containers for HPC Specializations



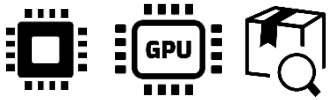
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



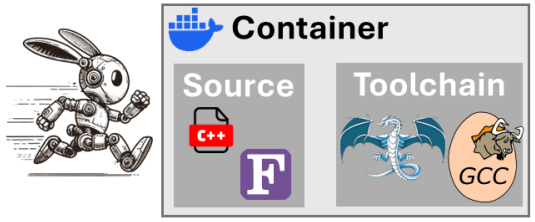
Specialization Discovery

```
{
 "CPU Info": {
   "Architecture": "x86_64",
   "Vectorization": [
    "avx512f", "avx", "avx2", "sse4_1"
   ]
 },
 "GPU Backends": {
   "CUDA": { "version": "12.1",
     "lib": ["/lib/libcuda.so.1"],
   }
 }
}
```



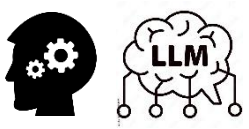
System Discovery

Configuring Containers for HPC Specializations



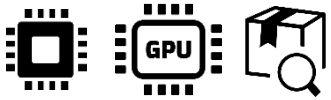
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



Specialization Discovery

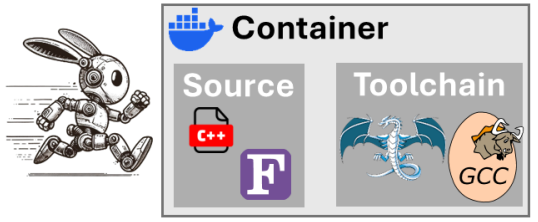
```
{
 "CPU Info": {
   "Architecture": "x86_64",
   "Vectorization": [
    "avx512f", "avx", "avx2", "sse4_1"
   ]
 },
 "GPU Backends": {
   "CUDA": { "version": "12.1",
     "lib": ["/lib/libcuda.so.1"],
   }
 }
}
```



System Discovery

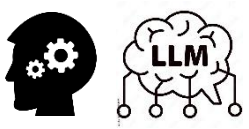
```
"vectorization_flags": {
  "SSE4.1": "-DGMX_SIMD=SSE4.1",
  "AVX_512": "-DGMX_SIMD=AVX_512",
  "AVX2_256": "-DGMX_SIMD=AVX2_256"
},
```

Configuring Containers for HPC Specializations



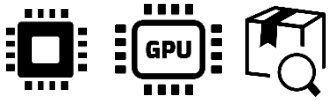
XaaS Source Container

```
{
  "gpu_build": {"value": true, "build_flag": "-DGMX_GPU"},
  "gpu_backends": {
    "CUDA": {"min_version": "12.1", "flag": "-DGMX_GPU=CUDA"},
    "HIP": {"min_version": "5.4.3", "flag": "-DGMX_GPU=HIP"}
  },
  "vectorization": {
    "None": {"flag": "-DGMX_SIMD=None"},
    "SSE4.1": {"flag": "-DGMX_SIMD=SSE4.1"},
    "AVX2_256": {"flag": "-DGMX_SIMD=AVX2_256"},
    "AVX_512": {"flag": "-DGMX_SIMD=AVX_512"},
    "ARM_NEON_ASIMD": {"flag": "-DGMX_SIMD=ARM_NEON_ASIMD"}
}}
```



Specialization Discovery

```
{
 "CPU Info": {
   "Architecture": "x86_64",
   "Vectorization": [
    "avx512f", "avx", "avx2", "sse4_1"
   ]
 },
 "GPU Backends": {
   "CUDA": { "version": "12.1",
     "lib": ["/lib/libcuda.so.1"],
   }
 }
}
```

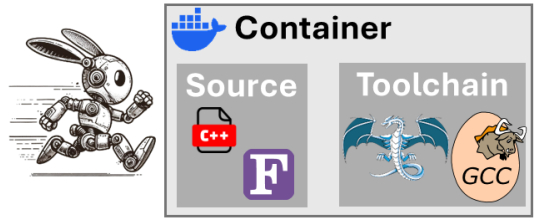


System Discovery

```
{
  "common_specialization": {
    "vectorization_flags": {
      "SSE4.1": "-DGMX_SIMD=SSE4.1",
      "AVX_512": "-DGMX_SIMD=AVX_512",
      "AVX2_256": "-DGMX_SIMD=AVX2_256"
    },
    "gpu_backends": {
      "CUDA": { "version": "12.1",
        "flag": "-DGMX_GPU=CUDA",
      },
    }}
}
```

Feature Intersection

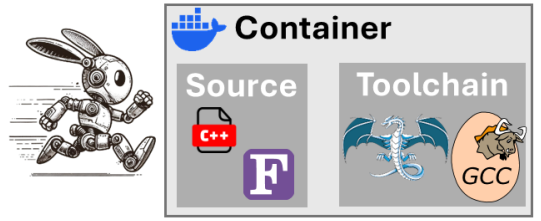
XaaS Source Containers Deployment



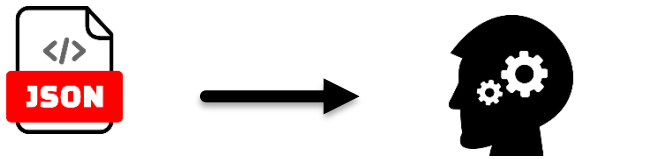
XaaS Source Container

Feature Intersection

XaaS Source Containers Deployment



XaaS Source Container

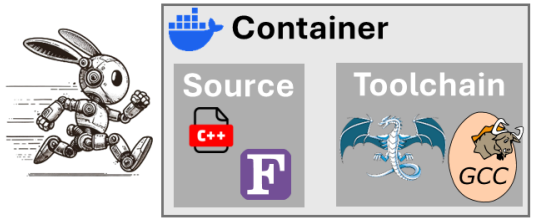


Feature Intersection

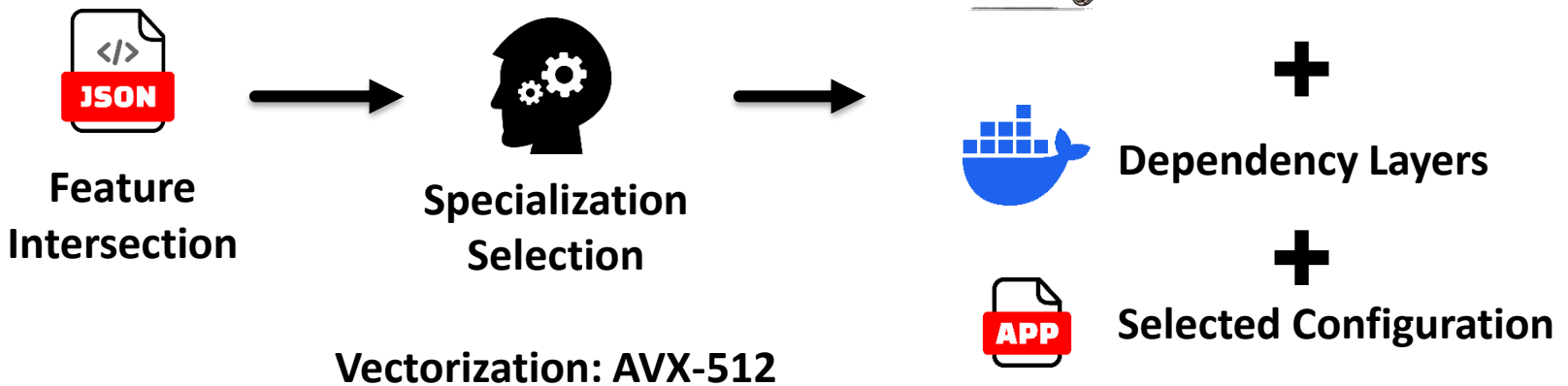
Specialization Selection

Vectorization: AVX-512
GPU: CUDA
...

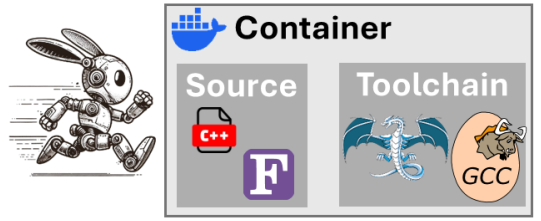
XaaS Source Containers Deployment



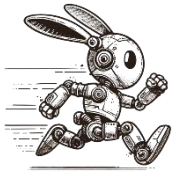
XaaS Source Container



XaaS Source Containers Deployment



XaaS Source Container



XaaS Source Container
Source Code + Dev Tools

+

Dependency Layers

+

Selected Configuration

JSON

Feature Intersection

Specialization Selection

Vectorization: AVX-512
GPU: CUDA
…

Deployed XaaS Container

How to Provide Portability for HPC Software?

Spack

EasyBuild

NVIDIA
PTX

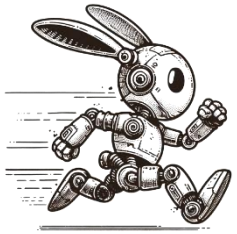
SARUS

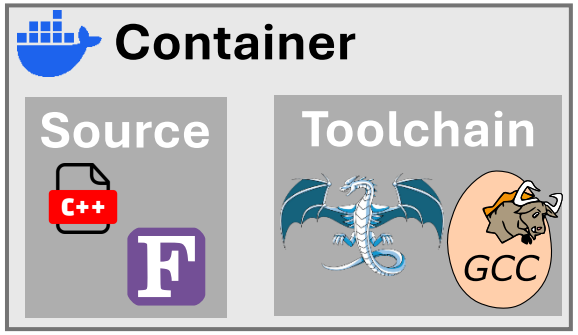
podman

Building
Source

(Re)lowering
Application

Relinking
Dependencies





XaaS Source Container

How to Provide Portability for HPC Software?

Spack

EasyBuild

NVIDIA
PTX

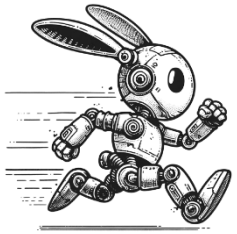
SARUS

podman

Building
Source

(Re)lowering
Application

Relinking
Dependencies



Container

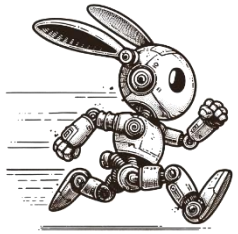
Source

C++

F

Toolchain

GCC



Container

IR

Source

C++

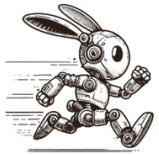
F

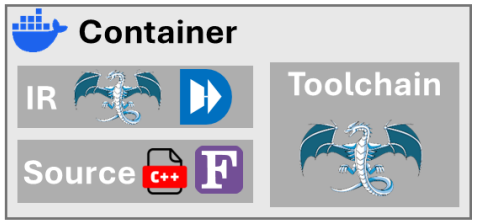
Toolchain

XaaS Source Container

XaaS IR Container

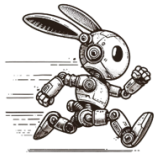
XaaS IR Containers: Eliminating Redundancy

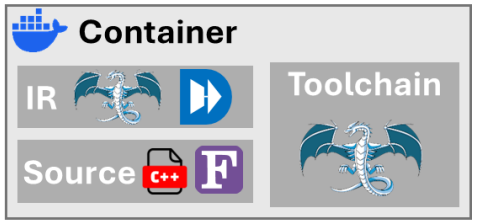




XaaS IR Container

XaaS IR Containers: Eliminating Redundancy



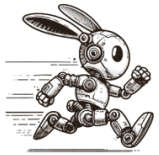


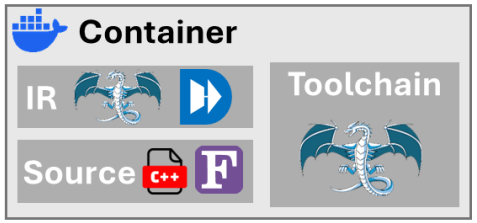
XaaS IR Container

Config № 1

AVX-256, no GPU, OpenMP

XaaS IR Containers: Eliminating Redundancy

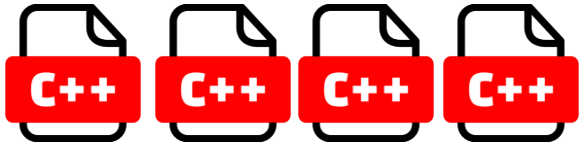




XaaS IR Container

Config № 1

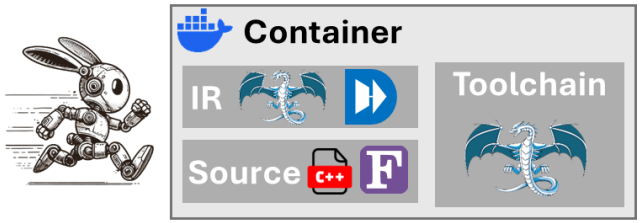
AVX-256, no GPU, OpenMP



Config № 2

AVX-512, CUDA, MPI

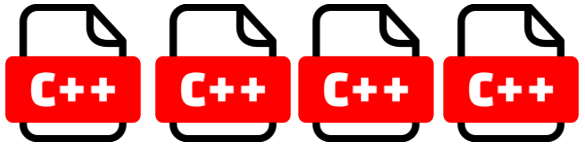
XaaS IR Containers: Eliminating Redundancy



XaaS IR Container

Config № 1

AVX-256, no GPU, OpenMP



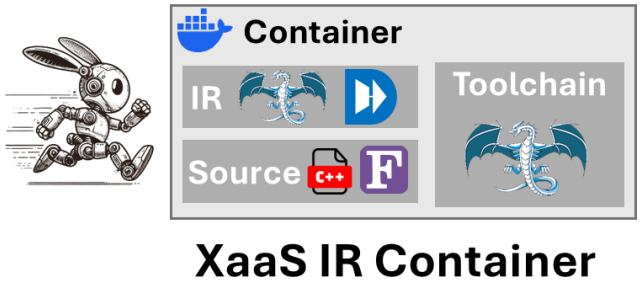
Config № 2

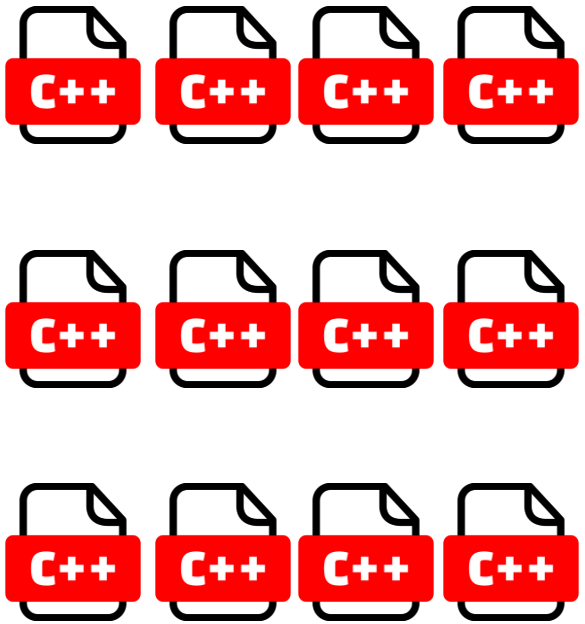
AVX-512, CUDA, MPI

Config № 3

AVX-256, CUDA, MPI

XaaS IR Containers: Eliminating Redundancy





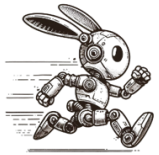
Config № 1
AVX-256, no GPU, OpenMP

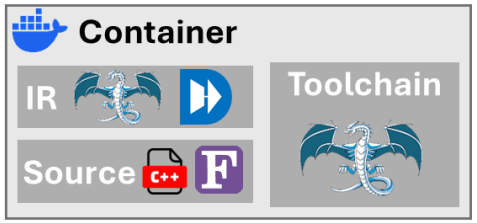
Config № 2
AVX-512, CUDA, MPI

Config № 3
AVX-256, CUDA, MPI

✕ Compile-time definitions.

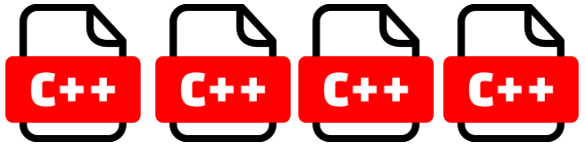
XaaS IR Containers: Eliminating Redundancy



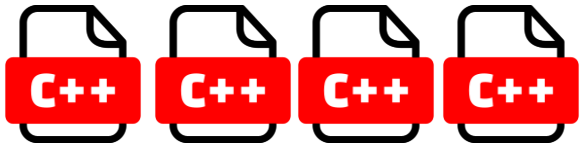


XaaS IR Container

Config № 1
AVX-256, no GPU, OpenMP



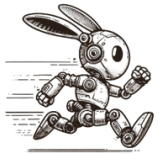
Config № 2
AVX-512, CUDA, MPI

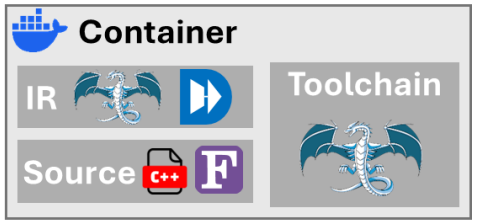


Config № 3
AVX-256, CUDA, MPI

- Compile-time definitions.
- Include paths in build directories.

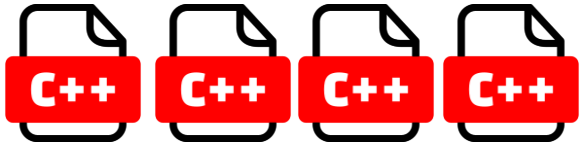
XaaS IR Containers: Eliminating Redundancy



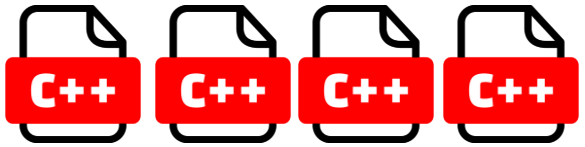


XaaS IR Container

Config № 1
AVX-256, no GPU, OpenMP



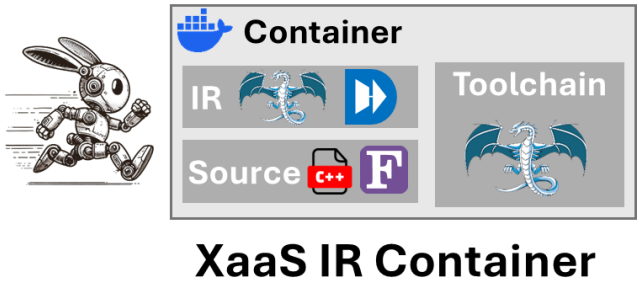
Config № 2
AVX-512, CUDA, MPI



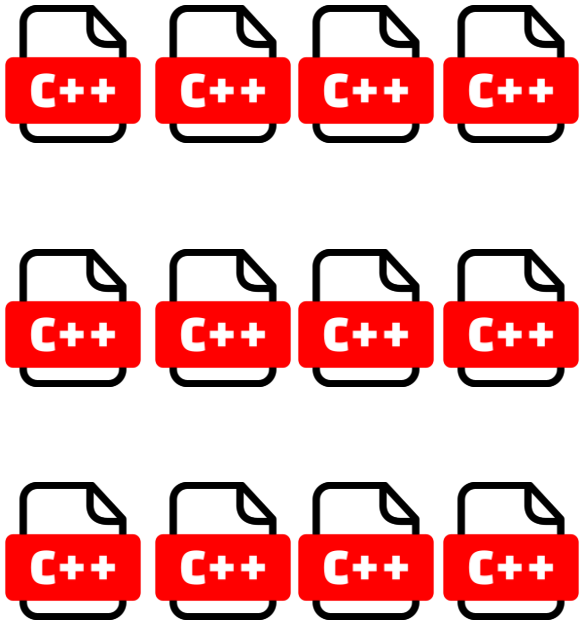
Config № 3
AVX-256, CUDA, MPI

- ✗ Compile-time definitions.
- ✗ Include paths in build directories.
- ✗ Vectorization and architecture flags.

XaaS IR Containers: Eliminating Redundancy



XaaS IR Container



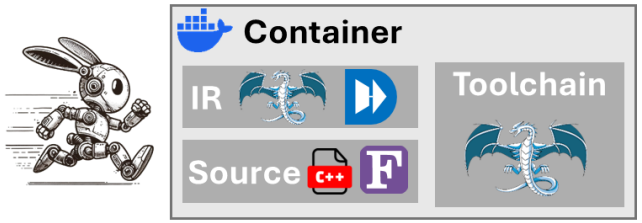
Config № 1
AVX-256, no GPU, OpenMP

Config № 2
AVX-512, CUDA, MPI

Config № 3
AVX-256, CUDA, MPI

- Compile-time definitions.
- Include paths in build directories.
- Vectorization and architecture flags.
- OpenMP flags.

XaaS IR Containers: Eliminating Redundancy



XaaS IR Container

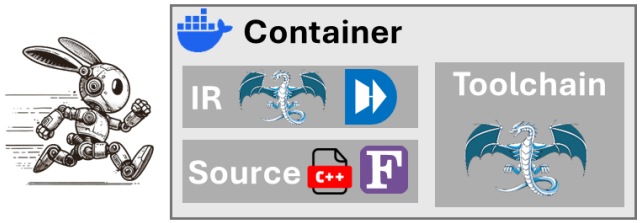
Config № 1
AVX-256, no GPU, OpenMP

Config № 2
AVX-512, CUDA, MPI

Config № 3
AVX-256, CUDA, MPI

- Compile-time definitions.
- Include paths in build directories.
- Vectorization and architecture flags.
- OpenMP flags.

XaaS IR Containers: Eliminating Redundancy



XaaS IR Container

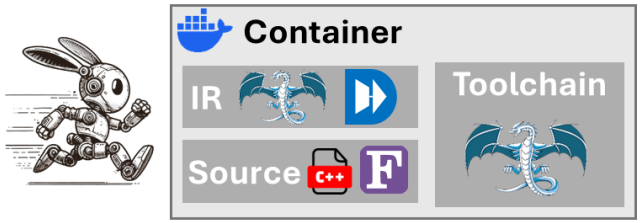
Config № 1
AVX-256, no GPU, OpenMP

Config № 2
AVX-512, CUDA, MPI

Config № 3
AVX-256, CUDA, MPI

- Compile-time definitions.
- Include paths in build directories.
- Vectorization and architecture flags.
- OpenMP flags.

XaaS IR Containers: Eliminating Redundancy



XaaS IR Container

Config № 1
AVX-256, no GPU, OpenMP

Config № 2
AVX-512, CUDA, MPI

Config № 3
AVX-256, CUDA, MPI

- ✗ Compile-time definitions.
- ✗ Include paths in build directories.
- ✗ Vectorization and architecture flags.
- ✗ OpenMP flags.

Example 1: GROMACS, CPU Vectorization (5)

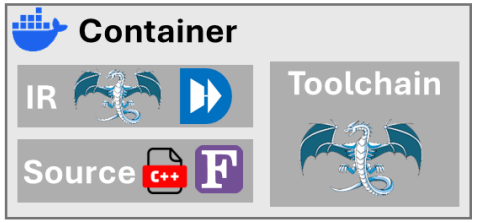
5 Build Configurations: 8710 compiled files
IR Container: 2695 IRs -> **69% reduction**

Example 2: GROMACS, CUDA + CPU Vectorization (2)

4 Build Configurations: 7052 compiled files
IR Container: 2694 IRs -> **62% reduction**

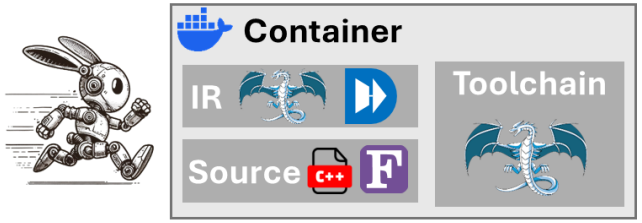
XaaS IR Containers: Build & Deploy

Container Build



XaaS IR Container

XaaS IR Containers: Build & Deploy



XaaS IR Container

Container Build

APP Config № 1

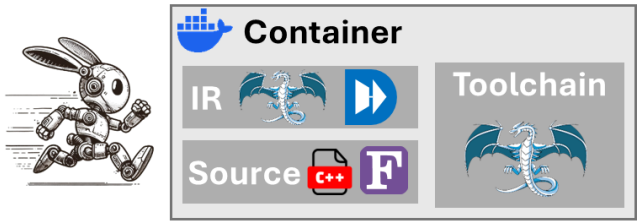
...

APP Config № N

↓

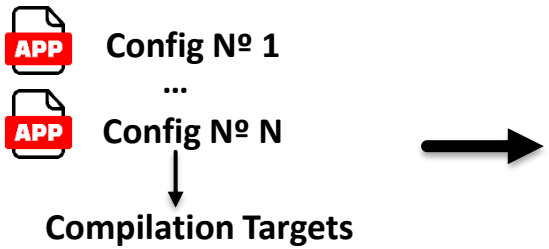
Compilation Targets

XaaS IR Containers: Build & Deploy

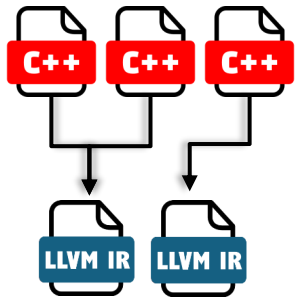


XaaS IR Container

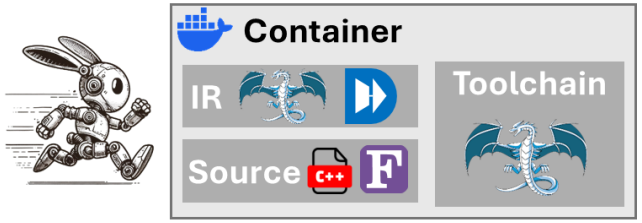
Container Build



- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs

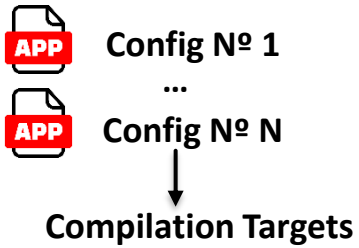


XaaS IR Containers: Build & Deploy

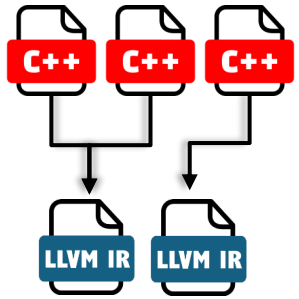


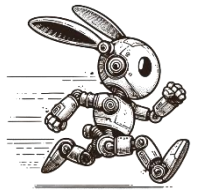
XaaS IR Container

Container Build

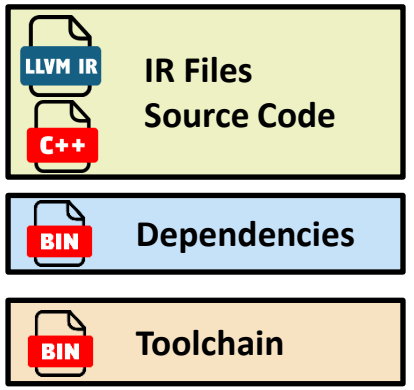


- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs

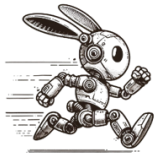


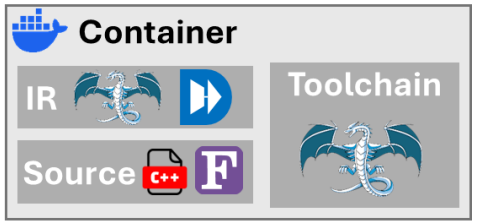


IR Container



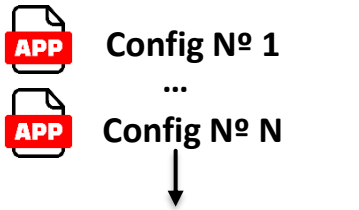
XaaS IR Containers: Build & Deploy



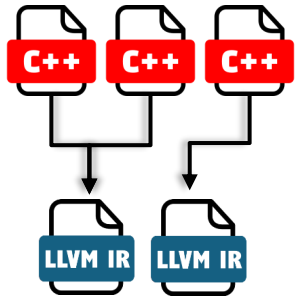


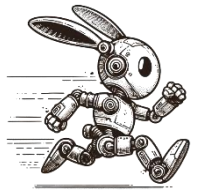
XaaS IR Container

Container Build

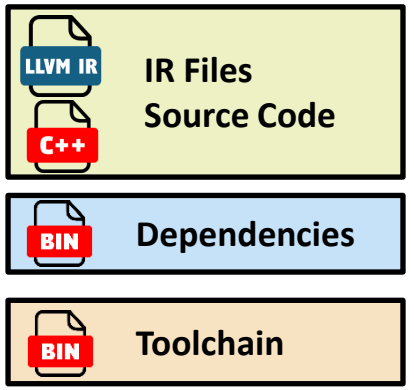


- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs



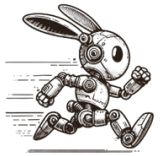


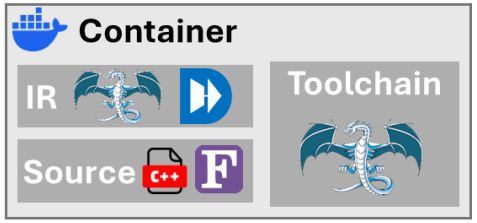
IR Container



Container Deployment

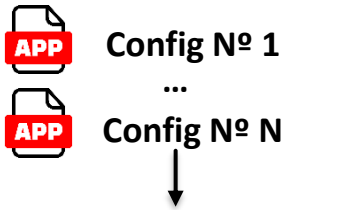
XaaS IR Containers: Build & Deploy



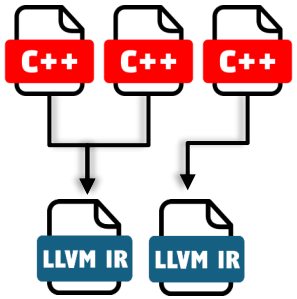


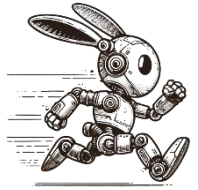
XaaS IR Container

Container Build

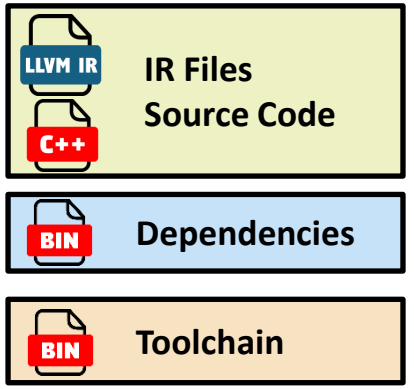


- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs

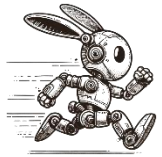




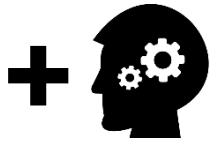
IR Container



Container Deployment

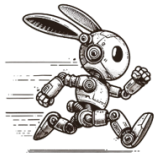


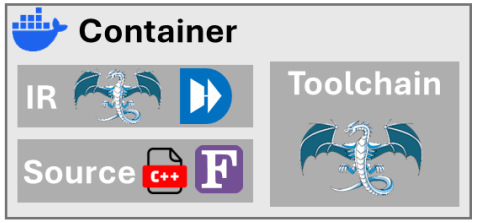
XaaS IR Container



Specialization Selection

XaaS IR Containers: Build & Deploy





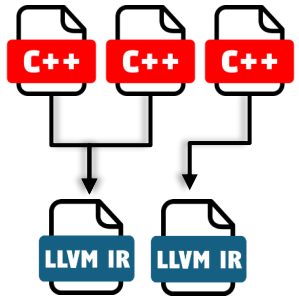
XaaS IR Container

Container Build

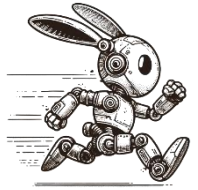
APP Config № 1
...
APP Config № N
↓
Compilation Targets

→

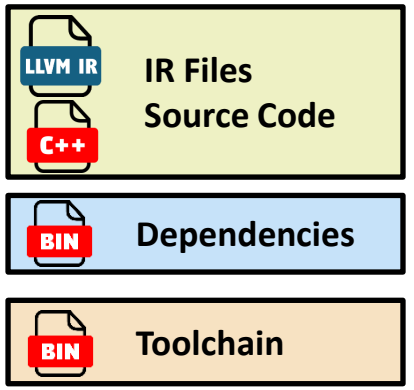
- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs



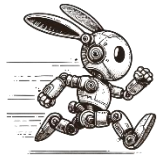
→



IR Container



Container Deployment



XaaS IR Container

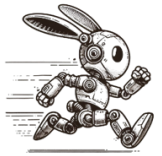
+

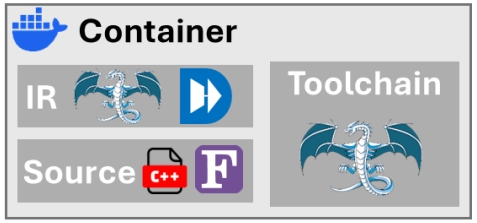
→

Specialization Selection

LLVM Vectorization, Lowering

XaaS IR Containers: Build & Deploy





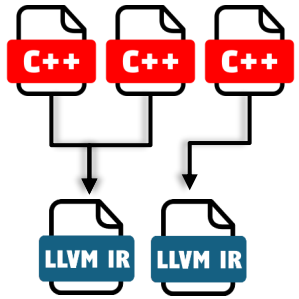
XaaS IR Container

Container Build

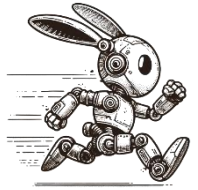
APP Config № 1
...
APP Config № N
↓
Compilation Targets

→

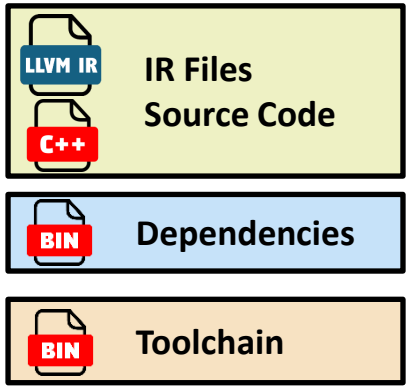
- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs



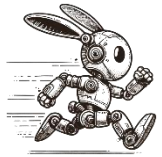
→



IR Container



Container Deployment



XaaS IR Container

+

→

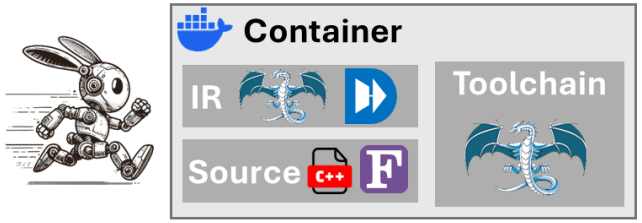
→

Specialization Selection

LLVM Vectorization, Lowering

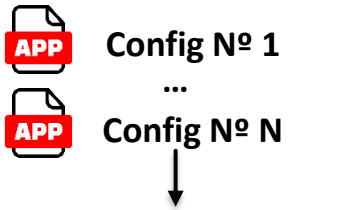
Linking, Installation

XaaS IR Containers: Build & Deploy

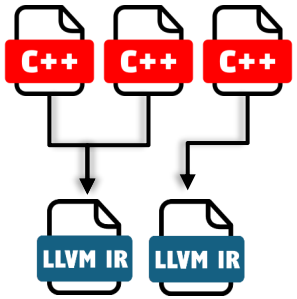


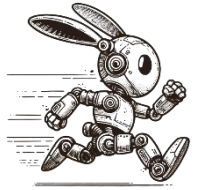
XaaS IR Container

Container Build

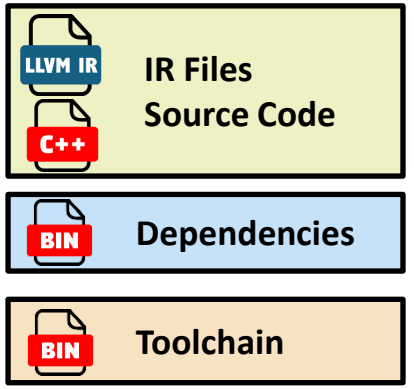


- Preprocessing
- OpenMP Detection
- Vectorization
- Building IRs





IR Container



Container Deployment

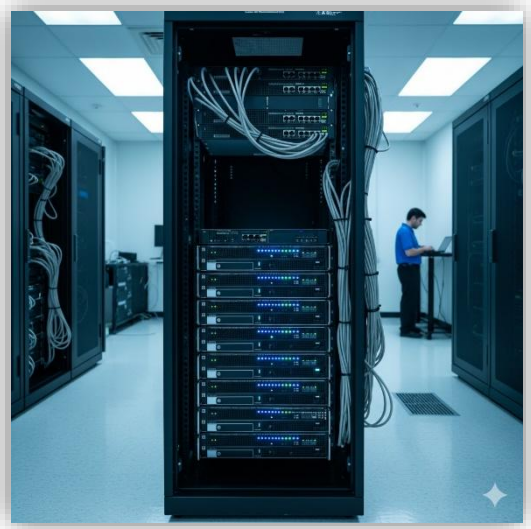
Specialization Selection

LLVM Vectorization, Lowering

Linking, Installation

Deployed IR Container

CSCS Ault



Intel/AMD CPU, NVIDIA GPU

CSCS Alps

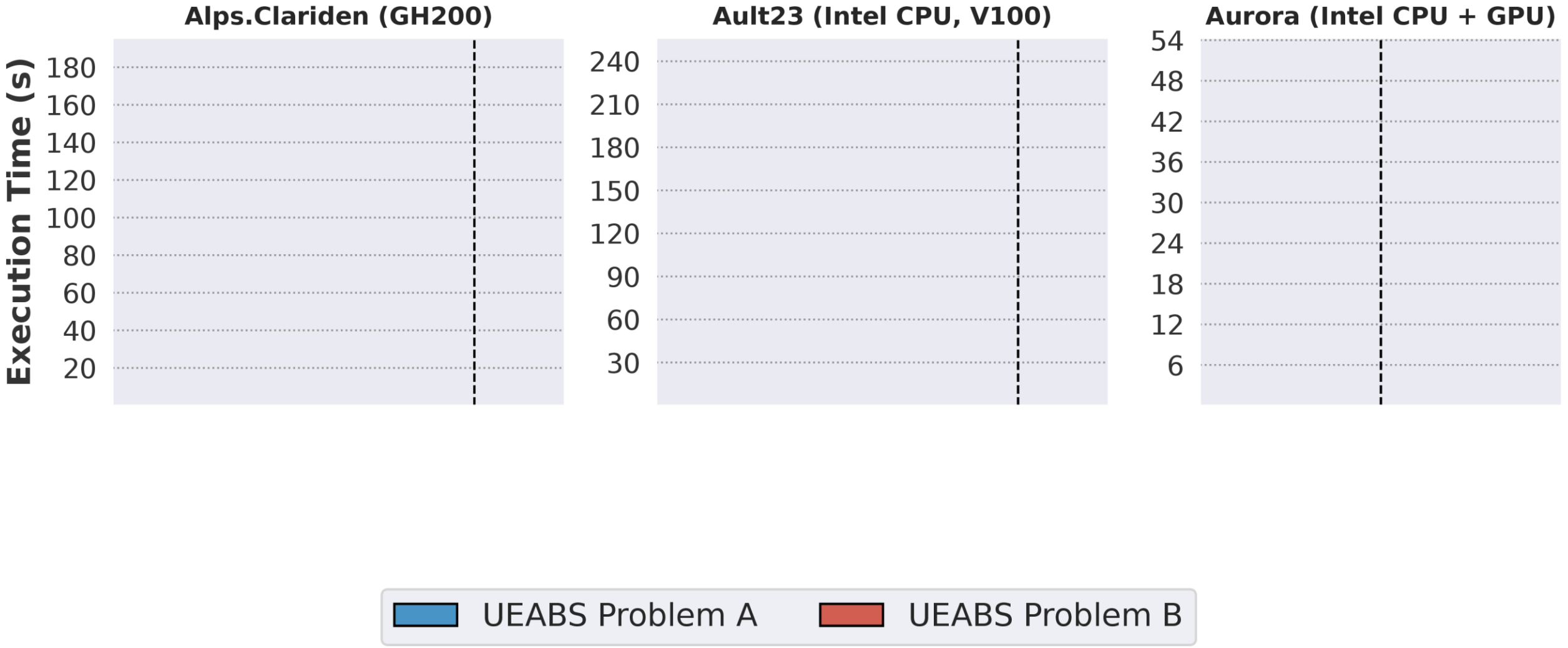
ARM CPU, NVIDIA GPU

ALCF Aurora

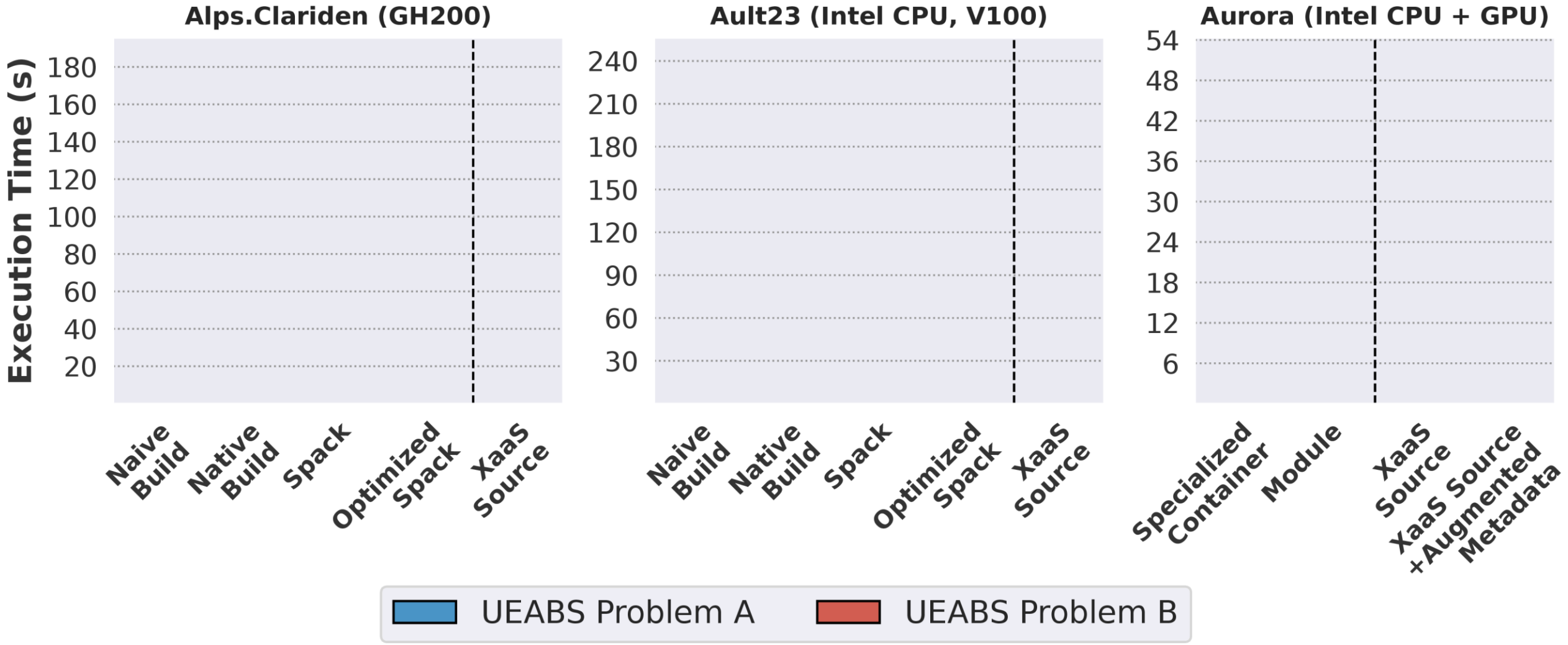
Intel CPU, Intel GPU

Can XaaS Source and IR Containers provide performance portability?

Evaluation – GROMACS in XaaS Source Containers

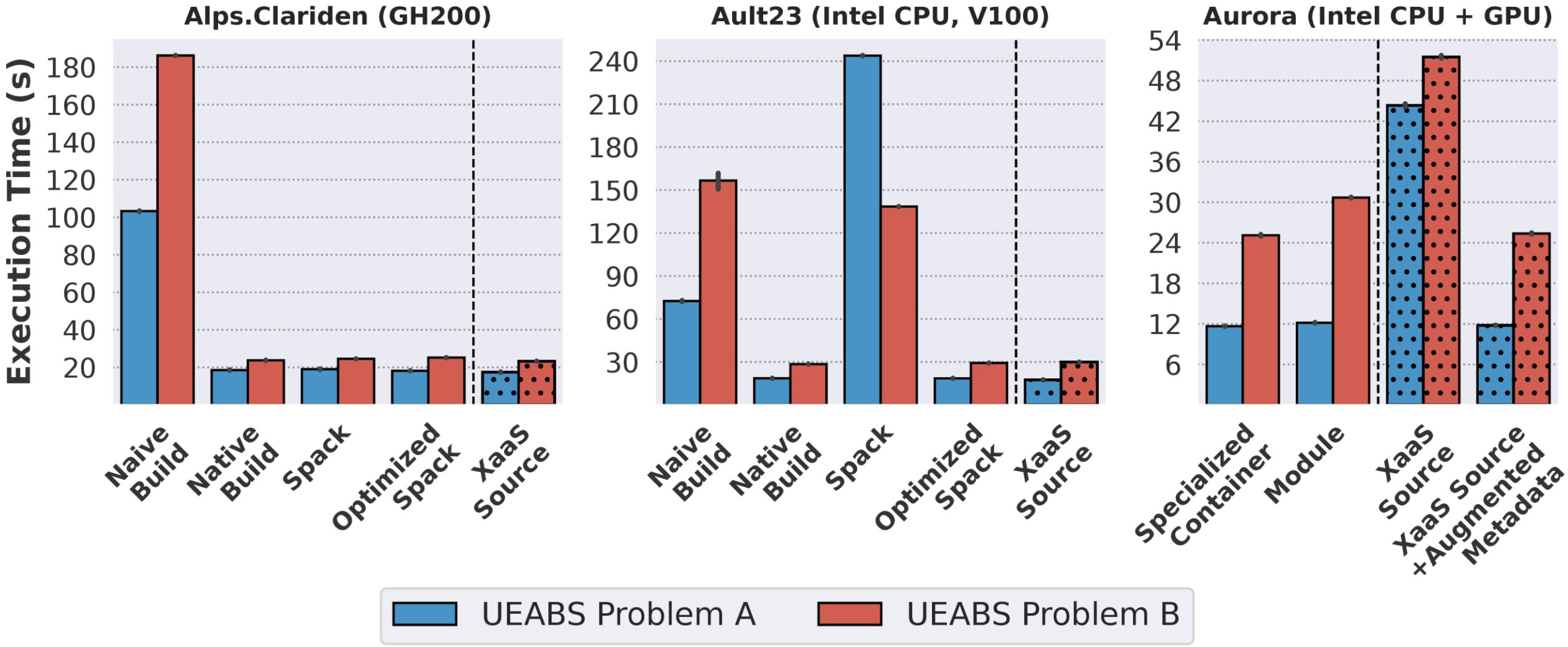


Evaluation – GROMACS in XaaS Source Containers



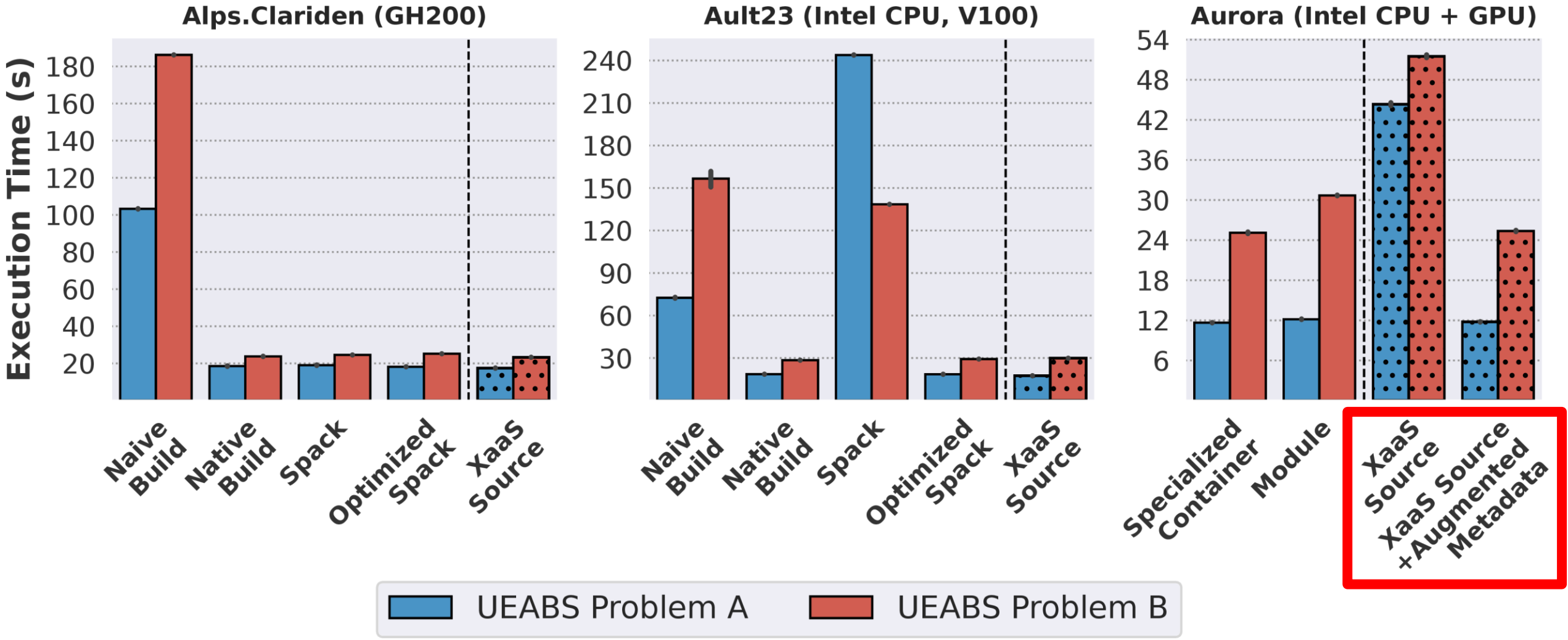
Baselines (depending on availability): “naïve” build, specialized native builds, specialized containers, system modules, Spack.

Evaluation – GROMACS in XaaS Source Containers



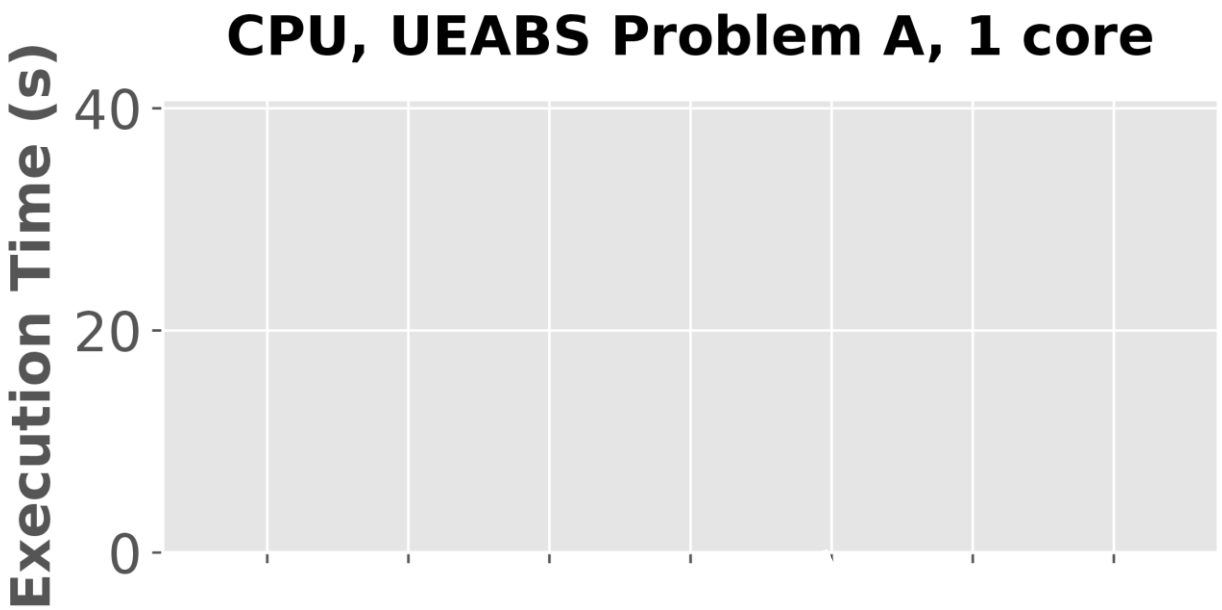
Baselines (depending on availability): “naïve” build, specialized native builds, specialized containers, system modules, Spack.

