Marcin Copik, Alexandru Calotoiu, Torsten Hoefler, Michał Podstawski, Laurin Brandner, Larissa Schmid, Nico Graf, Grzegorz Kwaśniewski, Kacper Janda, Mateusz Knapik, Jakub Czerski, Mahla Sarifi, Paweł Żuk, Sascha Kehrli, Abhishek Kumar, Prajin Khadka, Horia Mercan, and many others!

Demystifying Serverless Performance: A Hands-on Tutorial with Serverless Benchmark Suite SeBS

HiPEAC 2026
Cracow, Poland

Agenda

SeBS Slack

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ **Part II**
  - ✓ **Communication and Data**
  - ✓ **Serverless Workflows**
  - ✓ **Experiments in SeBS**
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ **Part II**
  - ✓ **Communication and Data**
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

SeBS Slack

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

From Cloud to the Edge

From Cloud to the Edge

Introducing Cloudflare Workers: Run JavaScript Service Workers at the Edge

2017-09-29

From Cloud to the Edge

Introducing Cloudflare Workers:

Ru

th

2017-0

Bringing Python to Workers using Pyodide and WebAssembly

2024-04-02

From Cloud to the Edge

Introducing Cloudflare Workers:
Ru
th
2017-0

Bringing Python to Workers using
Pyo
2024-04-02

Python Workers redux: fast cold starts, packages, and a uv-first workflow
2025-12-08

From Cloud to the Edge

Introducing Cloudflare Workers:

Ru

th

2017-0

Bringing Python to Workers using

Pyo

Python Workers redux: fast cold

-first

Simple, scalable, and global:

Containers are coming to

Cloudflare Workers in June 2025

2025-04-11

From Cloud to the Edge

Cloudflare Workers

From Cloud to the Edge

		Cloudflare Workers
Runtime	Container or microVM.	V8 Isolates.

From Cloud to the Edge

		Cloudflare Workers
Runtime	Container or microVM.	V8 Isolates.
Languages	Many, interpreted and compiled.	Node.js and Python, compiled to WASM.

From Cloud to the Edge

		Cloudflare Workers
Runtime	Container or microVM.	V8 Isolates.
Languages	Many, interpreted and compiled.	Node.js and Python, compiled to WASM.
Language Compatibility	Almost complete.	Partial.

From Cloud to the Edge

		Cloudflare Workers
Runtime	Container or microVM.	V8 Isolates.
Languages	Many, interpreted and compiled.	Node.js and Python, compiled to WASM.
Language Compatibility	Almost complete.	Partial.
Resource Allocation	Elastic, powerful instances.	128 MB memory, single thread.

From Cloud to the Edge

		Cloudflare Workers
Runtime	Container or microVM.	V8 Isolates.
Languages	Many, interpreted and compiled.	Node.js and Python, compiled to WASM.
Language Compatibility	Almost complete.	Partial.
Resource Allocation	Elastic, powerful instances.	128 MB memory, single thread.
Execution Limits	10-15+ minutes	10 ms / 5 minutes CPU time

From Cloud to the Edge: SeBS Compatibility

Sebs Benchmarks Cloudflare Compatibility

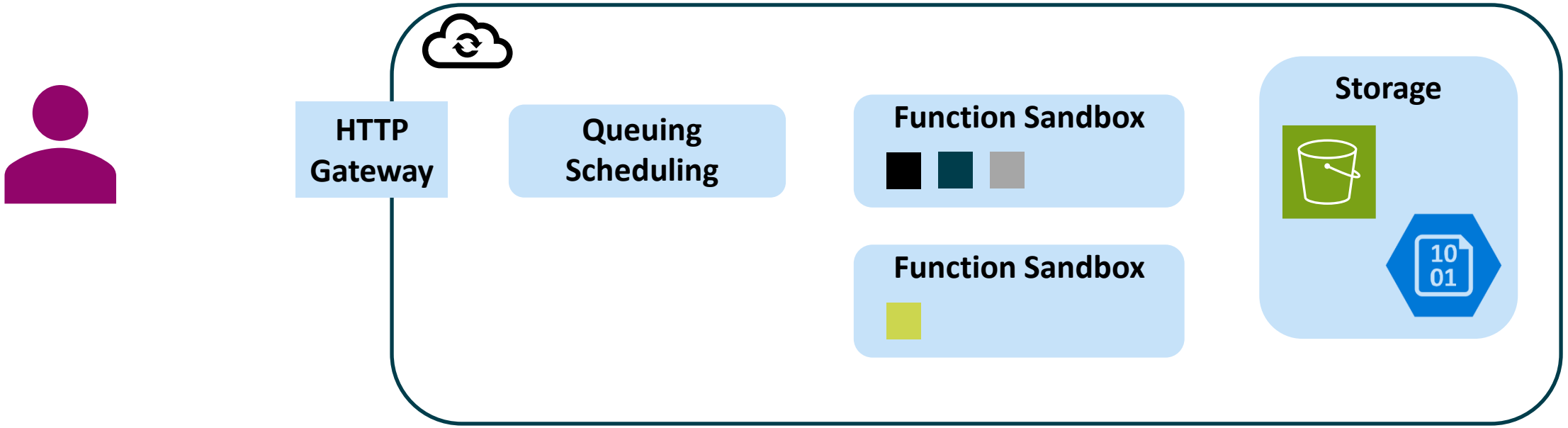
could be polyfilled could be translated not possible