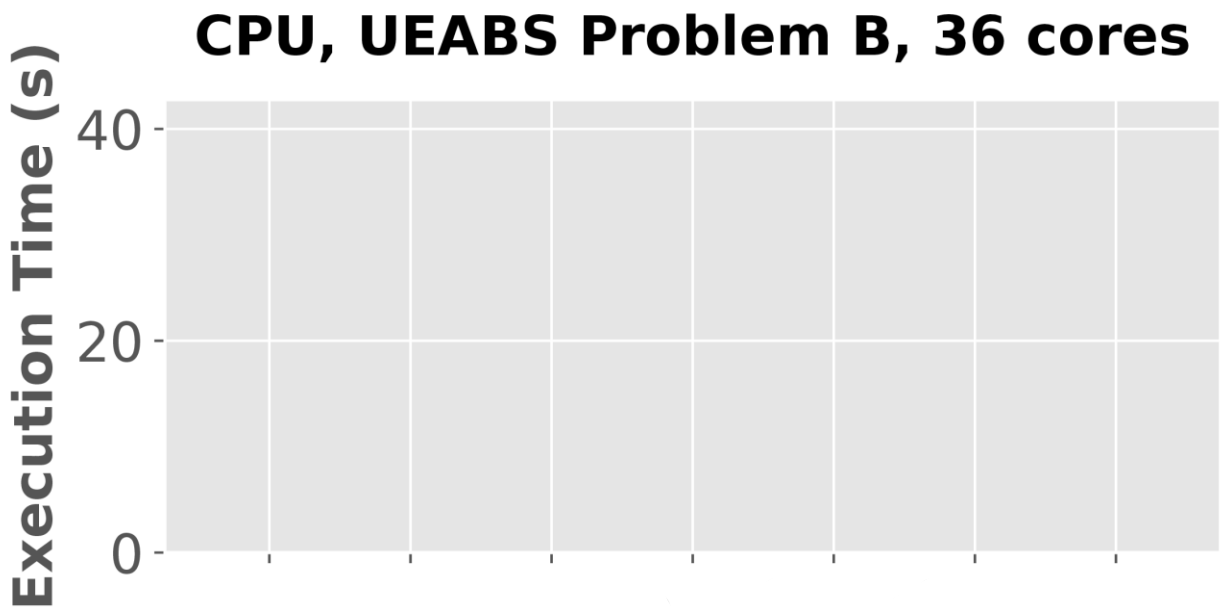
Evaluation – GROMACS in XaaS Source Containers



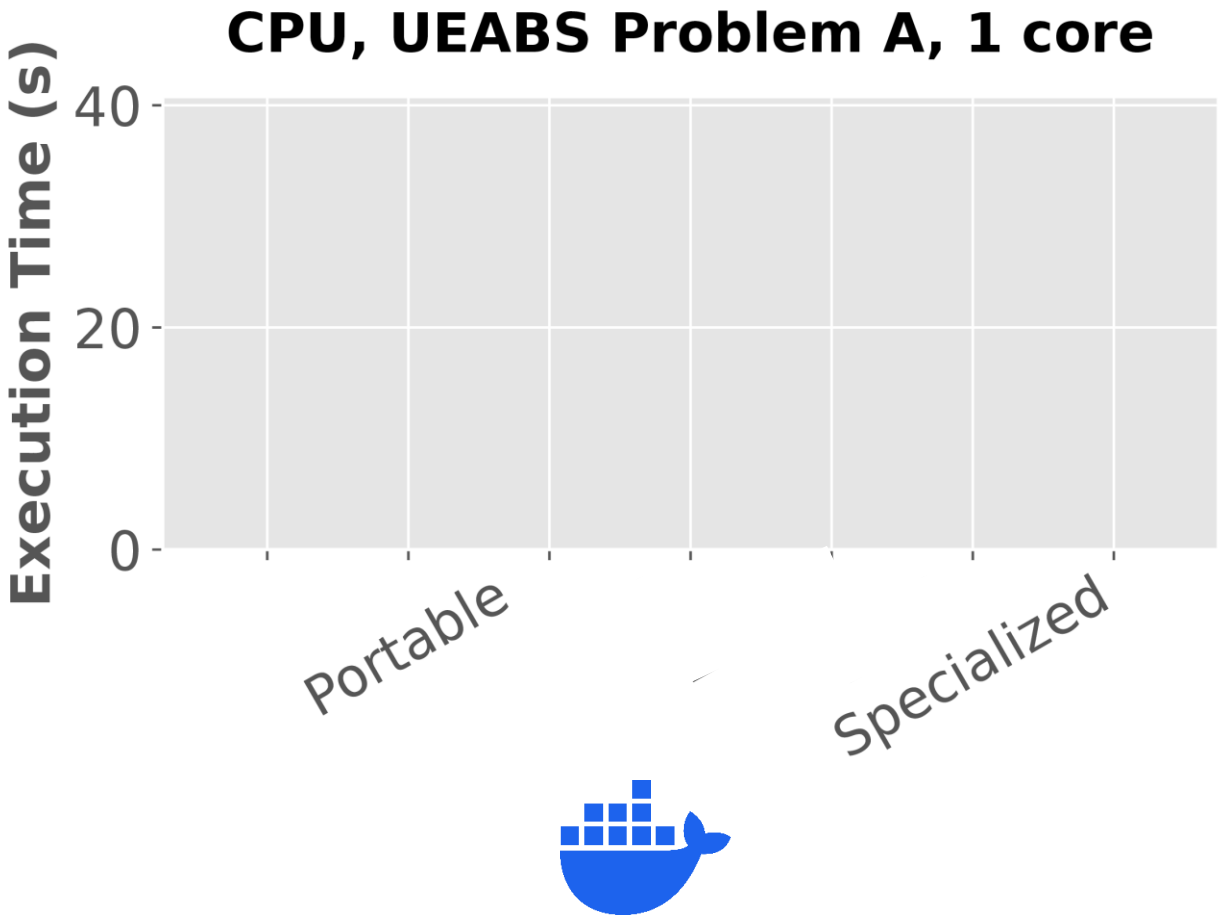
Baselines (depending on availability): “naïve” build, specialized native builds, specialized containers, system modules, Spack.

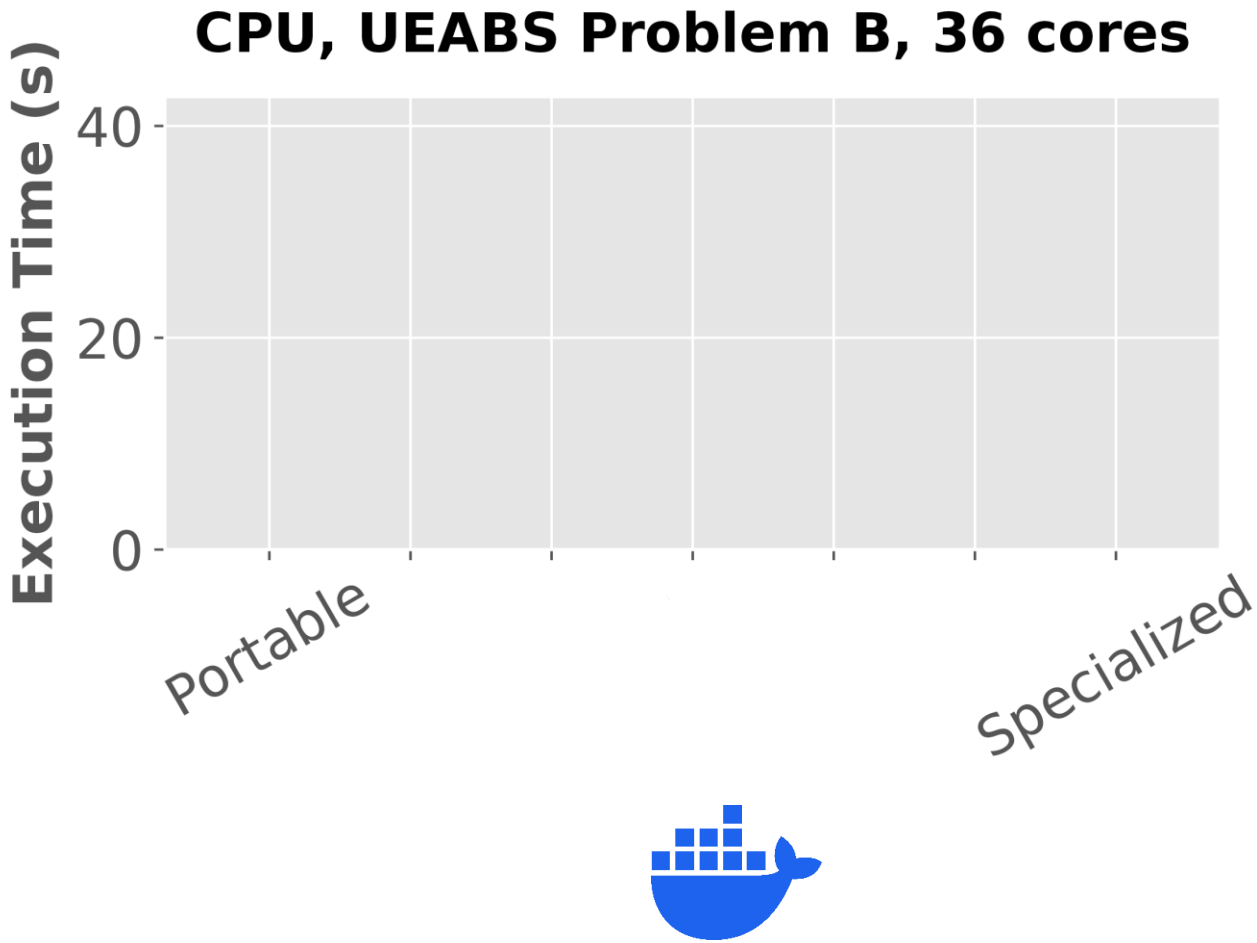
Evaluation – XaaS IR Containers with CPU Vectorization





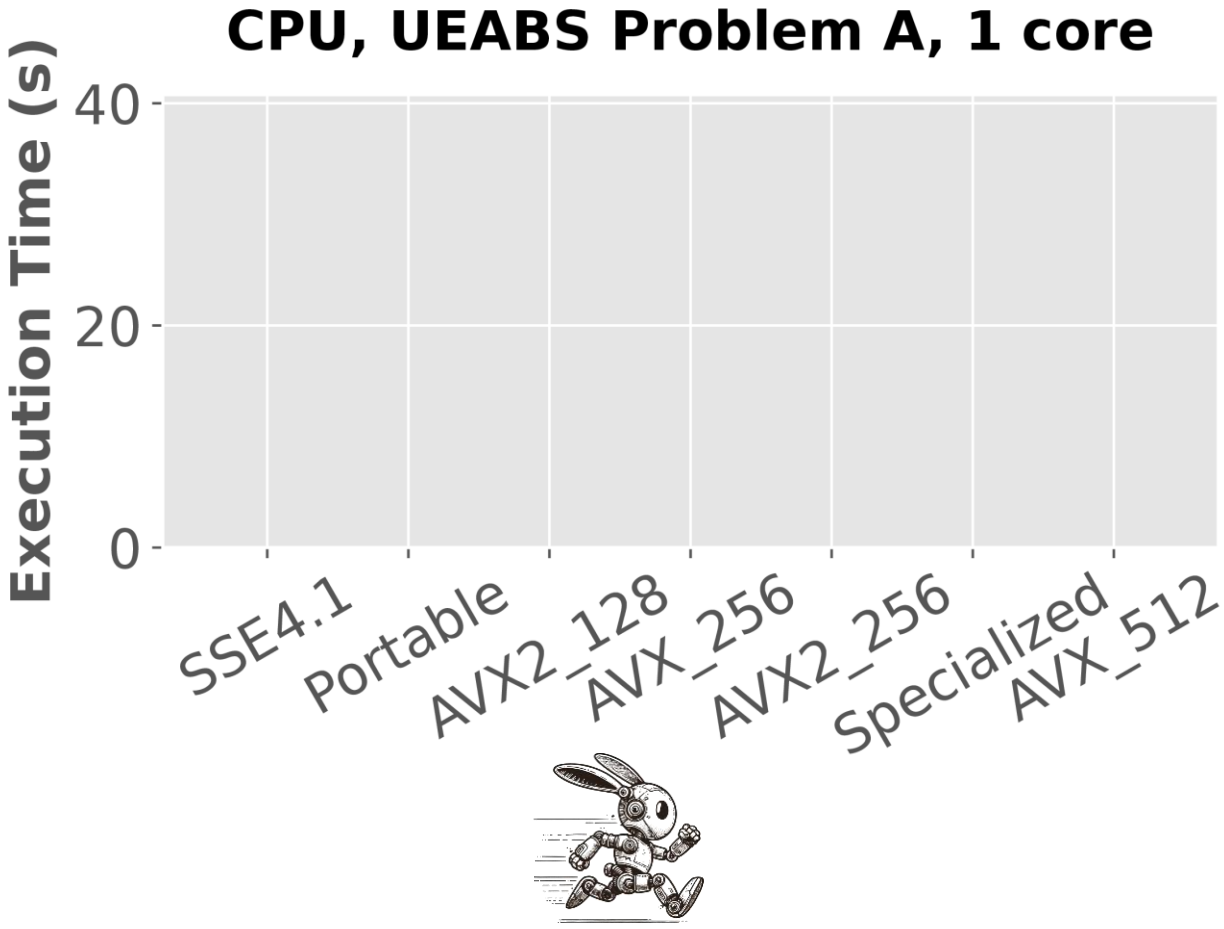
Evaluation – XaaS IR Containers with CPU Vectorization

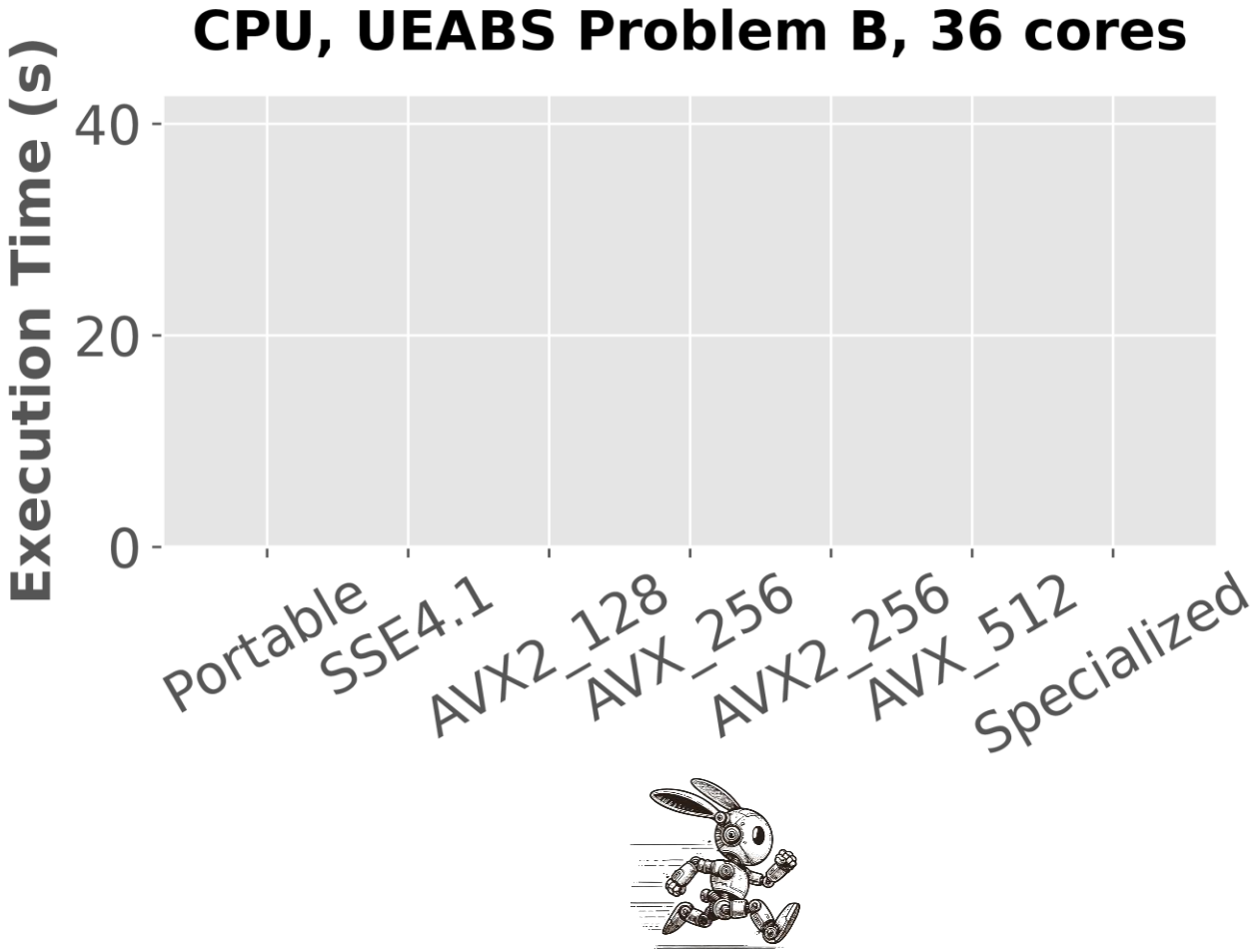




Baselines: portable (SSE4.1) and specialized (AVX-512) Docker containers.

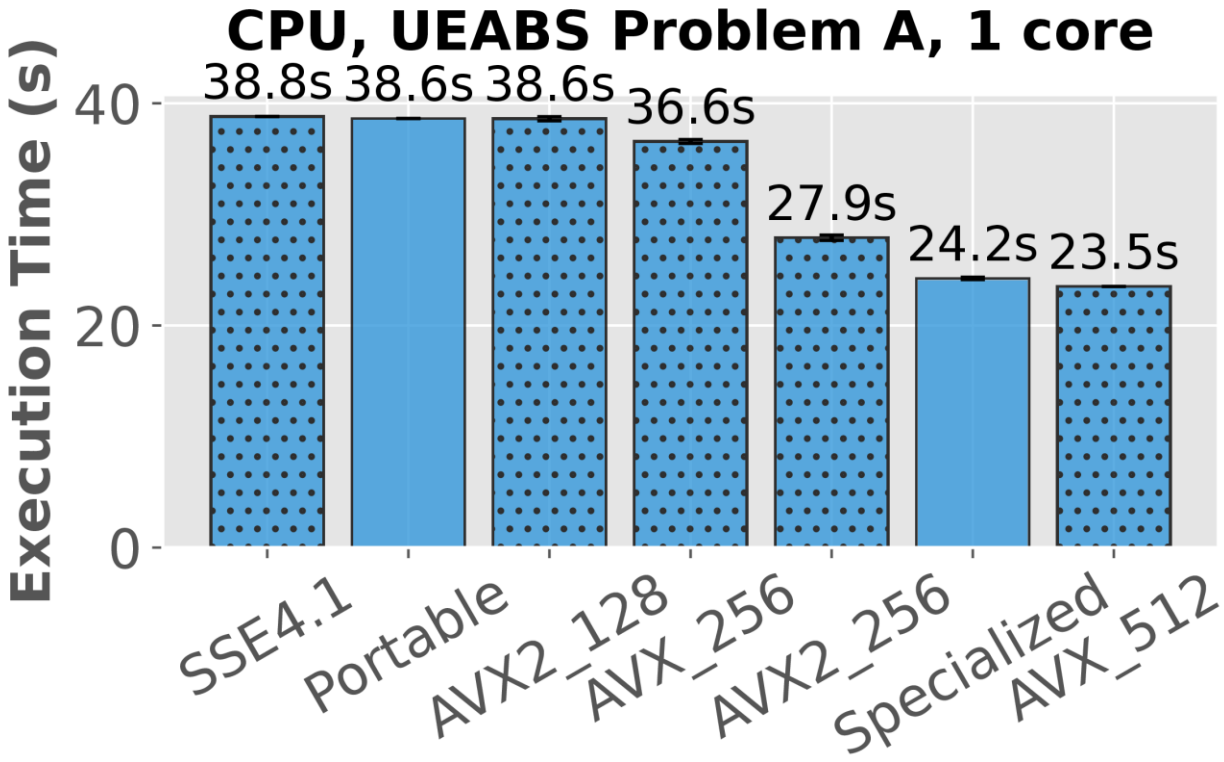
Evaluation – XaaS IR Containers with CPU Vectorization

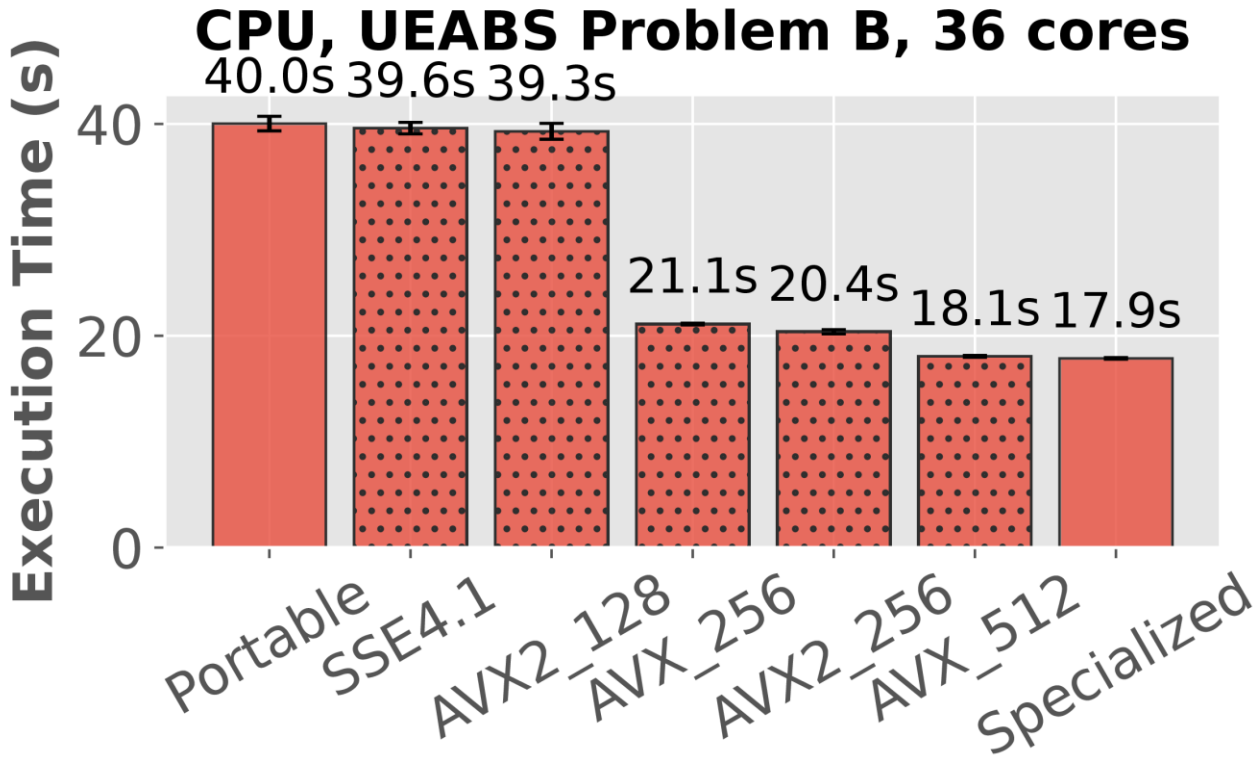




Baselines: portable (SSE4.1) and specialized (AVX-512) Docker containers.

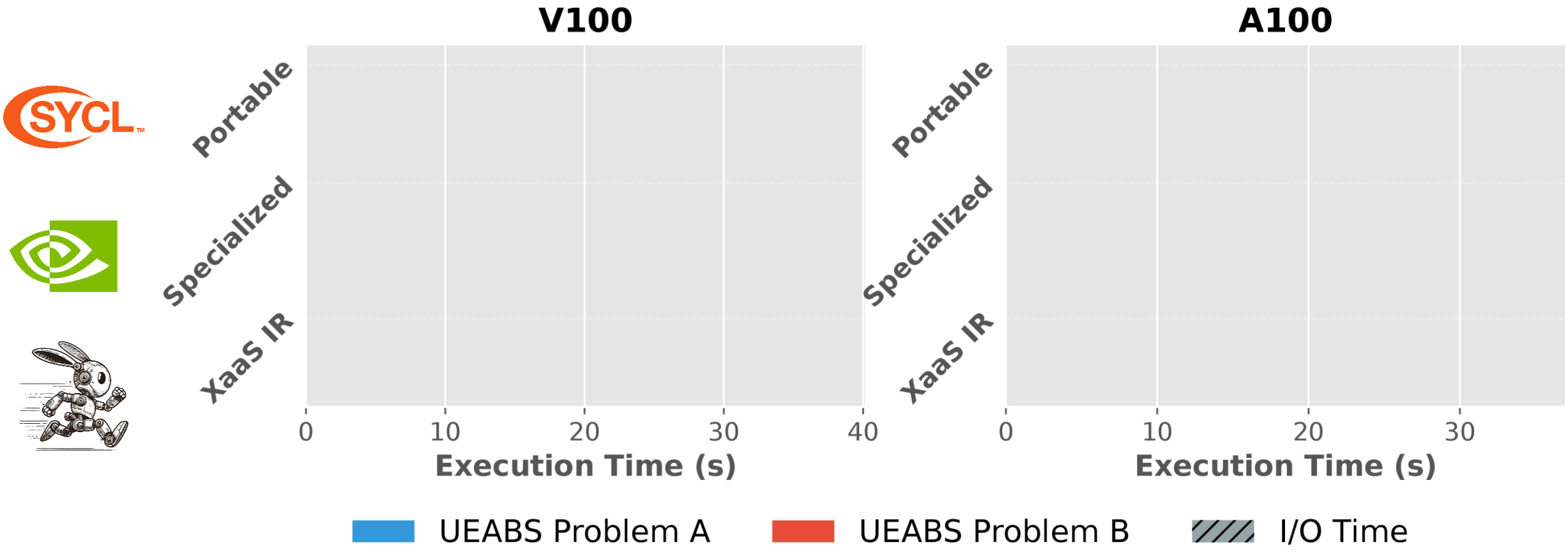
Evaluation – XaaS IR Containers with CPU Vectorization





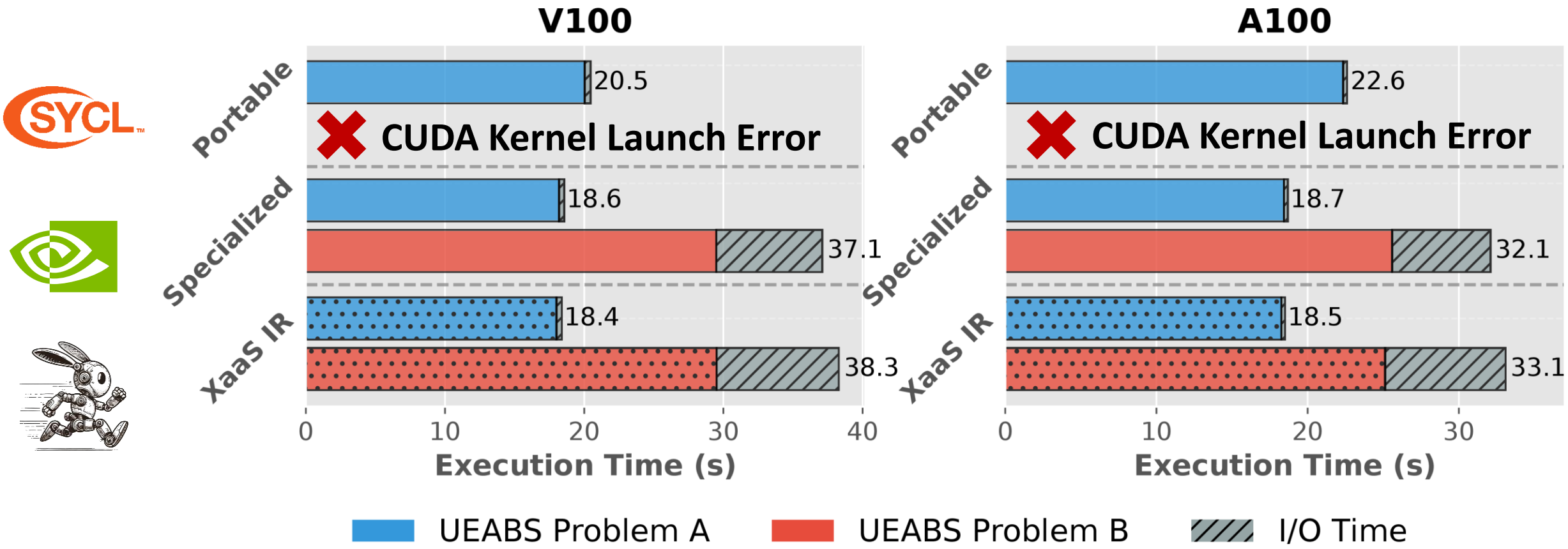
Baselines: portable (SSE4.1) and specialized (AVX-512) Docker containers.

Evaluation – XaaS IR Containers with CUDA Support



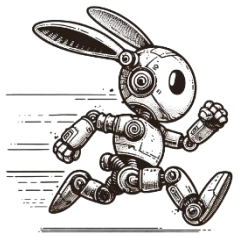
Baselines: specialized Docker containers with CUDA, AVX-512 (V100) or AVX-256 (A100).
Portable containers with SYCL + CUDA, compiled with Intel's icpx compiler.

Evaluation – XaaS IR Containers with CUDA Support

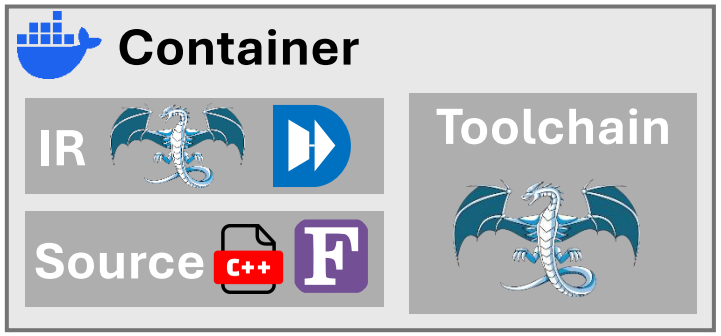


Baselines: specialized Docker containers with CUDA, AVX-512 (V100) or AVX-256 (A100).
Portable containers with SYCL + CUDA, compiled with Intel's icpx compiler.

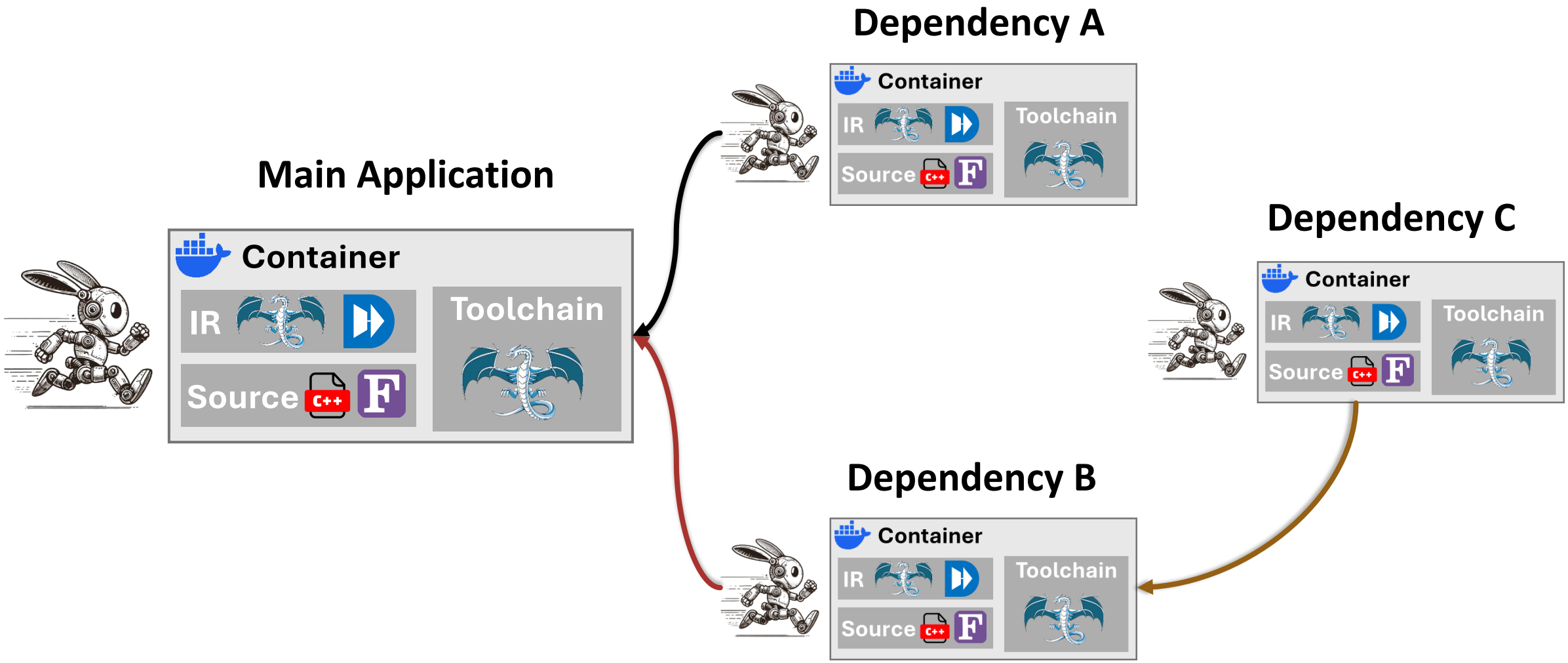
Where Do We Go Next?



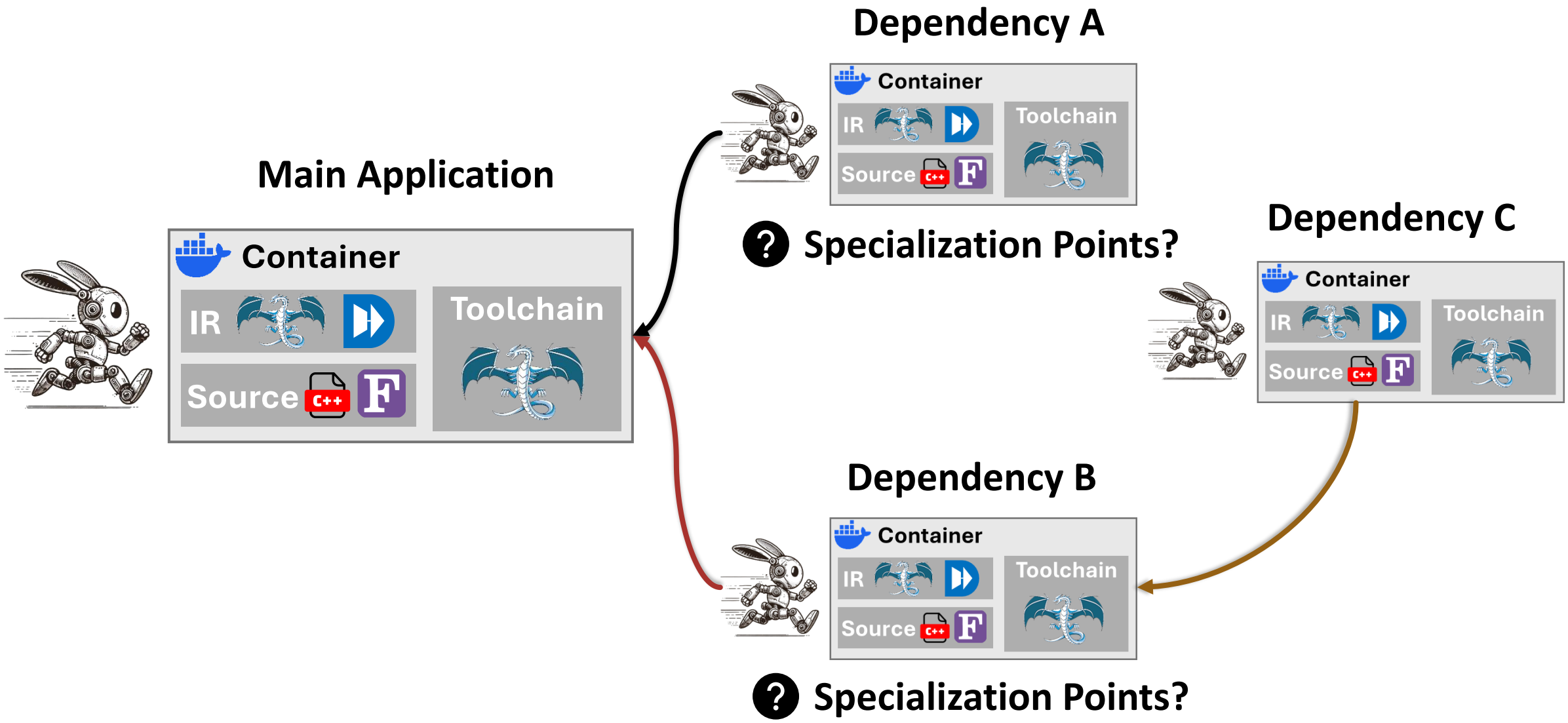
Main Application



Where Do We Go Next?



Where Do We Go Next?



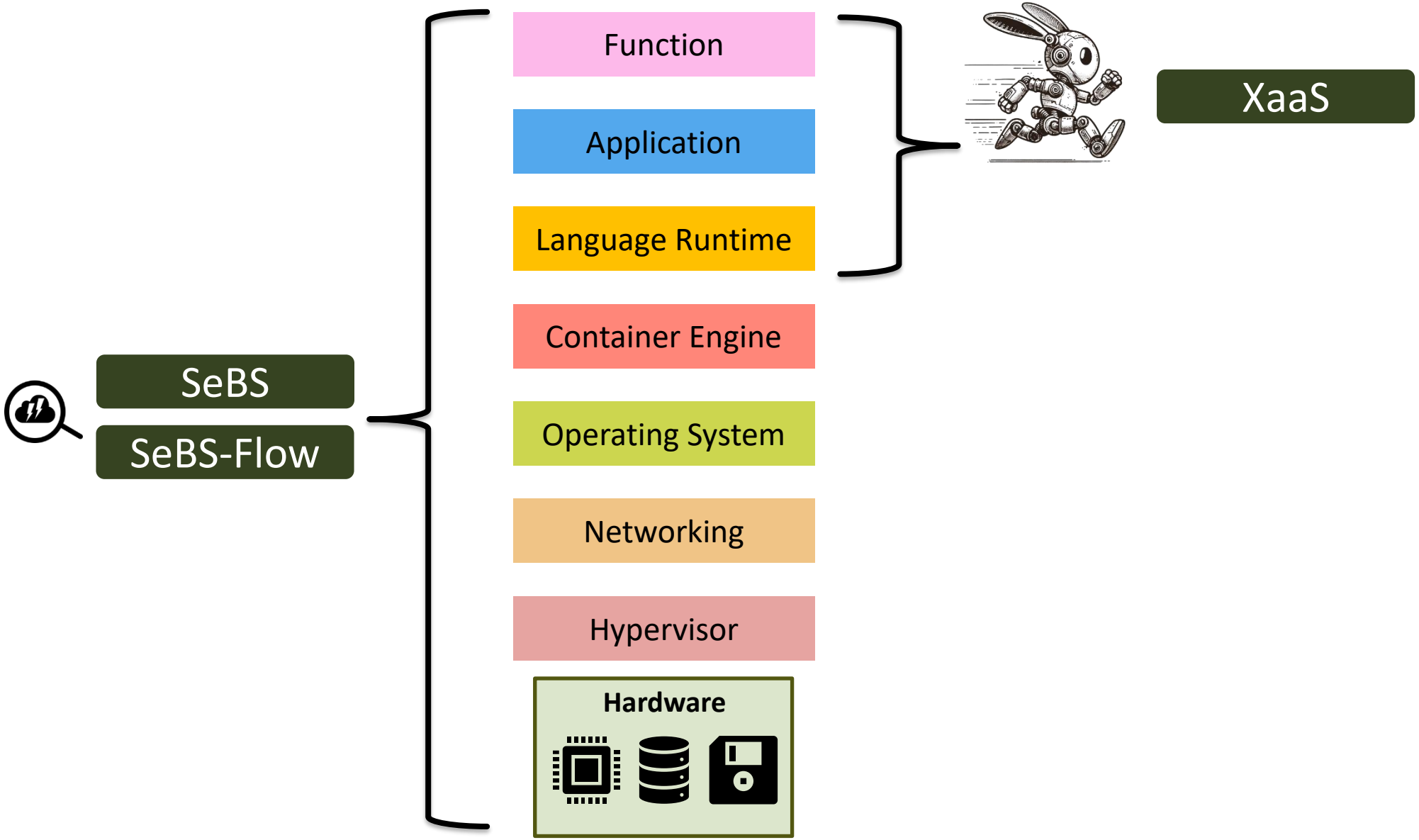
Where Do We Go Next?

(1) Which build configurations matter the most?

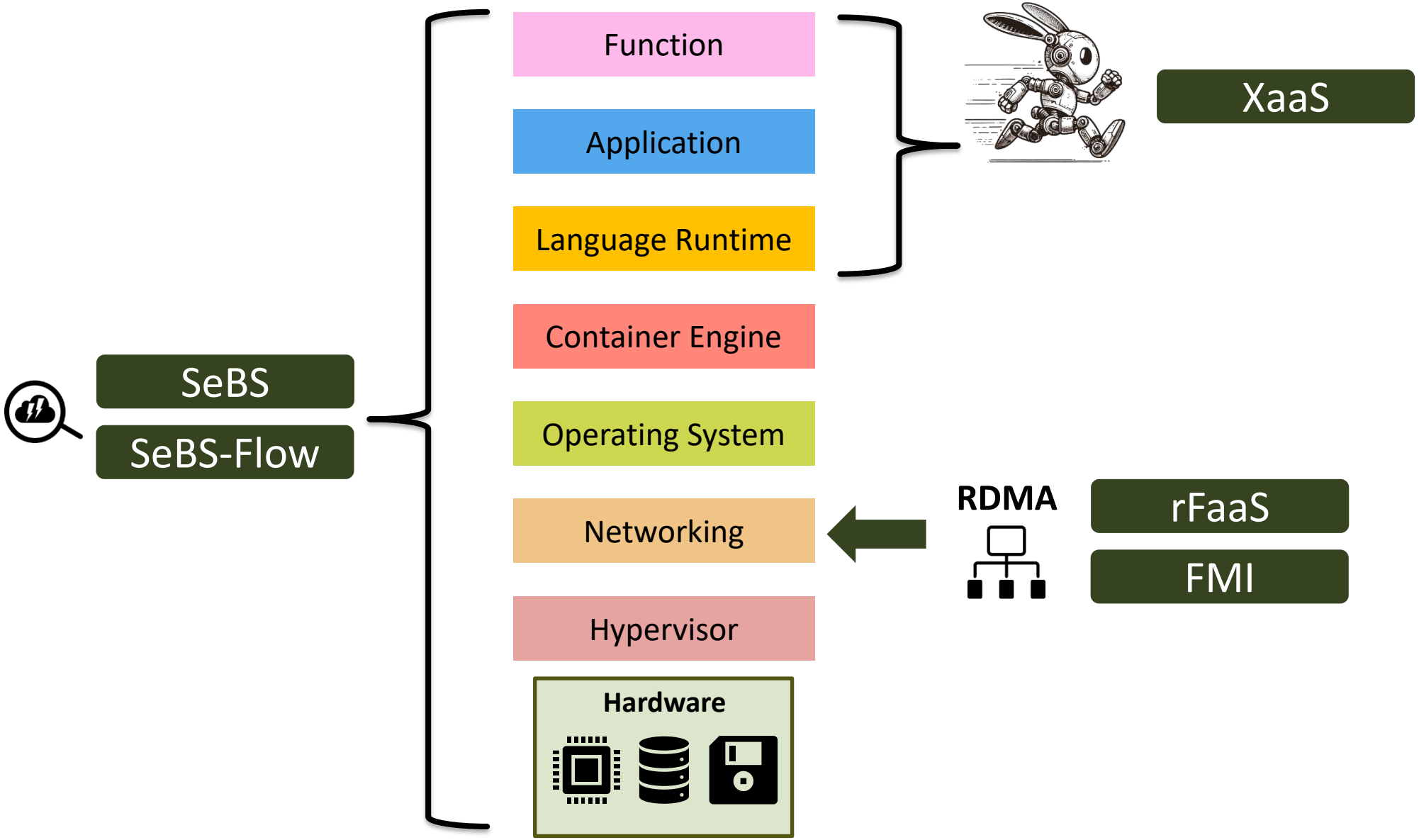
(2) Which software is not worth a rebuild?

GitHub **spcl/XaaS-Containers**

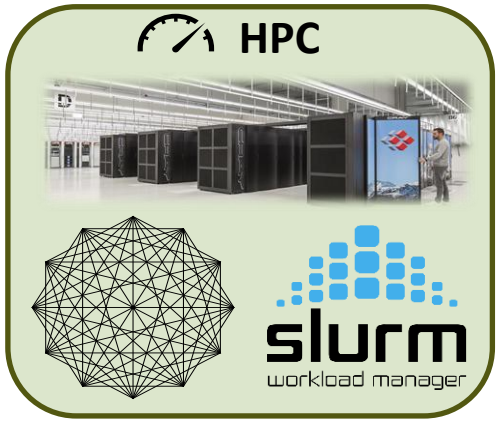
AXelerating Serverless Computing: Communicate!



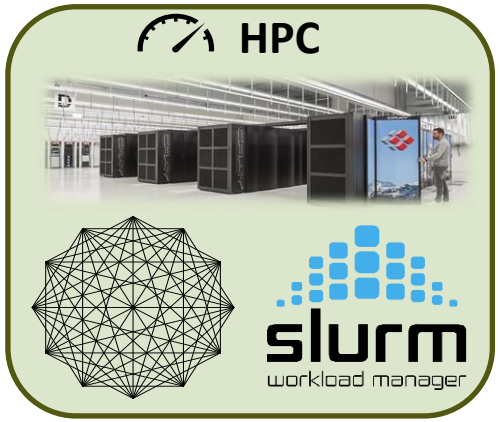
AXelerating Serverless Computing: Communicate!

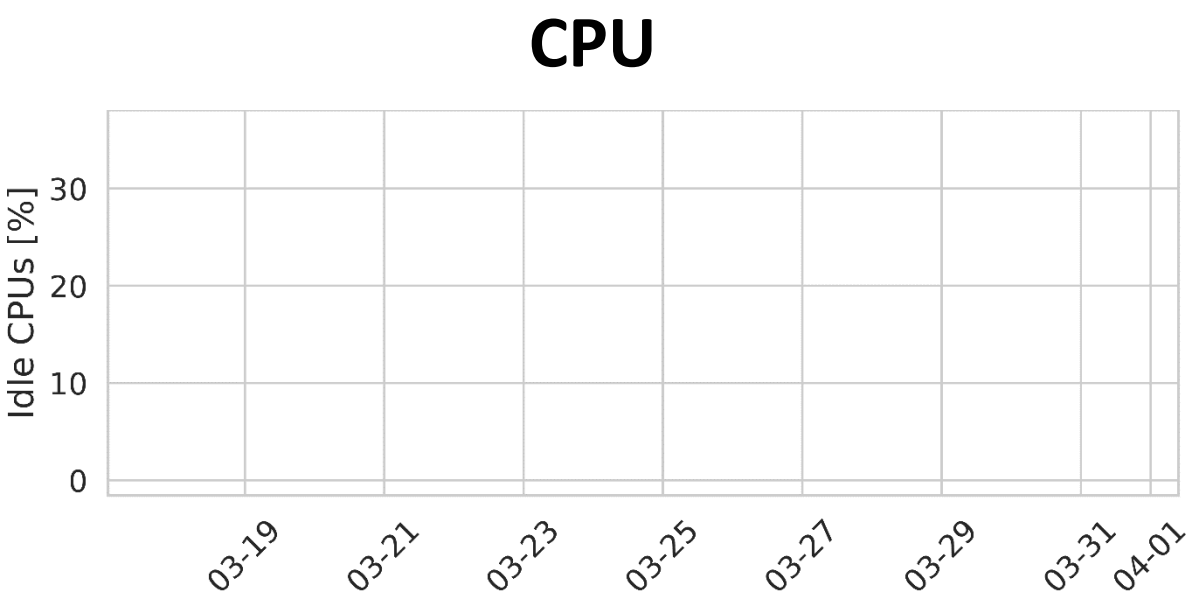


Tracking Wasted Resources in HPC

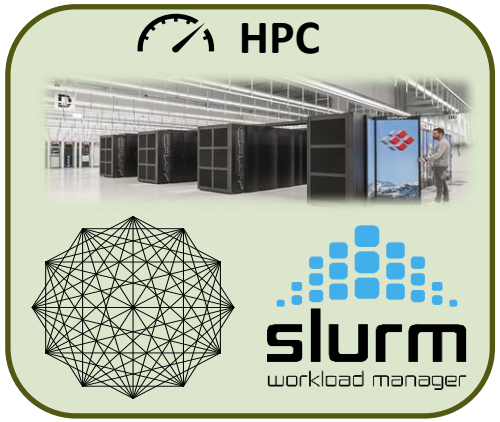


Tracking Wasted Resources in HPC

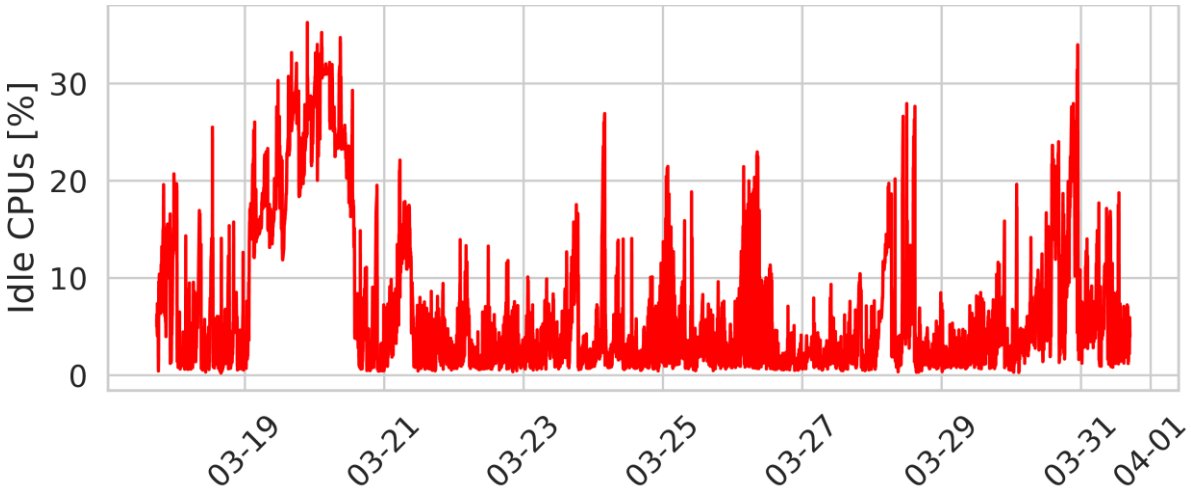




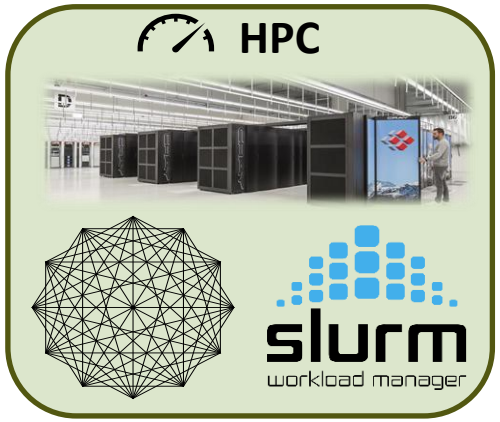
Tracking Wasted Resources in HPC



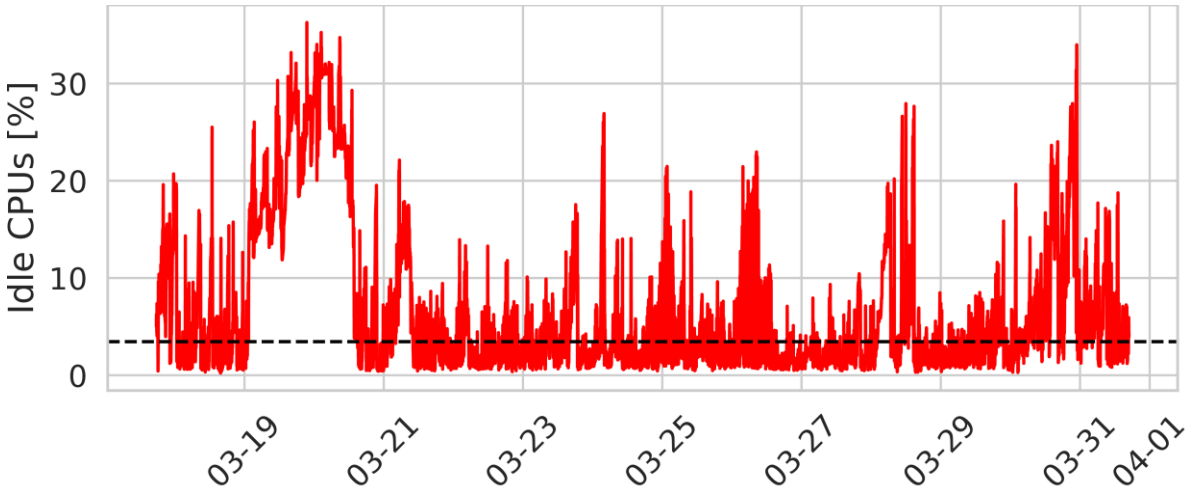
CPU



Tracking Wasted Resources in HPC

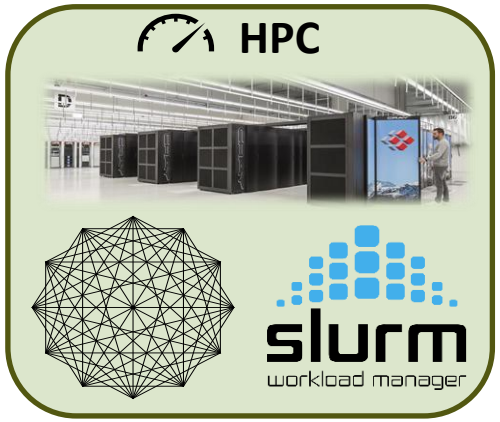


CPU

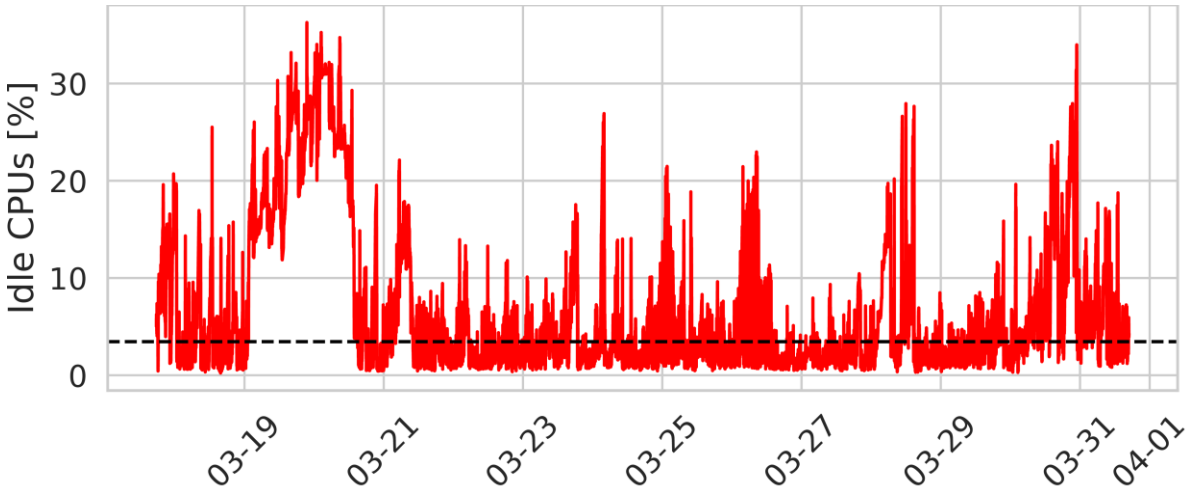


Mean idle CPUs: 6.6%

Tracking Wasted Resources in HPC

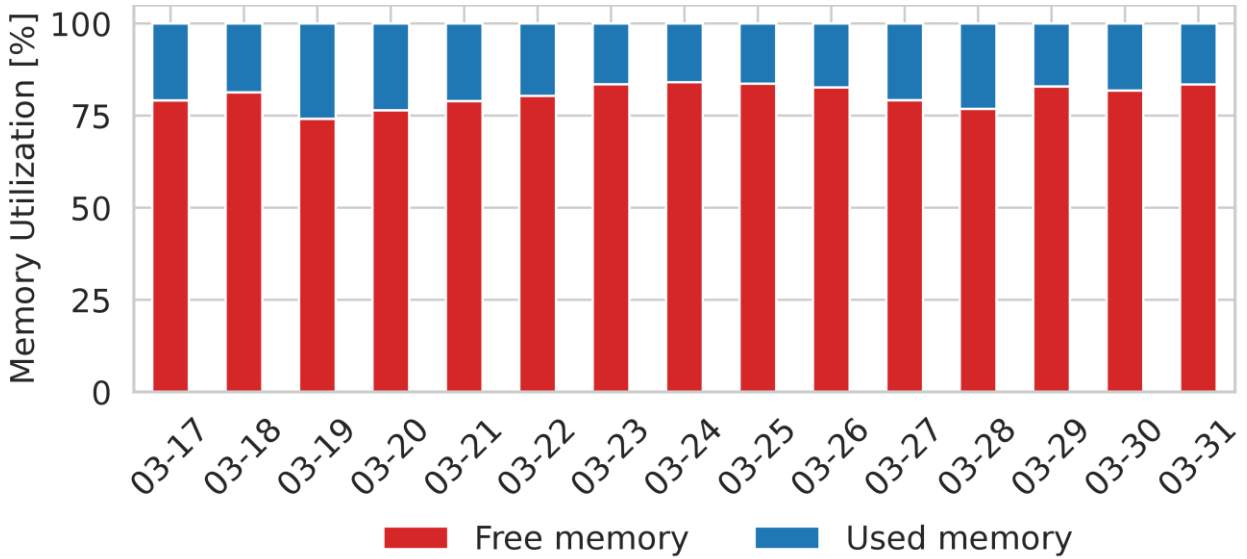


CPU



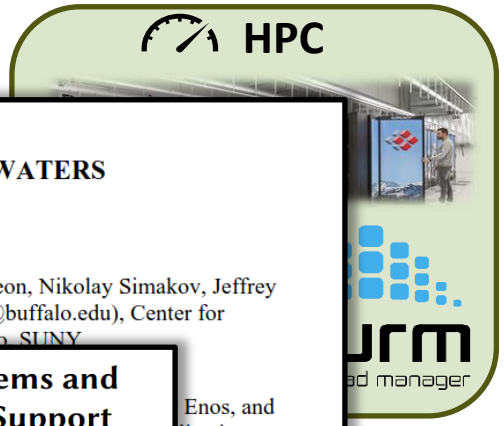
Mean idle CPUs: 6.6%

Memory



Mean free memory: 80.5%

Tracking Wasted Resources in HPC



Learning from Five-year Resource-Utilization Data of Titan System

Feiyi Wang*, Sarp Oral†, Satyabrata Sen ‡ and Neena Imam§

Oak Ridge National Laboratory

CLUSTER, 2019

FINAL REPORT
WORKLOAD ANALYSIS OF BLUE WATERS
(ACI 1650758)