	110	120	130	210	220	311	411	501	502	503	504
Nodejs Worker	✓	✓	✓	sharp n/a	sub-process n/a & too big	✓	pytorch n/a & too big	igraph n/a	igraph n/a	igraph n/a	too big
Python Worker	✓	✓	✓	✓	too big	✓	too big	✓	✓	✓	too big
Nodejs Container	✓	✓	✓	✓	Not Imple-mented	✓	pytorch n/a	igraph n/a	igraph n/a	igraph n/a	igraph n/a
Python Container	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

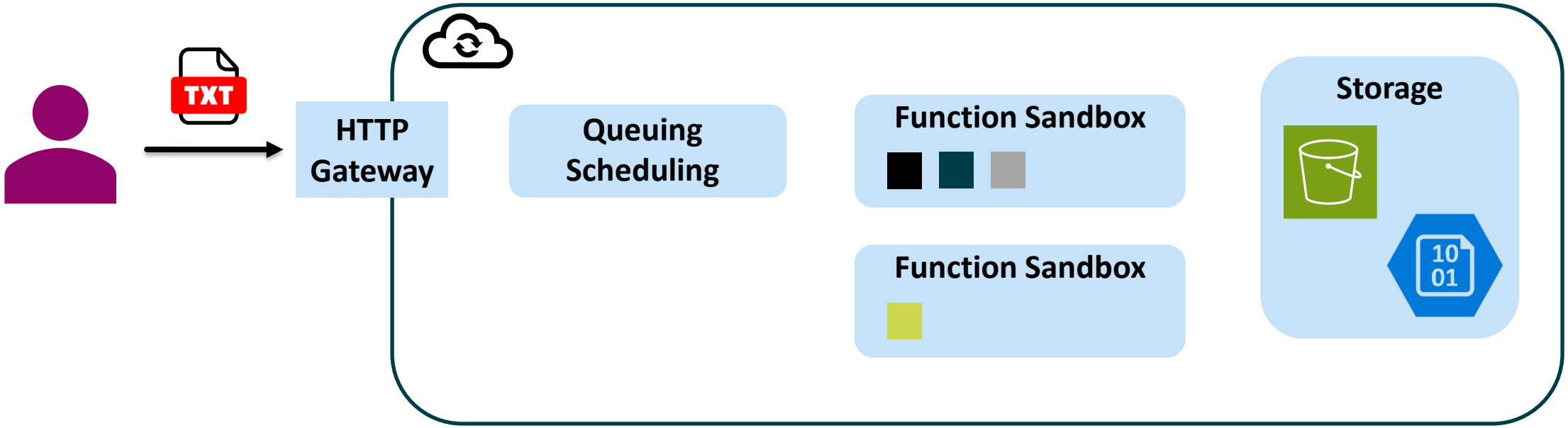
11

Project from ETH students Livio D'Agostini, Laurin Jahns, and Marouan Mbarek.