Matthew D. Jones, Joseph P. White, Martins Innus, Robert L. DeLeon, Nikolay Simakov, Jeffrey T. Palmer, Steven M. Gallo, and Thomas R. Furlani (furlani@buffalo.edu), Center for Computational Research, University at Buffalo, SUNY

Enos, and
lications

Xiv, 2017

Quantifying Memory Underutilization in HPC Systems and Using it to Improve Performance via Architecture Support

Gagandeep Panwar*
Virginia Tech
Blacksburg, USA
gpanwar@vt.edu

Da Zhang*
Virginia Tech
Blacksburg, USA
daz3@vt.edu

Yihan Pang*
Virginia Tech
Blacksburg, USA
pyihan1@vt.edu

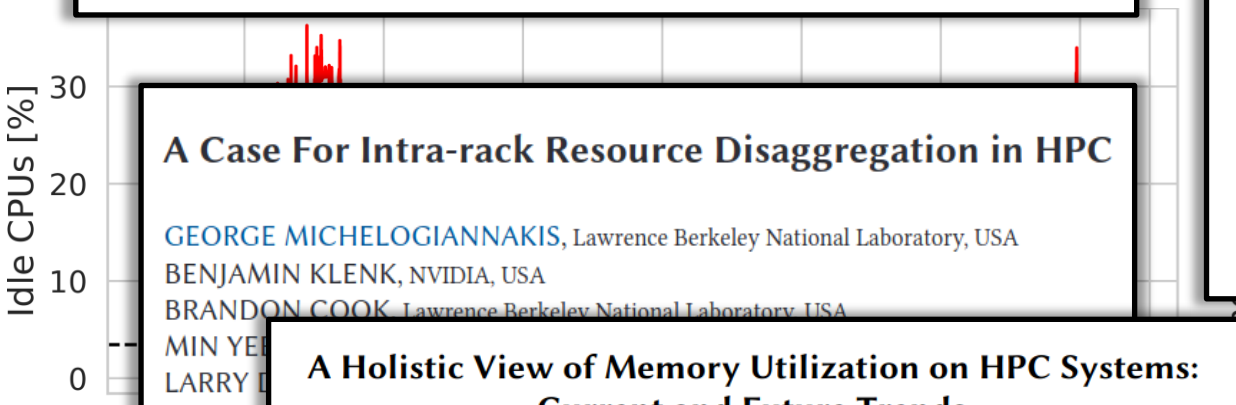
Mai Dahshan
Virginia Tech
Blacksburg, USA
mdahshan@vt.edu

Nathan DeBardeleben
Los Alamos National Laboratory
Los Alamos, USA
ndebard@lanl.gov

Binoy Ravindran
Virginia Tech
Blacksburg, USA
binoy@vt.edu

Xun Jian
Virginia Tech
Blacksburg, USA
xunj@vt.edu

MICRO, 2019



A Case For Intra-rack Resource Disaggregation in HPC

GEORGE MICHELOGIANNAKIS, Lawrence Berkeley National Laboratory, USA
BENJAMIN KLENK, NVIDIA, USA
BRANDON COOK, Lawrence Berkeley National Laboratory, USA
MIN YE
LARRY D
KEREN B
JOHN SH

A Holistic View of Memory Utilization on HPC Systems: Current and Future Trends

Ivy B. Peng*
peng8@llnl.gov
Lawrence Livermore National Laboratory
USA

Ian Karlin
karlin1@llnl.gov
Lawrence Livermore National Laboratory
USA

Maya B. Gokhale
gokhale2@llnl.gov
Lawrence Livermore National Laboratory
USA

Kathleen Shoga
Shoga1@llnl.gov
Lawrence Livermore National Laboratory
USA

Matthew Legendre
legendre1@llnl.gov
Lawrence Livermore National Laboratory
USA

Todd Gamblin
gamblin2@llnl.gov
Lawrence Livermore National Laboratory
USA

MEMSYS, 2021

Comprehensive Workload Analysis and Modeling of a Petascale Supercomputer

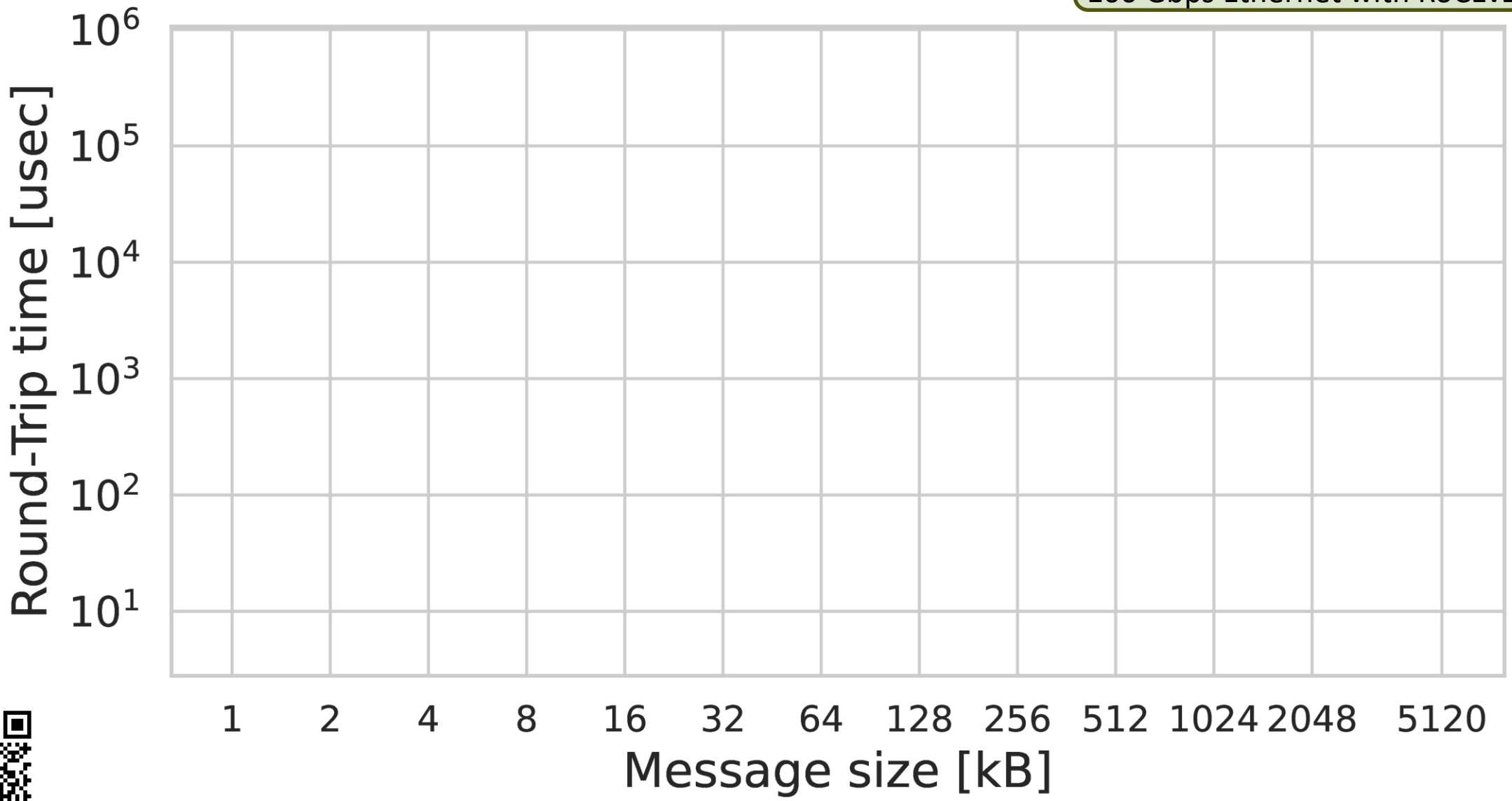
Haihang You[1] and Hao Zhang[2]

[1] National Institute for Computational Sciences,
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2] Department of Electrical Engineering and Computer Science,
University of Tennessee, Knoxville, TN 37996, USA
{hyou,haozhang}@utk.edu

JSSPP, 2012

rFaaS: How Fast are Invocations in FaaS?

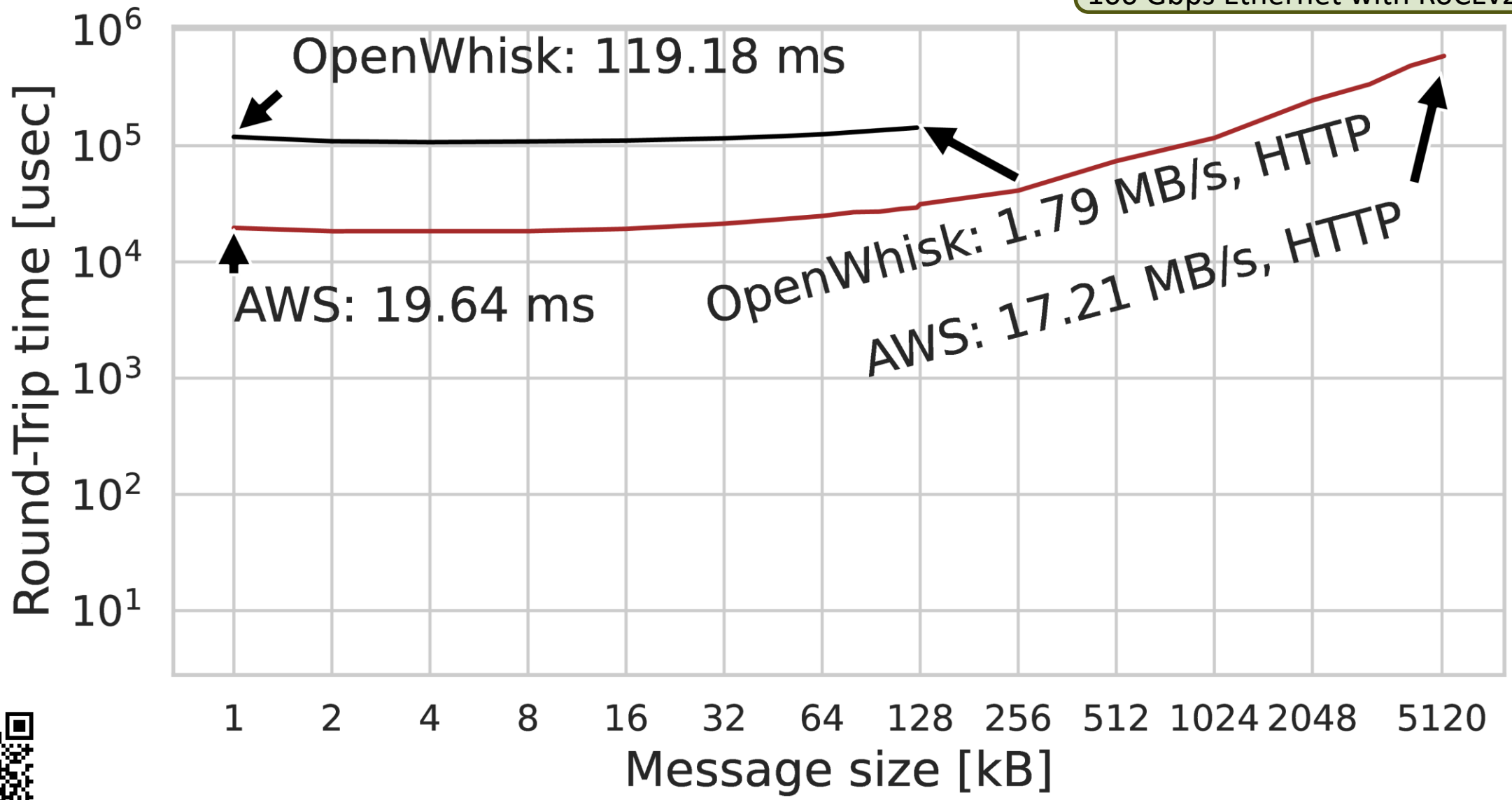
36 CPU cores, 377 GB memory.
100 Gbps Ethernet with RoCEv2 support.



“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: How Fast are Invocations in FaaS?

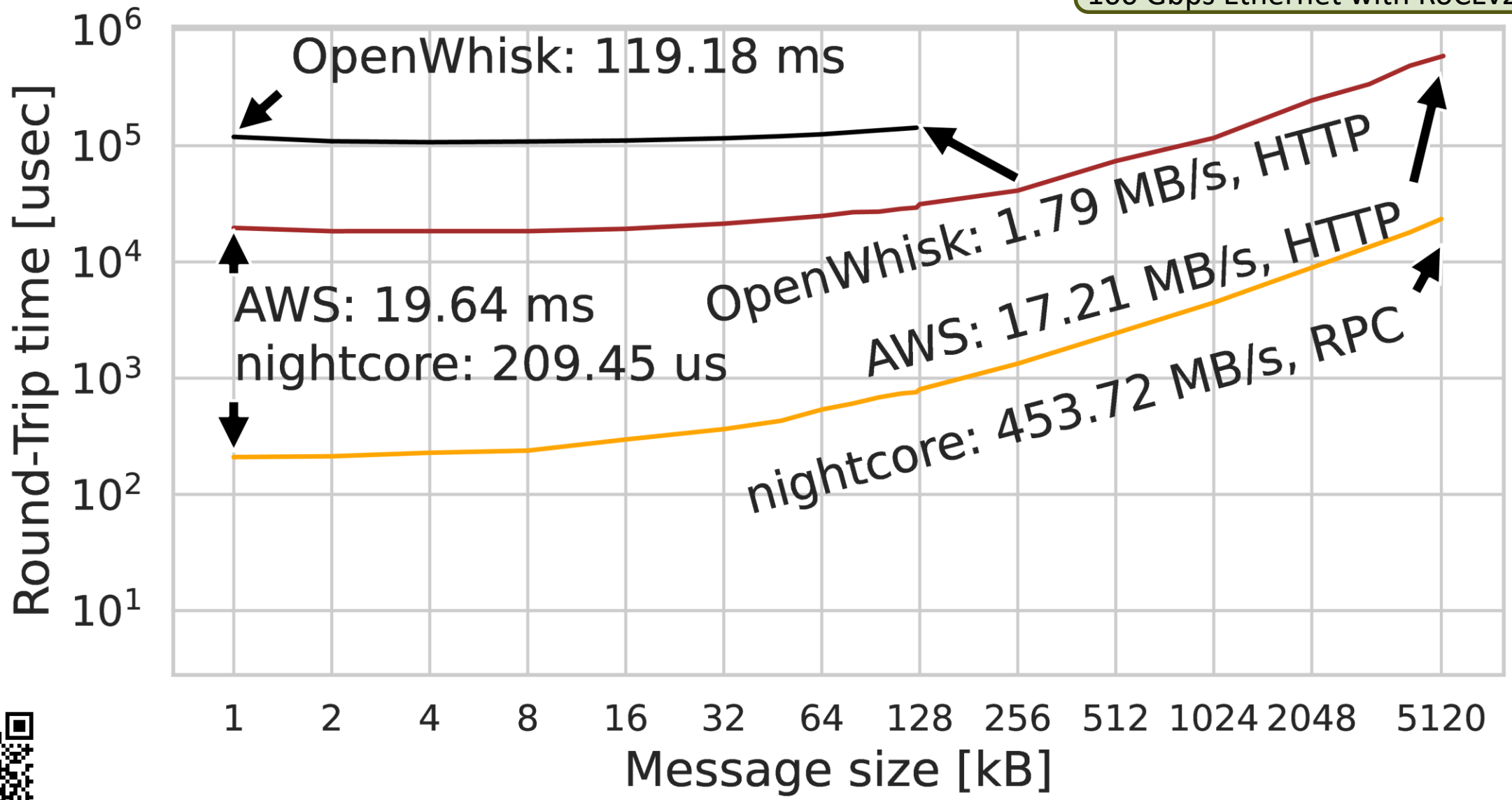
36 CPU cores, 377 GB memory.
100 Gbps Ethernet with RoCEv2 support.



“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

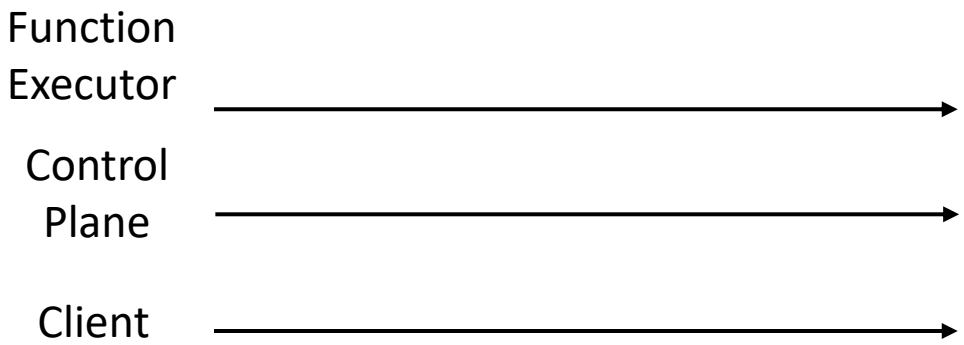
rFaaS: How Fast are Invocations in FaaS?

36 CPU cores, 377 GB memory.
100 Gbps Ethernet with RoCEv2 support.



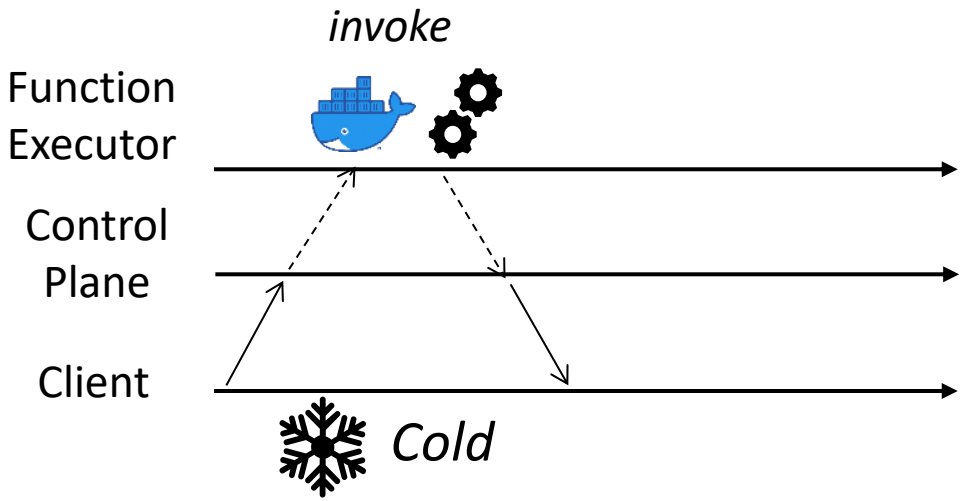
"rFaaS: Enabling High Performance Serverless with RDMA and Leases", IPDPS'23

rFaaS: Invocations with Leases



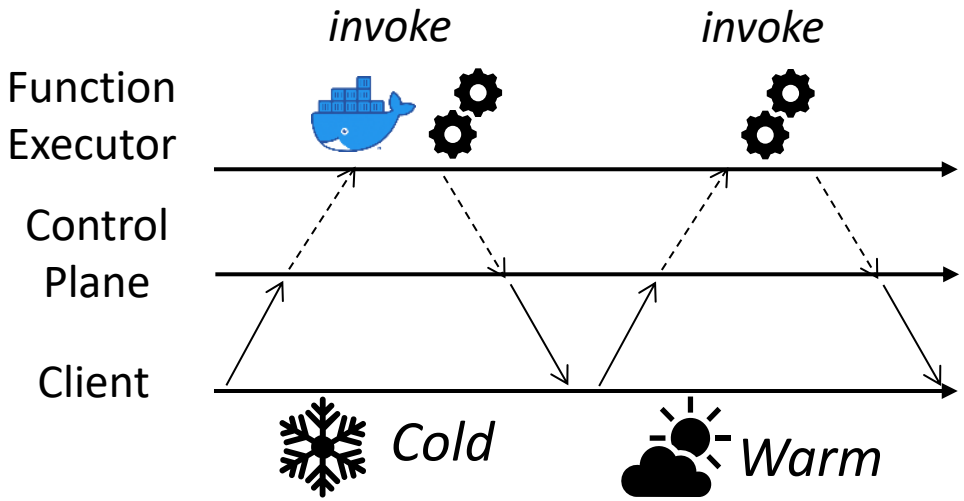
“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: Invocations with Leases



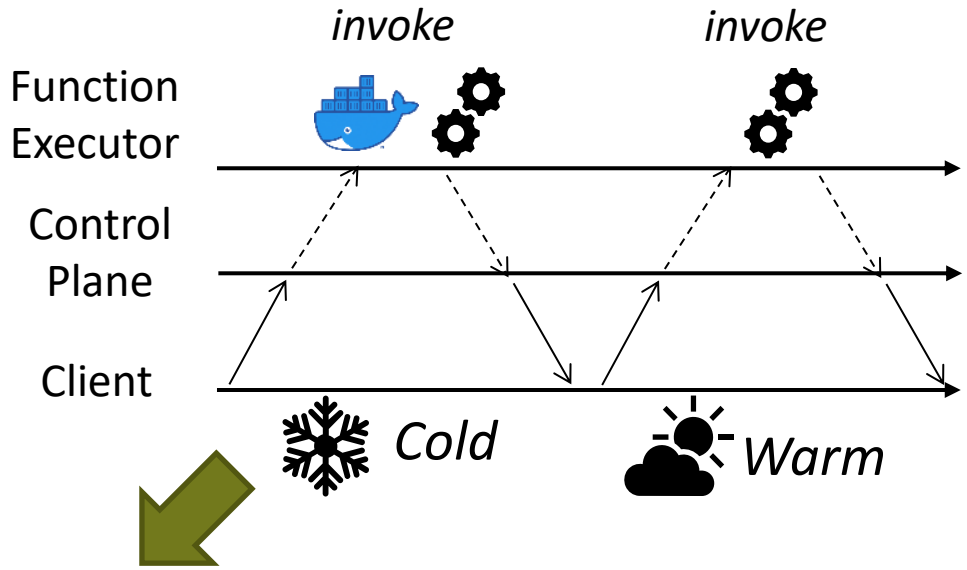
“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: Invocations with Leases



“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: Invocations with Leases



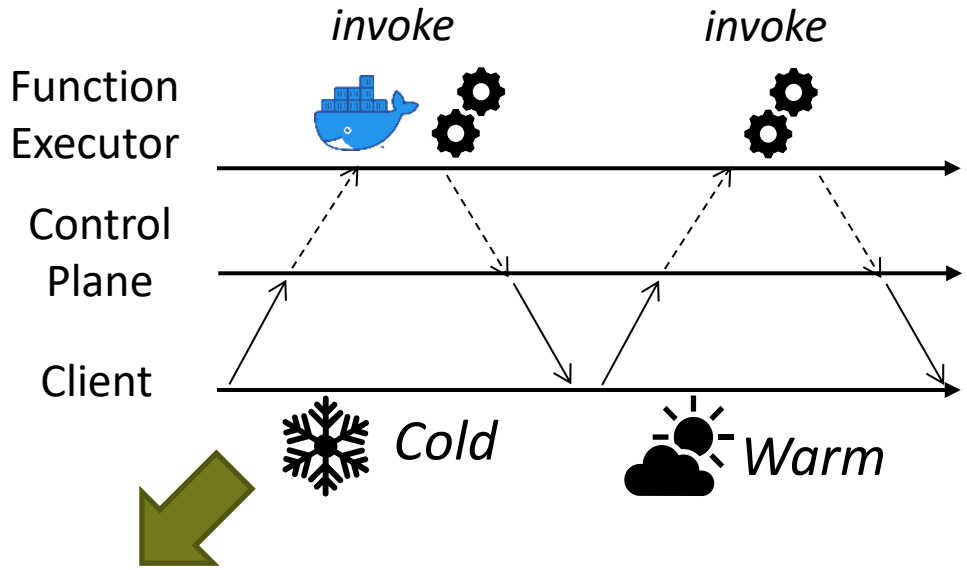
Executor

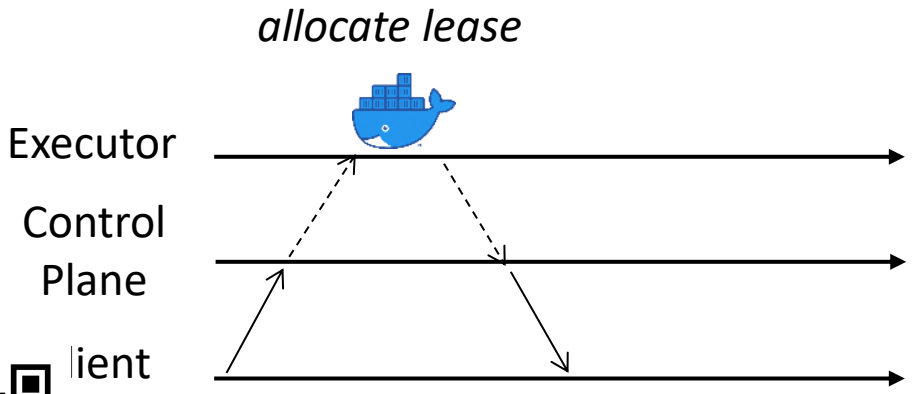
Control Plane

lient

“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

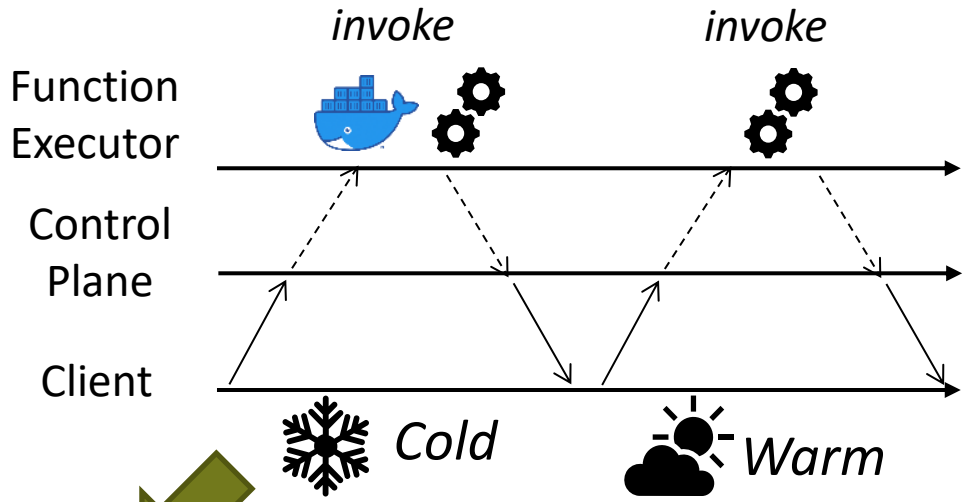
rFaaS: Invocations with Leases

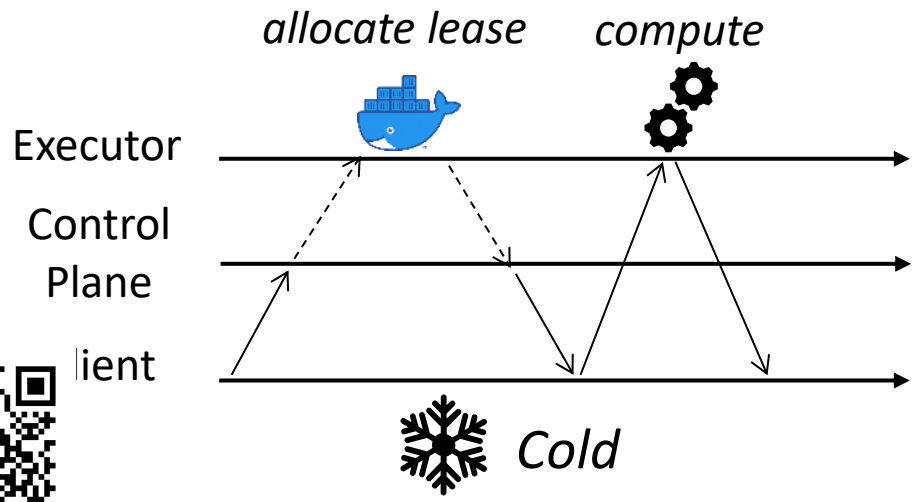




“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

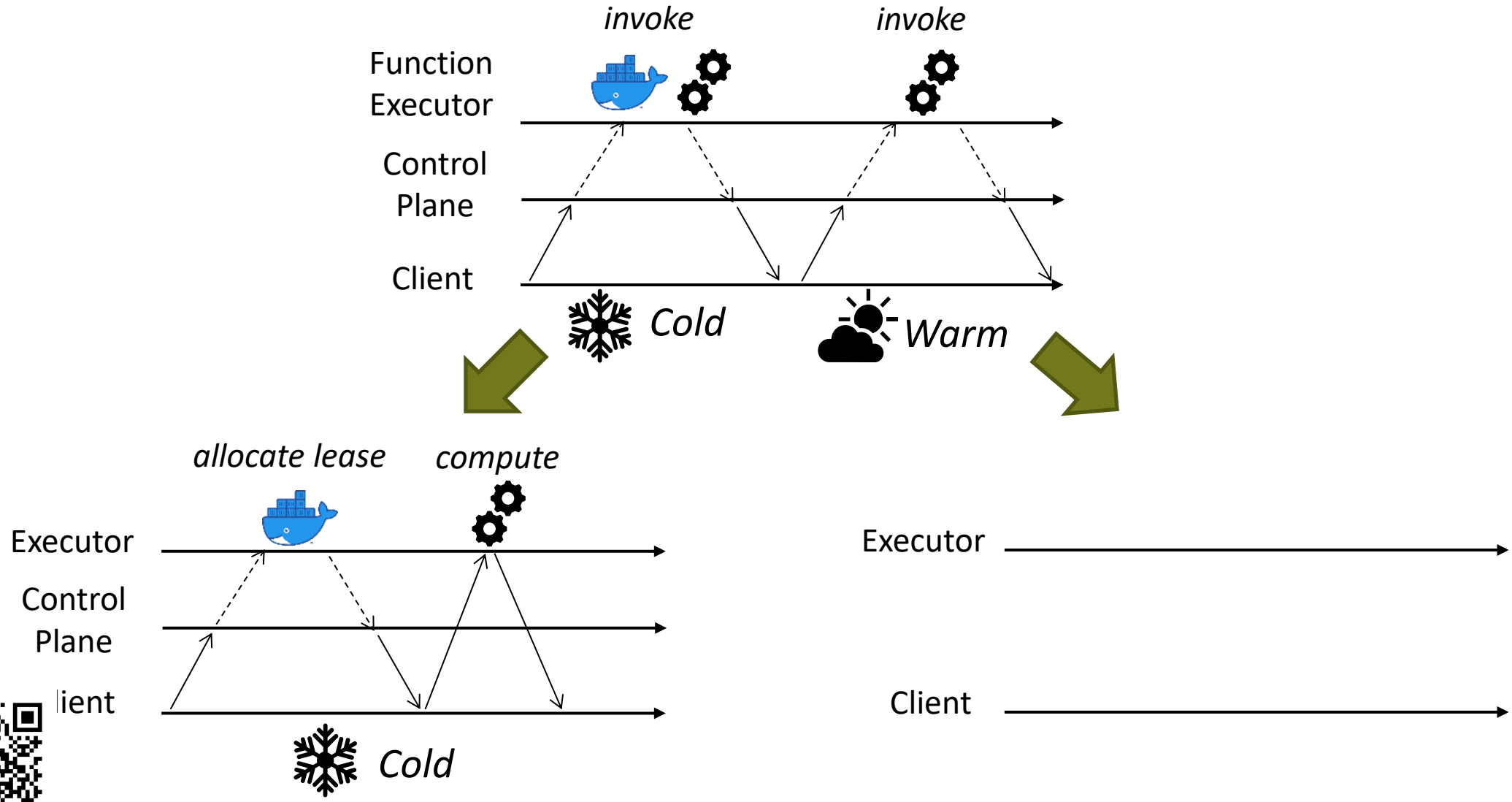
rFaaS: Invocations with Leases





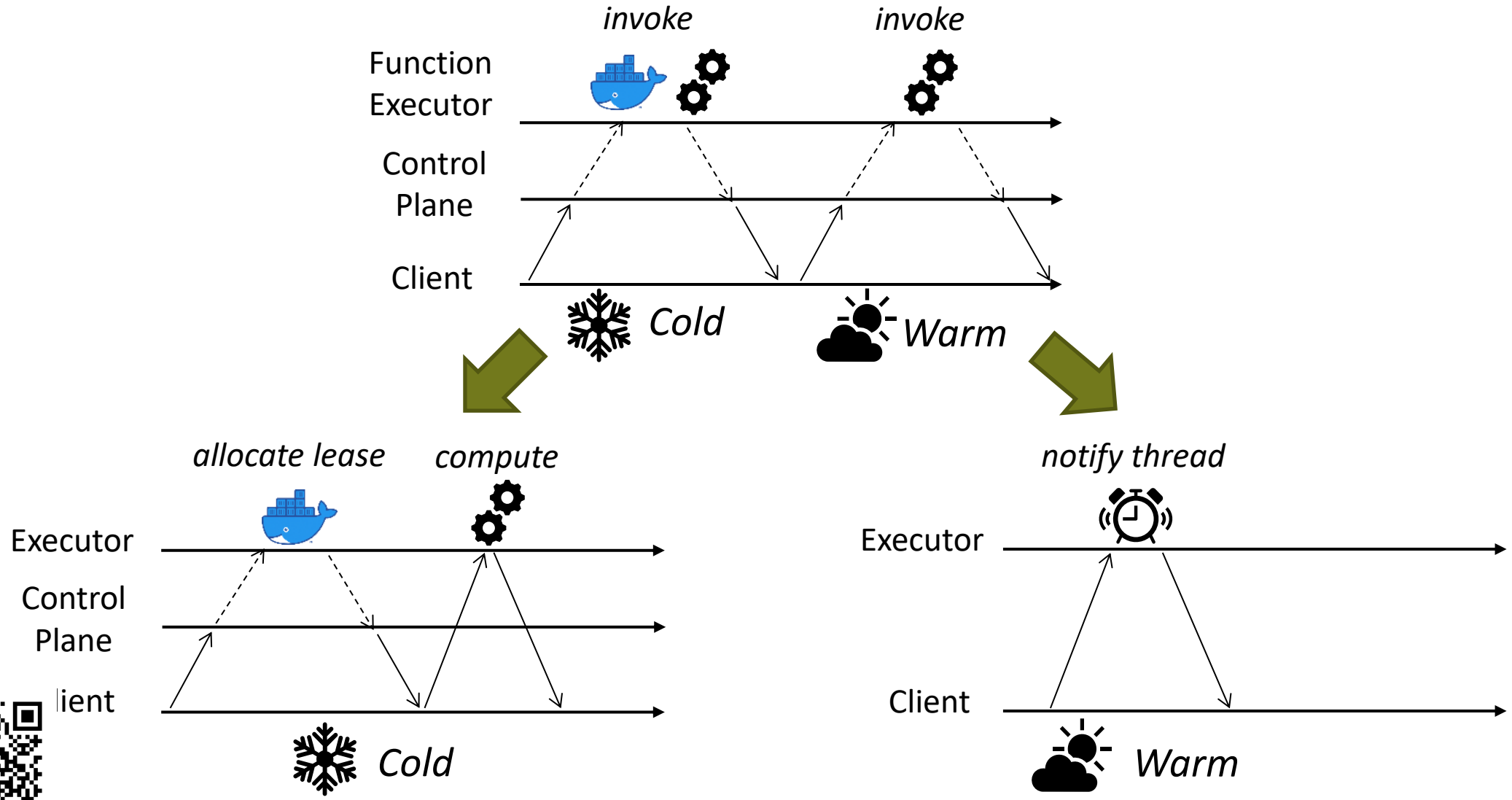
“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: Invocations with Leases

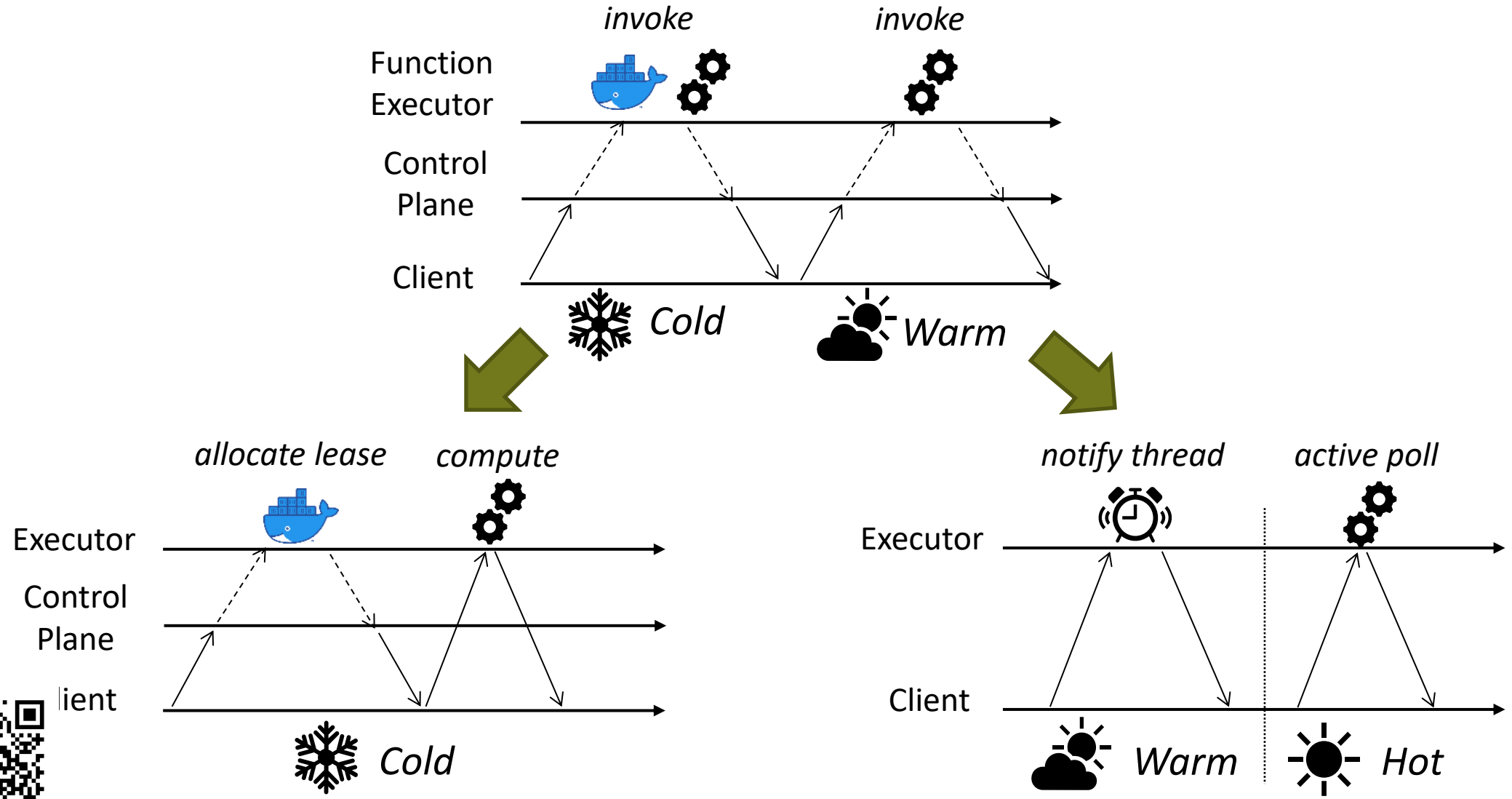


“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: Invocations with Leases



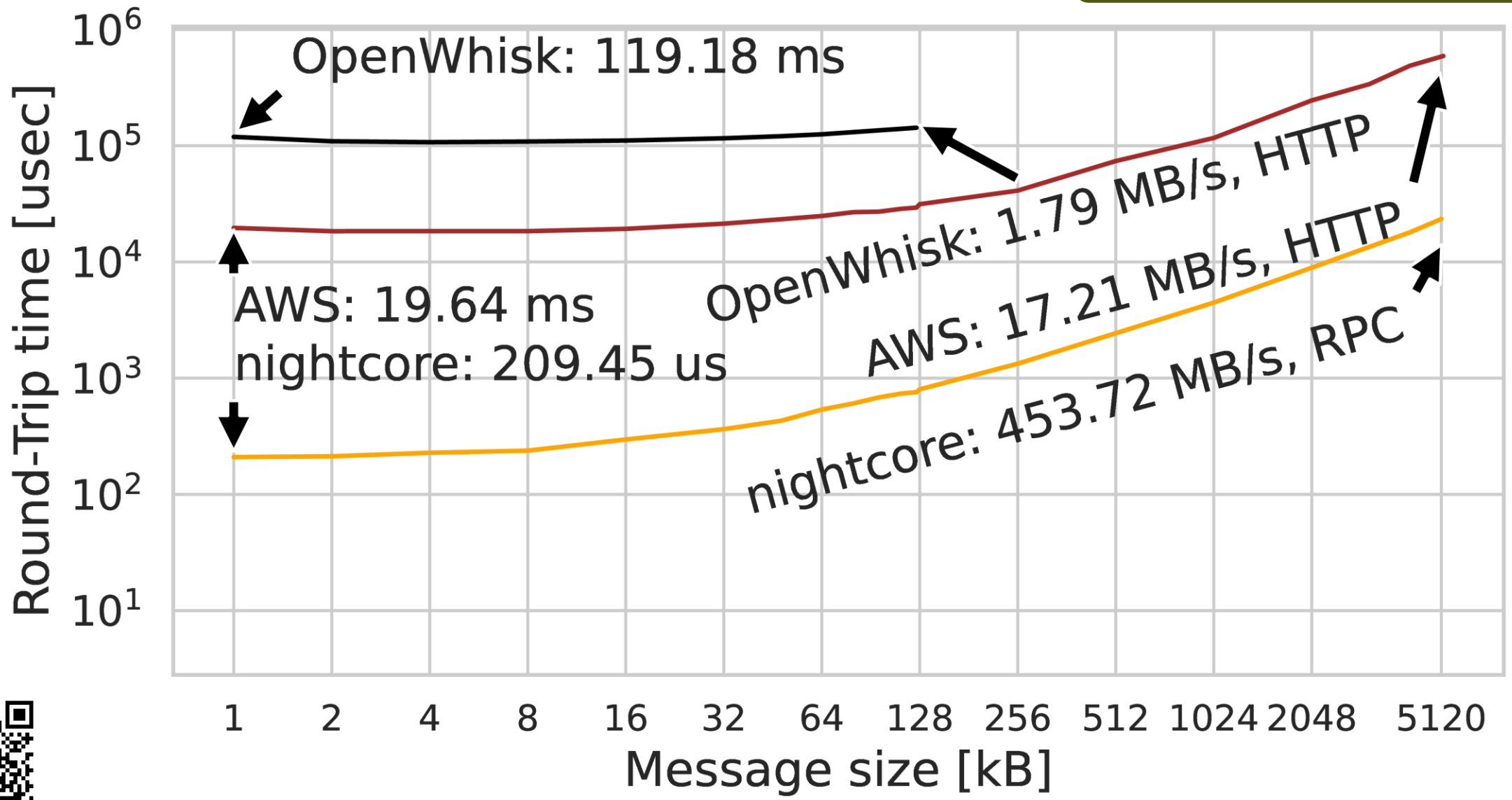
rFaaS: Invocations with Leases



“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: How Fast are Invocations in FaaS?

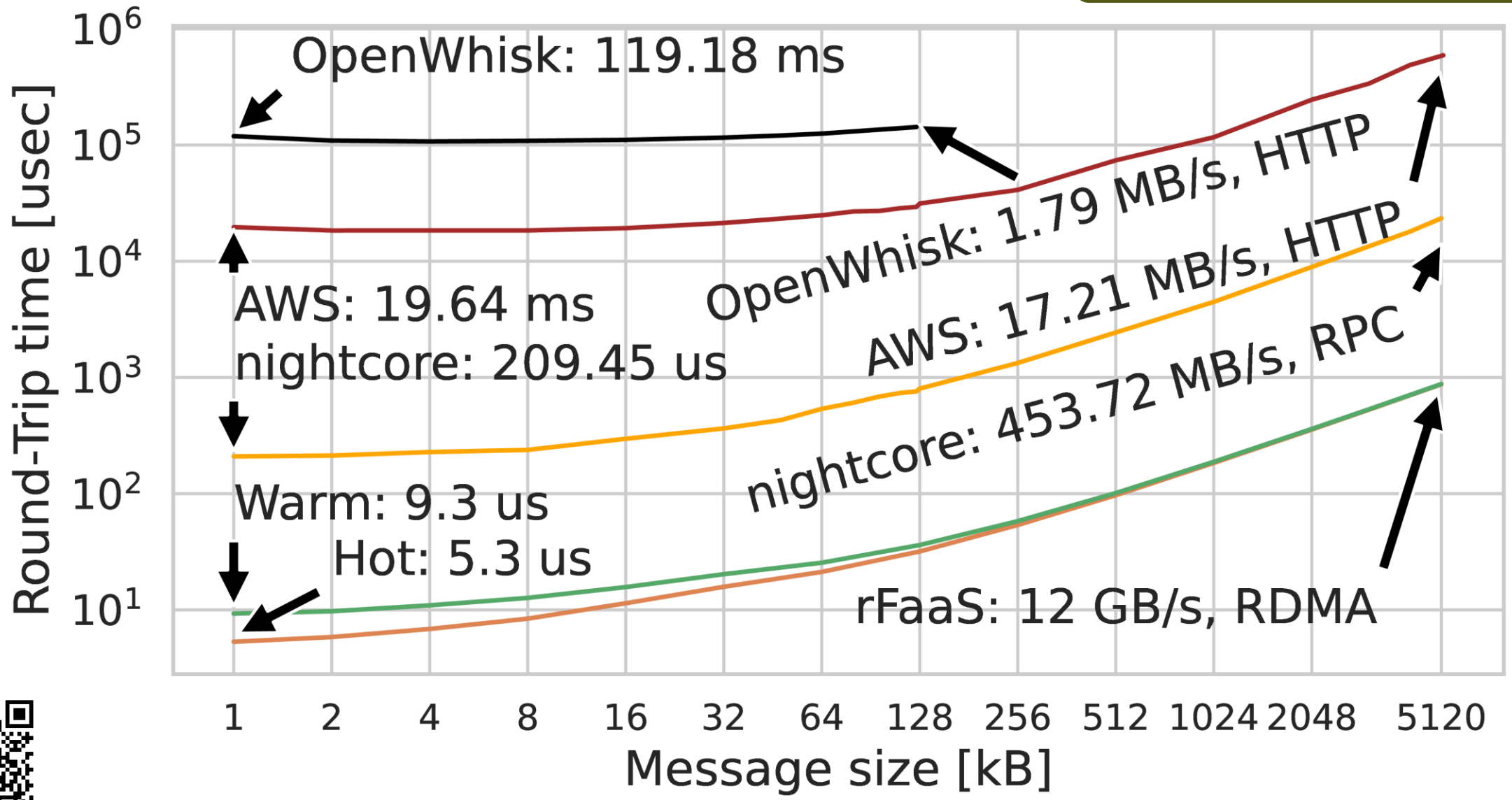
36 CPU cores, 377 GB memory.
100 Gbps Ethernet with RoCEv2 support.



“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

rFaaS: How Fast are Invocations in FaaS?

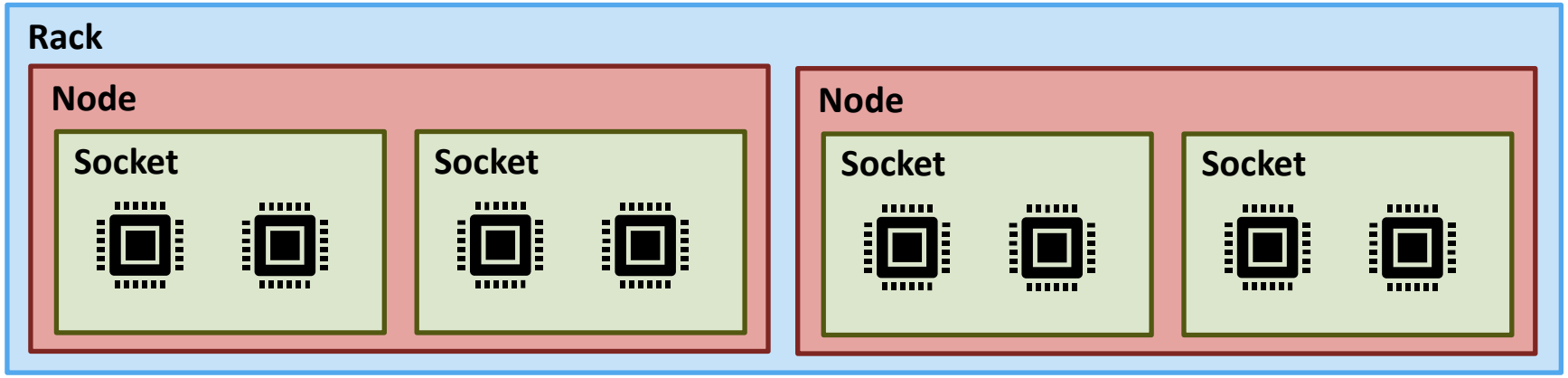
36 CPU cores, 377 GB memory.
100 Gbps Ethernet with RoCEv2 support.



"rFaaS: Enabling High Performance Serverless with RDMA and Leases", IPDPS'23

FMI: Serverless Communication

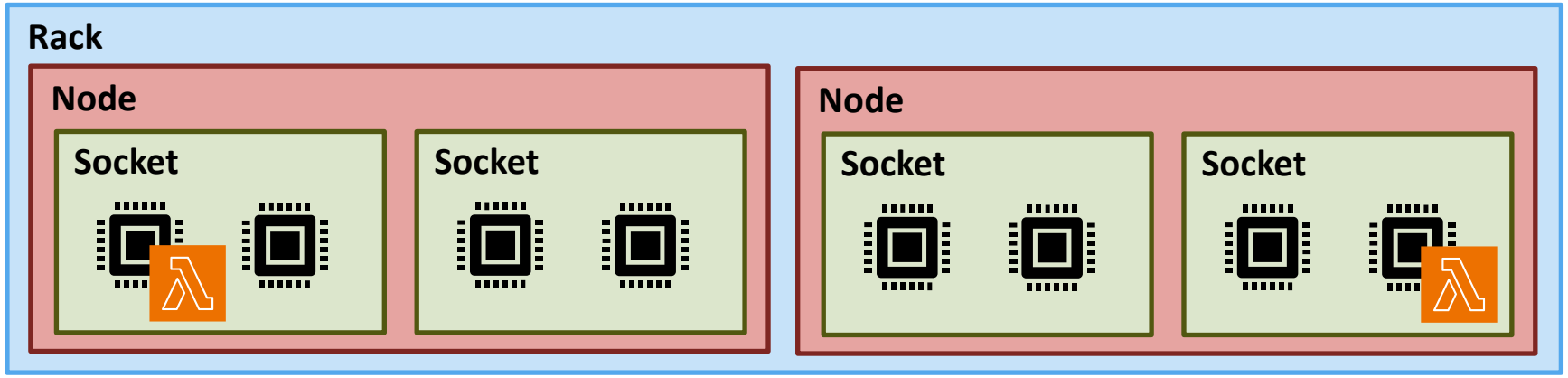
- We need **high performance.**
- We need **portable performance.**



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Serverless Communication

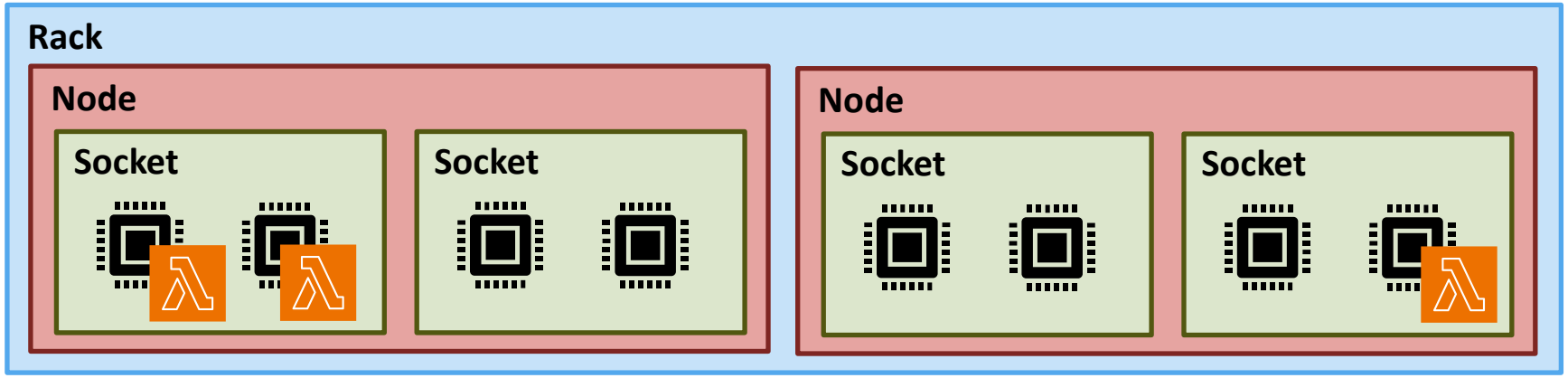
- We need **high performance.**
- We need **portable performance.**



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Serverless Communication

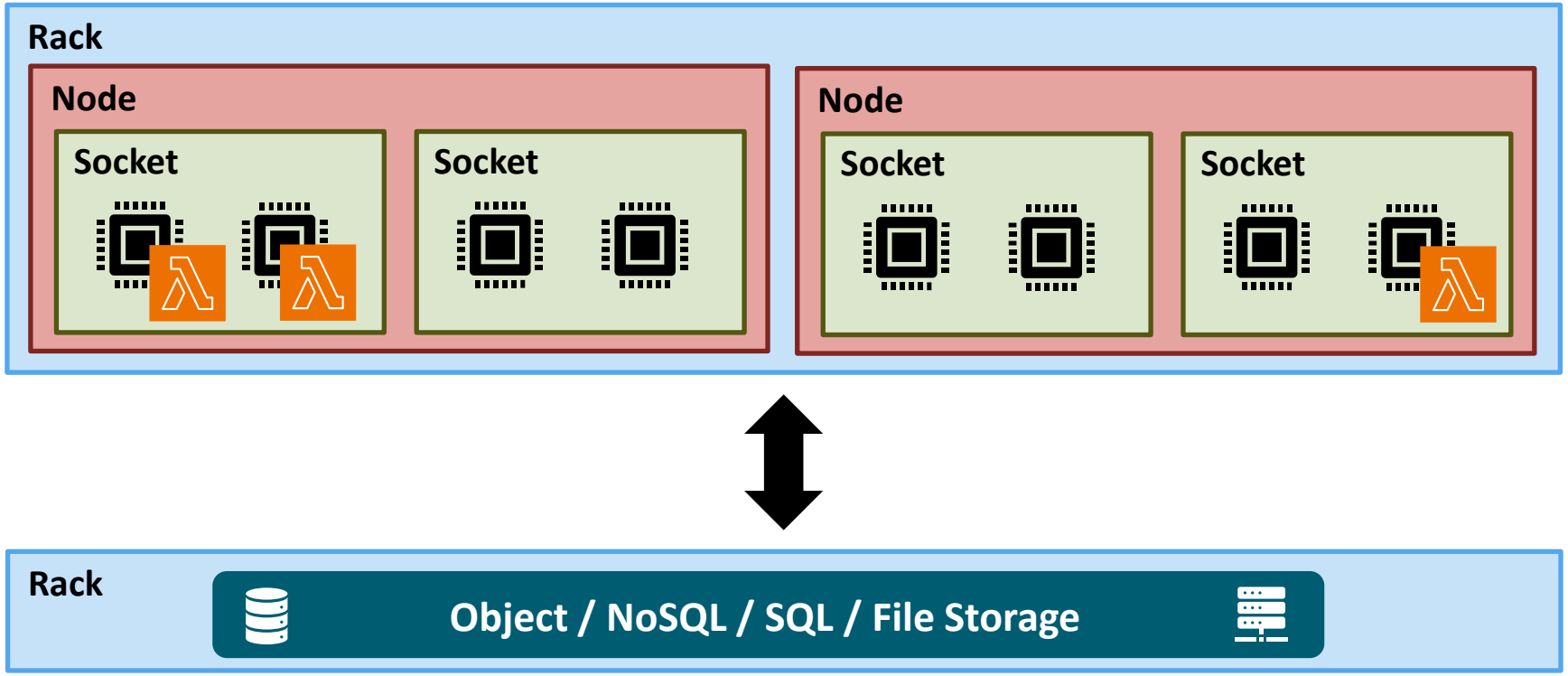
- We need **high performance.**
- We need **portable performance.**



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Serverless Communication

- We need **high performance.**
- We need **portable performance.**



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

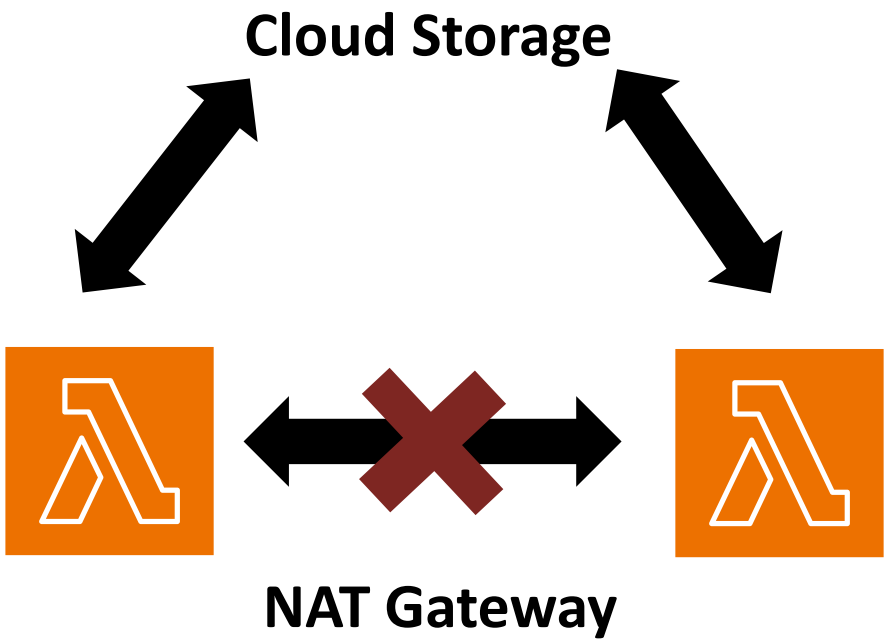
“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

NAT Gateway

“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

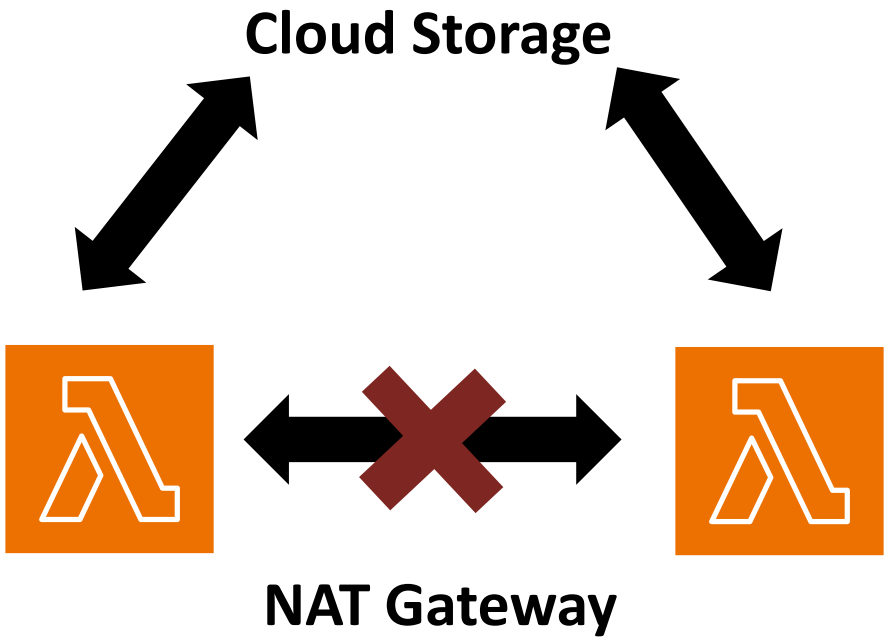


“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

High Latency
For Small Messages

S3



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

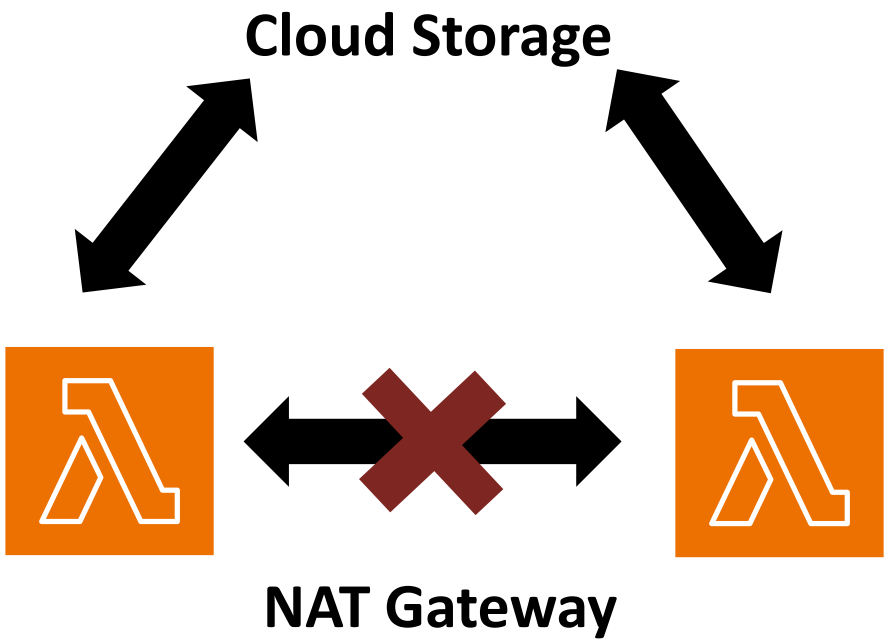
FMI: Slow Communication in Serverless

High Latency
For Small Messages

S3

Expensive for
Large Messages

DynamoDB



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

High Latency
For Small Messages

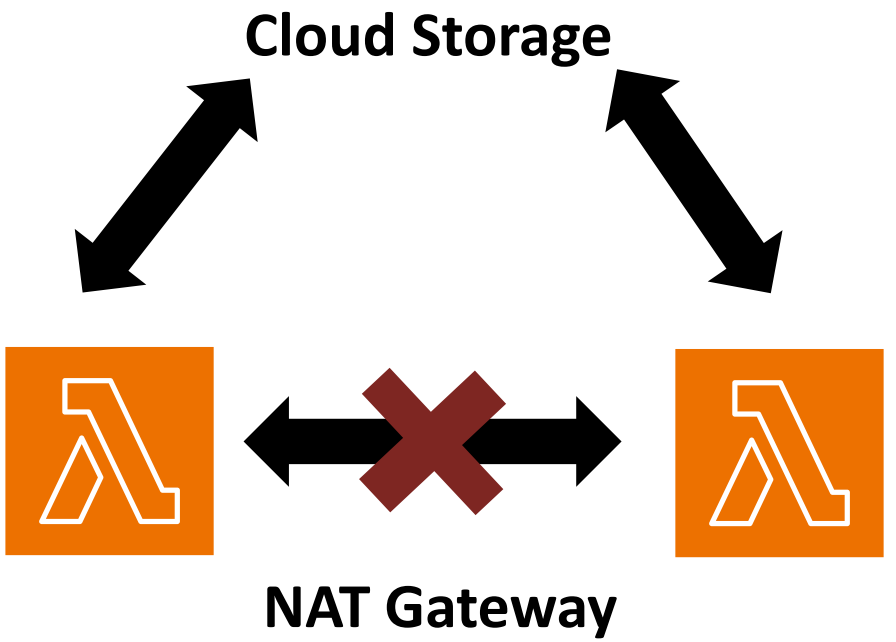
S3

Expensive for
Large Messages

DynamoDB

Not Serverless
Expensive

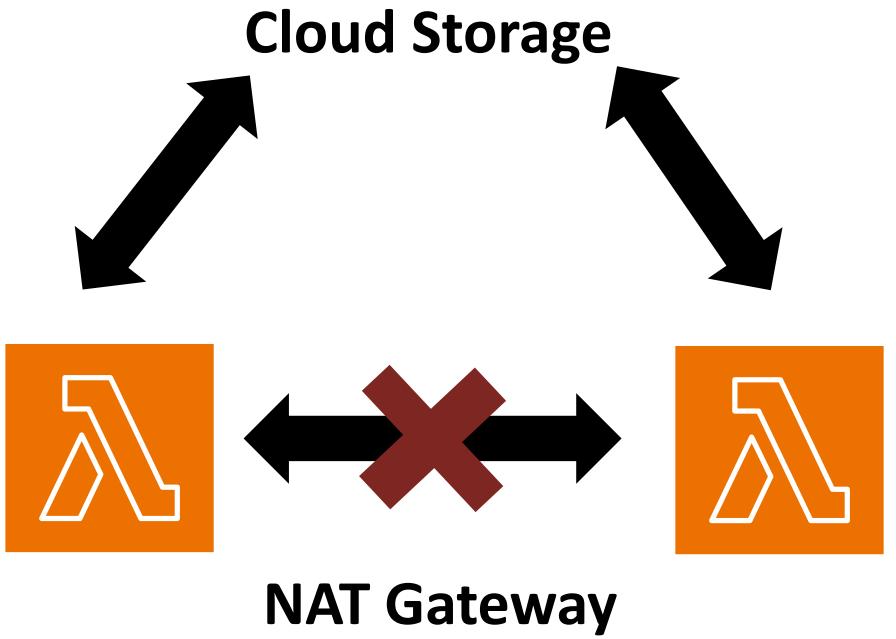
Redis



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

FMI: Slow Communication in Serverless

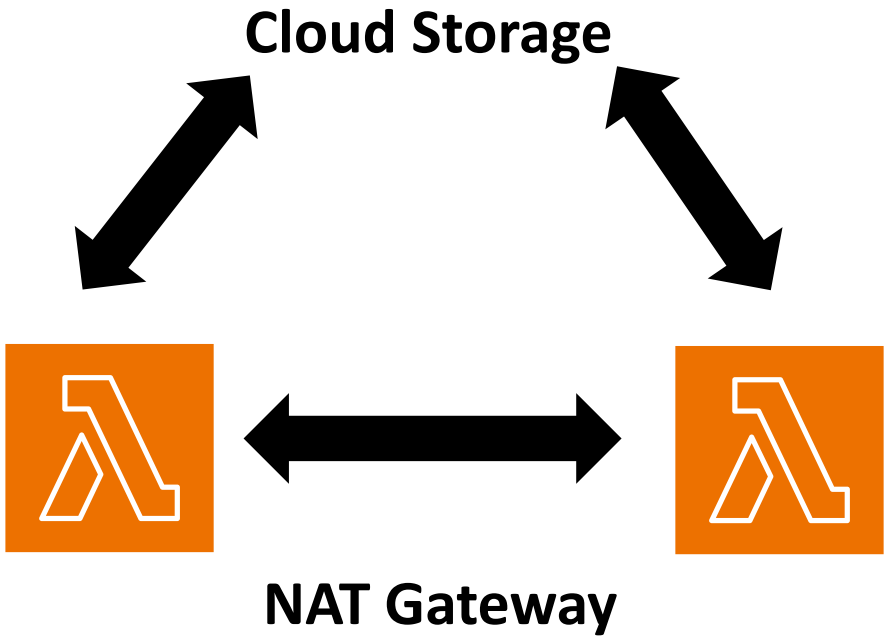
High Latency For Small Messages	Expensive for Large Messages	Not Serverless Expensive	Size Limits Good Latency
			
S3	DynamoDB	Redis	SQS



"FMI: Fast and Cheap Message Passing for Serverless Functions", ICS'23

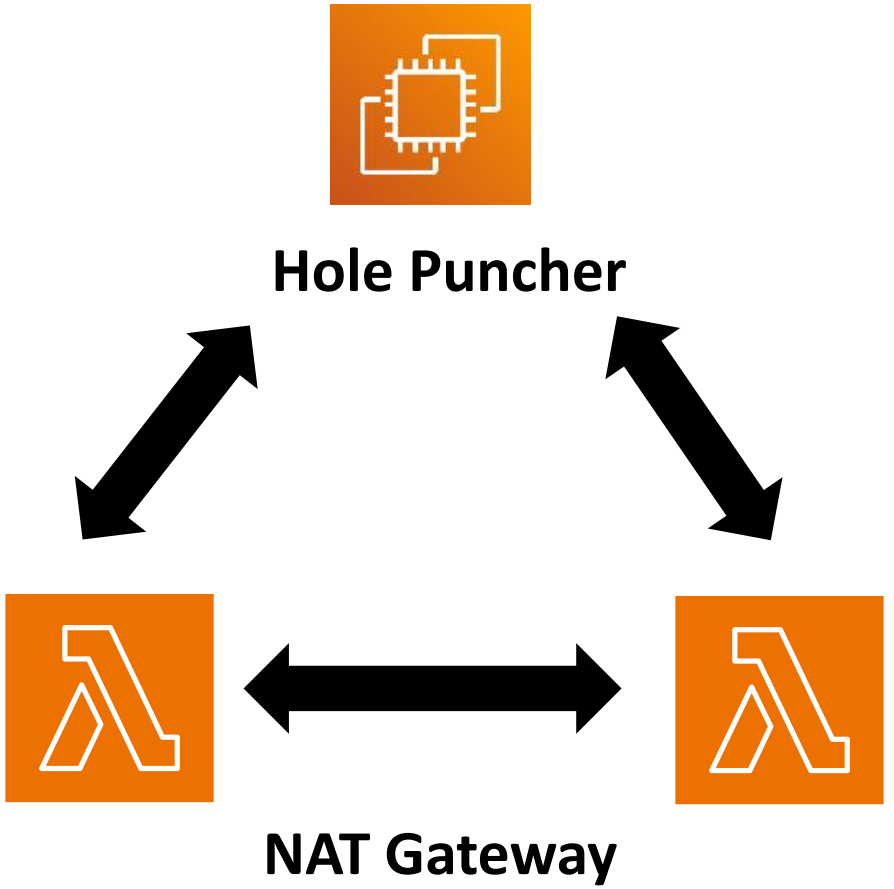
FMI: Slow Communication in Serverless

High Latency For Small Messages	Expensive for Large Messages	Not Serverless Expensive	Size Limits Good Latency
			
S3	DynamoDB	Redis	SQS



"FMI: Fast and Cheap Message Passing for Serverless Functions", ICS'23

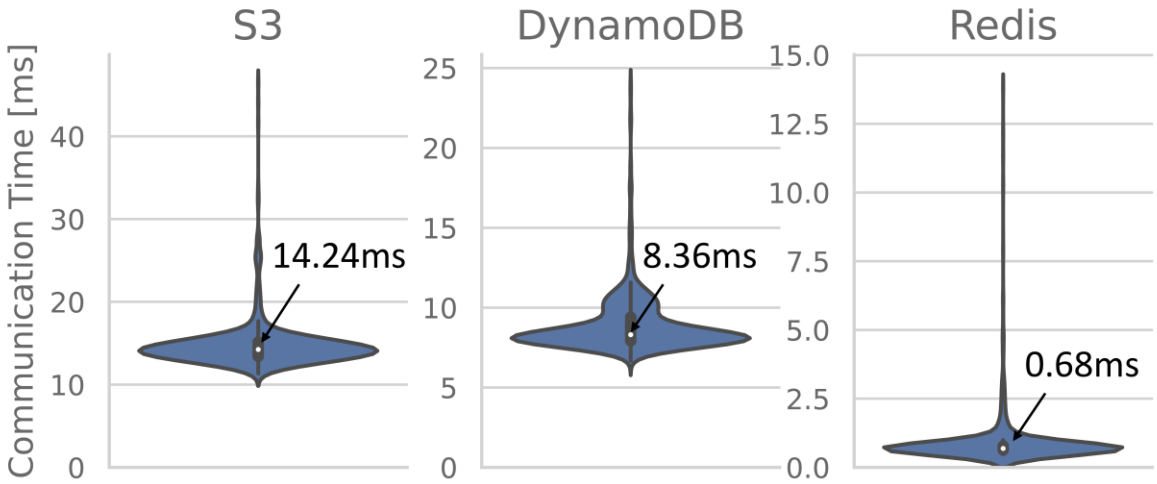
FMI: Fast Communication in Serverless



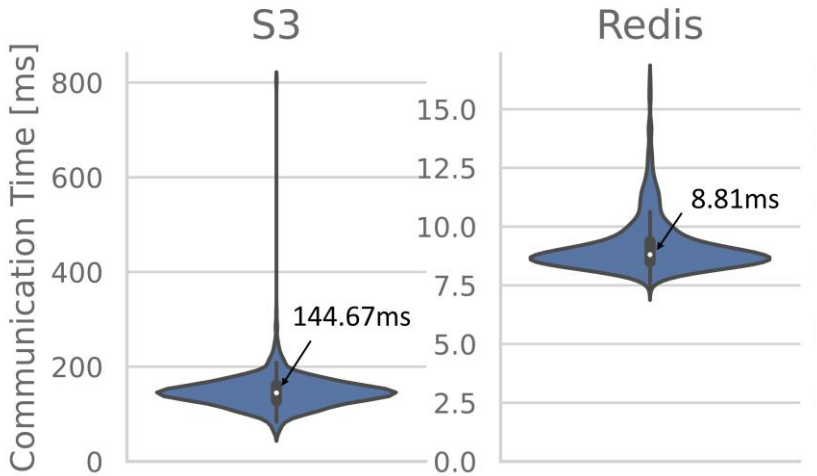
“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

Serverless Applications: Communication

1 B



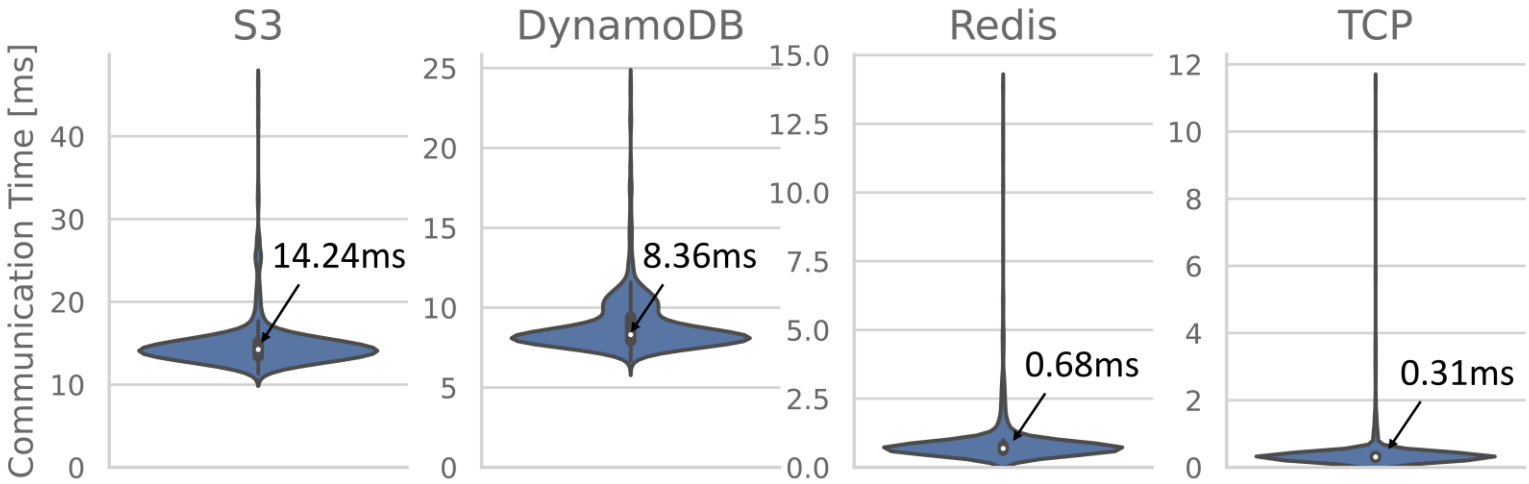
1 MB



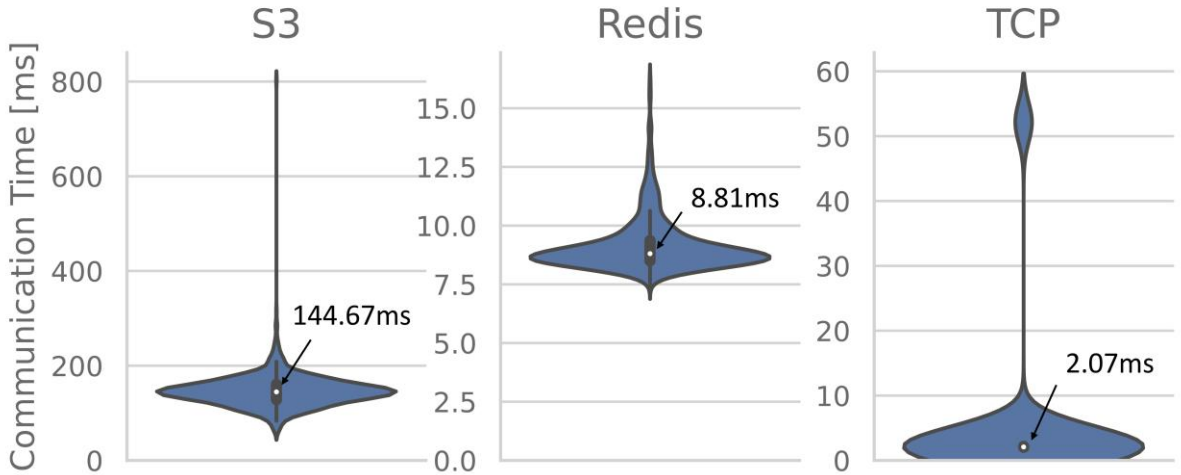
"FMI: Fast and Cheap Message Passing for Serverless Functions", ICS'23

Serverless Applications: Communication

1 B

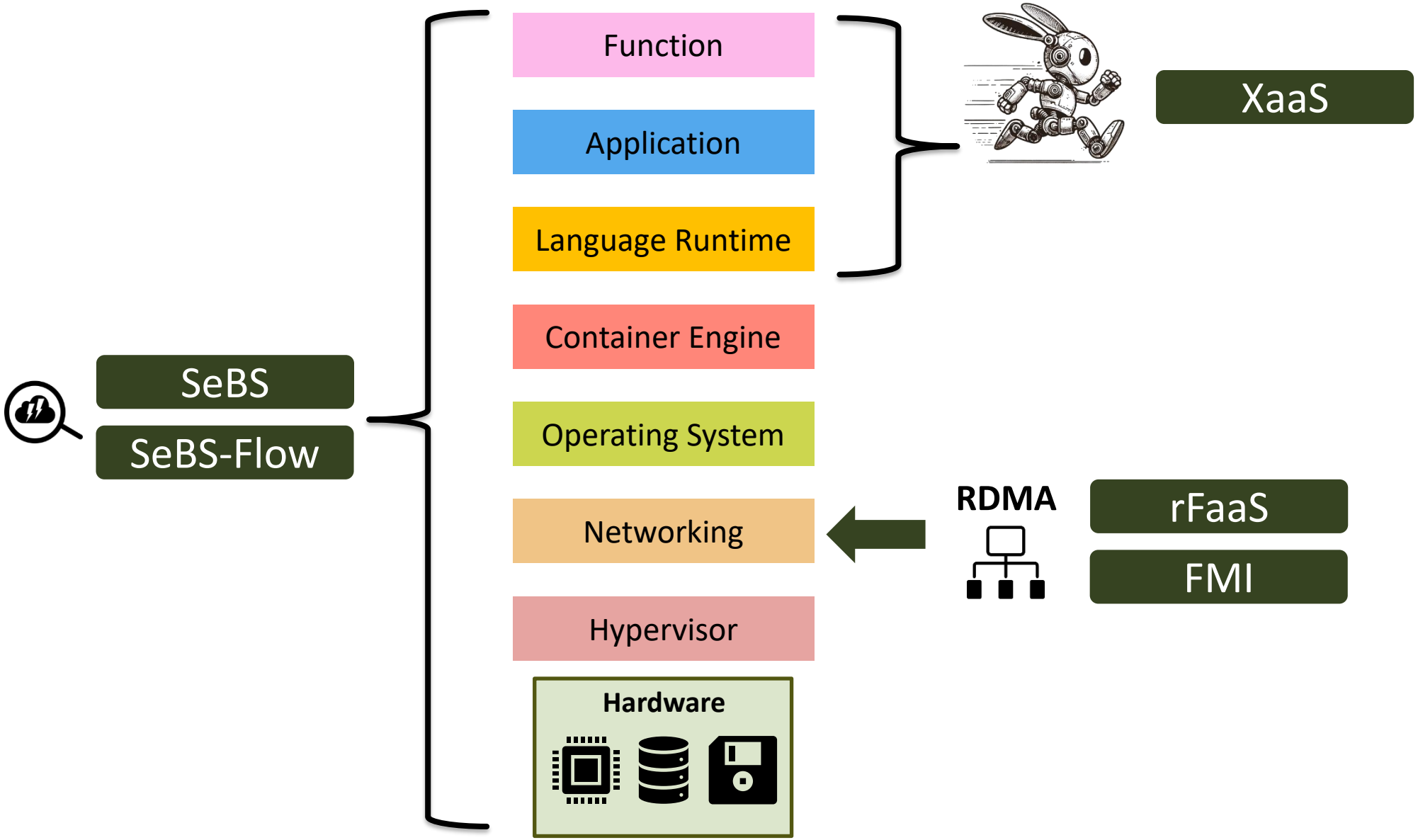


1 MB

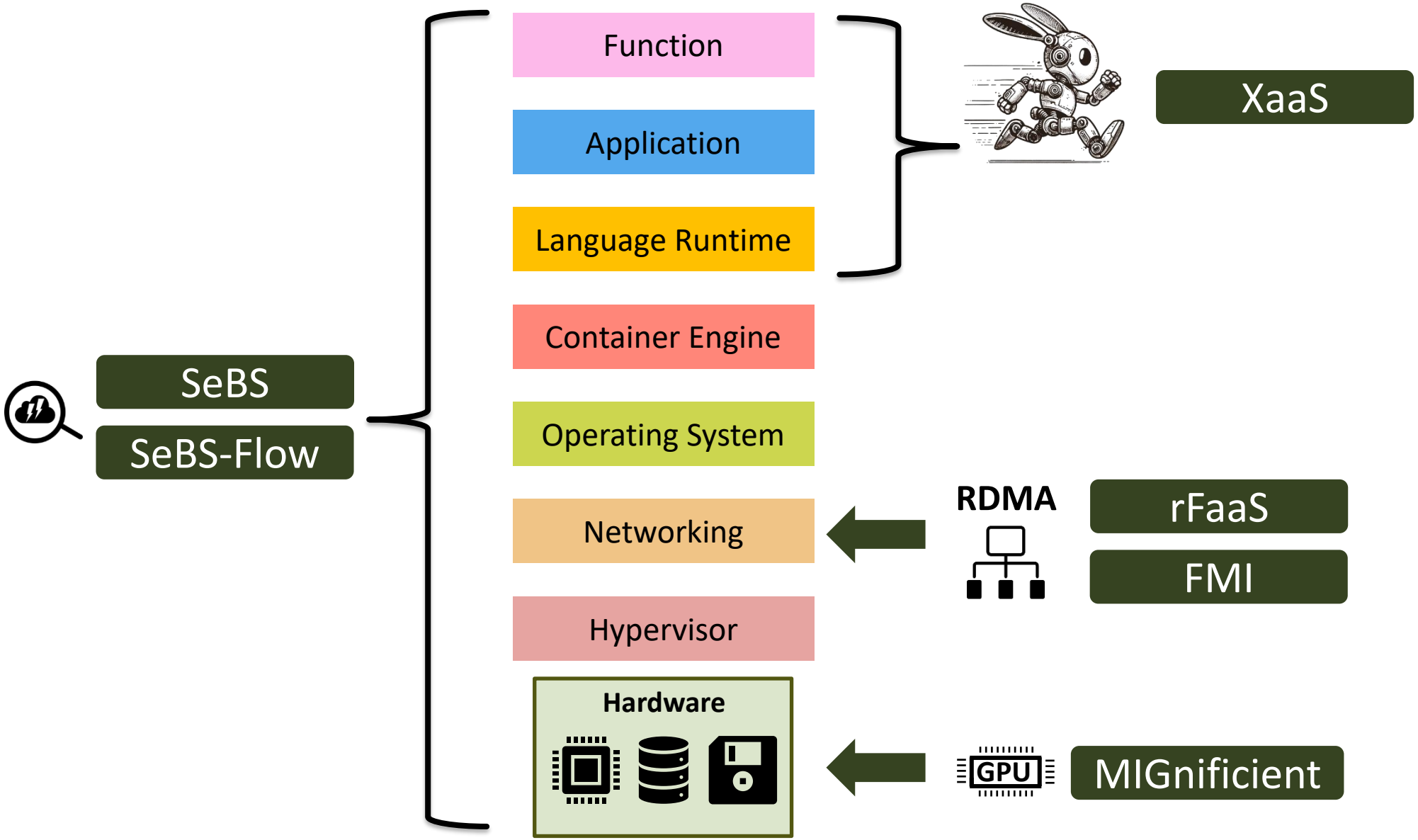


"FMI: Fast and Cheap Message Passing for Serverless Functions", ICS'23

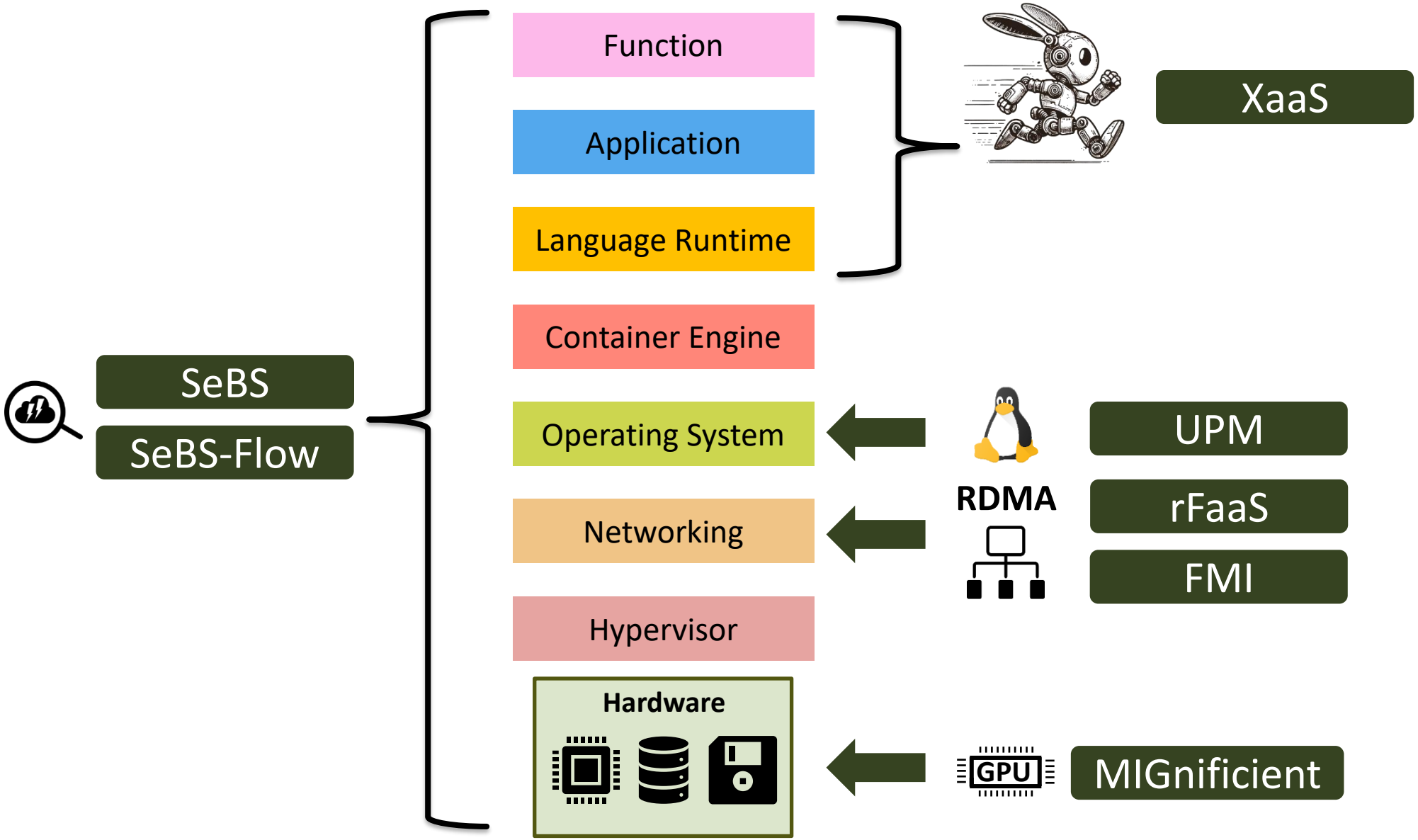
AXelerating Serverless Computing



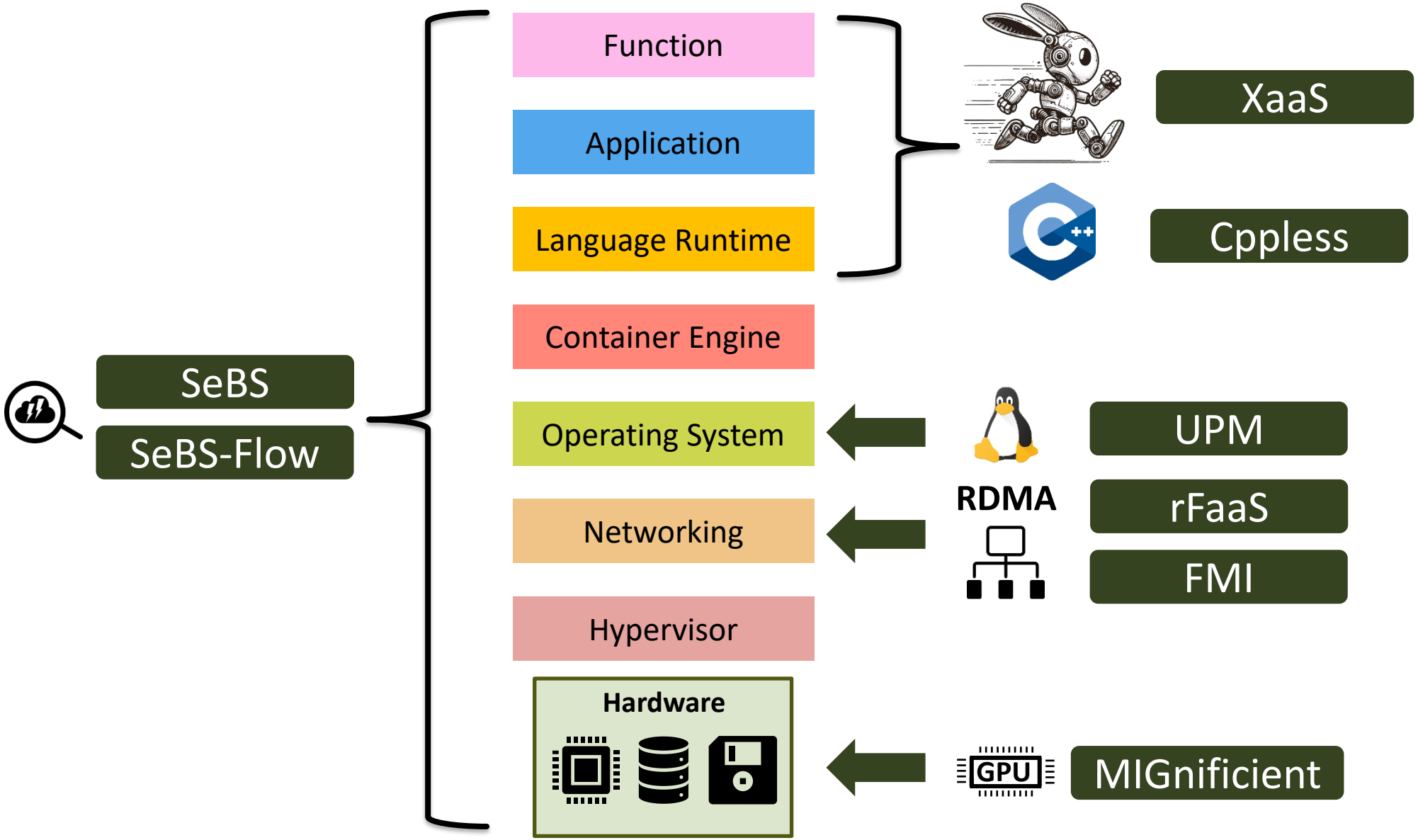
AXelerating Serverless Computing



AXelerating Serverless Computing



AXelerating Serverless Computing



Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{









}
```

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }









}
```

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }
  auto iterations = json.GetInt64("iterations");
  auto result = pi_estimation(iterations);
  auto response = std::to_string(result);



}
```

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }
  auto iterations = json.GetInt64("iterations");
  auto result = pi_estimation(iterations);
  auto response = std:::to_string(result);
  return invocation_response:::success(
    response, "application/json"
  );
}
```

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }
  auto iterations = json.GetInt64("iterations");
  auto result = pi_estimation(iterations);
  auto response = std:::to_string(result);
  return invocation_response:::success(
    response, "application/json"
  );
}
```

(Cross) Compile to shared library.

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }
  auto iterations = json.GetInt64("iterations");
  auto result = pi_estimation(iterations);
  auto response = std:::to_string(result);
  return invocation_response:::success(
    response, "application/json"
  );
}
```

(Cross) Compile to shared library.

Link with custom runtime.

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }
  auto iterations = json.GetInt64("iterations");
  auto result = pi_estimation(iterations);
  auto response = std:::to_string(result);
  return invocation_response:::success(
    response, "application/json"
  );
}
```

(Cross) Compile to shared library.

↓

Link with custom runtime.

↓

Upload to cloud with all dependencies.

Let's deploy C++ on AWS Lambda!

```cpp
#include <aws/lambda-runtime/runtime.h>
#include <aws/core/utils/json/JsonSerializer.h>
#include <aws/core/utils/memory/stl/SimpleStringStream.h>

using namespace aws::lambda_runtime;

invocation_response my_handler(invocation_request const& request)
{
  using namespace Aws::Utils::Json;
  JsonValue json(request.payload);
  if (!json.WasParseSuccessful()) {
    return invocation_response::failure(
      "Failed to parse input JSON", "InvalidJSON"
    );
  }
  auto iterations = json.GetInt64("iterations");
  auto result = pi_estimation(iterations);
  auto response = std:::to_string(result);
  return invocation_response:::success(
    response, "application/json"
  );
}
```

(Cross) Compile to shared library.

↓

Link with custom runtime.

↓

Upload to cloud with all dependencies.

Let's deploy C++ on AWS Lambda!

```cpp
bool InvokeFunction(
  const Aws::String& functionName,
  std::shared_ptr<Aws::Lambda::LambdaClient> client,
  int invocations, int &result
)
{
  Aws::Lambda::Model::InvokeRequest invokeRequest;
  invokeRequest.SetFunctionName(functionName);
  invokeRequest.SetInvocationType(
  Aws:::Lambda::Model::InvocationType::RequestResponse);

  std::shared_ptr<Aws::IOStream> payload
        = Aws::MakeShared<Aws::StringStream>();
  Aws::Utils::Json::JsonValue jsonPayload;
  jsonPayload.WithInt64("iterations", iterations);
  *payload <<< jsonPayload.View().WriteReadable();
  invokeRequest.SetBody(payload);
  invokeRequest.SetContentType("application/json");

  …

}
```

(Cross) Compile to shared library.

Link with custom runtime.

Upload to cloud with all dependencies.

Let's deploy C++ on AWS Lambda!

```
bool InvokeFunction(
)
{
  {
    …

    auto outcome = client->Invoke(invokeRequest);

    if (outcome.IsSuccess()) {
      auto &result = outcome.GetResult();
      Aws::IOStream &payload = result.GetPayload();
      Aws::String functionResult;
      std::getline(payload, functionResult);
      result = std::stoi(functionResult);
      return true;
    } else {
      return false;
    }

  }

  …

}
```

(Cross) Compile to shared library.

Link with custom runtime.

Upload to cloud with all dependencies.

Let's deploy C++ on AWS Lambda!

```cpp
Aws::SDKOptions options;
Aws::InitAPI(options);
Aws::Client::ClientConfiguration clientConfig;
auto m_client = Aws::MakeShared<Aws::Lambda::LambdaClient>(
  ALLOCATION_TAG,
  clientConfig
);

int n = 10000;
int np = 10;
std::vector<int> results(np);
std::vector<std::thread> threads;
for (int i = 0; i < np; i++) {
  threads.emplace_back([&, i]() {
    InvokeFunction("pi-mc-worker", n / np, results[i]);
  });
}

for (auto &thread : threads) {
  thread.join();
}
auto pi = std:::reduce(results.begin(), results.end()) / np;
```

(Cross) Compile to shared library.

↓

Link with custom runtime.

↓

Upload to cloud with all dependencies.

Challenges of C++ in Serverless

Challenges of C++ in Serverless

Complex, multi-source project setup

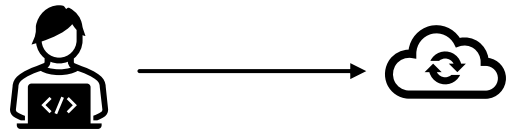
```
#pragma omp but in serverless
for(int i = 0; i < n; ++i)
  pi_mc(i);
```

Challenges of C++ in Serverless

Complex, multi-source project setup

```
#pragma omp but in serverless
for(int i = 0; i < n; ++i)
  pi_mc(i);
```

Cross-compiled environments

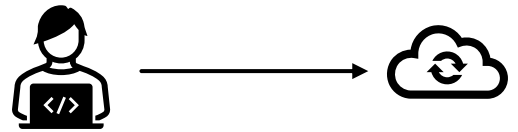


Challenges of C++ in Serverless

Complex, multi-source project setup

```
#pragma omp but in serverless
for(int i = 0; i < n; ++i)
  pi_mc(i);
```

Cross-compiled environments



Lack of static typing

Manual verification?
JSON Schema?

Cppless: Single-source C++ Compiler for Serverless

```
double pi_mc(int n);

double pi_estimate()
{
  const int n = 100000000;
  const int np = 128;

  cppless::aws_dispatcher dispatcher;
  auto aws = dispatcher.create_instance();

  std:::vector<double> results(np);
  auto fn = [=] { return pi_mc(n / np); };

  for (auto& result : results)
        cppless:::dispatch(aws, fn, result);
  cppless:::wait(aws, np);

  auto pi = std:::reduce(
    results.begin(), results.end()
  ) / np;

  return pi;
}
```

GitHub spcl/cppless

Cppless: Single-source C++ Compiler for Serverless

```
double pi_mc(int n);

double pi_estimate()
{
  const int n = 100000000;
  const int np = 128;

  cppless::aws_dispatcher dispatcher;
  auto aws = dispatcher.create_instance();

  std:::vector<double> results(np);
  auto fn = [=] { return pi_mc(n / np); };

  for (auto& result : results)
        cppless:::dispatch(aws, fn, result);
  cppless:::wait(aws, np);

  auto pi = std:::reduce(
    results.begin(), results.end()
  ) / np;

  return pi;
}
```

- **C++ abstraction for cloud provider APIs.**

  Avoids the vendor lock-in and simplifies invocations.

GitHub **spcl/cppless**

Cppless: Single-source C++ Compiler for Serverless

```
double pi_mc(int n);

double pi_estimate()
{
  const int n = 100000000;
  const int np = 128;

  cppless::aws_dispatcher dispatcher;
  auto aws = dispatcher.create_instance();

  std:::vector<double> results(np);
  auto fn = [=] { return pi_mc(n / np); };

  for (auto& result : results)
        cppless:::dispatch(aws, fn, result);
  cppless:::wait(aws, np);

  auto pi = std:::reduce(
    results.begin(), results.end()
  ) / np;

  return pi;
}
```

- **C++ abstraction for cloud provider APIs.**
  Avoids the vendor lock-in and simplifies invocations.
- **Serverless function as C++ lambda expression.**
  Automatically compiled to a cloud function.

GitHub **spcl/cppless**

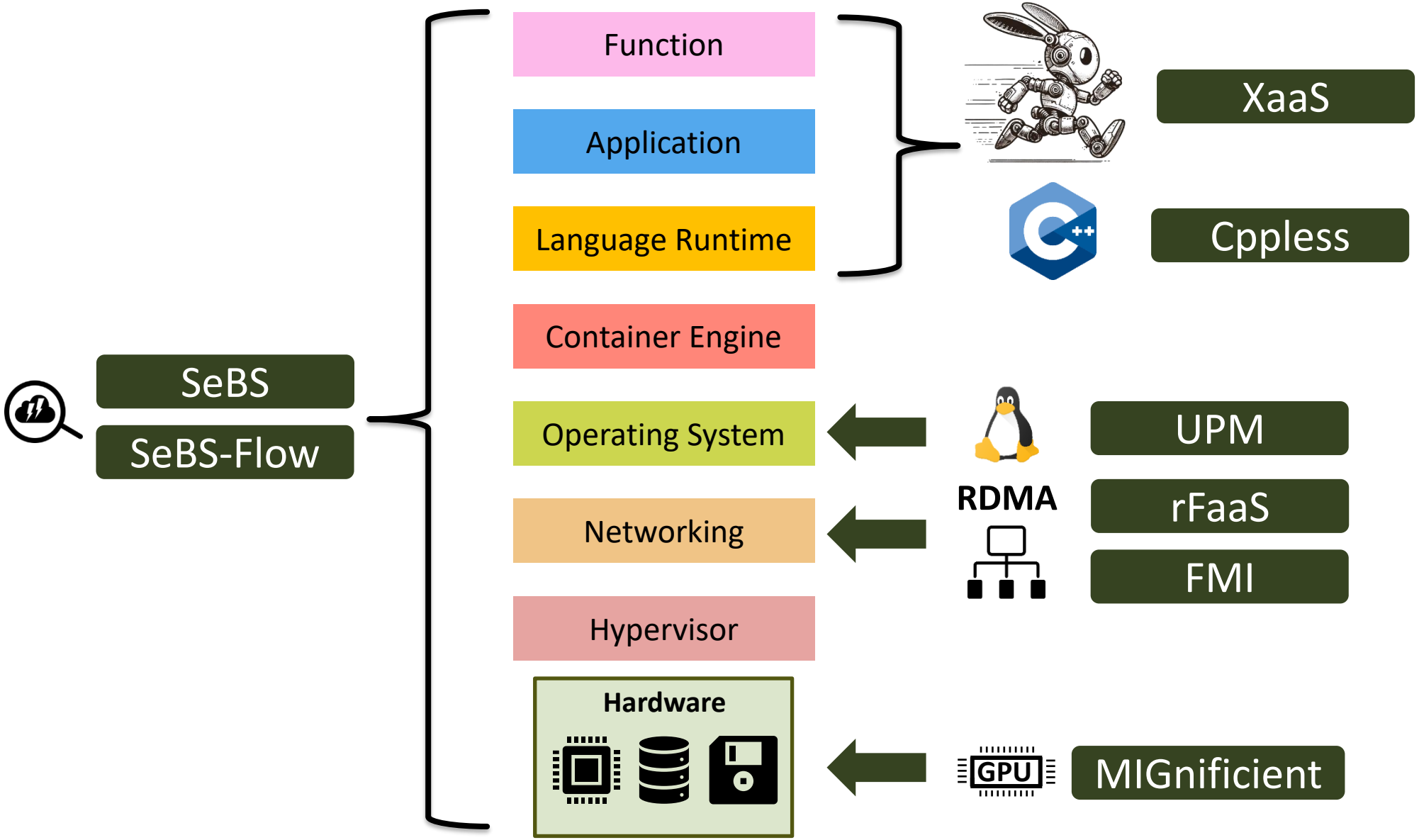
Cppless: Single-source C++ Compiler for Serverless

```
double pi_mc(int n);

double pi_estimate()
{
  const int n = 100000000;
  const int np = 128;

  cppless::aws_dispatcher dispatcher;
  auto aws = dispatcher.create_instance();

  std:::vector<double> results(np);
  auto fn = [=] { return pi_mc(n / np); };

  for (auto& result : results)
        cppless:::dispatch(aws, fn, result);
  cppless:::wait(aws, np);

  auto pi = std:::reduce(
    results.begin(), results.end()
  ) / np;

  return pi;
}
```

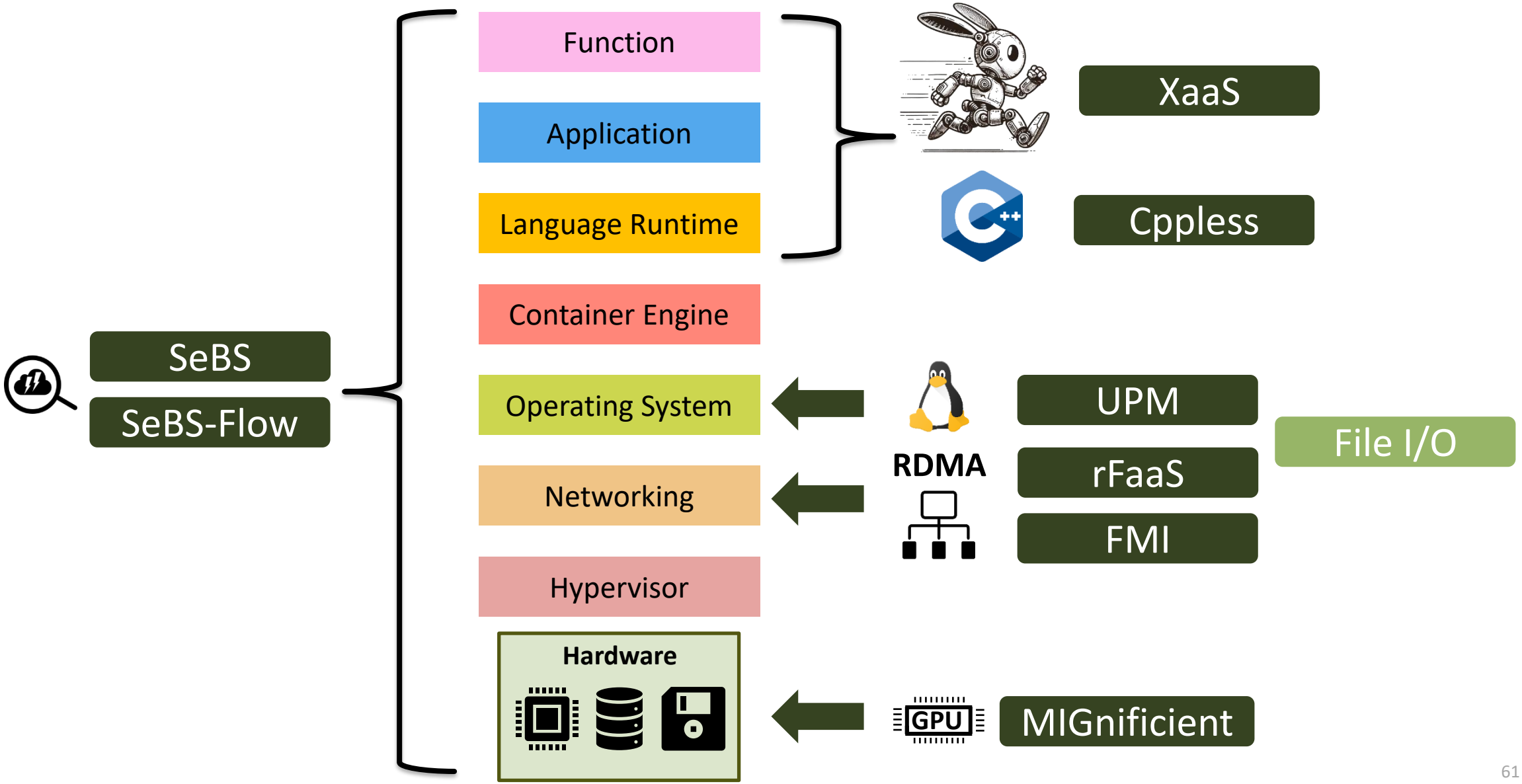
- **C++ abstraction for cloud provider APIs.**
  Avoids the vendor lock-in and simplifies invocations.
- **Serverless function as C++ lambda expression.**
  Automatically compiled to a cloud function.
- **Integrated invocation of the function.**
  Automatic serialization and type checking.

GitHub **spcl/cppless**

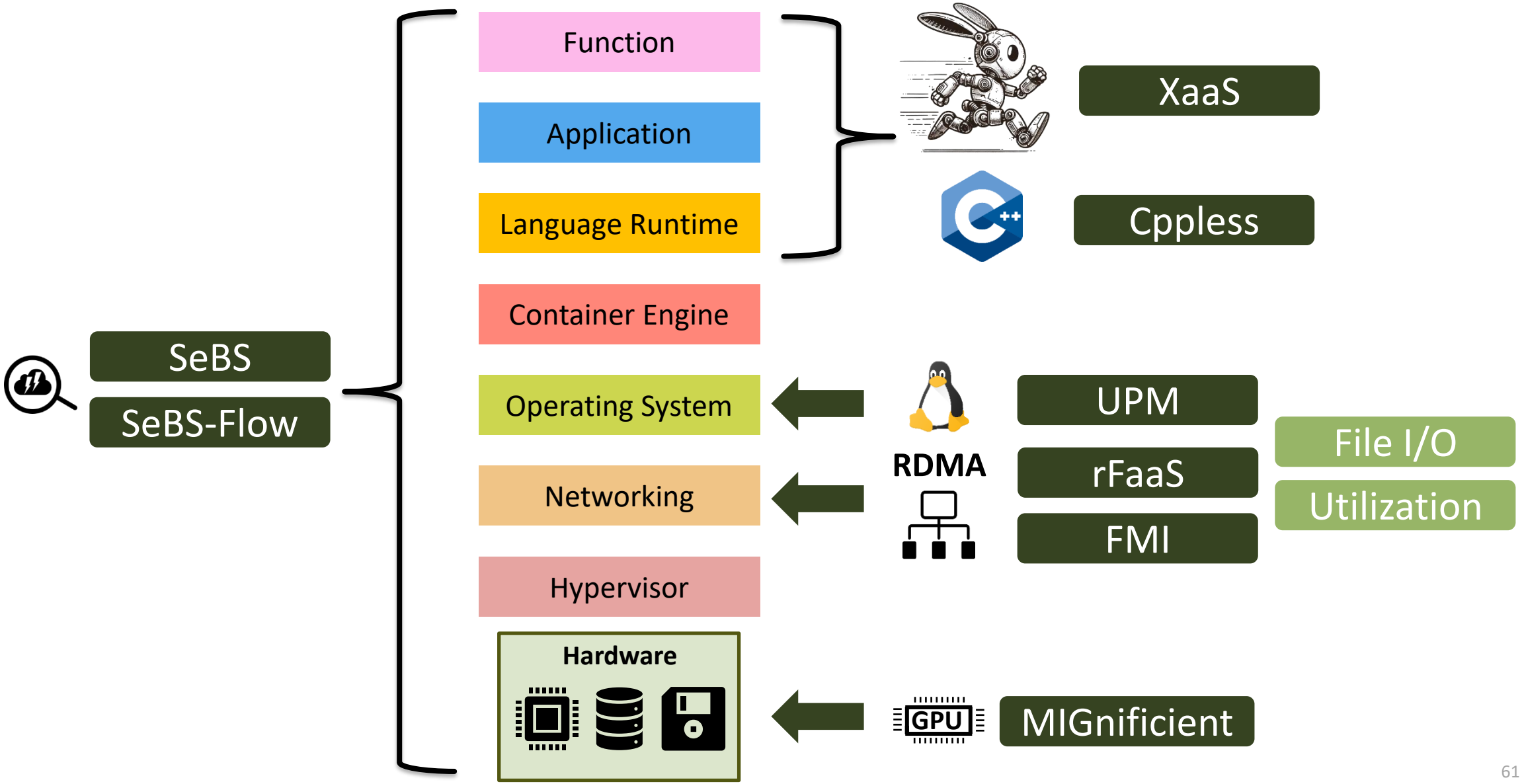
AXelerating Serverless Computing: Ongoing Work & Future



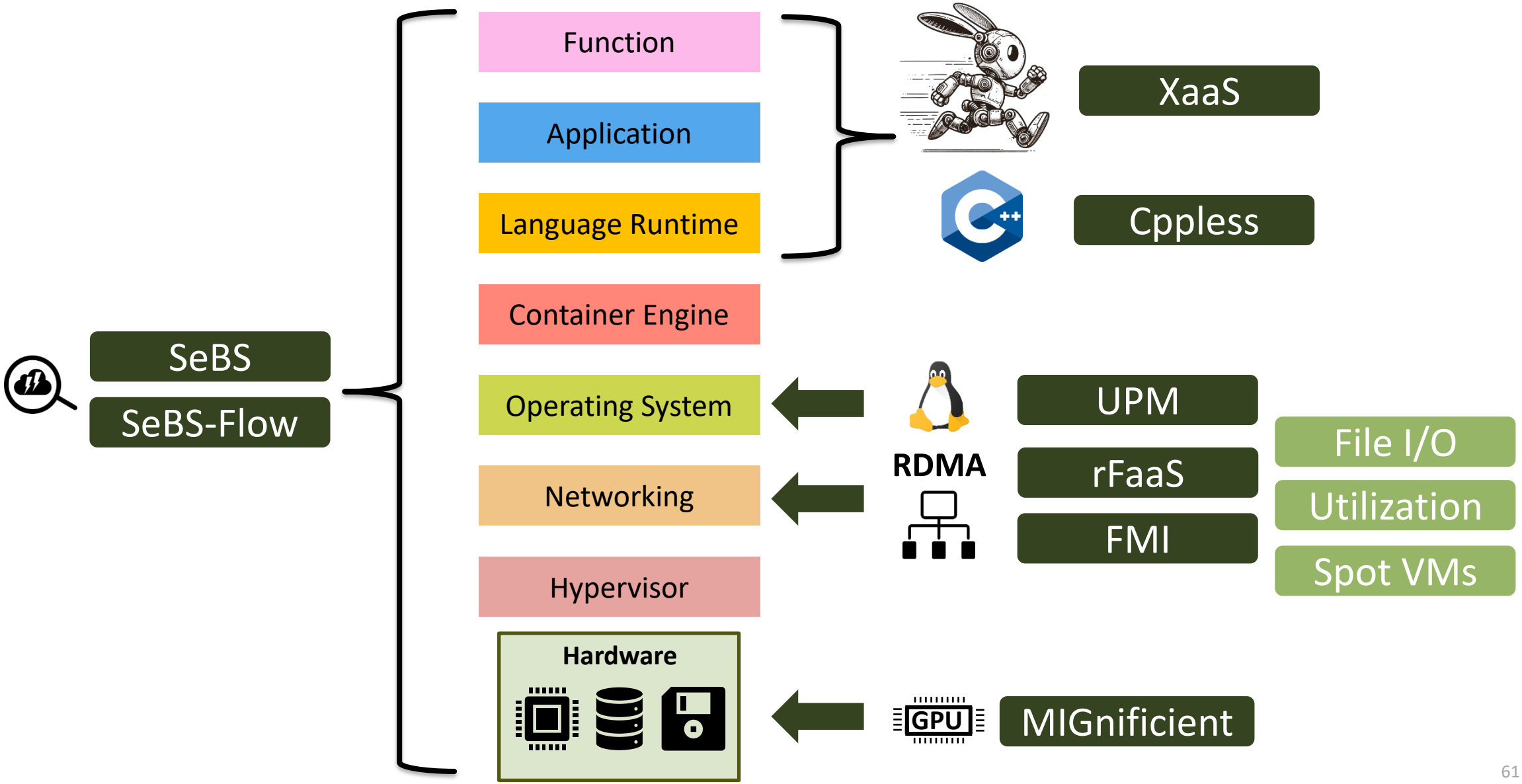
AXelerating Serverless Computing: Ongoing Work & Future



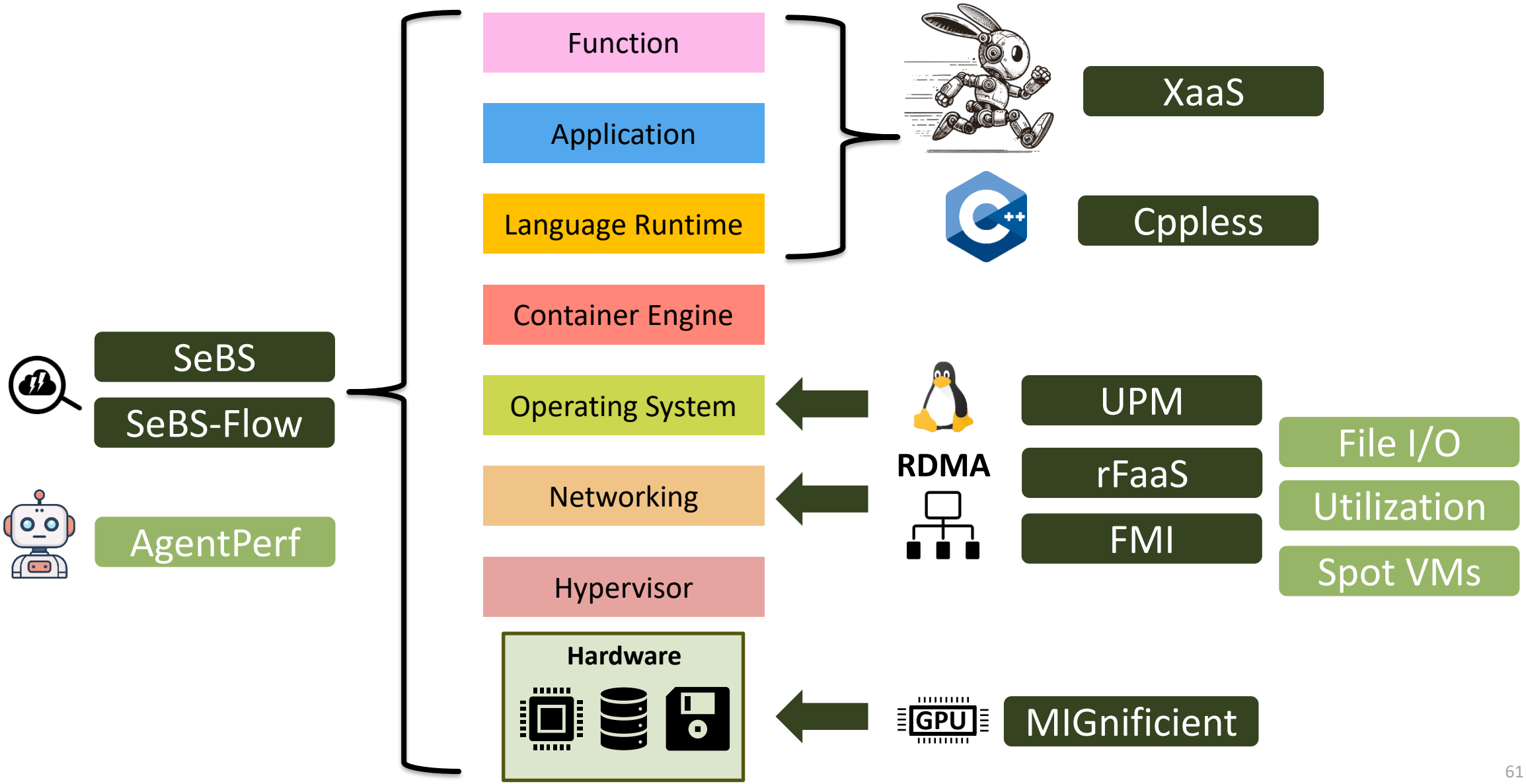
AXelerating Serverless Computing: Ongoing Work & Future



AXelerating Serverless Computing: Ongoing Work & Future



AXelerating Serverless Computing: Ongoing Work & Future



High-Performance Solutions for Serverless

 spcl/serverless-benchmarks

 spcl/rFaaS

 spcl/PraaS

 spcl/FaaSKeeper

 spcl/XaaS

 spcl/FMI

 spcl/cppless

Conclusions

With contributions from: Alexandru Calotoiu, Torsten Hoefler, Maciej Besta, Laurin Brandner, Rodrigo Bruno, Roman Böhringer, Tiancheng Chen, Marcin Chrapek, Jakub Czerski, Nico Graf, Tobias Grosser, Laurin Jahns, Kacper Janda, Sascha Kehrli, Prajin Khadka, Mateusz Knapik, Abhishek Kumar, Grzegorz Kwaśniewski, Horia Mercan, Michał Podstawski, Wei Qiu, Gyorgy Rethy, Oana Rosca, Mahla Sarifi, Larissa Schmid, Konstantin Taranov, Nicolas Wicki, Felix Wolf, Yuejian Yu, Pengyu Zhou, Paweł Żuk,

More of SPCL's research:

youtube.com/@spcl — 240+ Talks

twitter.com/spcl_eth — 1.7K+ Followers

github.com/spcl — 7.2K+ Stars

… or spcl.ethz.ch

Serverless HPC Projects

Conclusions

With contributions from: Alexandru Calotoiu, Torsten Hoefler, Maciej Besta, Laurin Brandner, Rodrigo Bruno, Roman Böhringer, Tiancheng Chen, Marcin Chrapek, Jakub Czerski, Nico Graf, Tobias Grosser, Laurin Jahns, Kacper Janda, Sascha Kehrli, Prajin Khadka, Mateusz Knapik, Abhishek Kumar, Grzegorz Kwaśniewski, Horia Mercan, Michał Podstawski, Wei Qiu, Gyorgy Rethy, Oana Rosca, Mahla Sarifi, Larissa Schmid, Konstantin Taranov, Nicolas Wicki, Felix Wolf, Yuejian Yu, Pengyu Zhou, Paweł Żuk,

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

More of SPCL's research:

youtube.com/@spcl — 240+ Talks

twitter.com/spcl_eth — 1.7K+ Followers

github.com/spcl — 7.2K+ Stars

… or spcl.ethz.ch

Serverless HPC Projects

Conclusions

With contributions from: Alexandru Calotoiu, Torsten Hoefler, Maciej Besta, Laurin Brandner, Rodrigo Bruno, Roman Böhringer, Tiancheng Chen, Marcin Chrapek, Jakub Czerski, Nico Graf, Tobias Grosser, Laurin Jahns, Kacper Janda, Sascha Kehrli, Prajin Khadka, Mateusz Knapik, Abhishek Kumar, Grzegorz Kwaśniewski, Horia Mercan, Michał Podstawski, Wei Qiu, Gyorgy Rethy, Oana Rosca, Mahla Sarifi, Larissa Schmid, Konstantin Taranov, Nicolas Wicki, Felix Wolf, Yuejian Yu, Pengyu Zhou, Paweł Żuk,

More of SPCL's research:

youtube.com/@spcl — 240+ Talks

twitter.com/spcl_eth — 1.7K+ Followers

github.com/spcl — 7.2K+ Stars

… or spcl.ethz.ch

Serverless HPC Projects

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

aws

Google Cloud

Google Summer of Code

Research into Serverless Performance

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Research into Serverless Performance

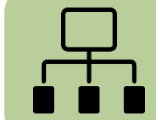
Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)

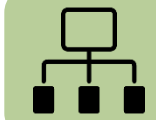


Networking and Communication
(Boxer, FMI, rFaaS)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



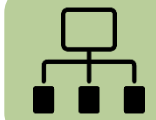
Networking and Communication
(Boxer, FMI, rFaaS)

HPC FaaS
(Globus Compute, rFaaS, Lithops)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



Networking and Communication
(Boxer, FMI, rFaaS)

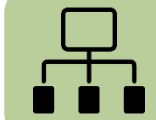
HPC FaaS
(Globus Compute, rFaaS, Lithops)

New HPC Workloads
(Cirrus, LambdaML, Mashup, DayDream)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



Networking and Communication
(Boxer, FMI, rFaaS)

HPC FaaS
(Globus Compute, rFaaS, Lithops)

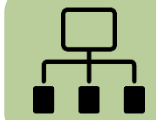
New HPC Workloads
(Cirrus, LambdaML, Mashup, DayDream)

Improved Cold Starts
(RainbowCake, IceBreaker, Medes)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



Networking and Communication
(Boxer, FMI, rFaaS)

HPC FaaS
(Globus Compute, rFaaS, Lithops)

New HPC Workloads
(Cirrus, LambdaML, Mashup, DayDream)

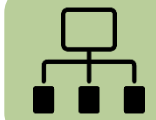
Improved Cold Starts
(RainbowCake, IceBreaker, Medes)

Improved Scheduling (Wukong, Palette, PraaS, ProPack, Pheromone)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



Networking and Communication
(Boxer, FMI, rFaaS)

HPC FaaS
(Globus Compute, rFaaS, Lithops)

New HPC Workloads
(Cirrus, LambdaML, Mashup, DayDream)

Improved Cold Starts
(RainbowCake, IceBreaker, Medes)

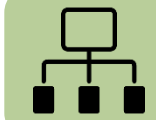
Improved Scheduling (Wukong, Palette, PraaS, ProPack, Pheromone)

HPC Utilization
(HPC-Whisk, Serverless Disaggregation)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



Networking and Communication
(Boxer, FMI, rFaaS)

HPC FaaS
(Globus Compute, rFaaS, Lithops)

New HPC Workloads
(Cirrus, LambdaML, Mashup, DayDream)

Improved Cold Starts
(RainbowCake, IceBreaker, Medes)

Improved Scheduling (Wukong, Palette, PraaS, ProPack, Pheromone)

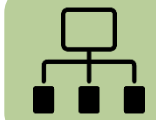
HPC Utilization
(HPC-Whisk, Serverless Disaggregation)

Benchmark Suites
(SeBS, Serverlessbench, FaaSDom)

Research into Serverless Performance

Fast and Lightweight Sandboxes
(gVisor, Catalyzer, SEUSS, Photon)

Stateful Functions
(Cloudburst, Faasm, Crucial, PraaS)



Networking and Communication
(Boxer, FMI, rFaaS)

HPC FaaS
(Globus Compute, rFaaS, Lithops)

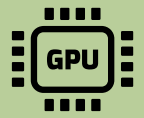
New HPC Workloads
(Cirrus, LambdaML, Mashup, DayDream)

Improved Cold Starts
(RainbowCake, IceBreaker, Medes)

Improved Scheduling (Wukong, Palette, PraaS, ProPack, Pheromone)

HPC Utilization
(HPC-Whisk, Serverless Disaggregation)

Benchmark Suites
(SeBS, Serverlessbench, FaaSDom)



Accelerated Functions
(DGSF, KaaS, MIGnificient)

Serverless: One Step Toward HPC – Cloud Convergence

Infrastructure

Serverless: One Step Toward HPC – Cloud Convergence

Infrastructure

Cloud

HPC

Serverless: One Step Toward HPC – Cloud Convergence

Infrastructure

Cloud


Virtual Machines


Object Storage


Virtual Networks

HPC

HPCaaS


HPC in the cloud

Serverless: One Step Toward HPC – Cloud Convergence

Infrastructure

Cloud


Virtual Machines


Object Storage


Virtual Networks

HPC

HPCaaS


HPC in the cloud

Cloud in HPC

Serverless: One Step Toward HPC – Cloud Convergence

Infrastructure

Deployment

Cloud


Virtual Machines


Object Storage


Virtual Networks


Containers


(Managed) Kubernetes


Services

HPC

HPCaaS


HPC in the cloud

Cloud in HPC

Serverless: One Step Toward HPC – Cloud Convergence

Infrastructure

Deployment

Cloud

- Virtual Machines
- Object Storage
- Virtual Networks

- Containers
- (Managed) Kubernetes
- Services

HPC

HPCaaS

- HPC in the cloud
- Cloud in HPC

- HPC
- Containers
- Flux
- XaaS

IaaS, SaaS, PaaS

Serverless: One Step Toward HPC – Cloud Convergence

	Infrastructure	Deployment	Compute
Cloud	Virtual Machines	Containers	Serverless Functions
	Object Storage	(Managed) Kubernetes	Serverless Containers
	Virtual Networks	Services	
HPC	HPCaaS	HPC	
	HPC in the cloud	Containers	
		Flux	
	Cloud in HPC	XaaS	

IaaS, SaaS, PaaS

GPU

Serverless: One Step Toward HPC – Cloud Convergence

	Infrastructure	Deployment	Compute
Cloud	Virtual Machines	Containers	Serverless Functions
	Object Storage	(Managed) Kubernetes	Serverless Containers
	Virtual Networks	Services	
HPC	HPCaaS	HPC Containers	Globus Compute (funcX)
	HPC in the cloud	Flux	RDMA rFaaS
	Cloud in HPC	XaaS	Lithops

Serverless: One Step Toward HPC – Cloud Convergence

	Infrastructure	Deployment	Compute	Applications
Cloud	Virtual Machines	Containers	Serverless Functions	Dask, Spark Ray
	Object Storage	(Managed) Kubernetes	Serverless Containers	Serverless Workflows
	Virtual Networks	Services		
HPC	HPCaaS	HPC Containers	Globus Compute (funcX)	
	HPC in the cloud	Flux	RDMA rFaaS	
	Cloud in HPC	XaaS	Lithops	

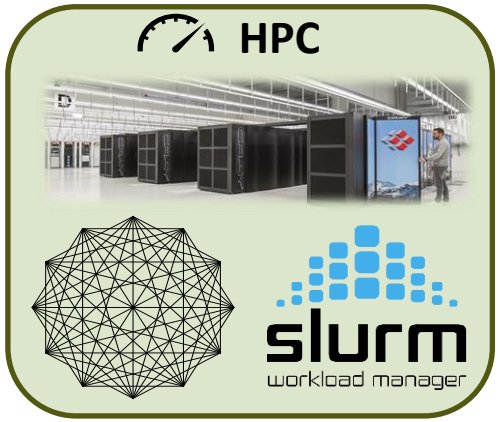
IaaS, SaaS, PaaS

Serverless: One Step Toward HPC – Cloud Convergence

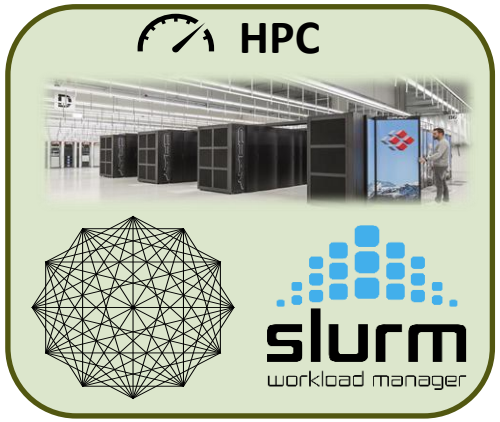
	Infrastructure	Deployment	Compute	Applications
Cloud	Virtual Machines	Containers	Serverless Functions	Dask, Spark Ray
	Object Storage	(Managed) Kubernetes	Serverless Containers	Serverless Workflows
	Virtual Networks	Services		
HPC	HPCaaS	HPC Containers	Globus Compute (funcX)	
	HPC in the cloud	Flux	RDMA rFaaS	
	Cloud in HPC	XaaS	Lithops	

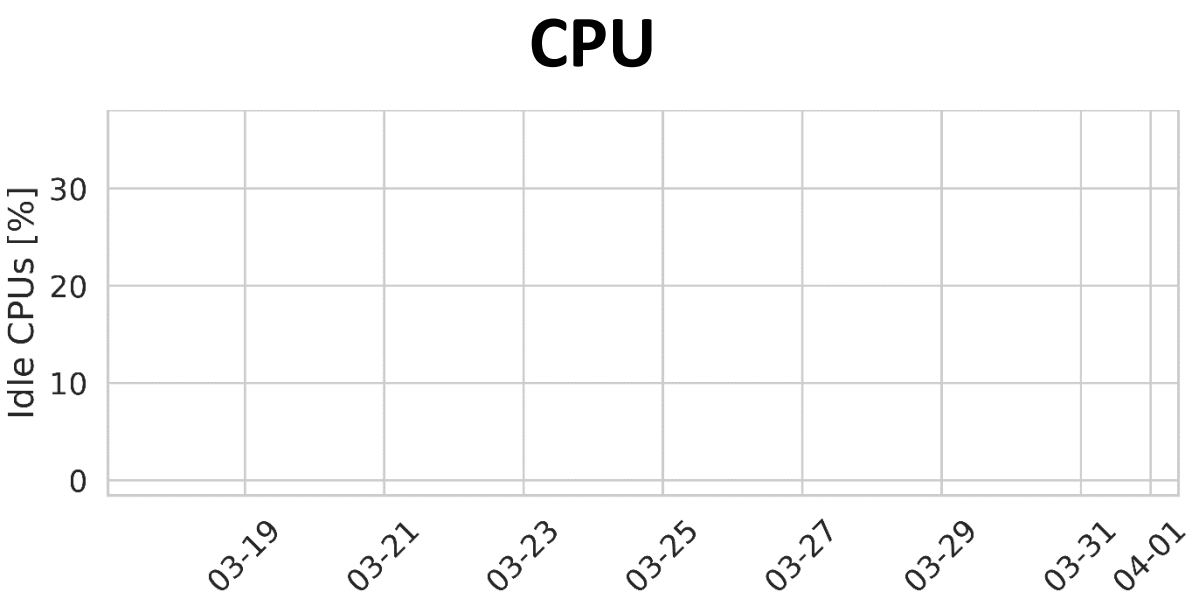
IaaS, SaaS, PaaS

Tracking Wasted Resources in HPC

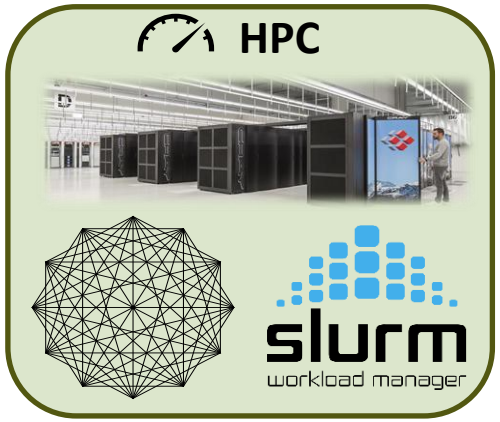


Tracking Wasted Resources in HPC

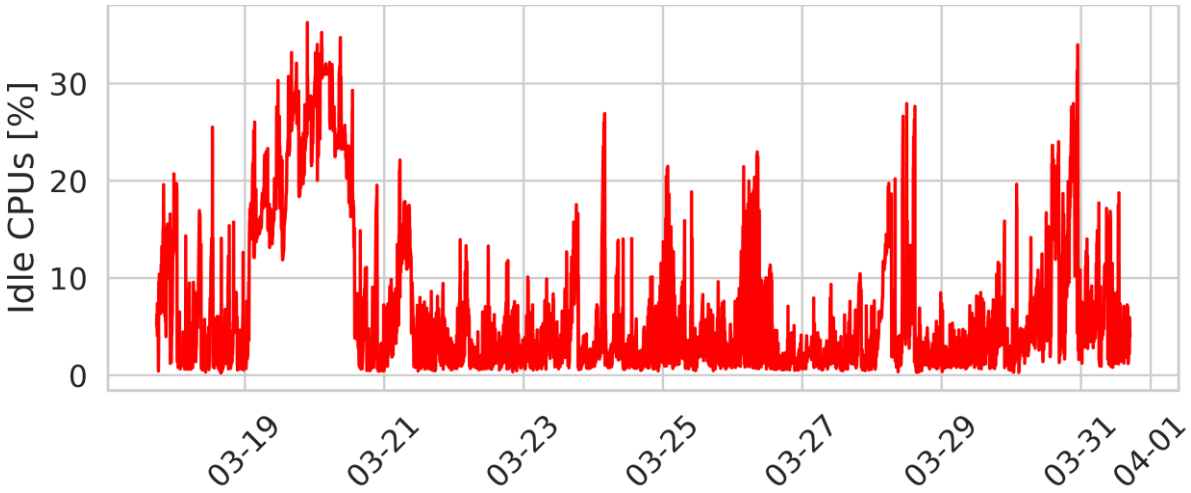




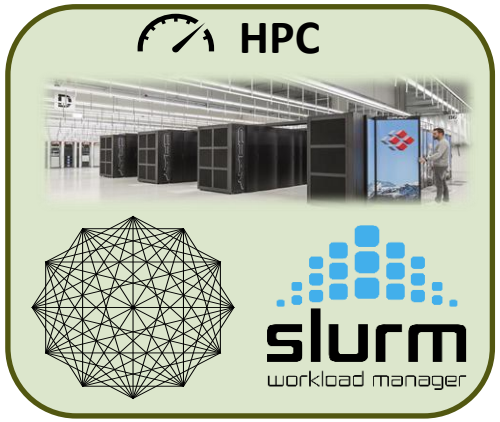
Tracking Wasted Resources in HPC



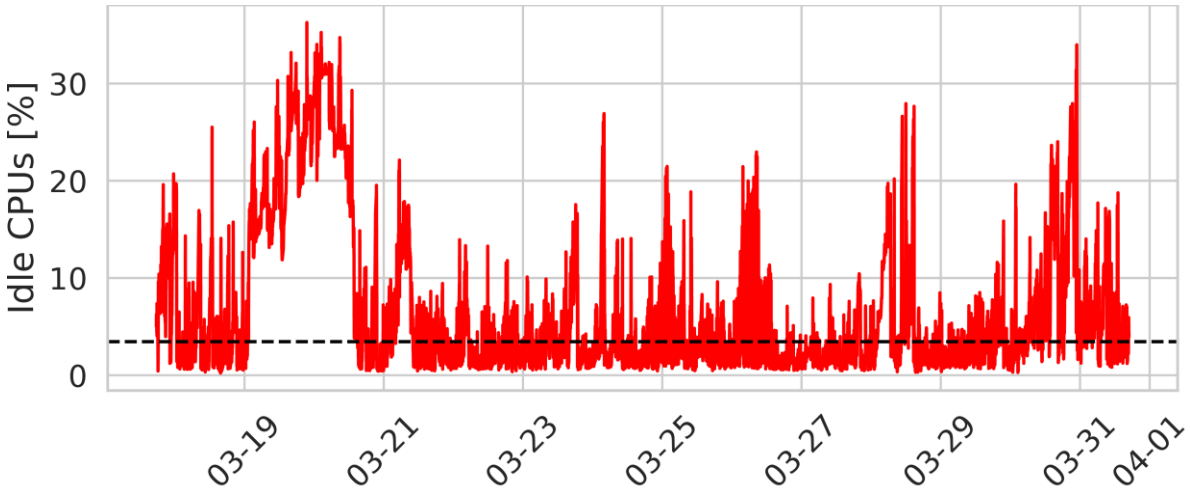
CPU



Tracking Wasted Resources in HPC

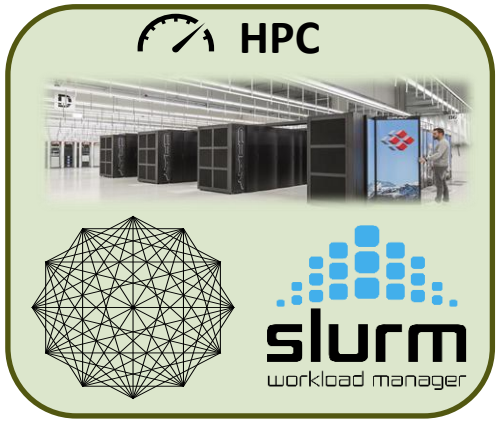


CPU

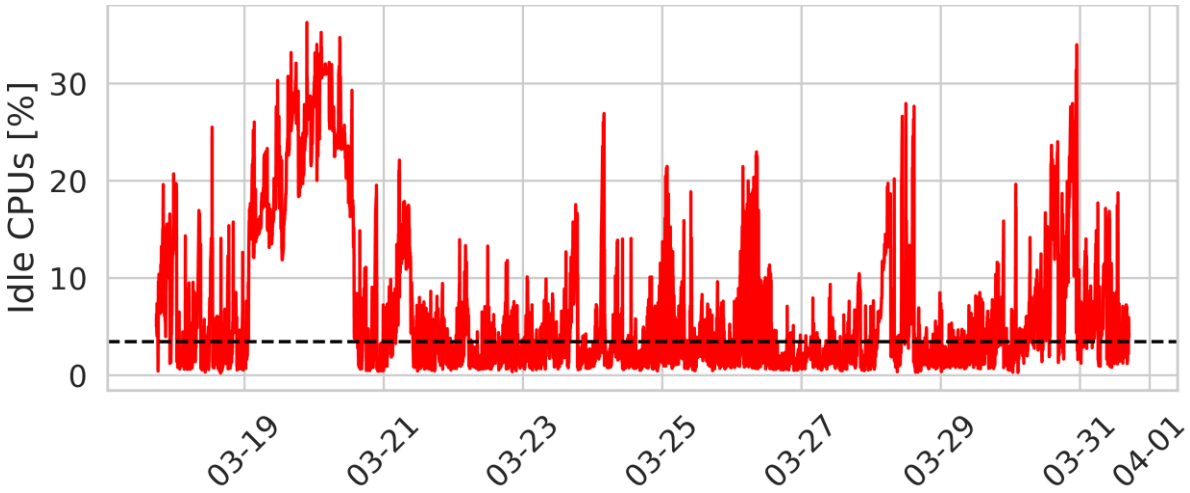


Mean idle CPUs: 6.6%

Tracking Wasted Resources in HPC

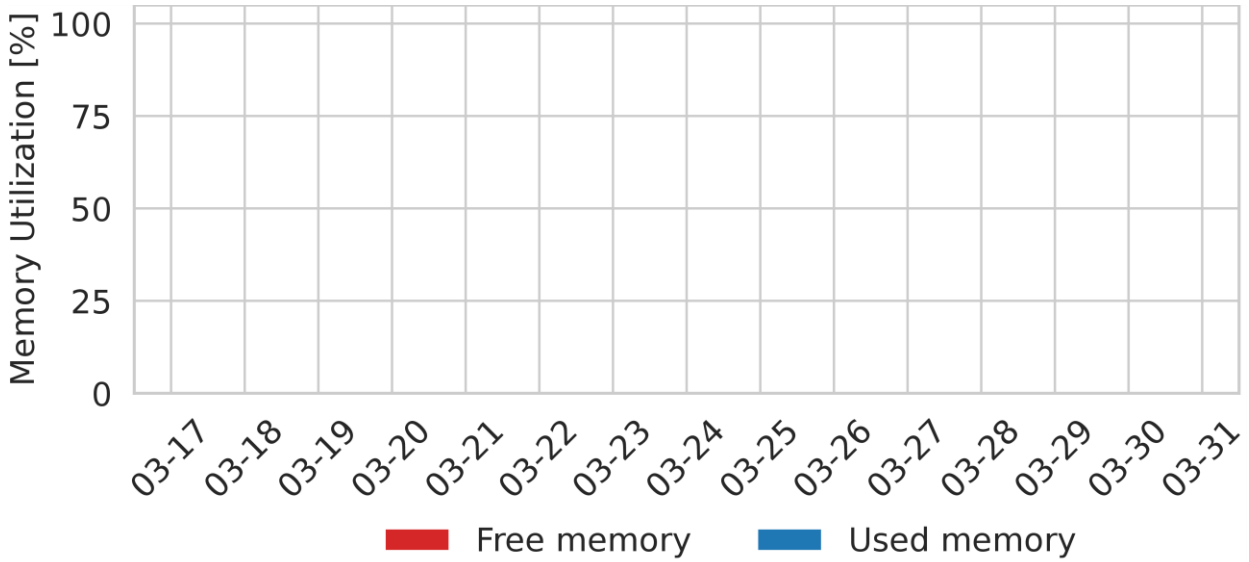


CPU

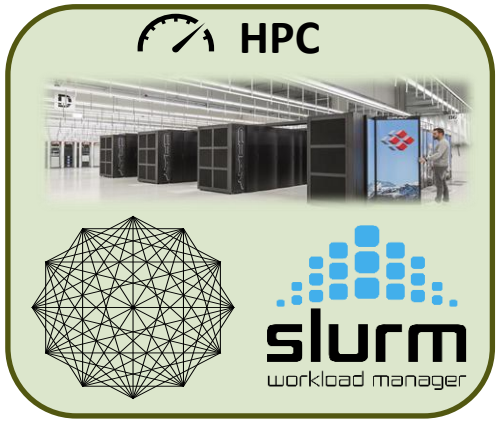


Mean idle CPUs: 6.6%

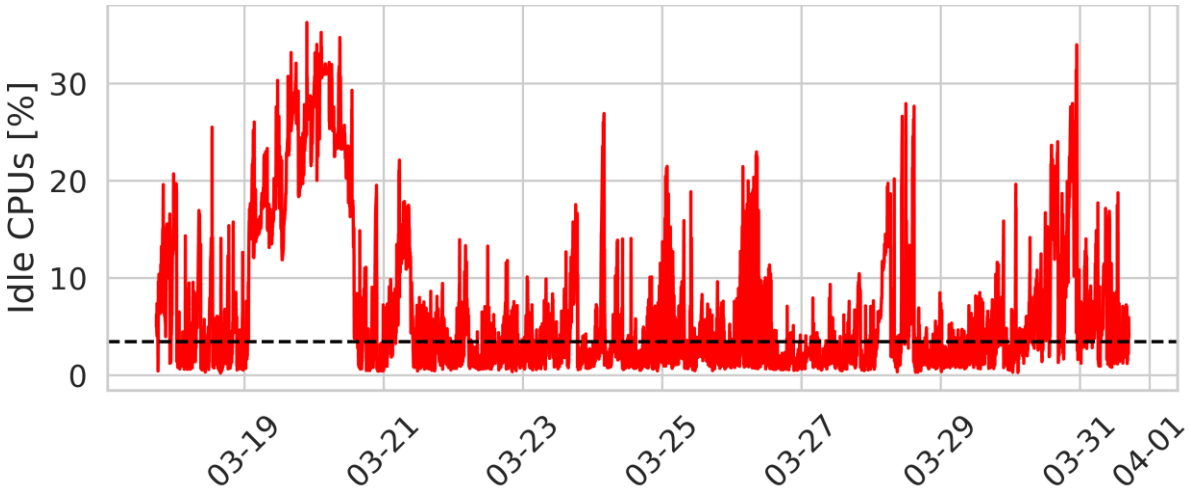
Memory



Tracking Wasted Resources in HPC

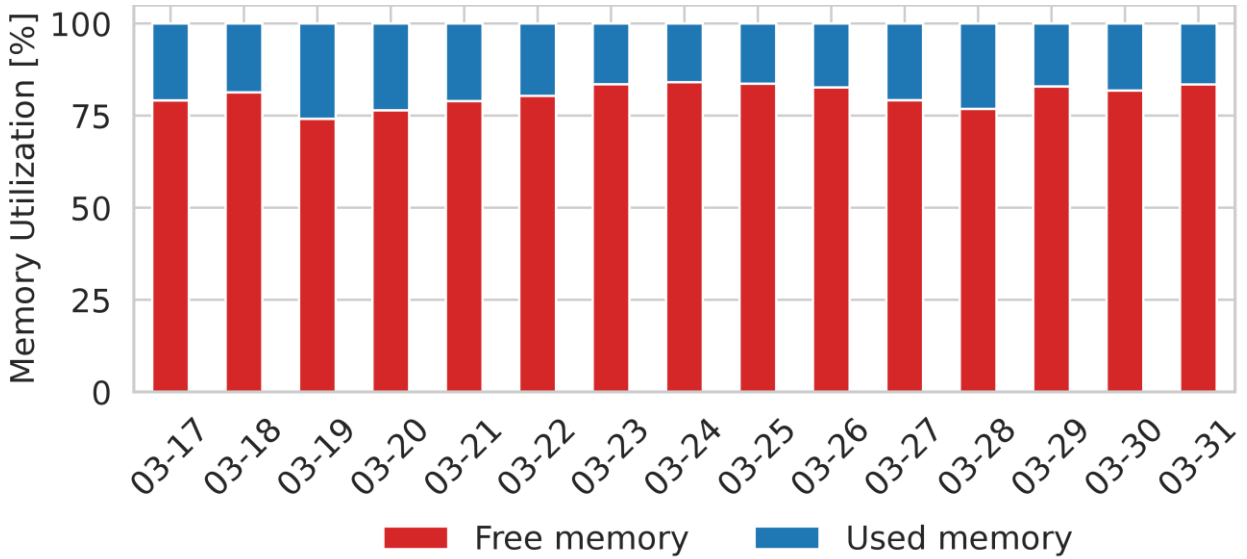


CPU



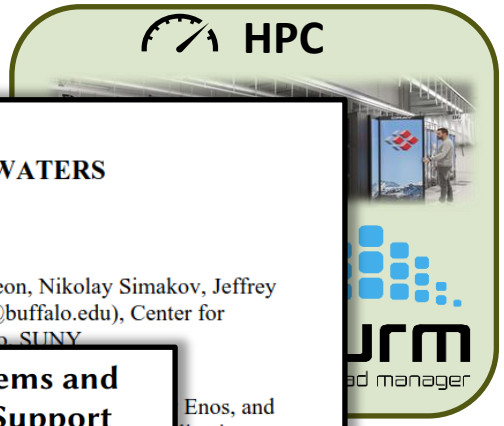
Mean idle CPUs: 6.6%

Memory



Mean free memory: 80.5%

Tracking Wasted Resources in HPC



Job Characteristics on Large-Scale Systems: Long-Term Analysis, Quantification, and Implications*

Tirthak Patel, Northeastern University
Zhengchun Liu, Raj Kettimuthu, Argonne National Laboratory
Paul Rich, William Allcock, Argonne National Laboratory
Devesh Tiwari, Northeastern University

SC, 2020

FINAL REPORT
WORKLOAD ANALYSIS OF BLUE WATERS
(ACI 1650758)

Matthew D. Jones, Joseph P. White, Martins Innus, Robert L. DeLeon, Nikolay Simakov, Jeffrey T. Palmer, Steven M. Gallo, and Thomas R. Furlani (furlani@buffalo.edu), Center for Computational Research, University at Buffalo, SUNY

Enos, and
lications

Xiv, 2017

Quantifying Memory Underutilization in HPC Systems and Using it to Improve Performance via Architecture Support

Gagandeep Panwar*, Virginia Tech, Blacksburg, USA, gpanwar@vt.edu
Da Zhang*, Virginia Tech, Blacksburg, USA, daz3@vt.edu
Yihan Pang*, Virginia Tech, Blacksburg, USA, pyihan1@vt.edu

A Case For Intra-rack Resource Disaggregation in HPC

GEORGE MICHELOGIANNAKIS, Lawrence Berkeley National Laboratory, USA
BENJAMIN KLENK, NVIDIA, USA
BRANDON COOK, Lawrence Berkeley National Laboratory, USA
MIN YE
LARRY D
KEREN
JOHN SH

Comprehensive Workload Analysis and Modeling of a Petascale Supercomputer

Haihang You[1] and Hao Zhang[2]

[1] National Institute for Computational Sciences, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2] Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996, USA

JSSPP, 2012

A Holistic View of Memory Utilization on HPC Systems: Current and Future Trends

Ivy B. Peng*, peng8@llnl.gov, Lawrence Livermore National Laboratory, USA
Ian Karlin, karlin1@llnl.gov, Lawrence Livermore National Laboratory, USA
Maya B. Gokhale, gokhale2@llnl.gov, Lawrence Liv
Lab
Kathleen Shoga, Shoga1@llnl.gov, Lawrence Livermore National Laboratory, USA
Matthew Legendre, legendre1@llnl.gov, Lawrence Livermore National Laboratory, USA
Todd
gamblin
Lawrence Liv
Lab

ME

Learning from Five-year Resource-Utilization Data of Titan System

Feiyi Wang*, Sarp Oral†, Satyabrata Sen ‡ and Neena Imam§
Oak Ridge National Laboratory

CLUSTER, 2019

Why C++ for Serverless?

Why C++ for Serverless?

Scientific Computing

rFaaS: Enabling High Performance Serverless with RDMA and Leases

Marcin Copik*, Konstantin Taranov†, Alexandru Calotoiu*, Torsten Hoefler*
*Department of Computer Science, ETH Zürich, Zürich, Switzerland
†Microsoft

Why C++ for Serverless?

Scientific Computing

Microservices

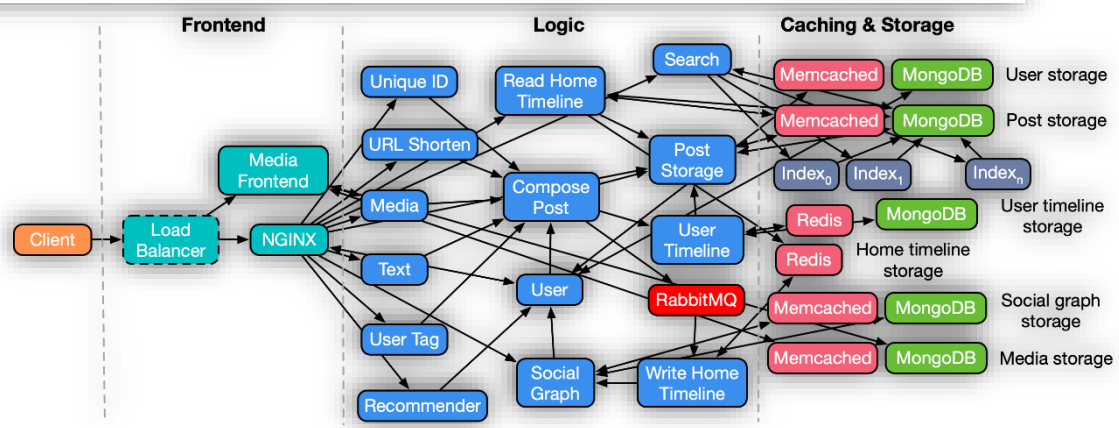
Nightcore: Efficient and Scalable Serverless Computing for Latency-Sensitive, Interactive Microservices

Zhipeng Jia
The University of Texas at Austin
Austin, TX, USA
zjia@cs.utexas.edu

Emmett Witchel
The University of Texas at Austin
Austin, TX, USA
witchel@cs.utexas.edu

rFaaS: Enabling High Performance Serverless with RDMA and Leases

Marcin Copik*, Konstantin Taranov†, Alexandru Calotoiu*, Torsten Hoefler*
*Department of Computer Science, ETH Zürich, Zürich, Switzerland
†Microsoft



Why C++ for Serverless?

Scientific Computing

Microservices

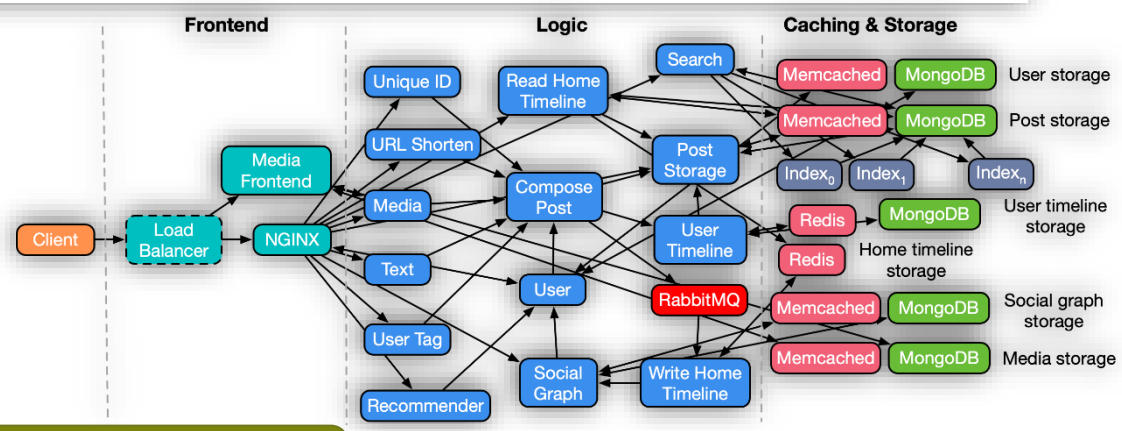
Nightcore: Efficient and Scalable Serverless Computing for Latency-Sensitive, Interactive Microservices

Zhipeng Jia
The University of Texas at Austin
Austin, TX, USA
zjia@cs.utexas.edu

Emmett Witchel
The University of Texas at Austin
Austin, TX, USA
witchel@cs.utexas.edu

rFaaS: Enabling High Performance Serverless with RDMA and Leases

Marcin Copik*, Konstantin Taranov†, Alexandru Calotoiu*, Torsten Hoefler*
*Department of Computer Science, ETH Zürich, Zürich, Switzerland
†Microsoft

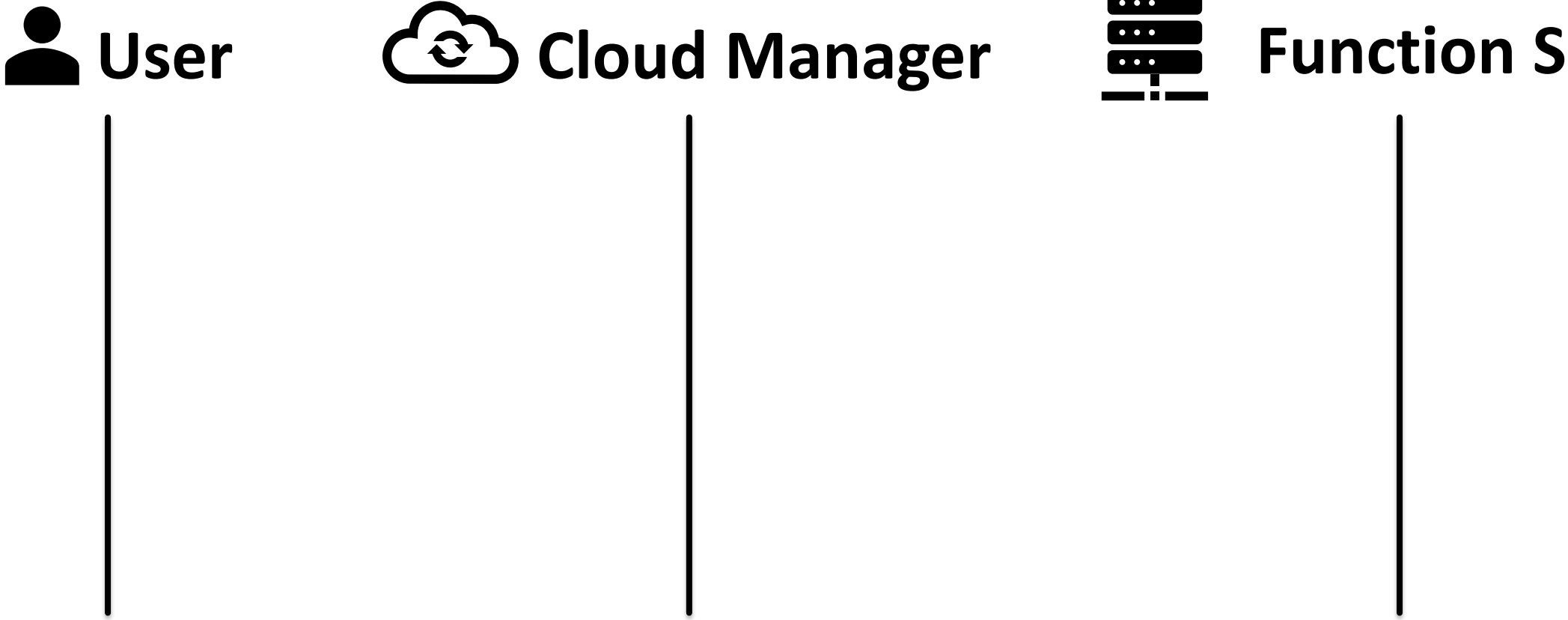


High-Performance Applications

R2E2: Low-Latency Path Tracing of Terabyte-Scale Scenes using Thousands of Cloud CPUs

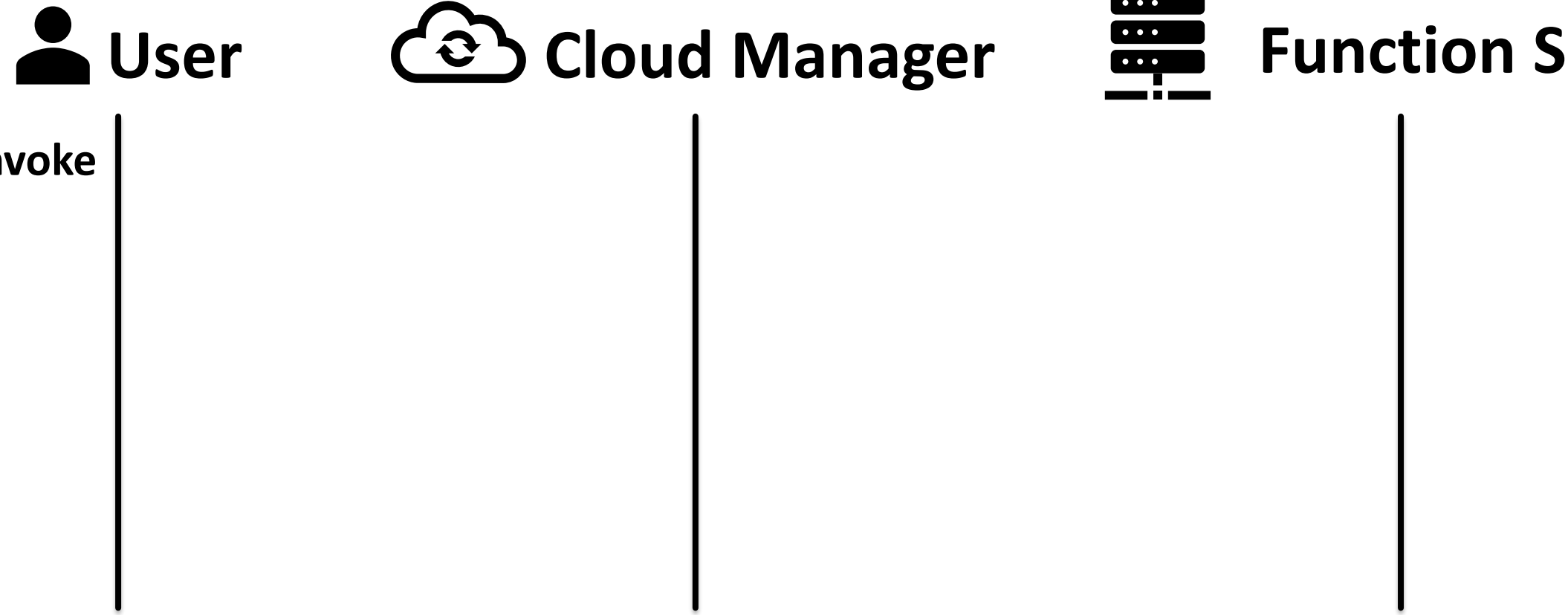
SADJAD FOULADI, Microsoft Research, USA and Stanford University, USA
BRENNAN SHACKLETT, FAIT POMS, ARJUN ARORA, ALEX OZDEMIR, DEEPTI RAGHAVAN, PAT HANRAHAN, KAYVON FATAHALIAN, and KEITH WINSTEIN, Stanford University, USA

FaaS Analysis: Invocation Overhead



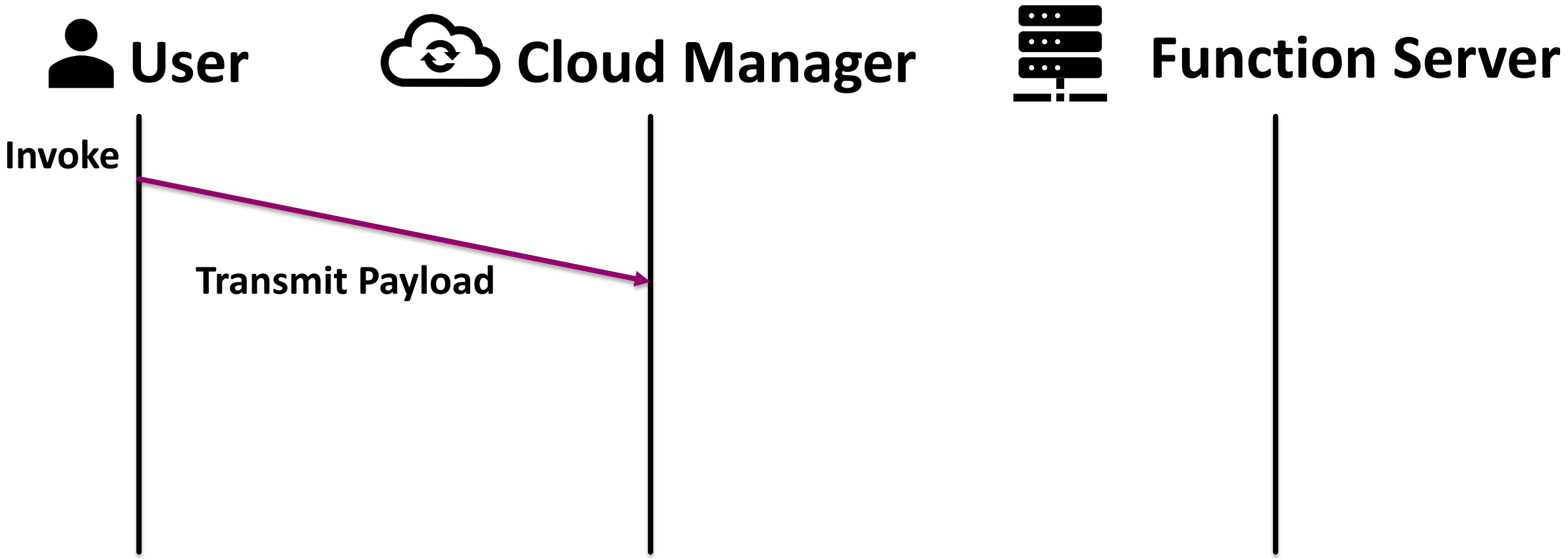
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

FaaS Analysis: Invocation Overhead



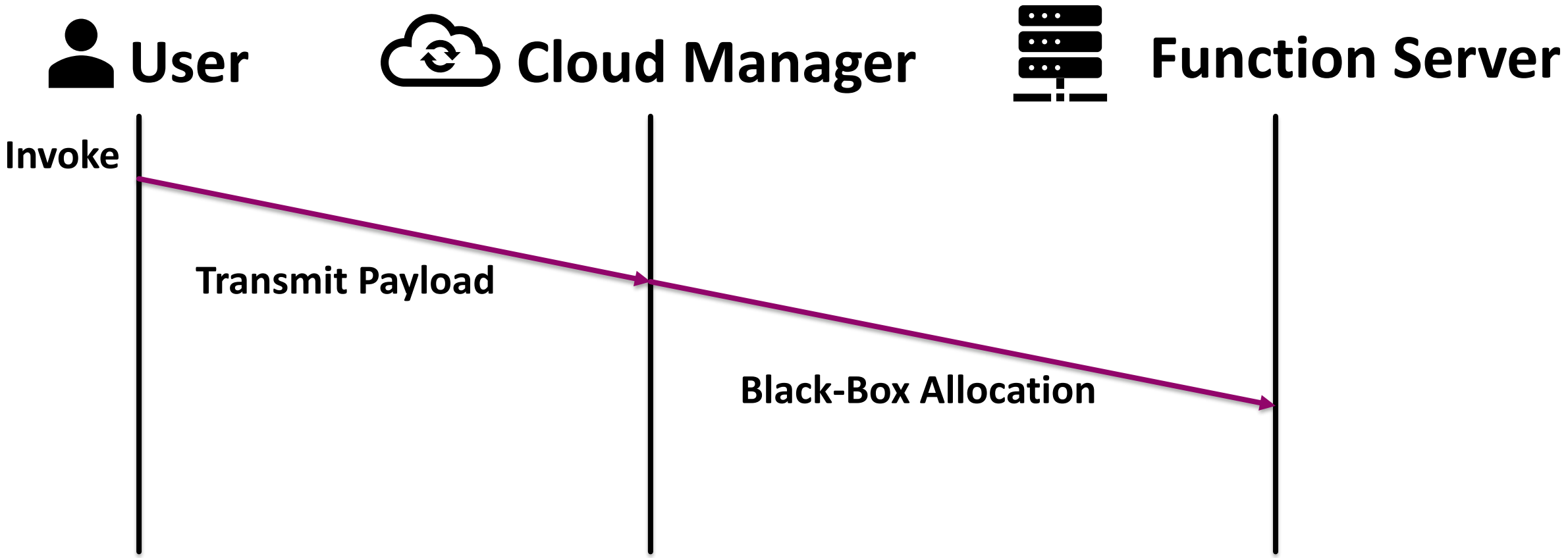
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

FaaS Analysis: Invocation Overhead



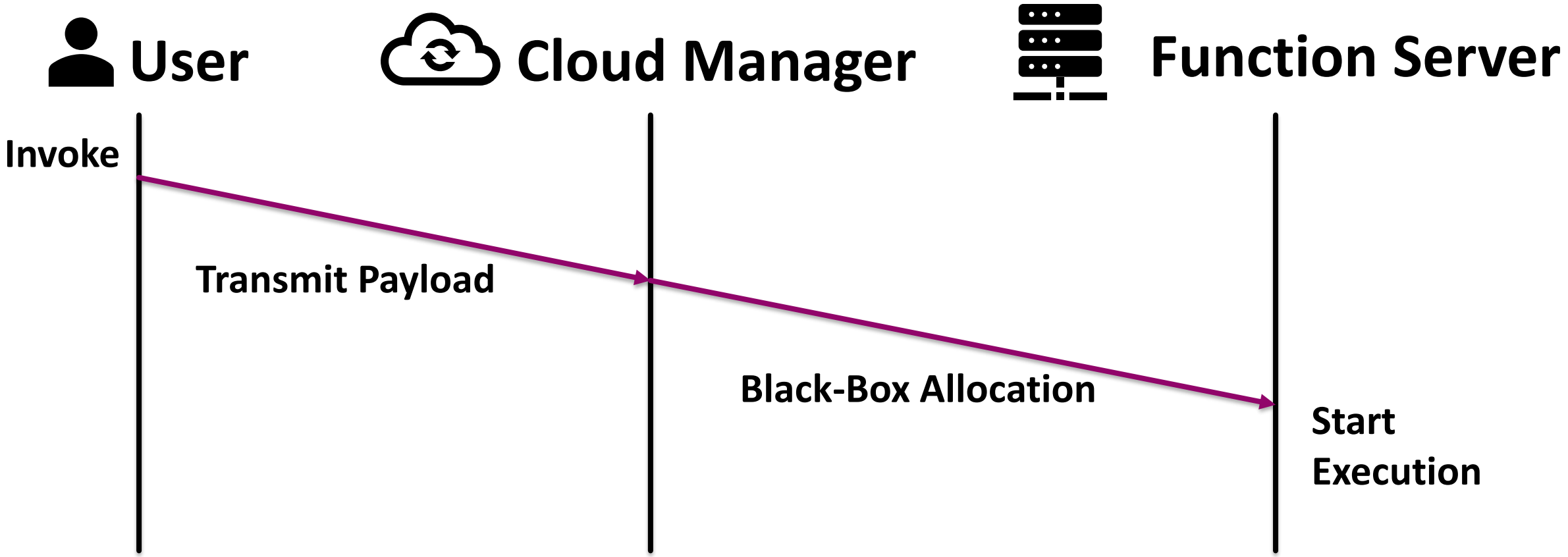
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

FaaS Analysis: Invocation Overhead



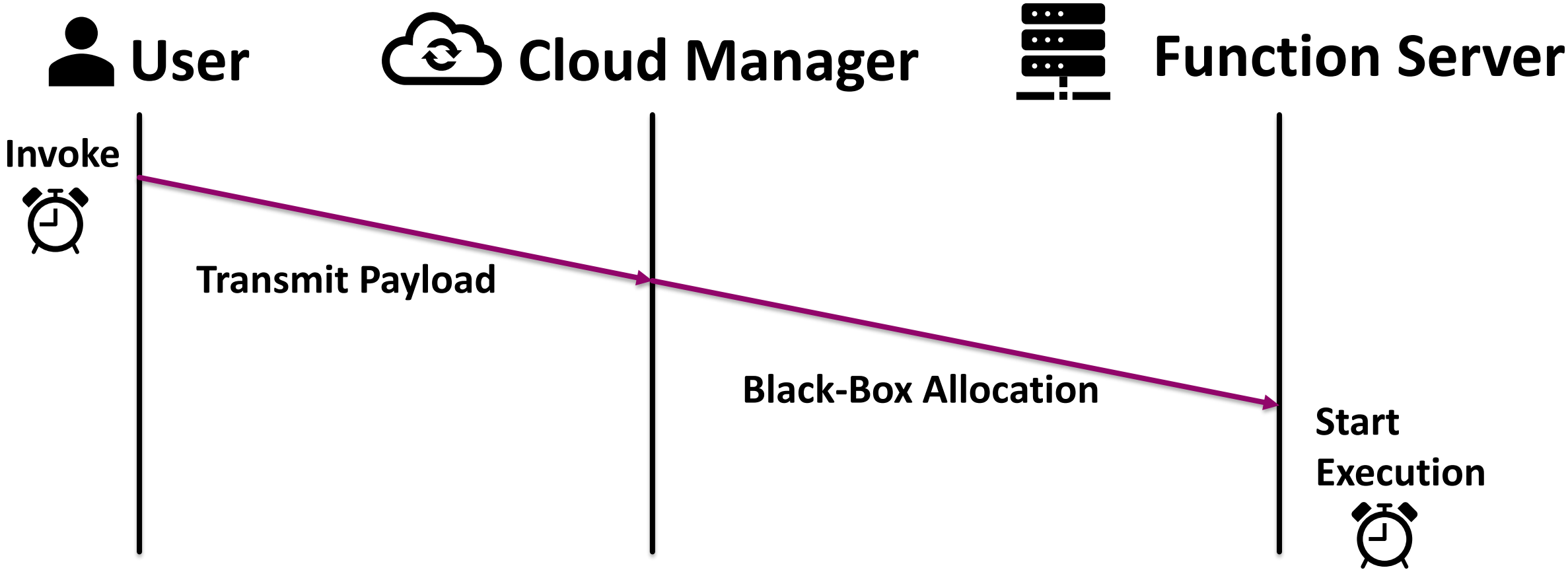
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

FaaS Analysis: Invocation Overhead



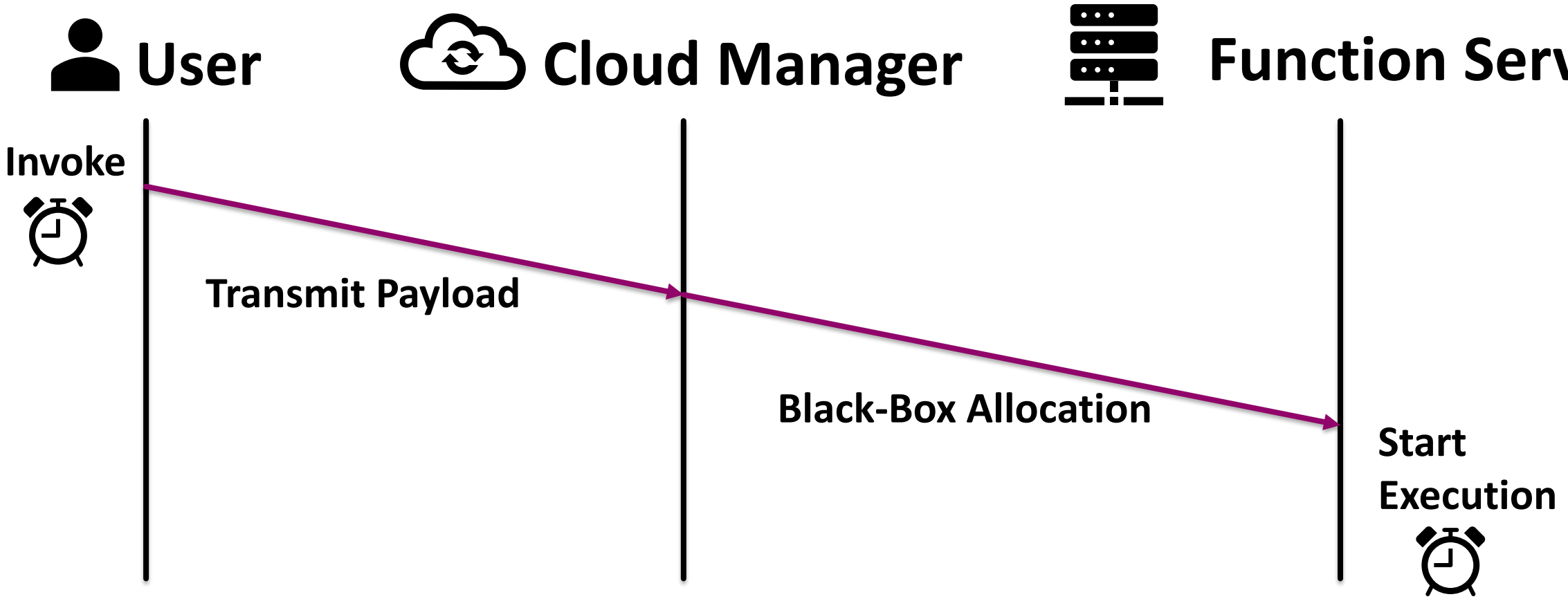
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

FaaS Analysis: Invocation Overhead



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

FaaS Analysis: Invocation Overhead



Solution: apply clock-drift estimation protocols!

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

FaaS Analysis: Invocation Overhead

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

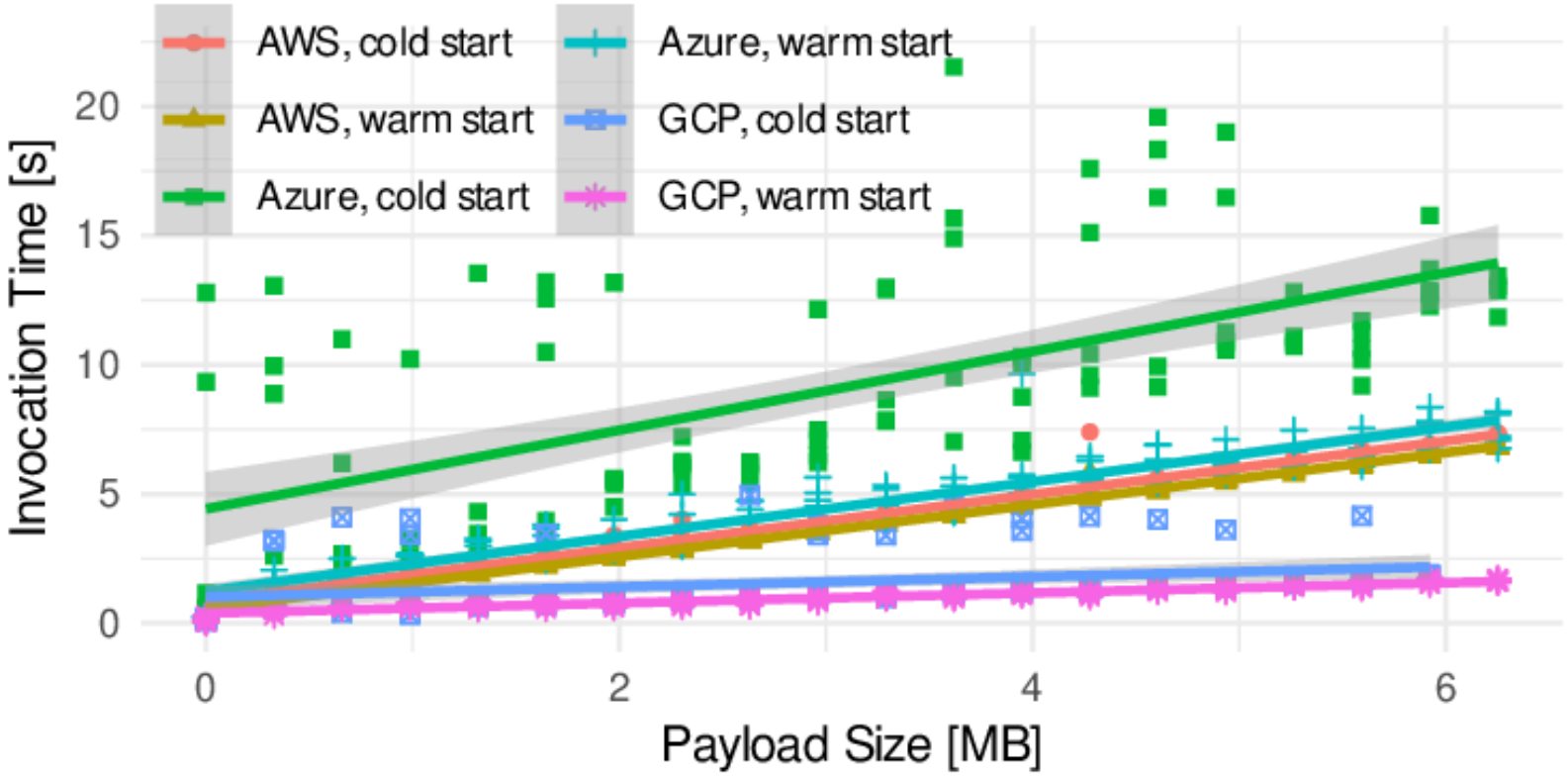
FaaS Analysis: Invocation Overhead

Configuration:

- Compare timestamps on client and function side.
- Clock drift estimation protocol.
- Payload: 1 kB – 5.9 MB

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

FaaS Analysis: Invocation Overhead



Configuration:

- Compare timestamps on client and function side.
- Clock drift estimation protocol.
- Payload: 1 kB – 5.9 MB

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

SeBS-Flow: Let the Work Flow in the Cloud

“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

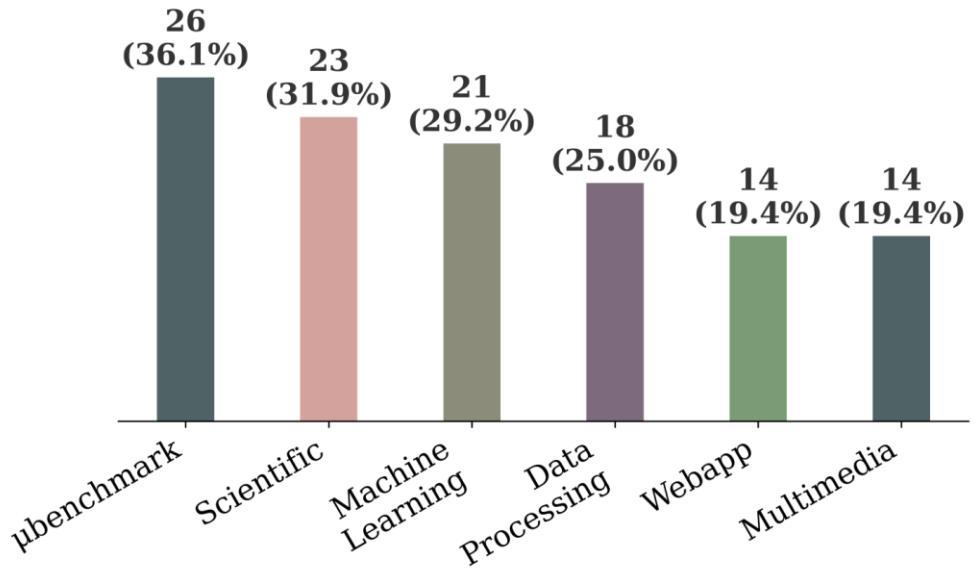
SeBS-Flow: Let the Work Flow in the Cloud

Analysis of 72 research papers: how are serverless workflows evaluated?

“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

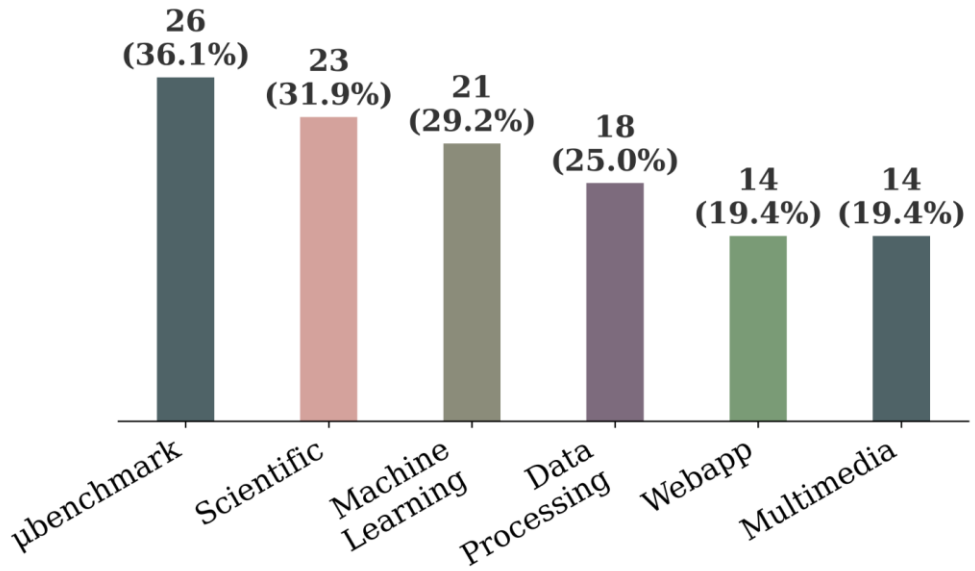
Analysis of 72 research papers: how are serverless workflows evaluated?

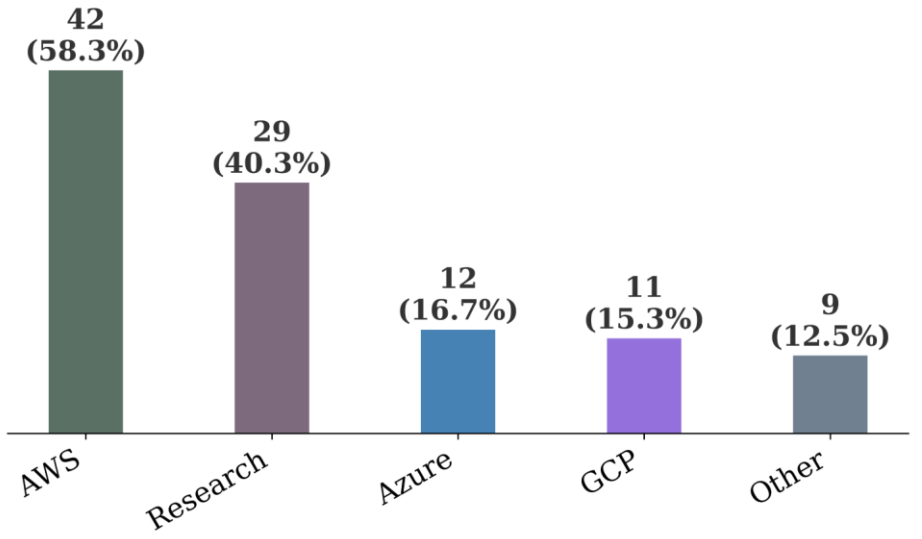


"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

Analysis of 72 research papers: how are serverless workflows evaluated?

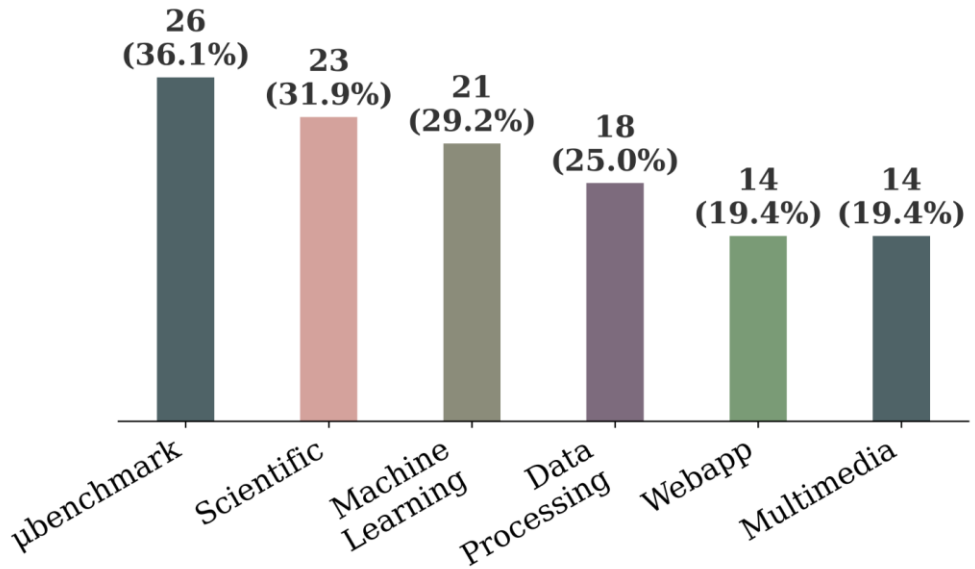


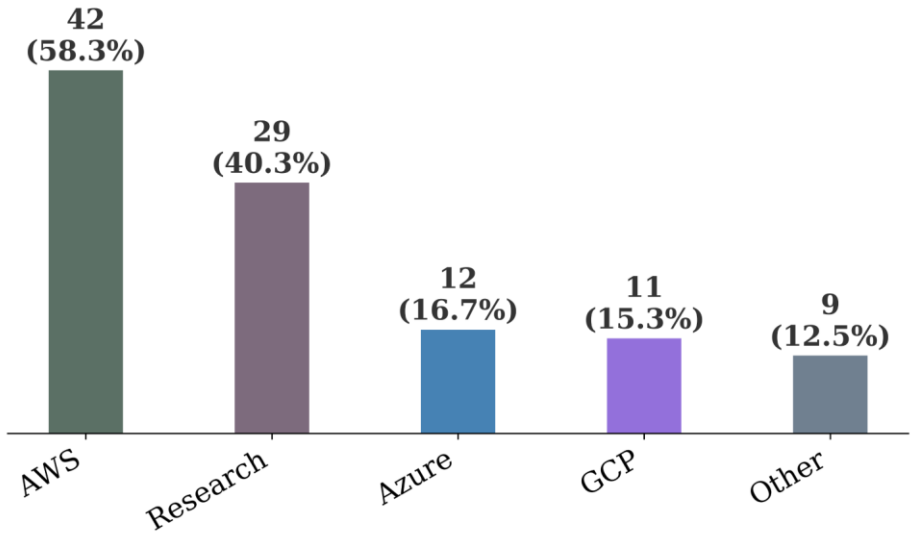


"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

Analysis of 72 research papers: how are serverless workflows evaluated?





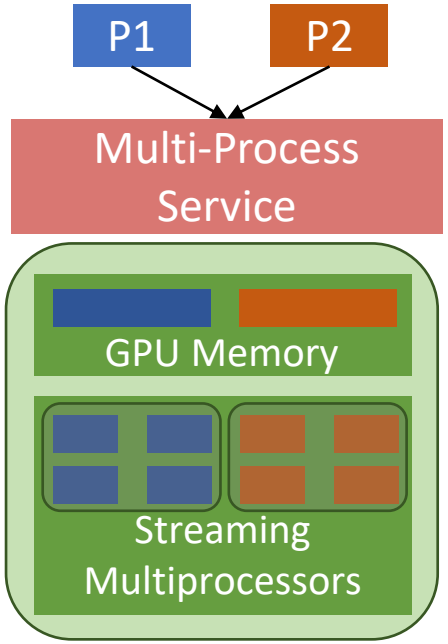
AWS Step Functions

Azure Durable Functions

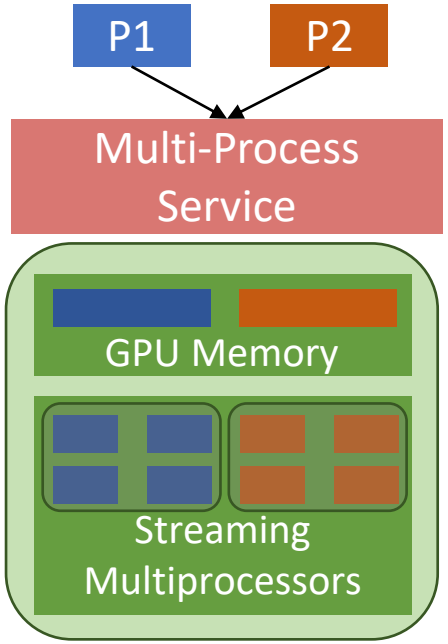
Google Cloud Workflows

"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

Serverless GPUs: Multi-Process Service (MPS) to the Rescue?

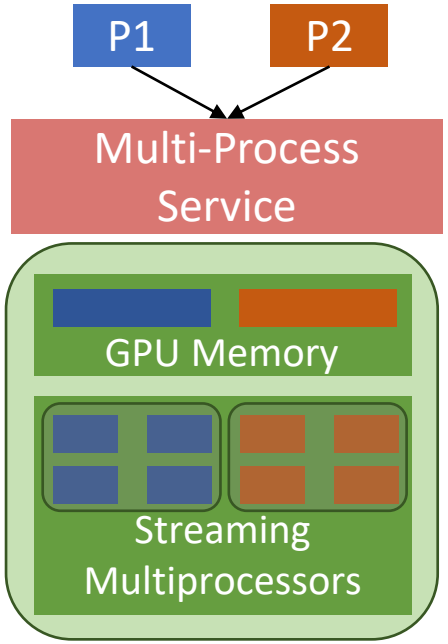


Serverless GPUs: Multi-Process Service (MPS) to the Rescue?



"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

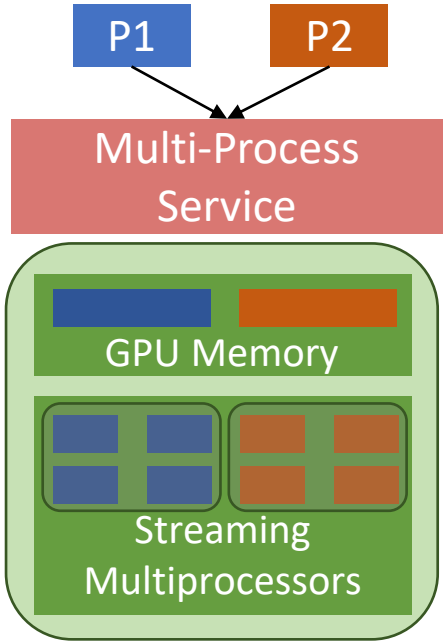
Serverless GPUs: Multi-Process Service (MPS) to the Rescue?



"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

⚠ Serverless is multi-tenant and executes arbitrary user code.

Serverless GPUs: Multi-Process Service (MPS) to the Rescue?

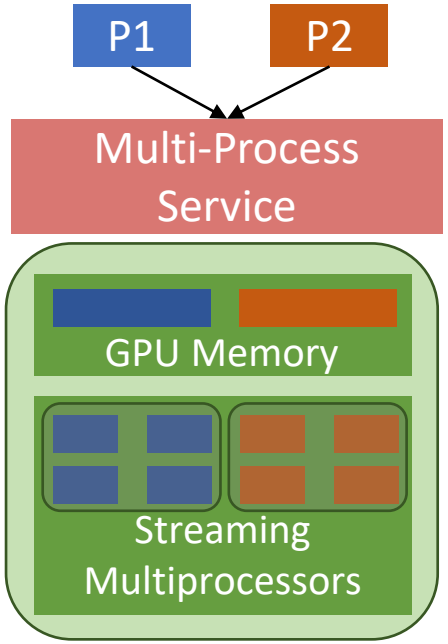


"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

Serverless is multi-tenant and executes arbitrary user code.

Limited security!
Function can conduct side-channel attack.

Serverless GPUs: Multi-Process Service (MPS) to the Rescue?



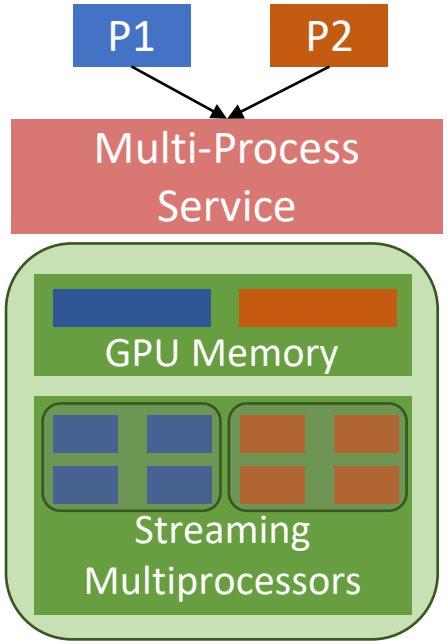
"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

Serverless is multi-tenant and executes arbitrary user code.

Limited security!
Function can conduct side-channel attack.

No performance isolation!
Function can hog the memory bandwidth.

Serverless GPUs: Multi-Process Service (MPS) to the Rescue?



"MPS is only recommended for running **cooperative processes** effectively acting as a **single application**, such as multiple ranks of the same MPI job, such that the severity of the following memory protection and error containment limitations is acceptable."

Serverless is multi-tenant and executes arbitrary user code.

Limited security!
Function can conduct side-channel attack.

No performance isolation!
Function can hog the memory bandwidth.

No error containment!
Function can maliciously kill GPU contexts.

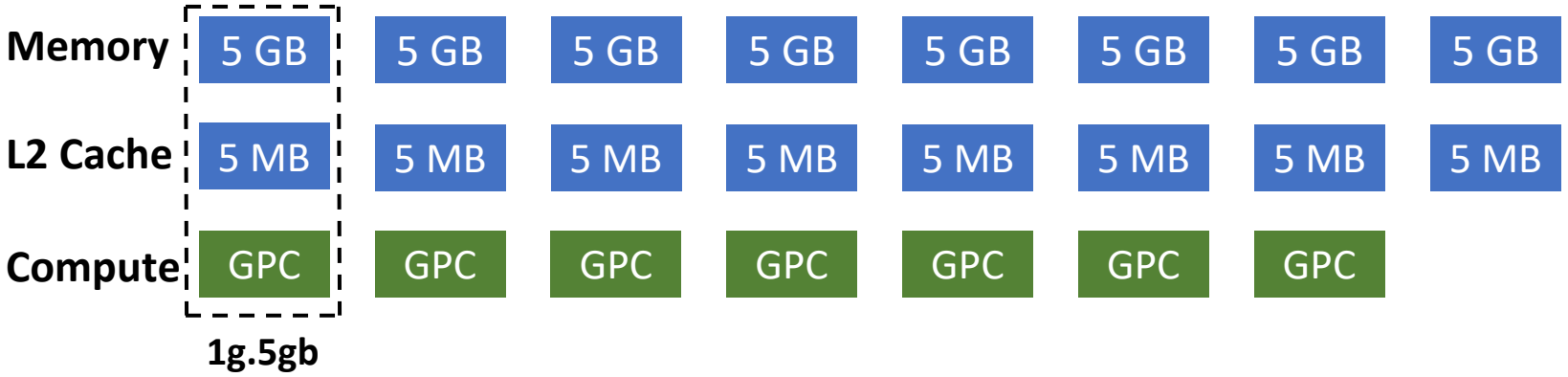
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)

Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)

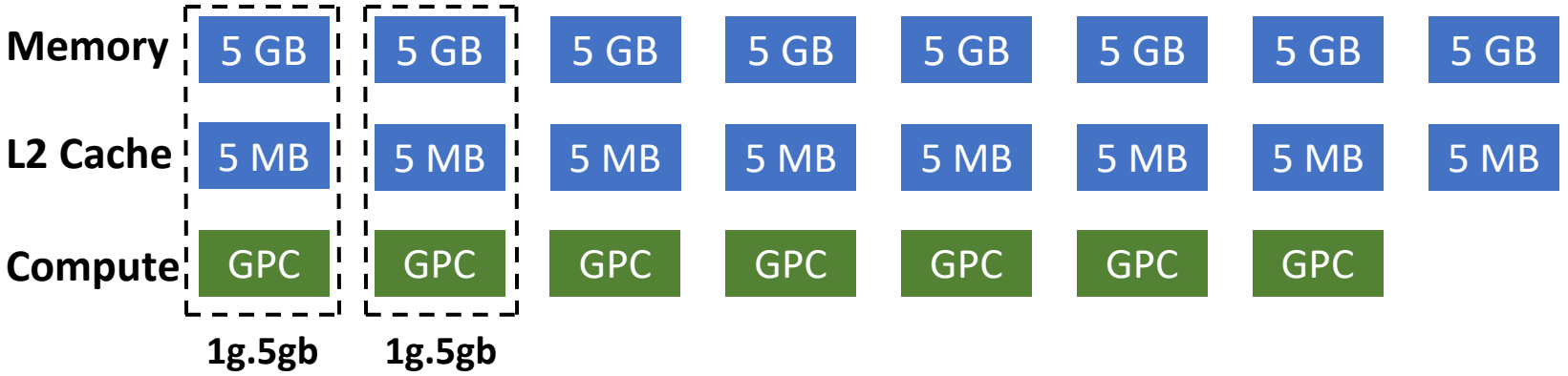
Memory	5 GB	5 GB	5 GB	5 GB	5 GB	5 GB	5 GB	5 GB
L2 Cache	5 MB	5 MB	5 MB	5 MB	5 MB	5 MB	5 MB	5 MB

Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)

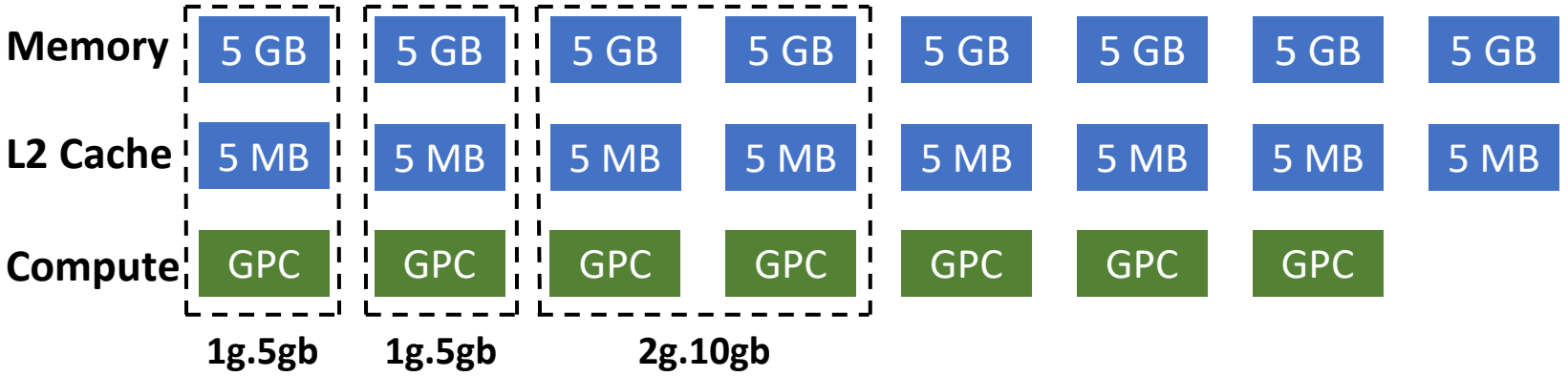
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)

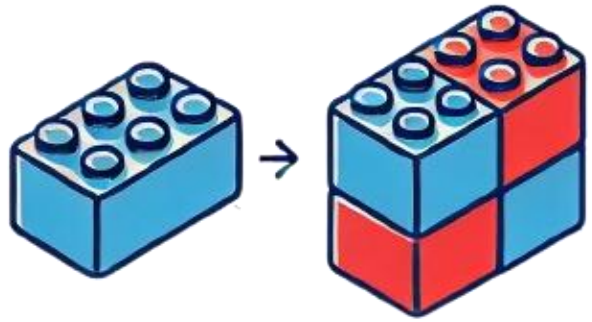


Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)

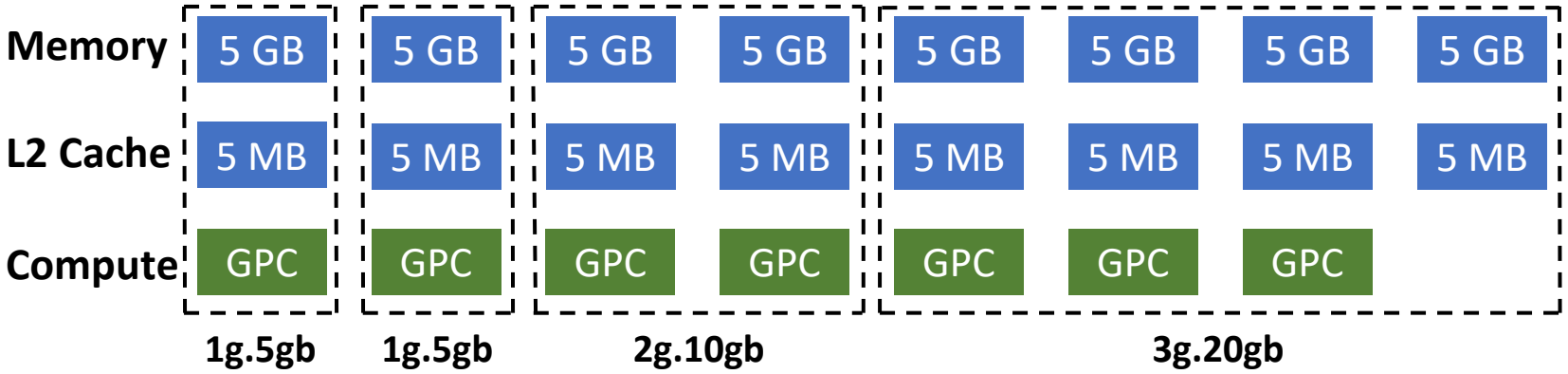


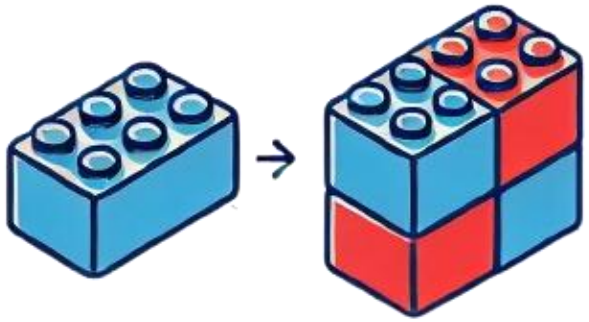
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)



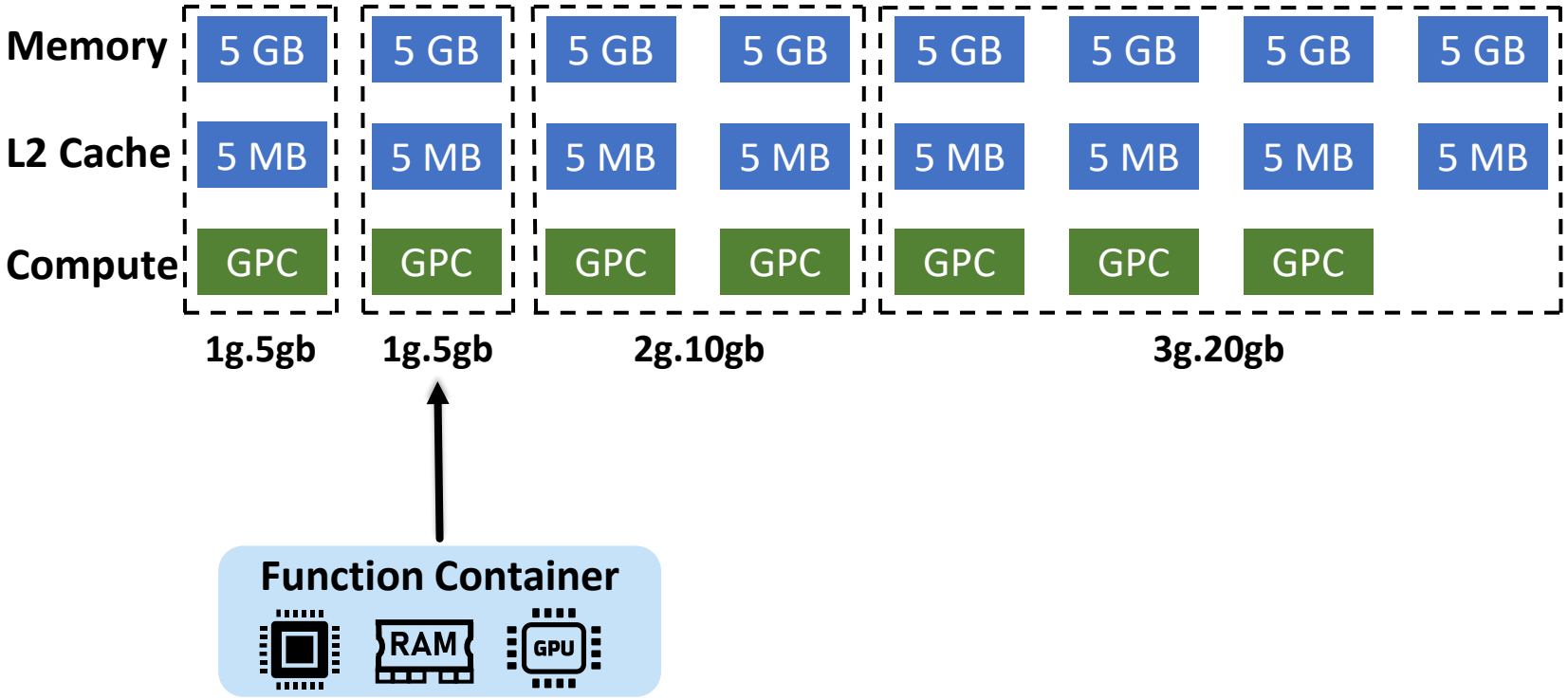


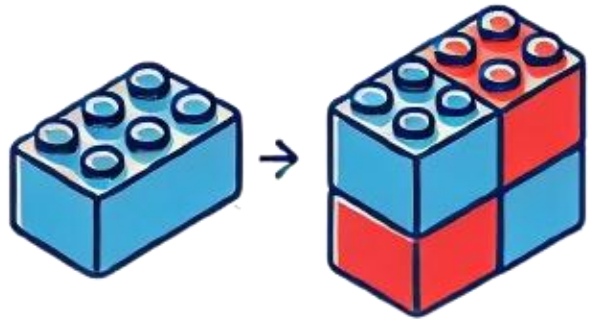
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)



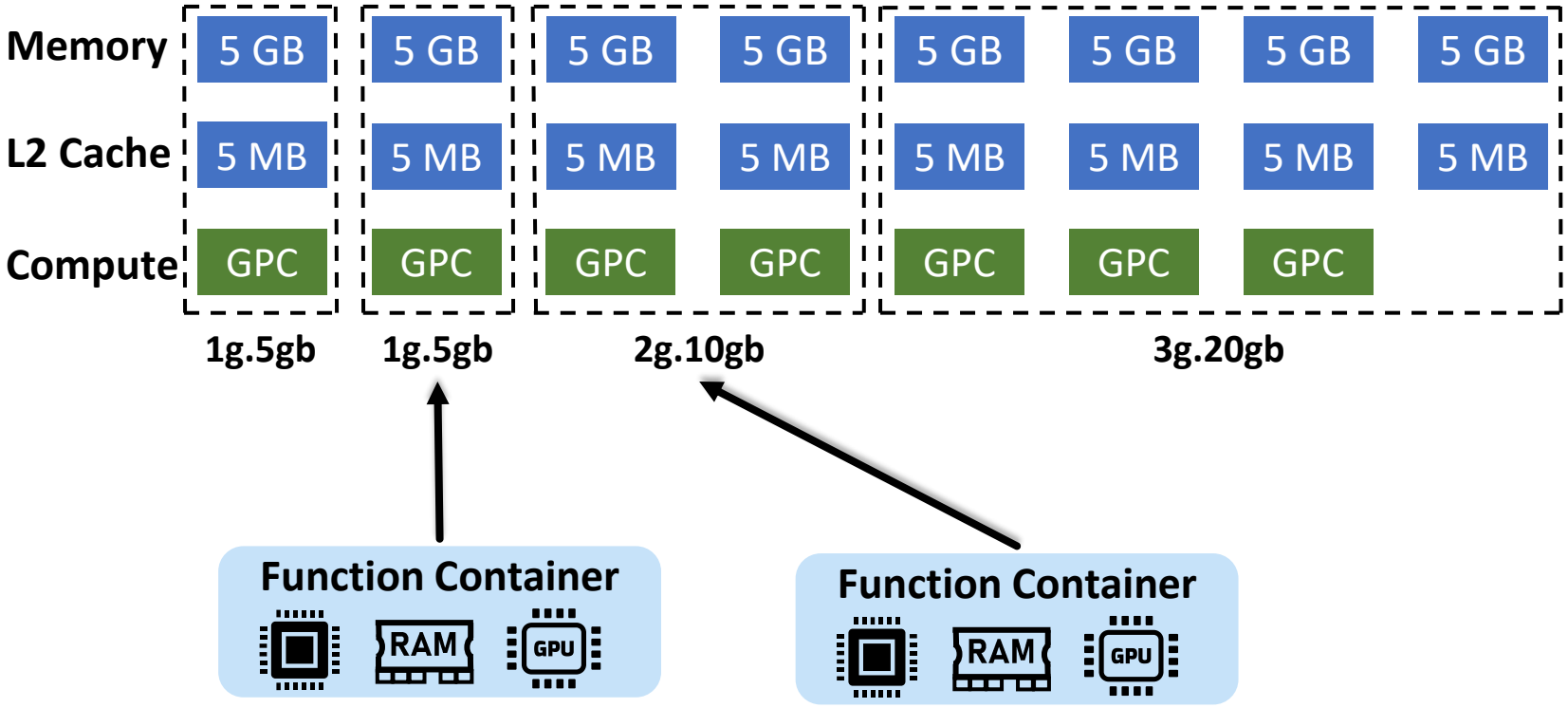


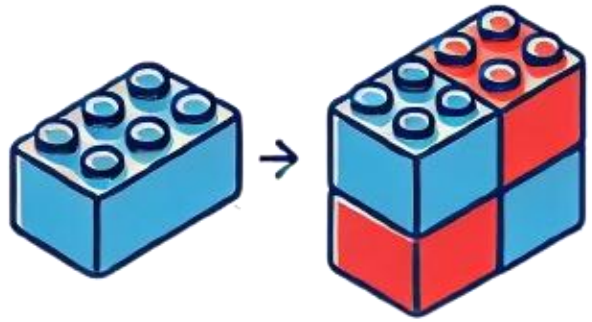
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)



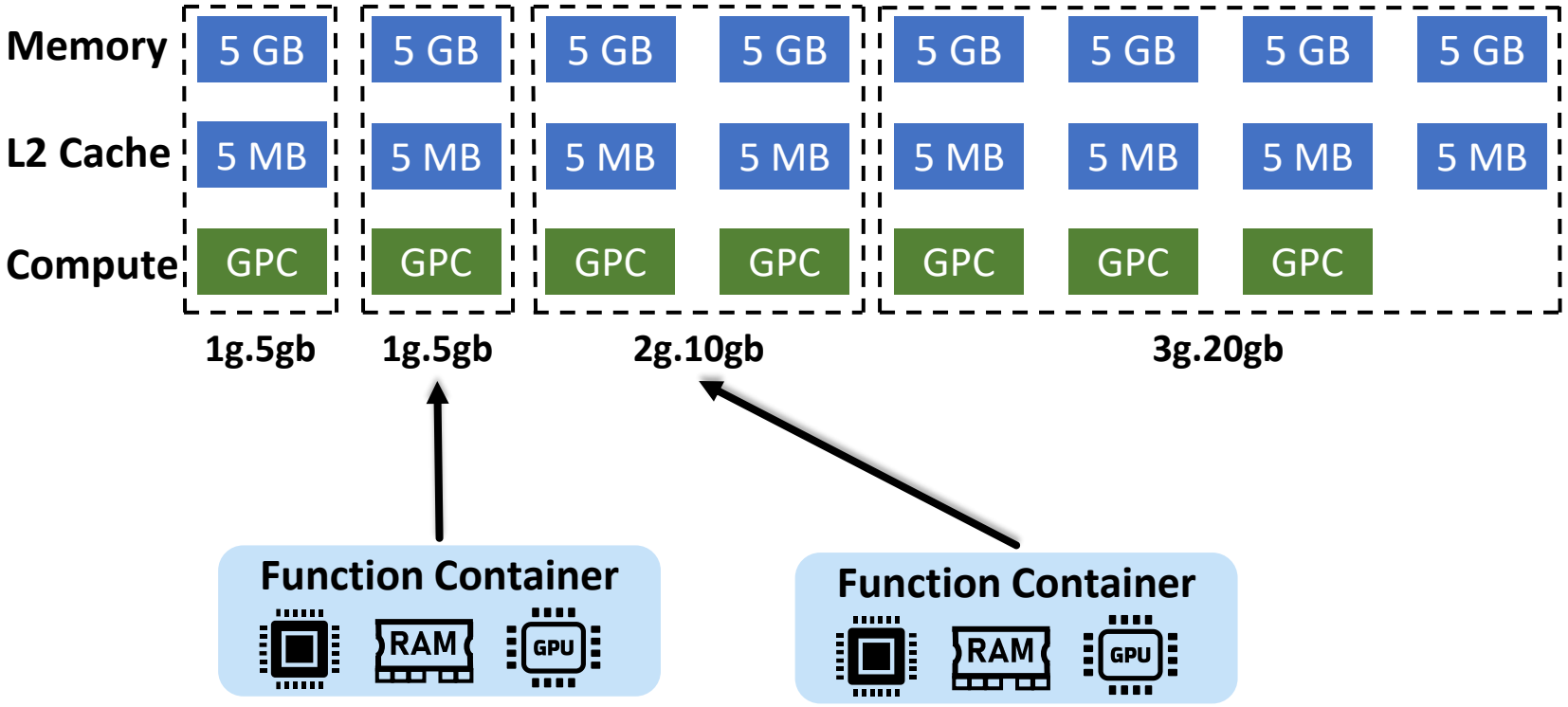


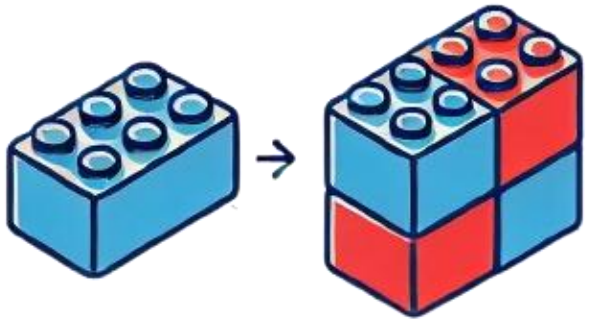
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)





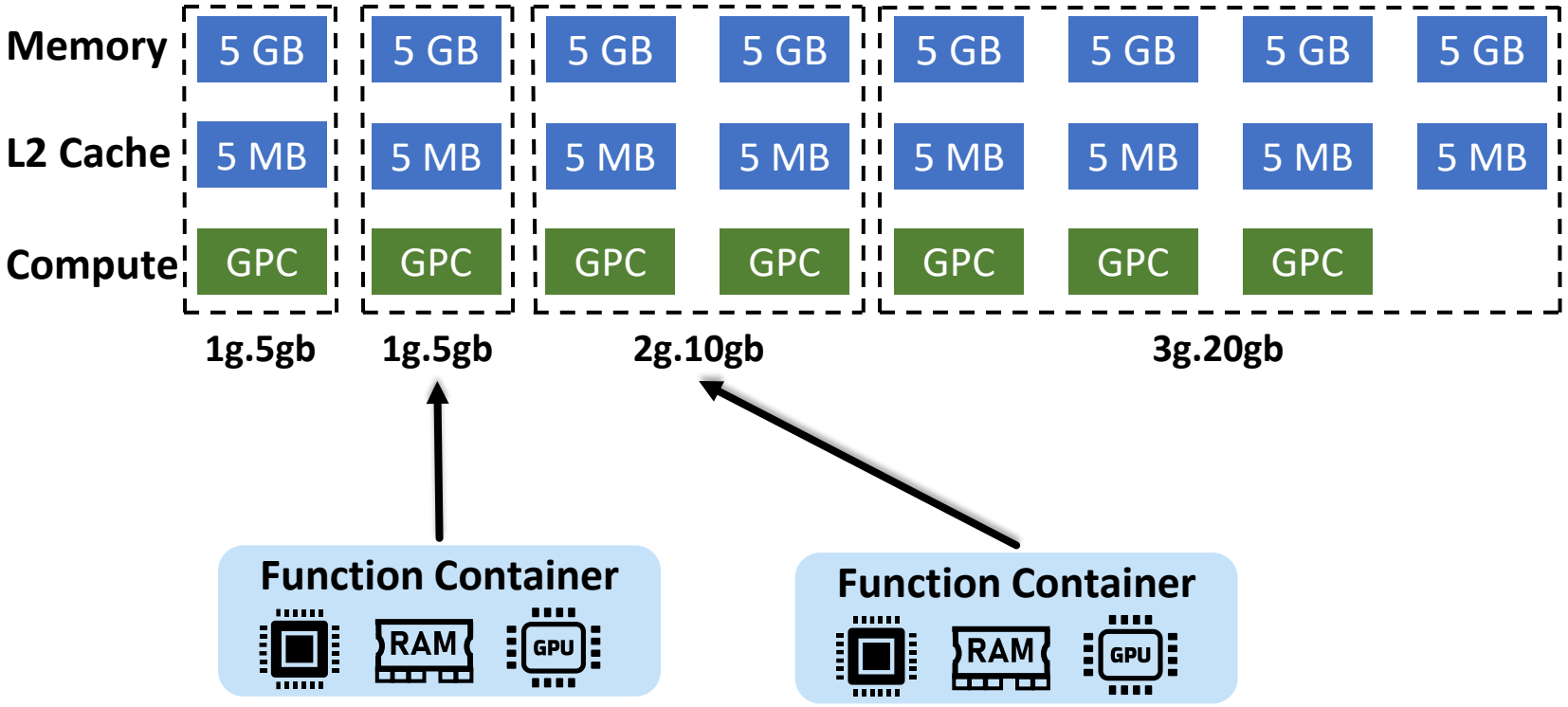
Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)

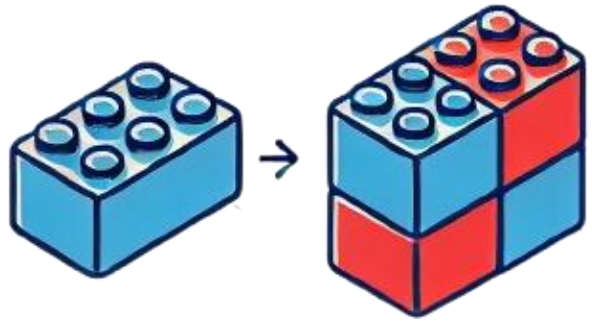




Not elastic!
Partitioning is quite static.

Serverless GPUs: Building from Blocks with Multi-Instance GPU (MIGs)



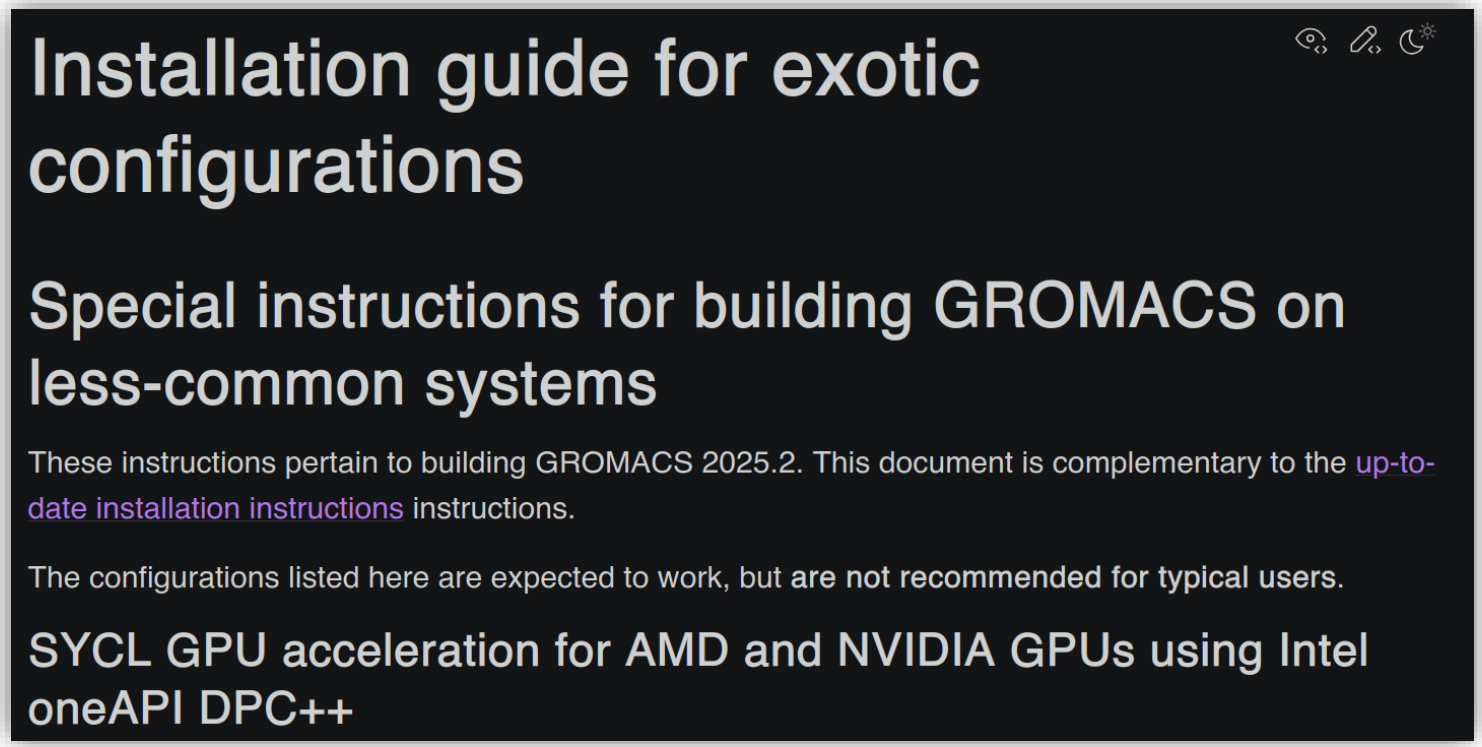


Not elastic!
Partitioning is quite static.

Static underutilization!
Difficult to migrate between partitions.

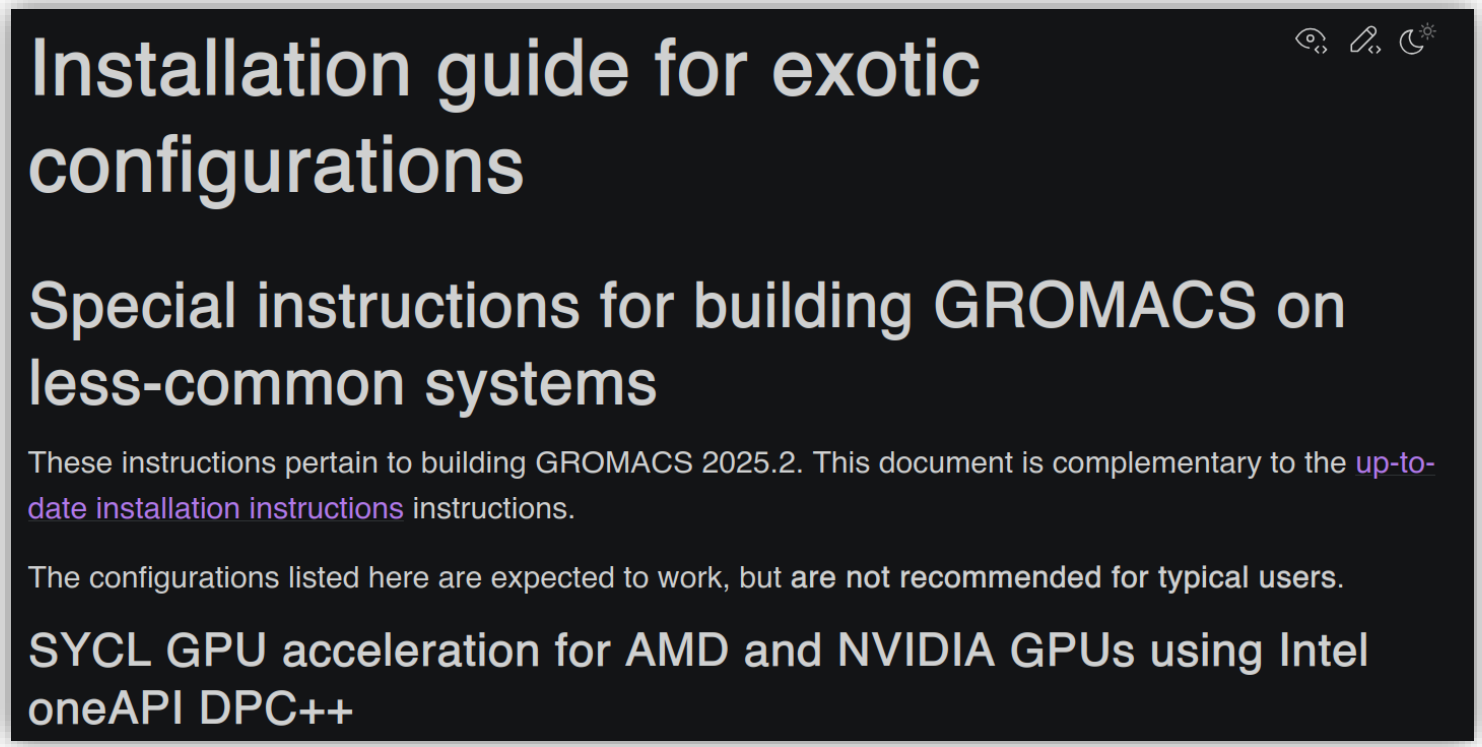
Evaluation – GROMACS Performance Portability

Why not just build one SYCL container for CUDA and Intel GPUs?



Evaluation – GROMACS Performance Portability

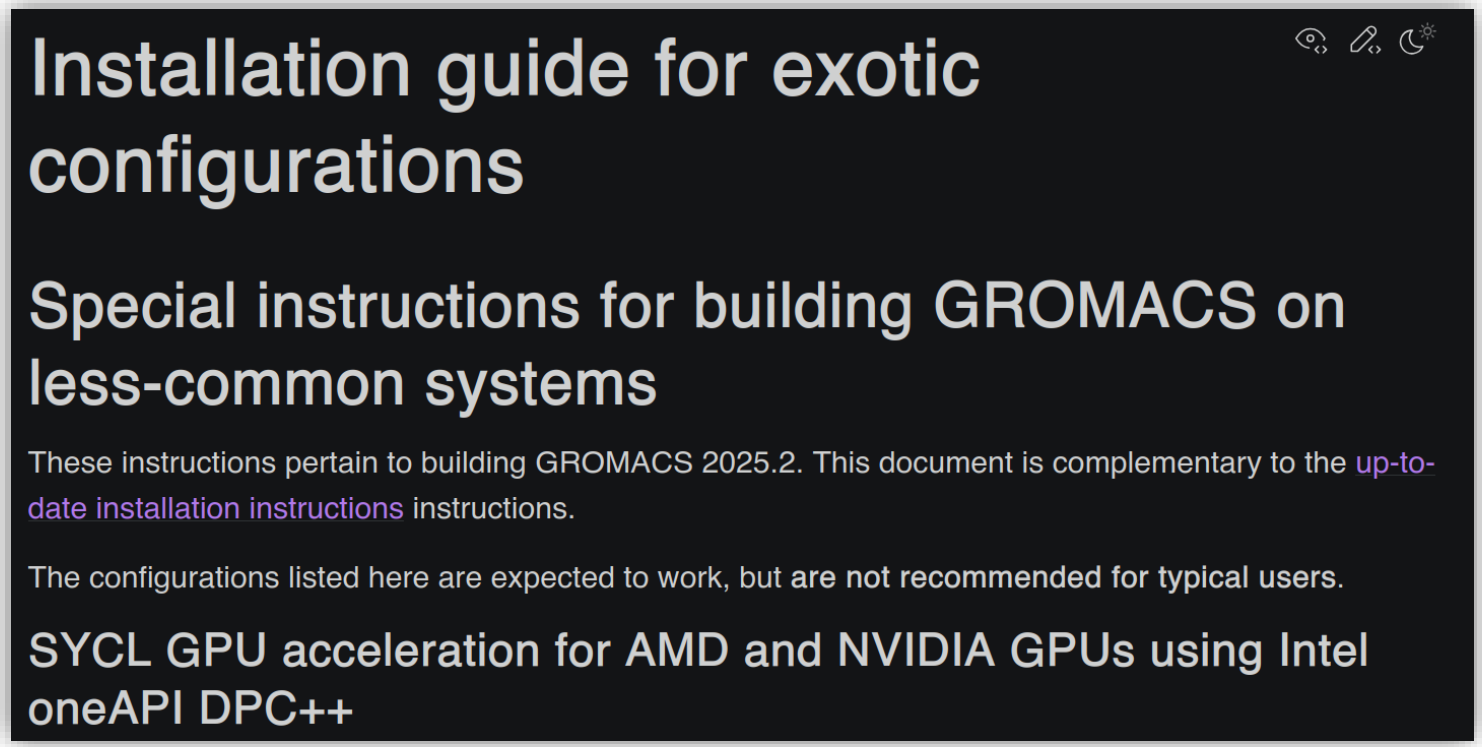
Why not just build one SYCL container for CUDA and Intel GPUs?



15-20% slowdown on test case A

Evaluation – GROMACS Performance Portability

Why not just build one SYCL container for CUDA and Intel GPUs?



Installation guide for exotic configurations

Special instructions for building GROMACS on less-common systems

These instructions pertain to building GROMACS 2025.2. This document is complementary to the up-to-date installation instructions instructions.

The configurations listed here are expected to work, but are not recommended for typical users.

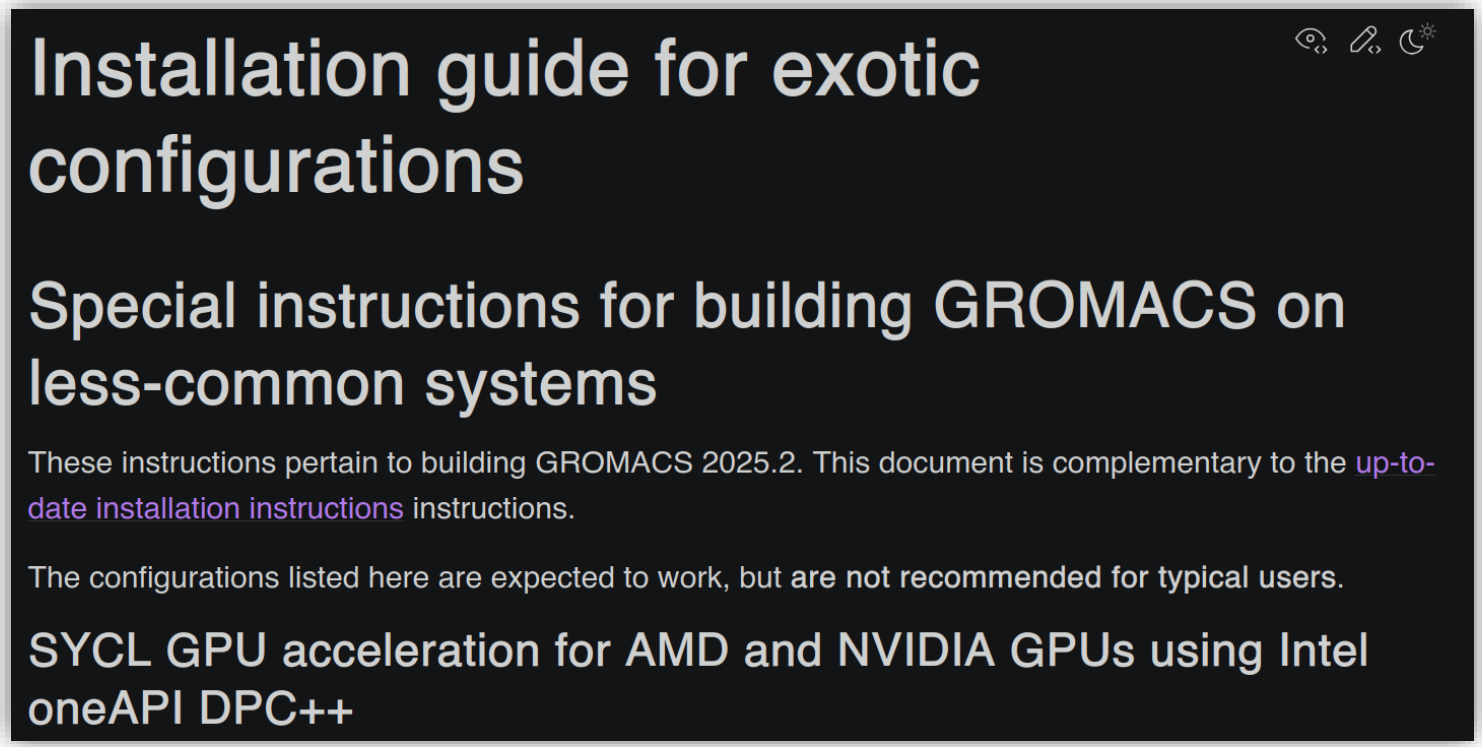
SYCL GPU acceleration for AMD and NVIDIA GPUs using Intel oneAPI DPC++

✗ **15-20% slowdown on test case A**

✗ **Failure on test case B**

Evaluation – GROMACS Performance Portability

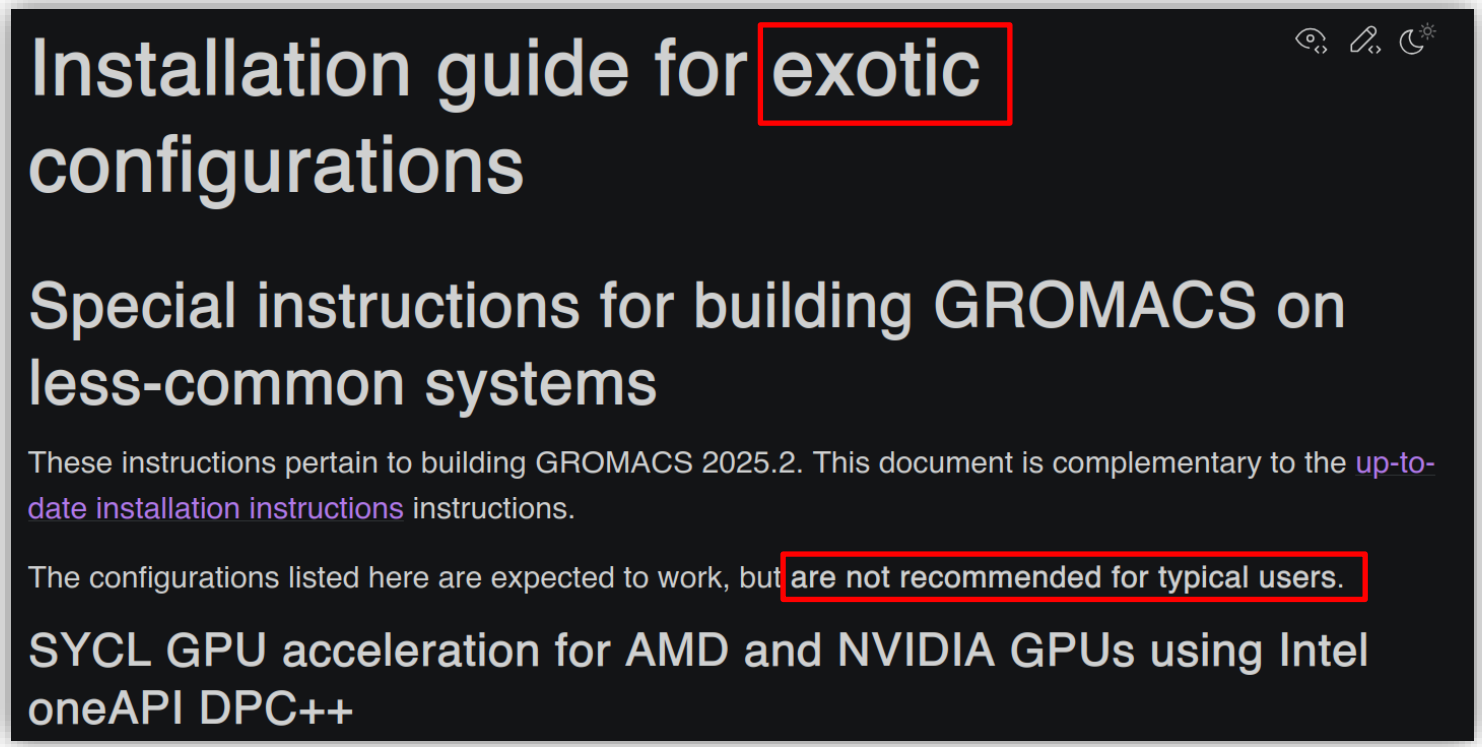
Why not just build one SYCL container for CUDA and Intel GPUs?



- **15-20% slowdown on test case A**
- **Failure on test case B**
- **Needs compile-time compatibility fix for Intel GPUs**

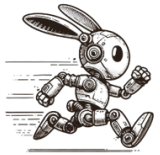
Evaluation – GROMACS Performance Portability

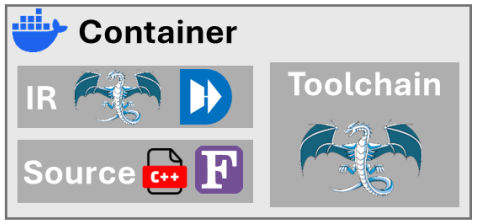
Why not just build one SYCL container for CUDA and Intel GPUs?



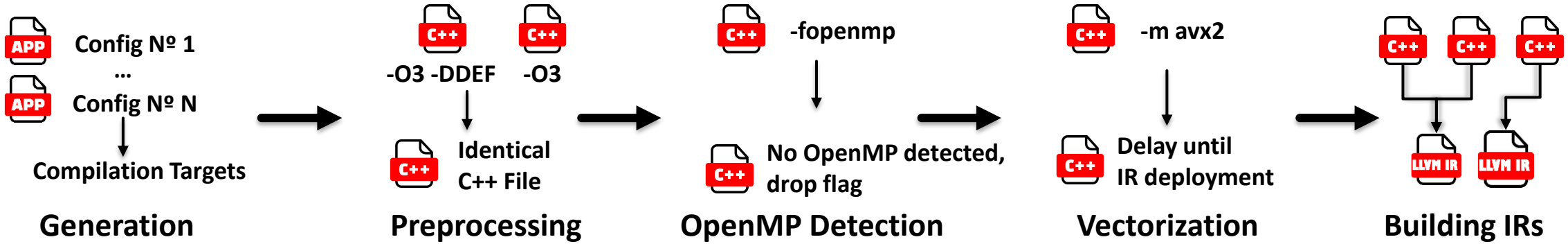
- **15-20% slowdown on test case A**
- **Failure on test case B**
- **Needs compile-time compatibility fix for Intel GPUs**

XaaS IR Containers: Build Containers

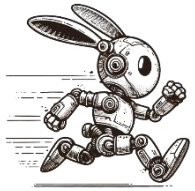


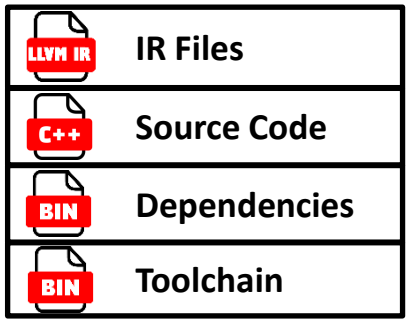


XaaS IR Container

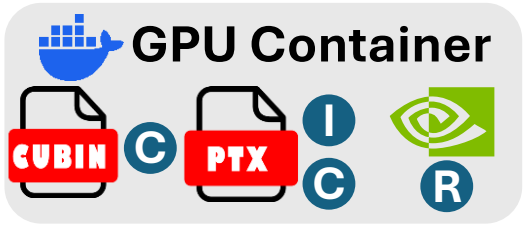


XaaS IR Container





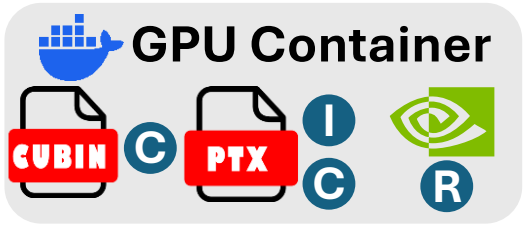
XaaS IR Containers: GPU Acceleration



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

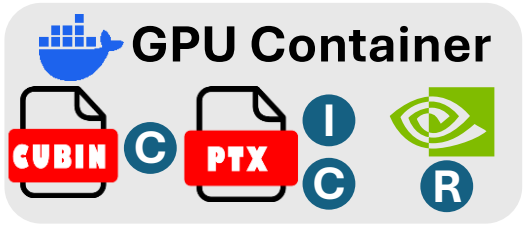
- **D** Driver Version



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

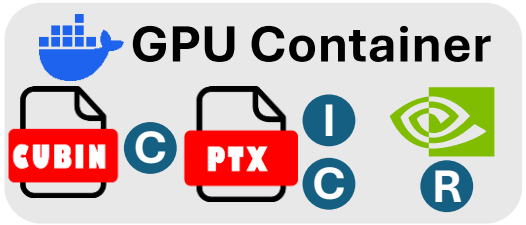
- **D** Driver Version
- **R** Runtime Version



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

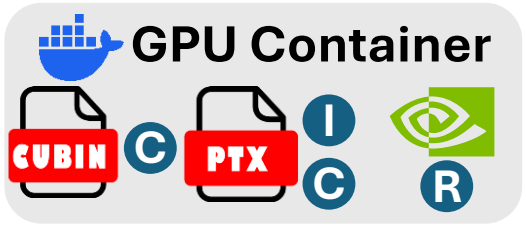
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

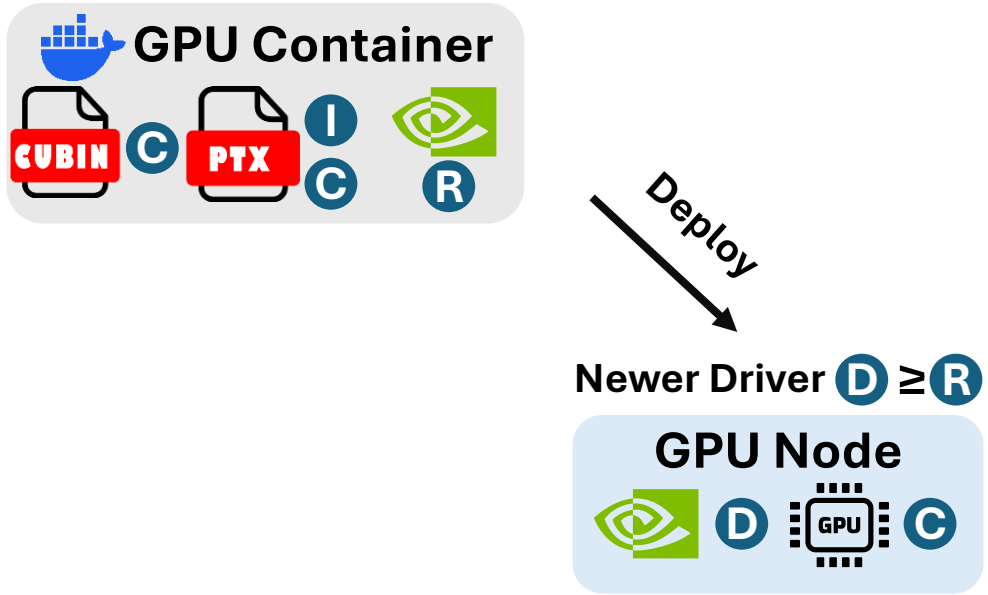
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

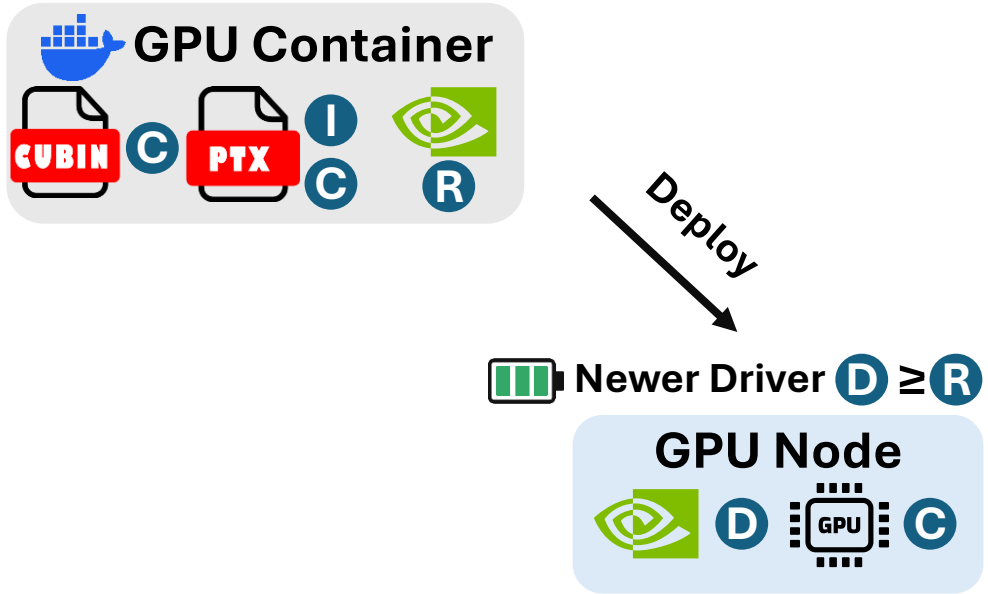
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

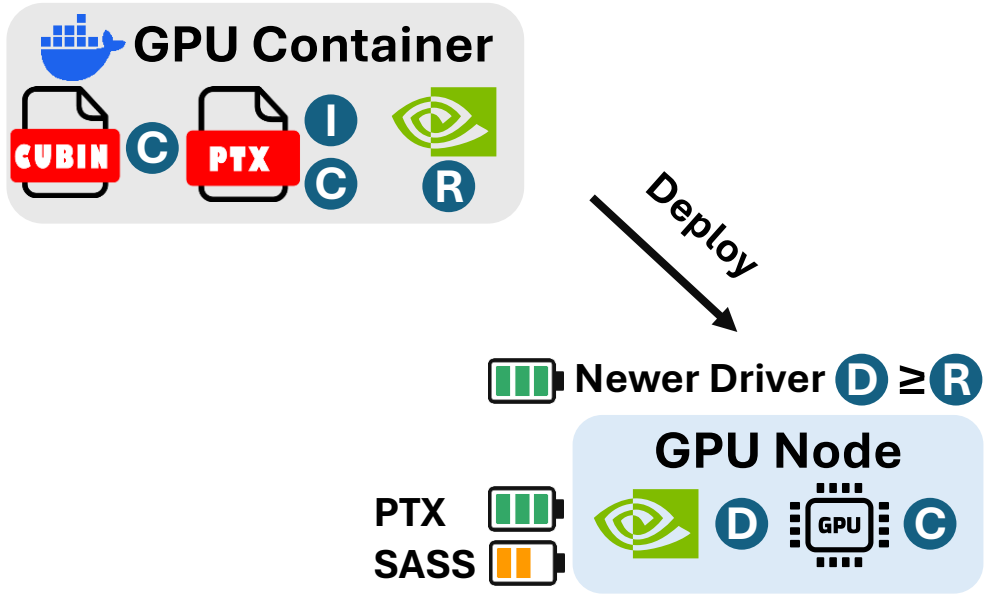
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

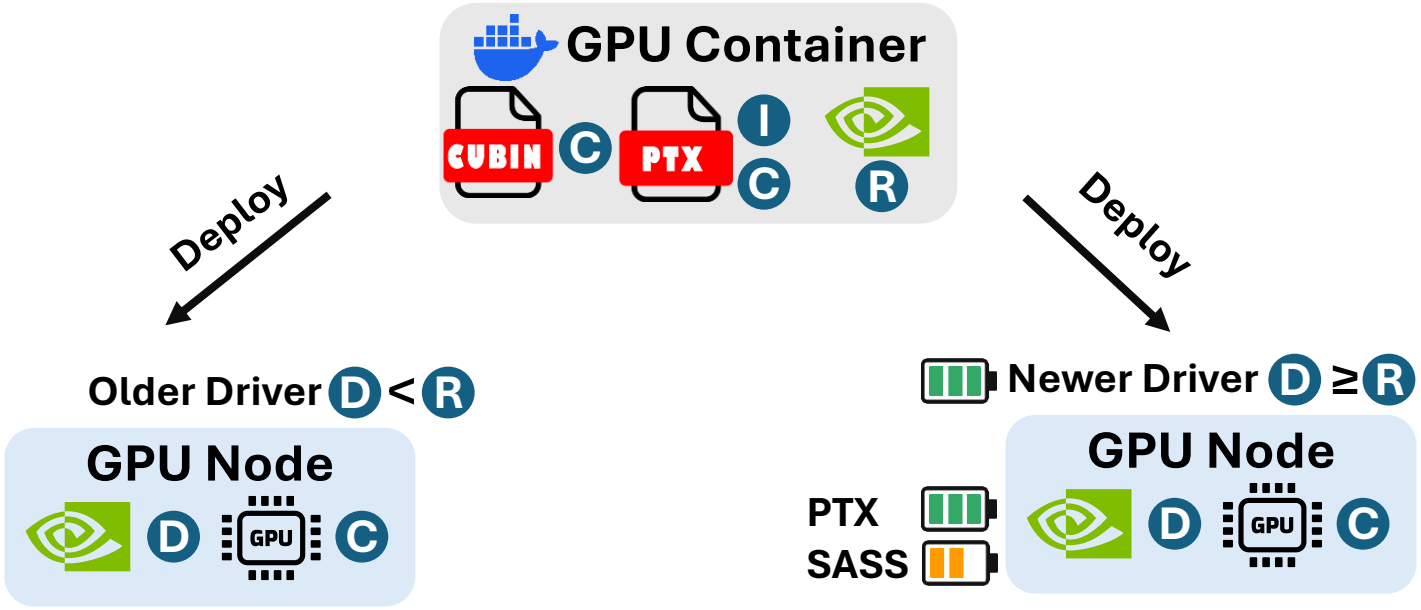
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

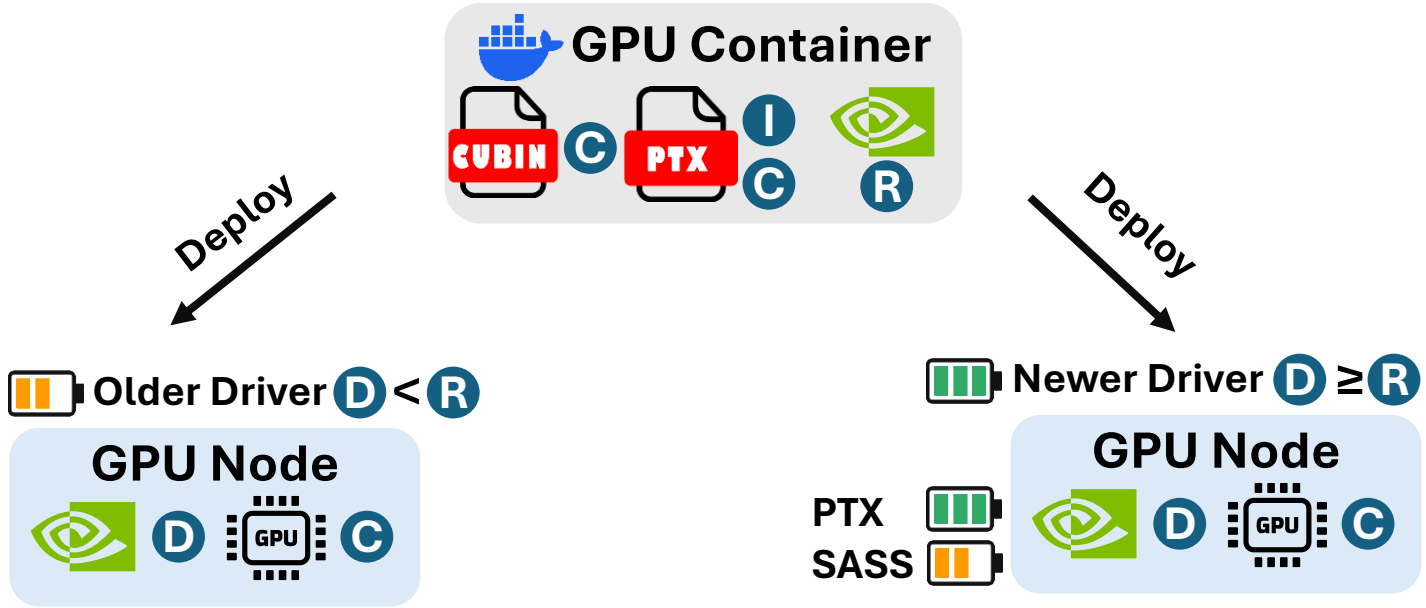
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

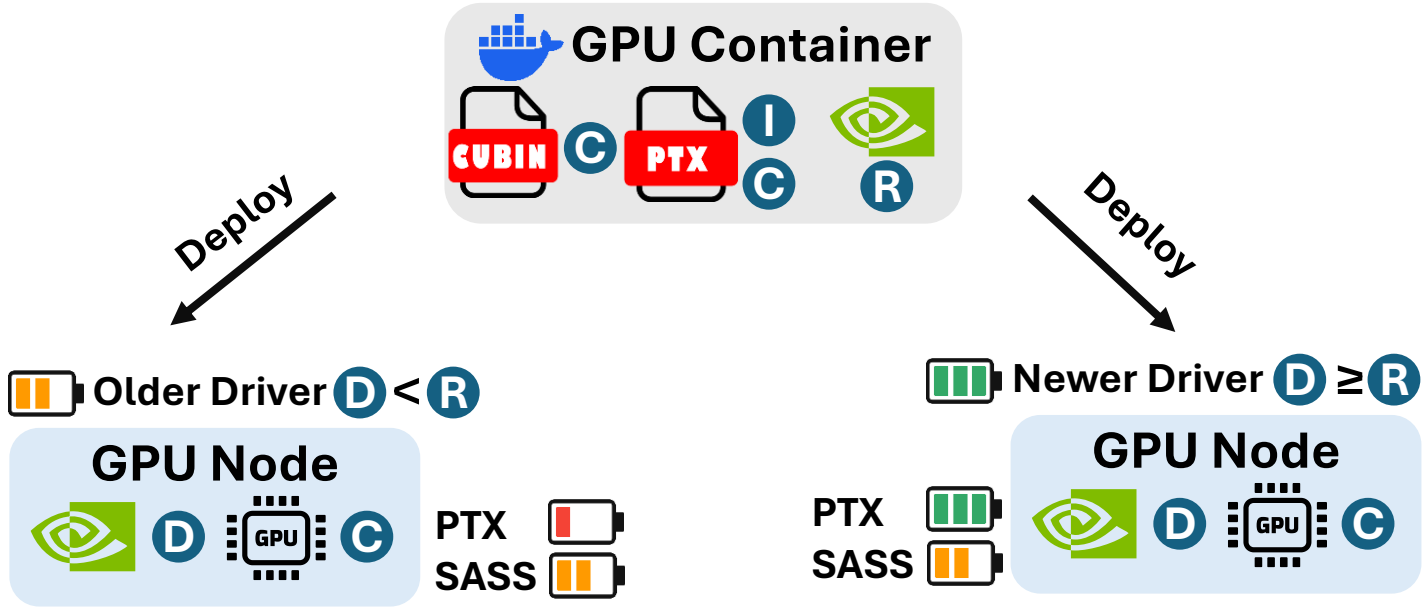
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

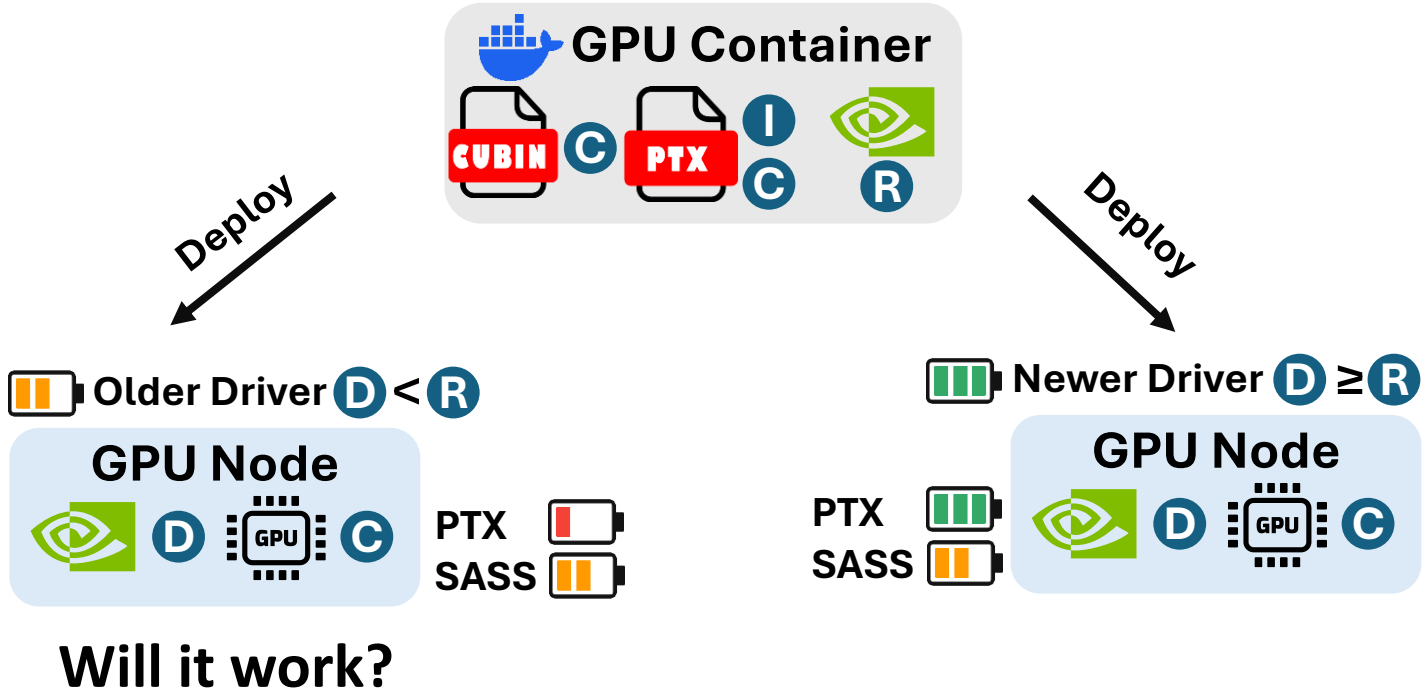
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

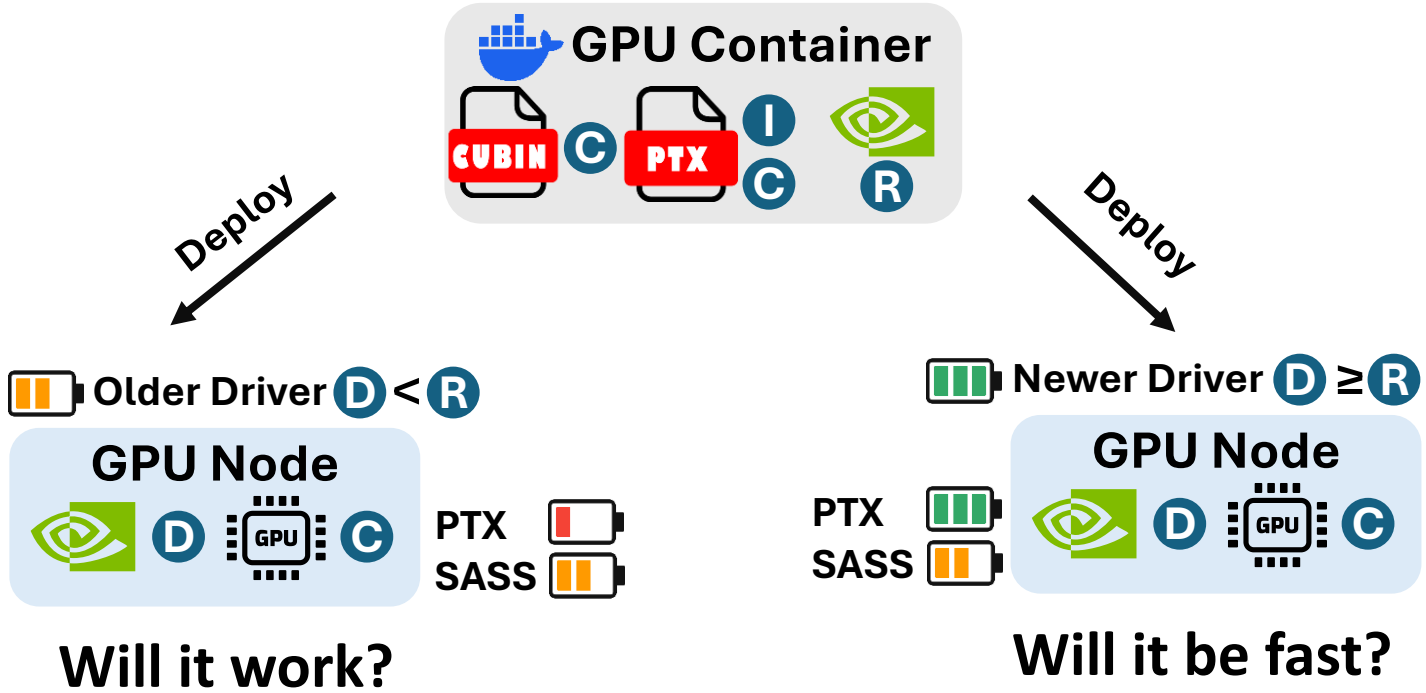
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

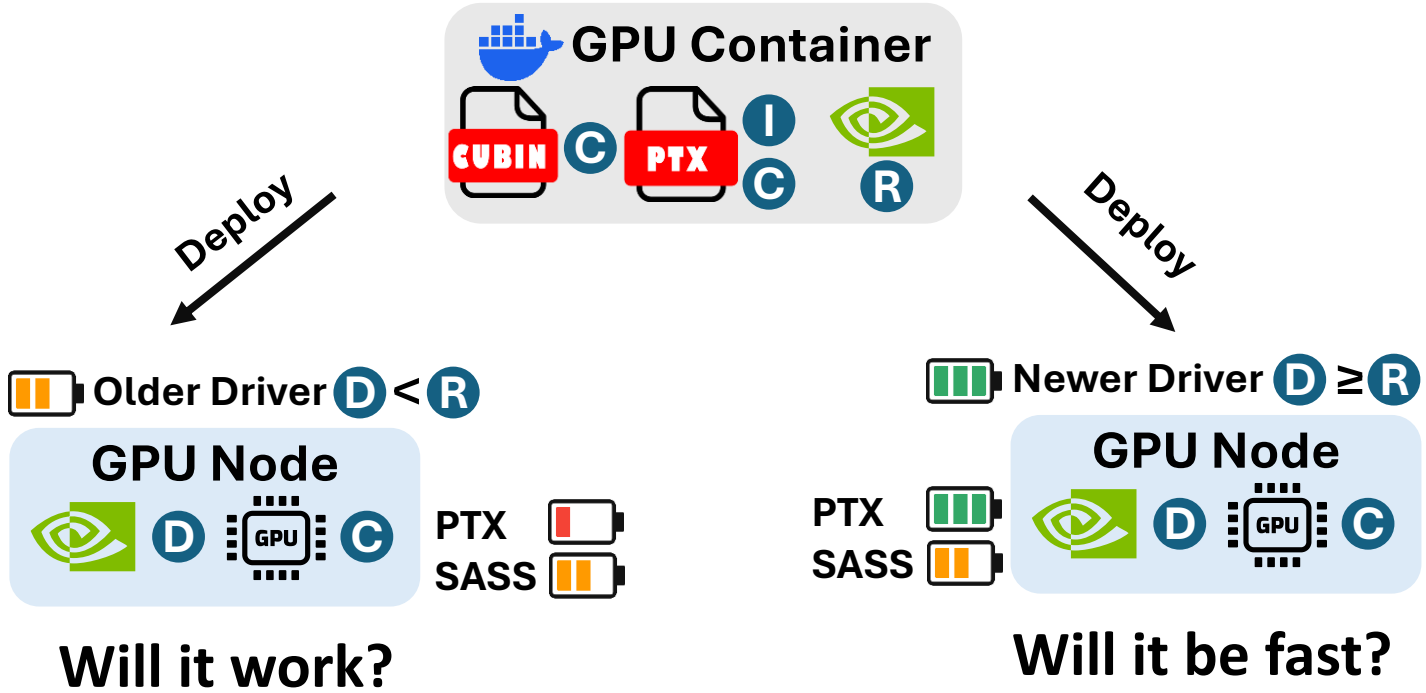
- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA

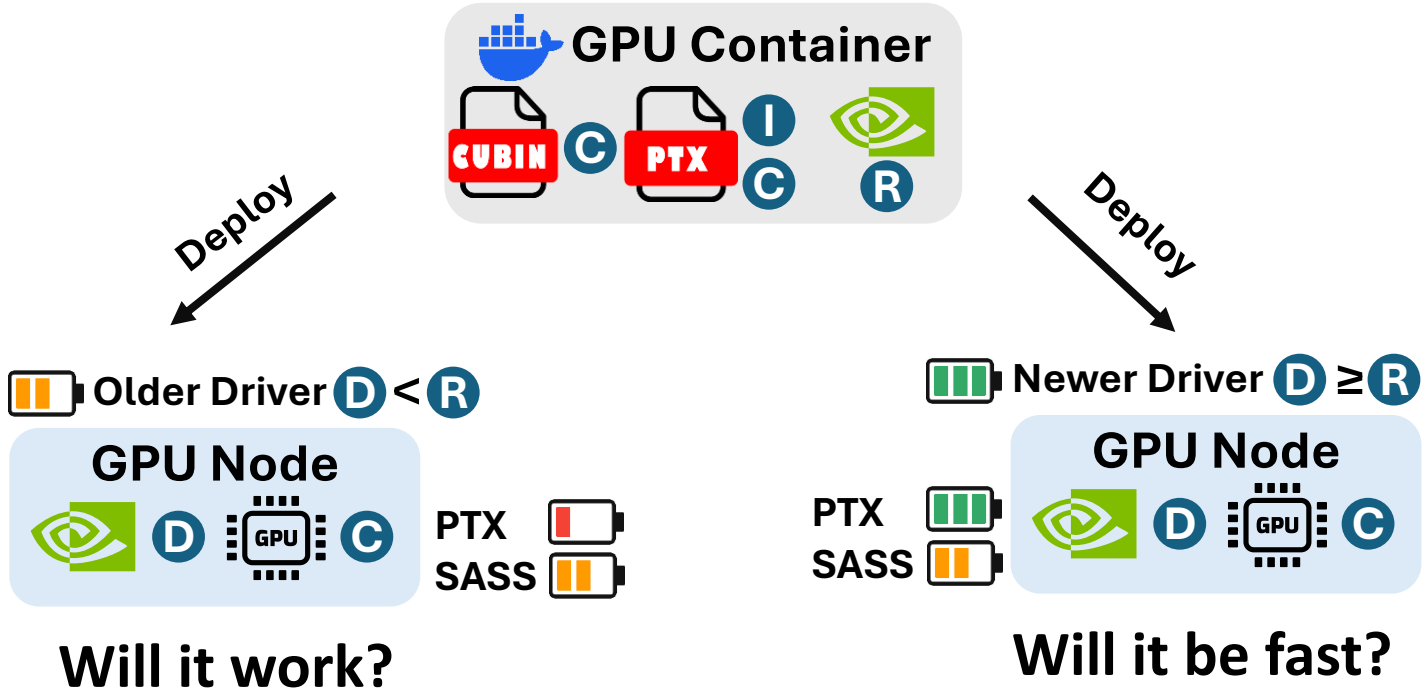


Generate SASS for all supported versions + PTX for the highest version.

XaaS IR Containers: GPU Acceleration

CUDA Compatibility Model

- **D** Driver Version
- **R** Runtime Version
- **C** Compute Capability (SASS, PTX)
- **I** PTX ISA



- Generate SASS for all supported versions + PTX for the highest version.
- Check for dependency on CUDA version.

Evaluation – Specialization Point Detection with LLMs

Prompt with Examples
(In-Context Learning)

JSON Schema

Build Configuration

Evaluation – Specialization Point Detection with LLMs

Prompt with Examples
(In-Context Learning)

JSON Schema

Build Configuration

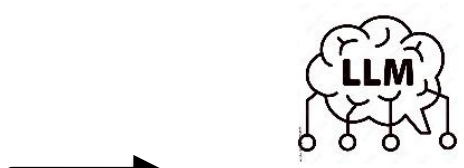
Sample LLM
10 times.

Evaluation – Specialization Point Detection with LLMs

Prompt with Examples (In-Context Learning)

JSON Schema

CMake Build Configuration



Sample LLM 10 times.

HPC Specialization Points

LLM Result

versus

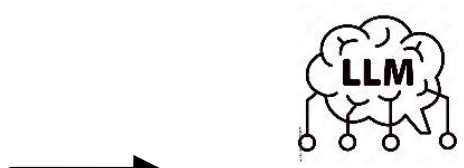
Ground Truth

Evaluation – Specialization Point Detection with LLMs

Prompt with Examples (In-Context Learning)

JSON Schema

CMake Build Configuration



Sample LLM 10 times.

HPC Specialization Points

LLM Result

versus

Ground Truth

ICL Test Case: GROMACS

Gemini 1.5 & 2.0

F1 Scores: 0.86 – 0.99

Evaluation – Specialization Point Detection with LLMs

Prompt with Examples (In-Context Learning)

JSON Schema

Build Configuration

Sample LLM 10 times.

HPC Specialization Points

LLM Result

versus

Ground Truth

ICL Test Case: GROMACS

Gemini 1.5 & 2.0

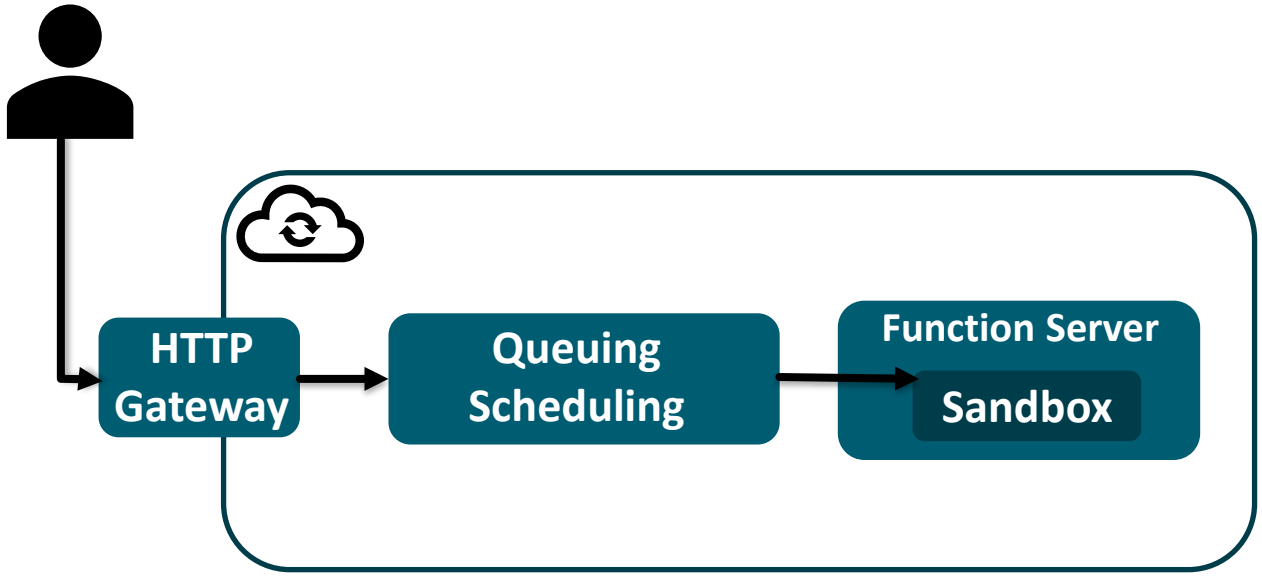
F1 Scores: 0.86 – 0.99

Generalization: llama.cpp

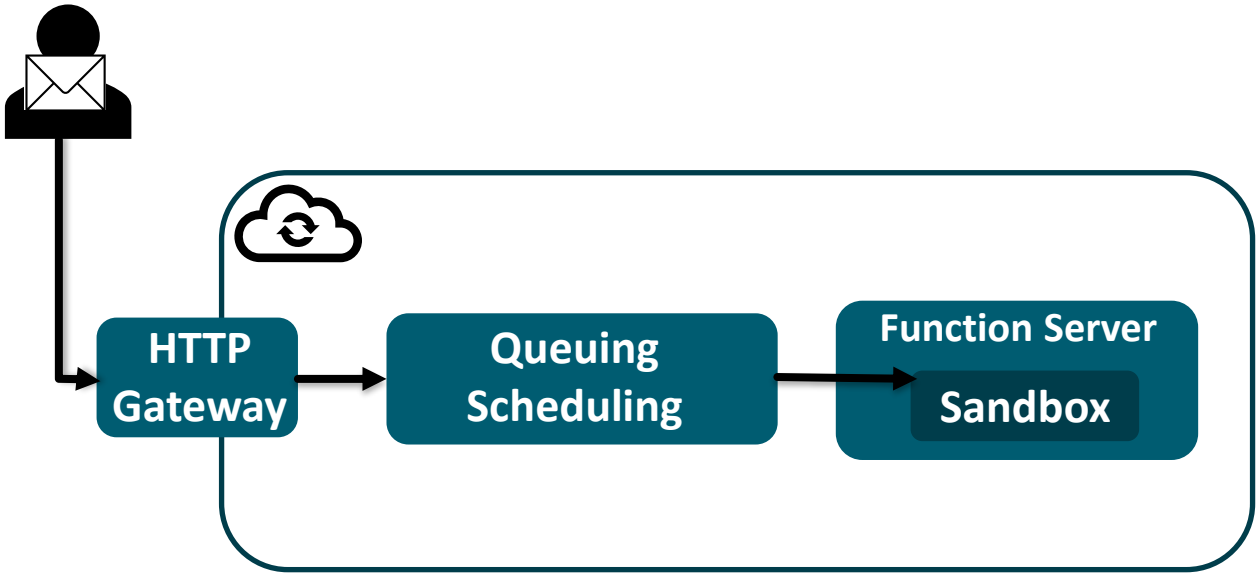
GPT-o3

F1 Scores: 0.73 - 0.79

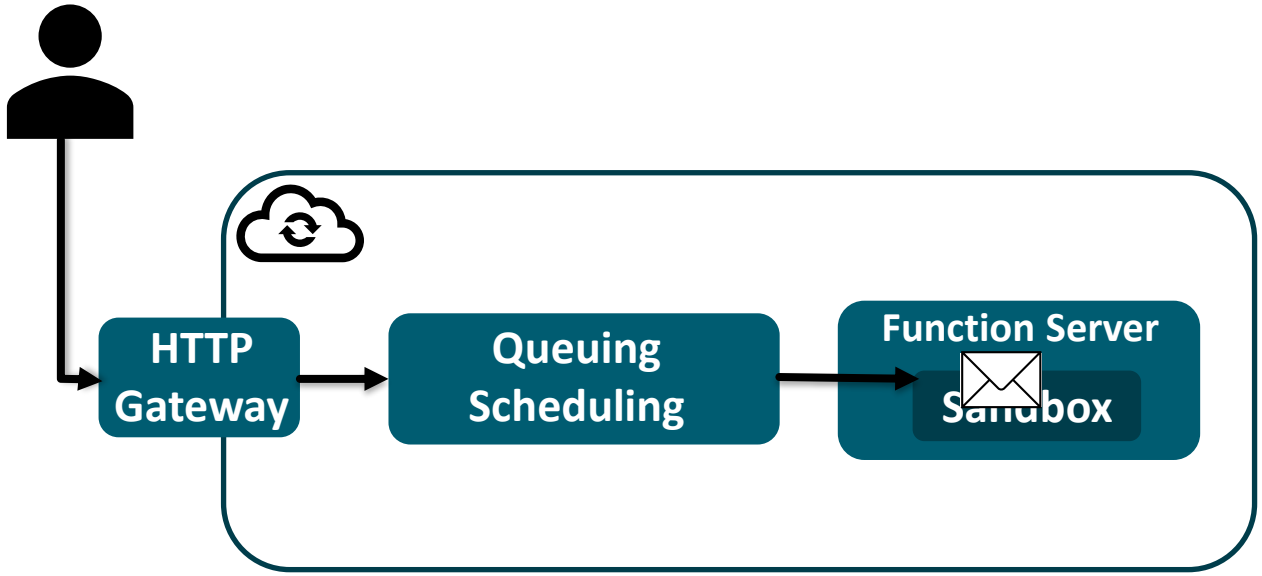
How does FaaS work?



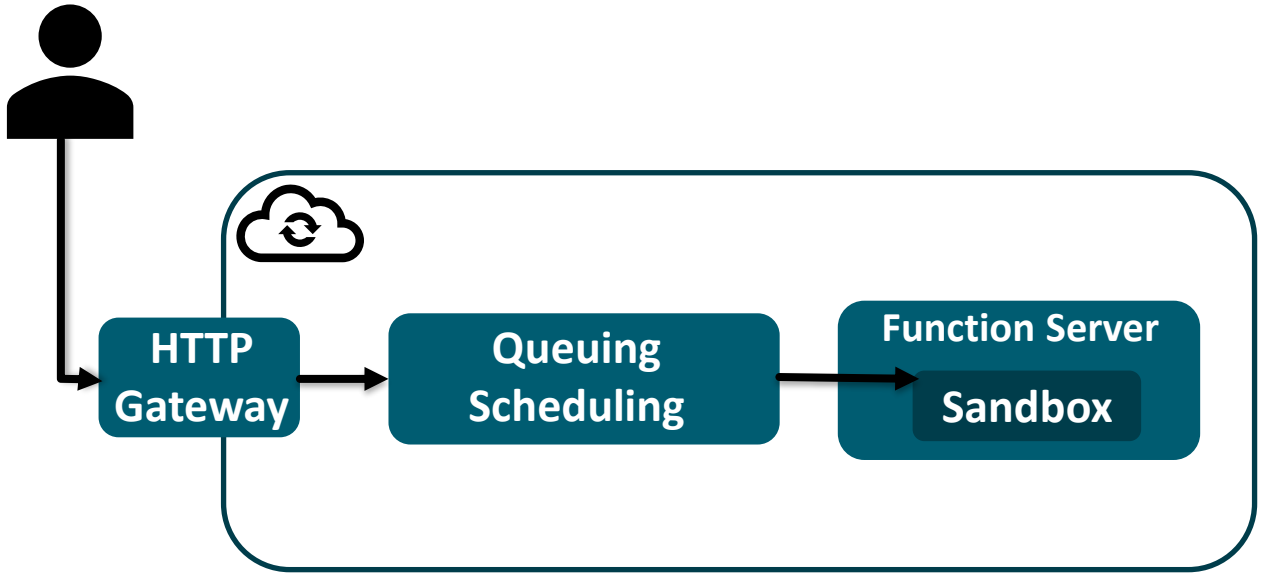
How does FaaS work?



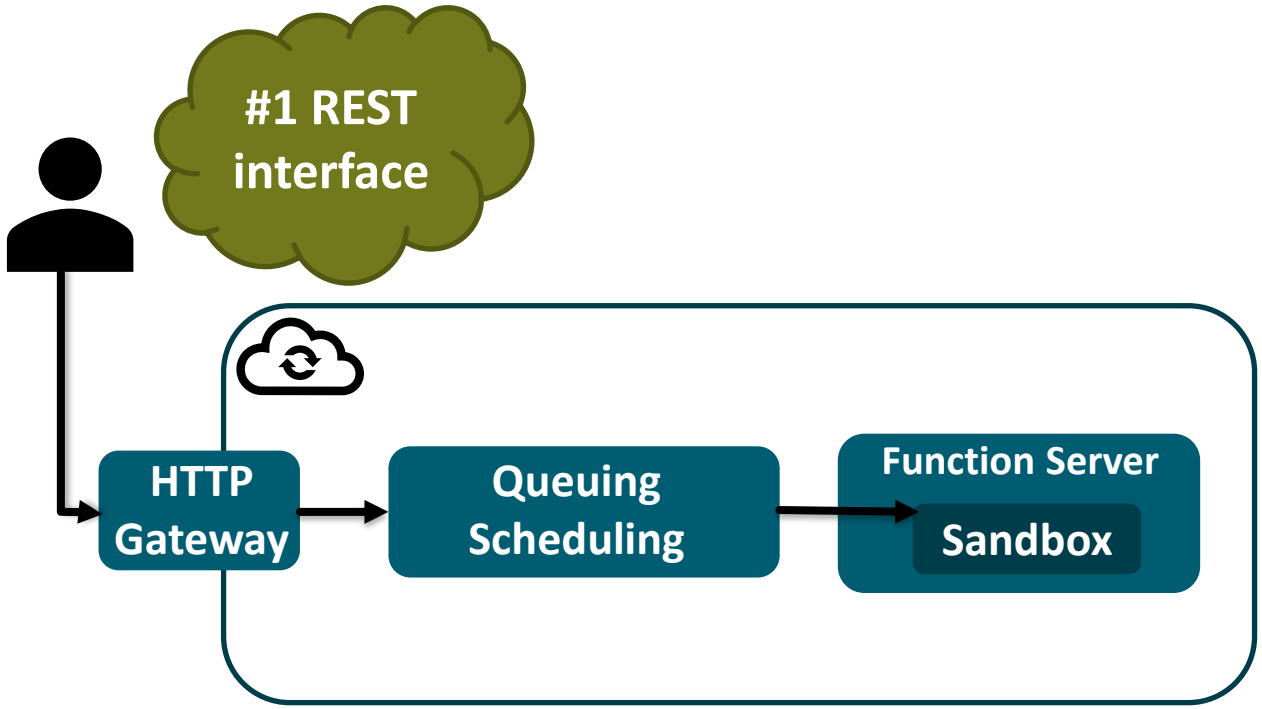
How does FaaS work?



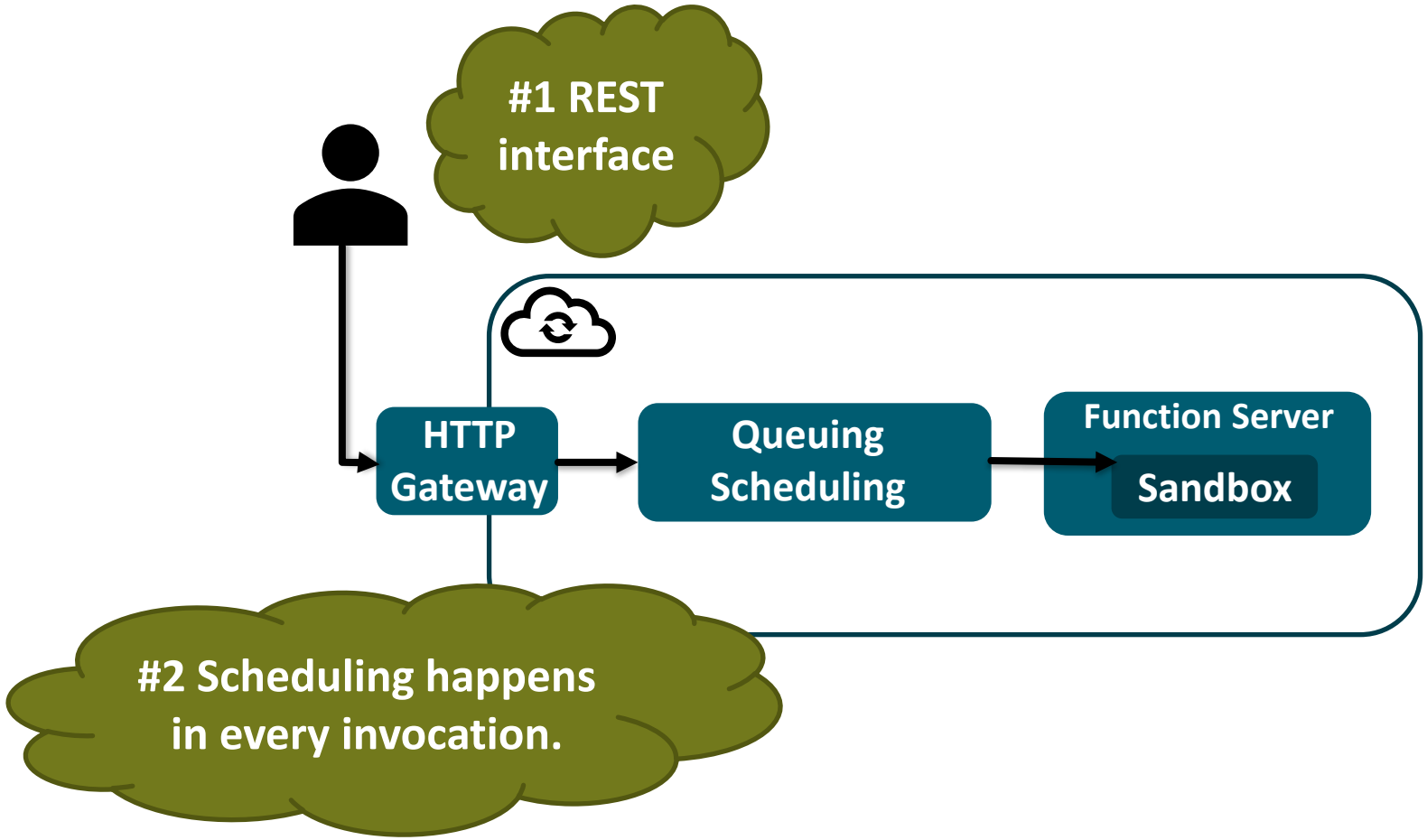
Why is FaaS slow?



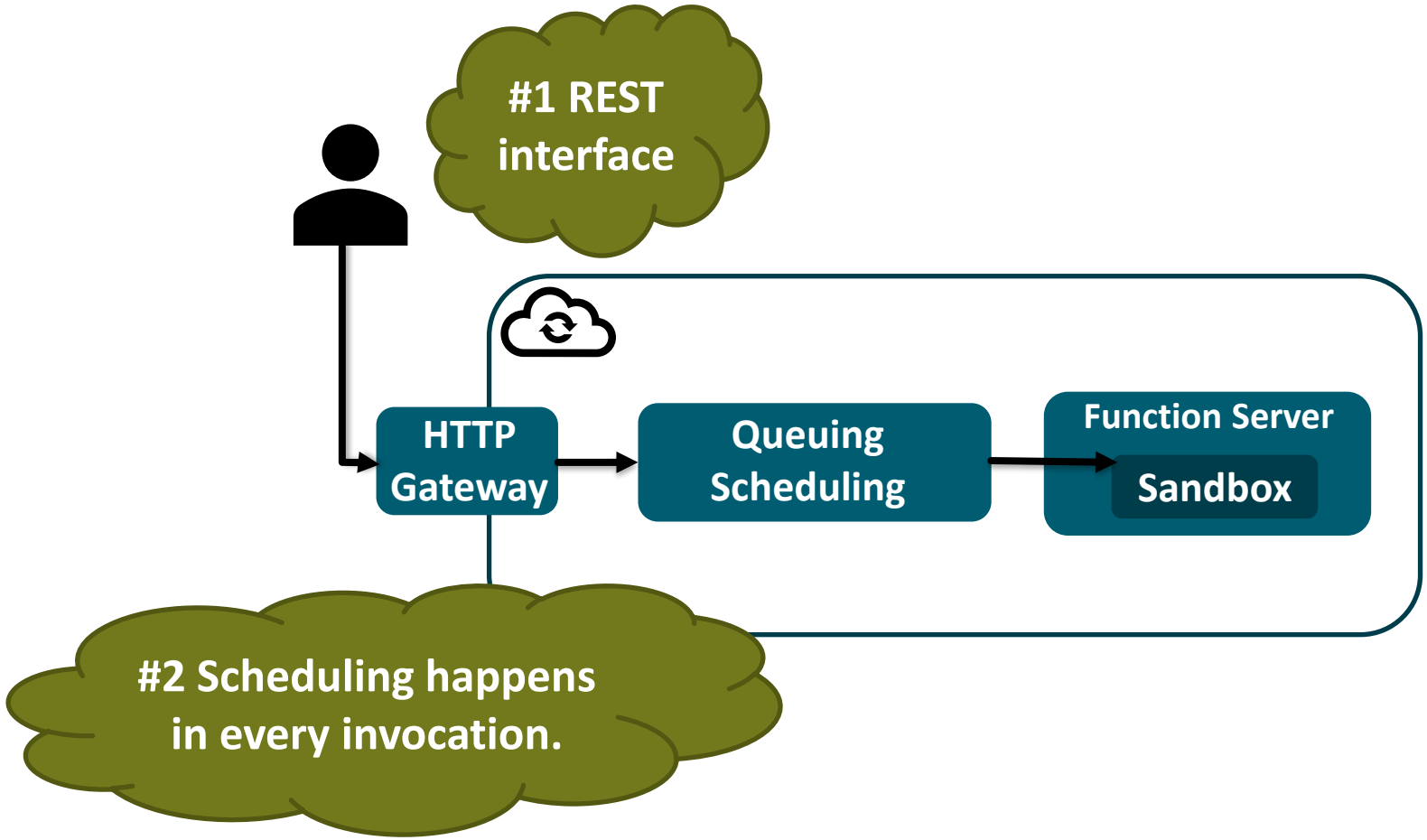
Why is FaaS slow?



Why is FaaS slow?

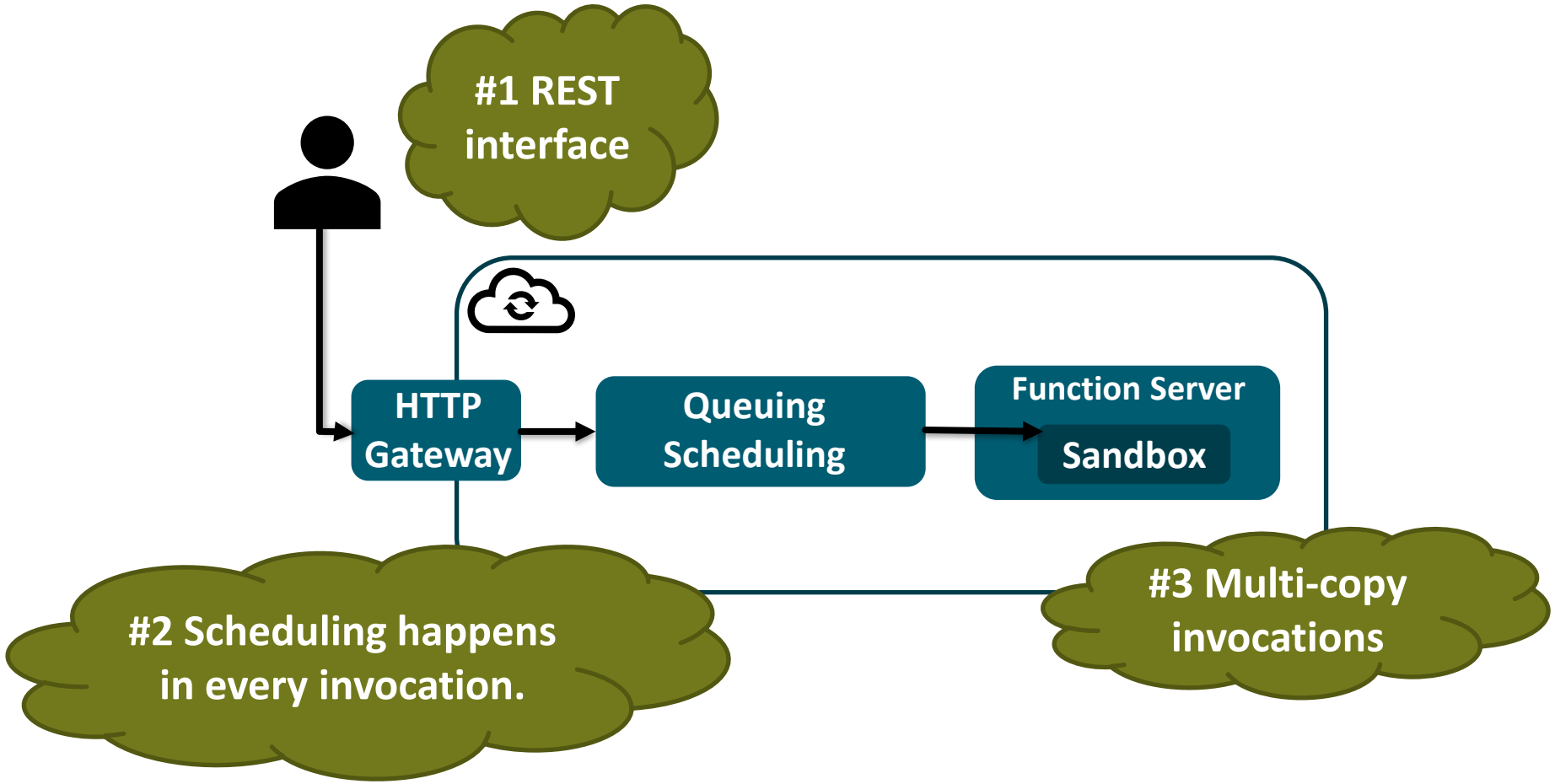


Why is FaaS slow?



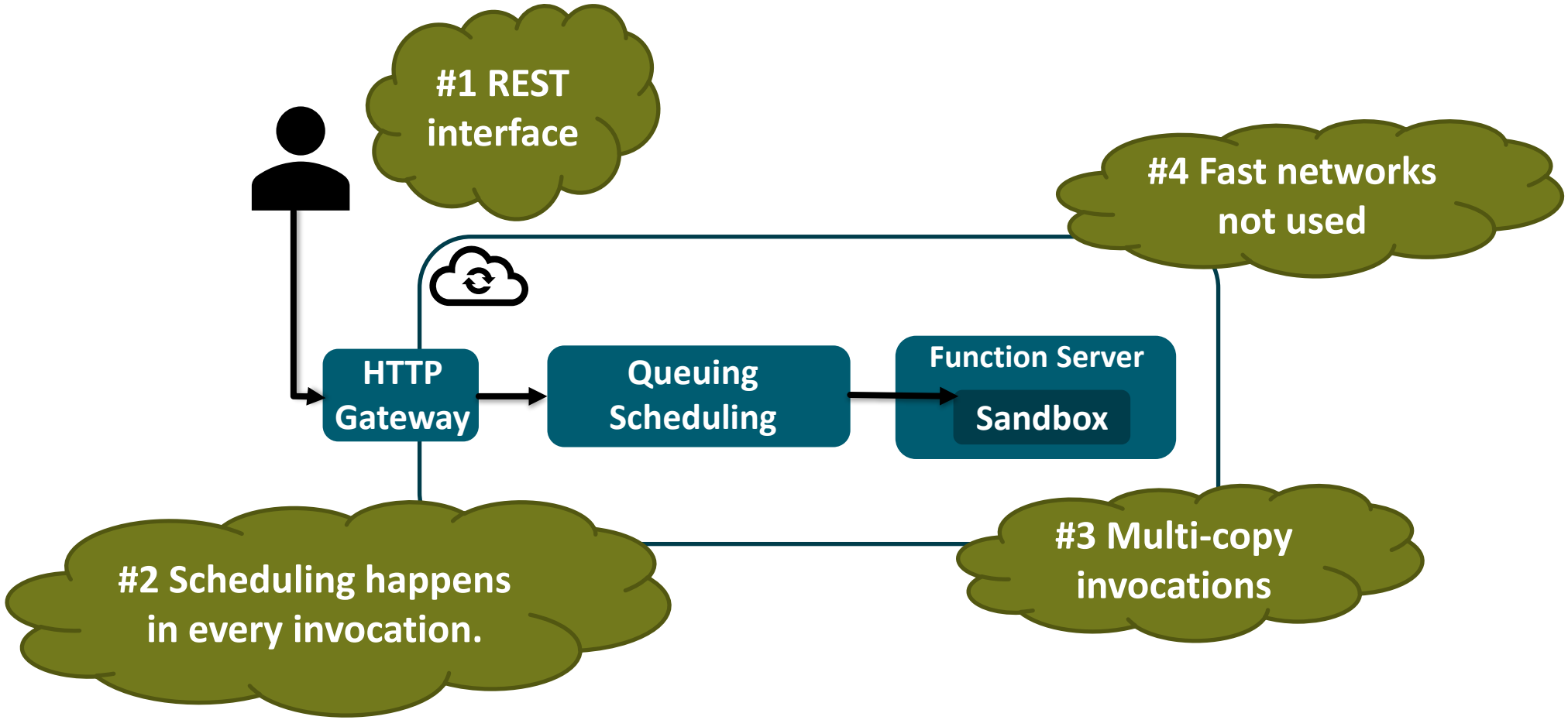
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing"

Why is FaaS slow?



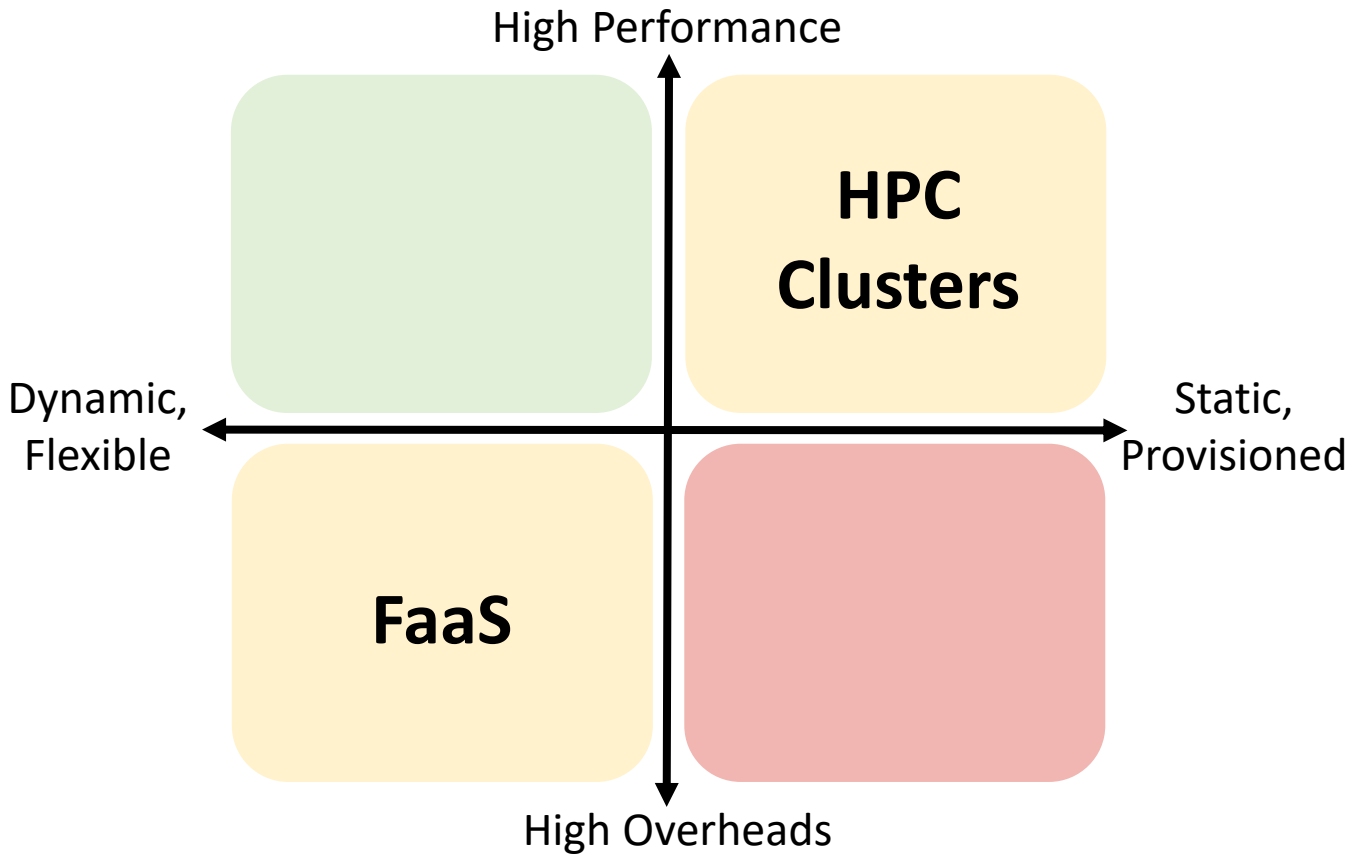
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing"

Why is FaaS slow?

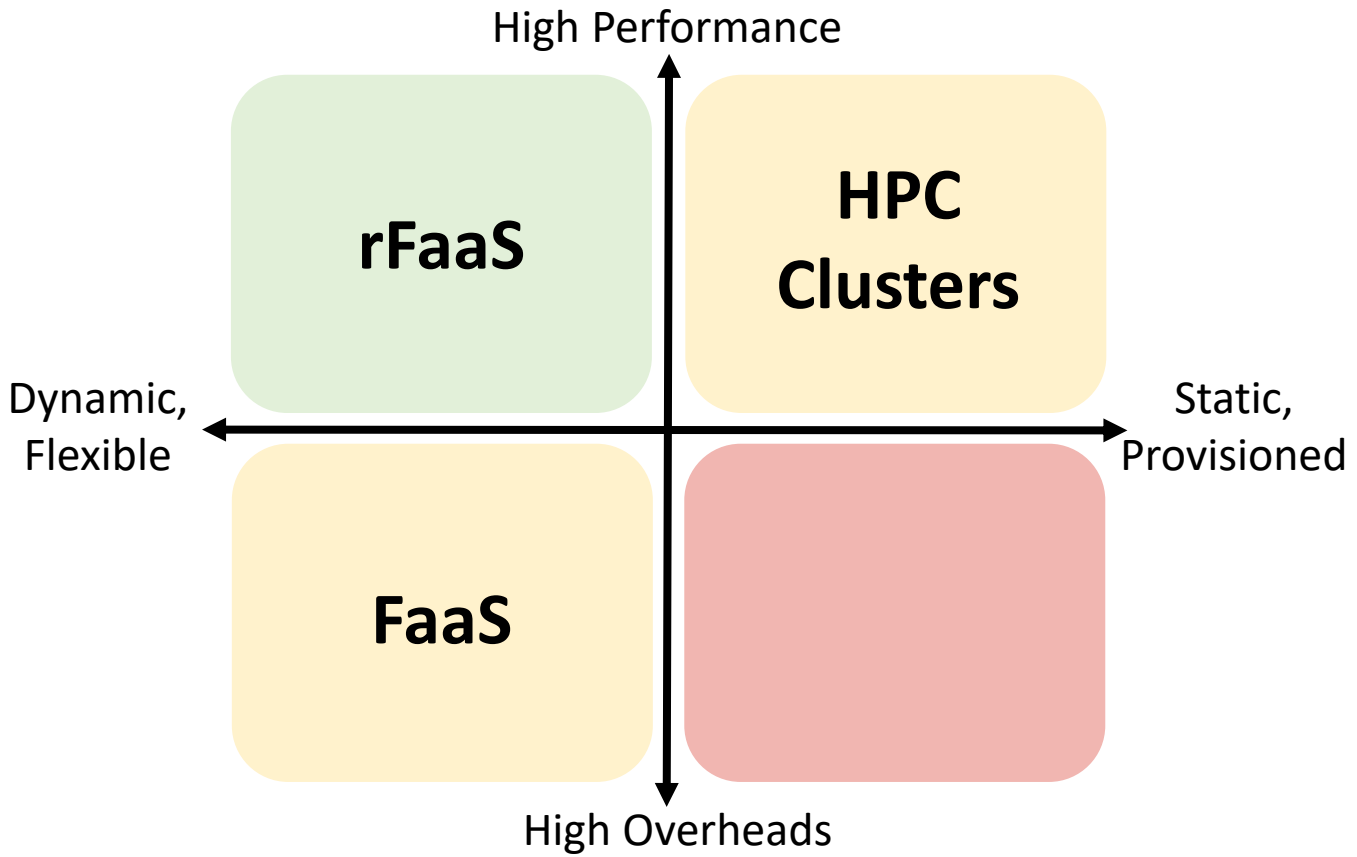


"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing"

Function-as-a-Service for HPC



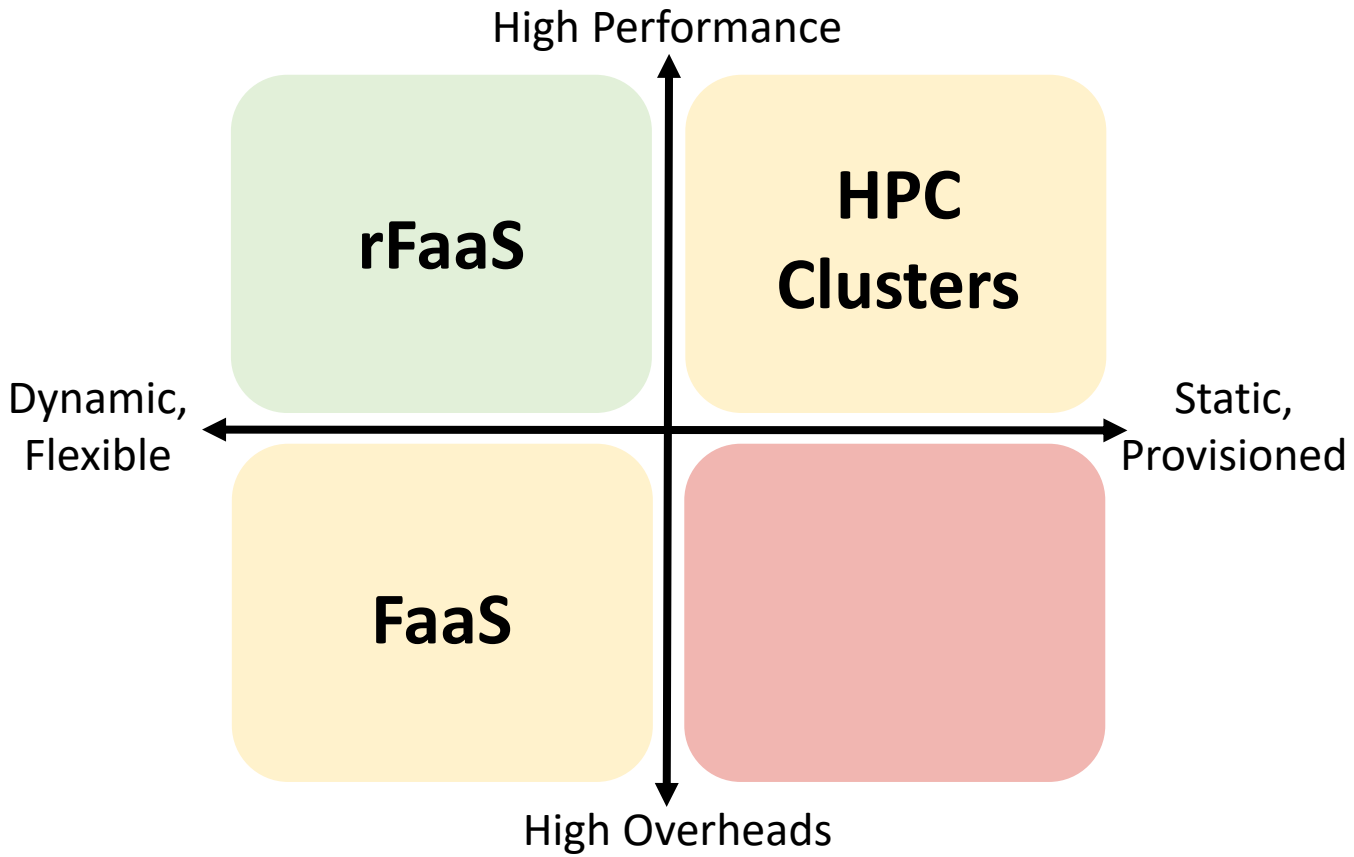
Function-as-a-Service for HPC



Function-as-a-Service for HPC

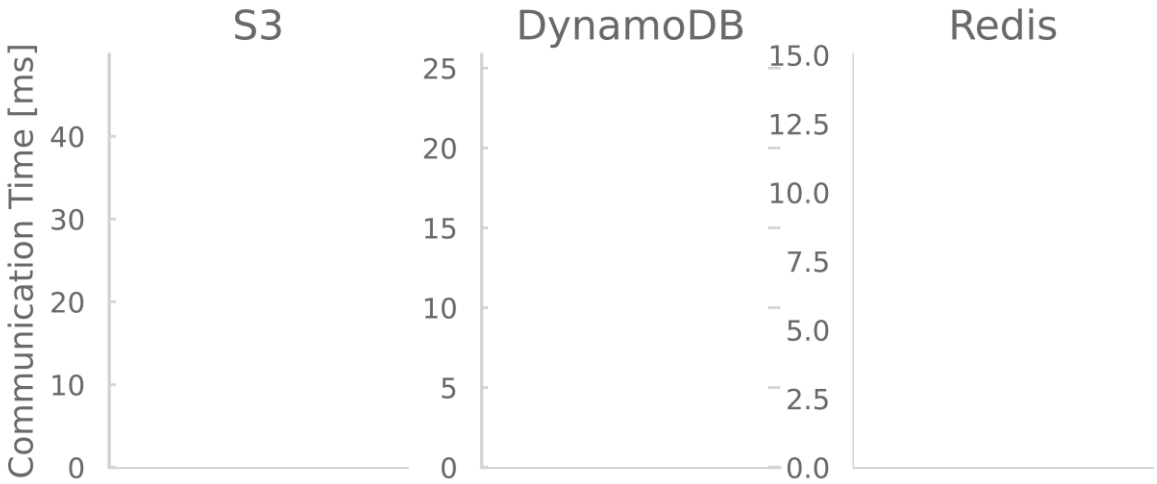
Reduced invocation critical path

Zero-copy RDMA

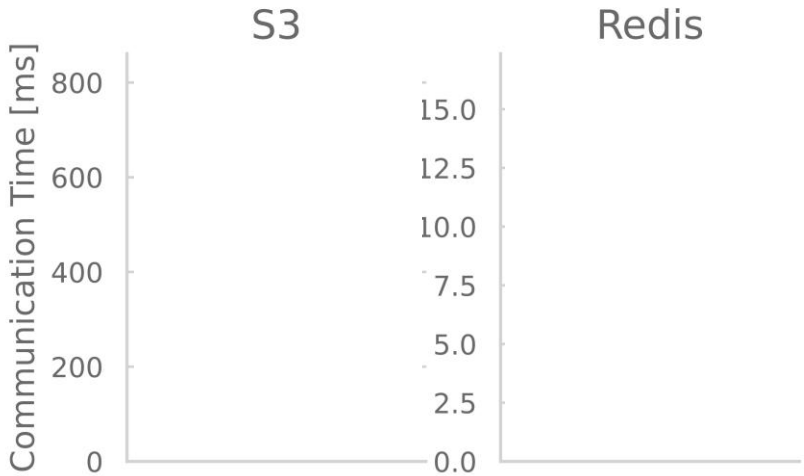


Serverless Applications: Communication

1 B



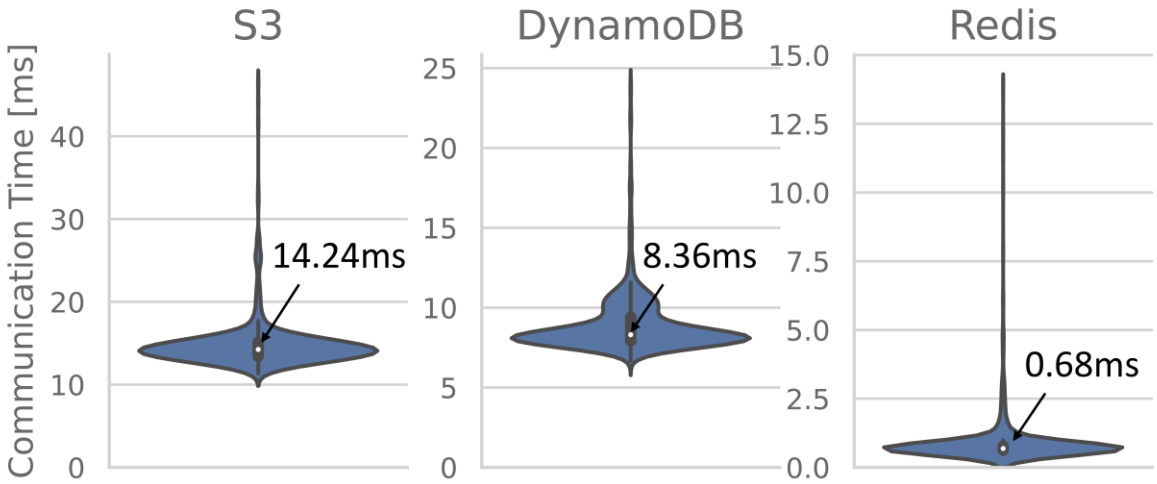
1 MB



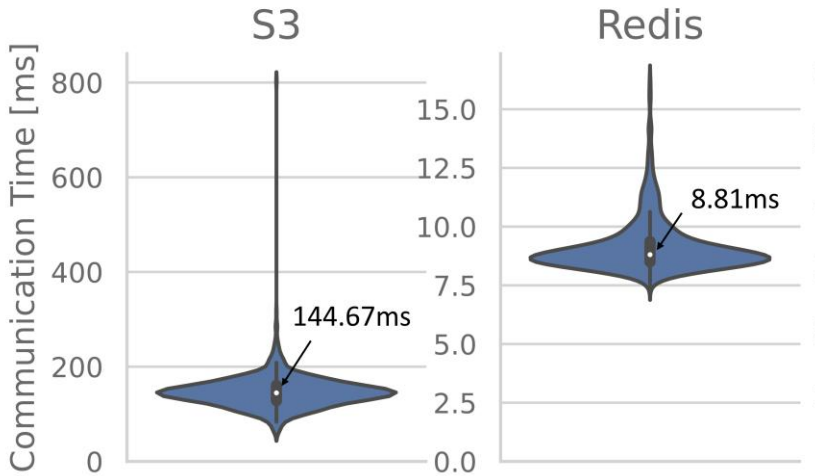
"FMI: Fast and Cheap Message Passing for Serverless Functions", ICS'23

Serverless Applications: Communication

1 B



1 MB



“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23