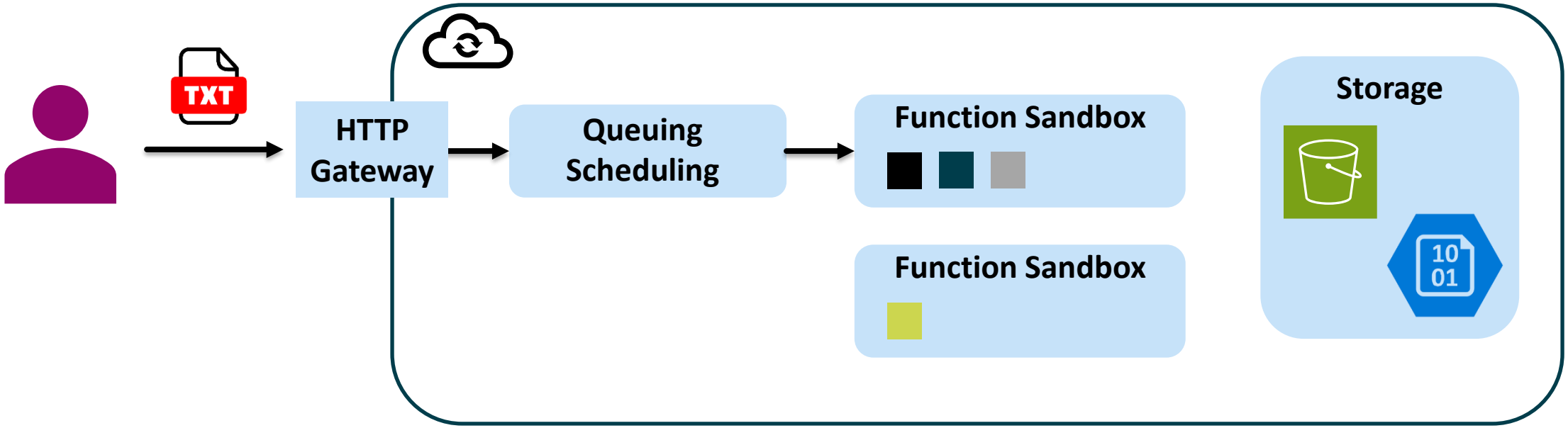
LaTeX Microservice in Serverless



LaTeX Microservice in Serverless

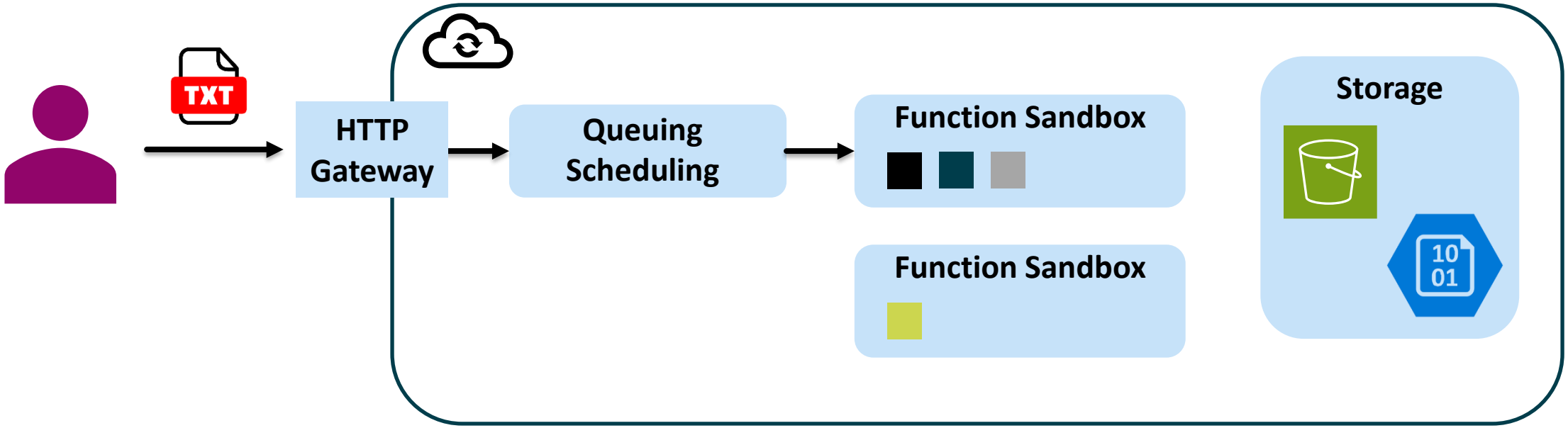


LaTeX Microservice in Serverless



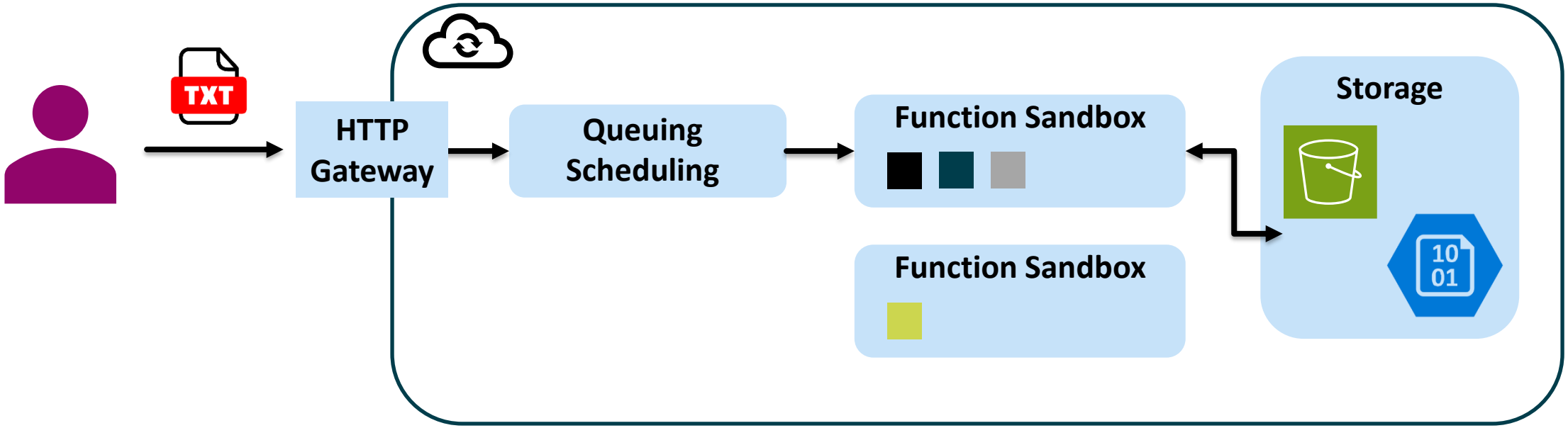
LaTeX Microservice in Serverless

Is cache up to date?
Did other function edit file of this client?

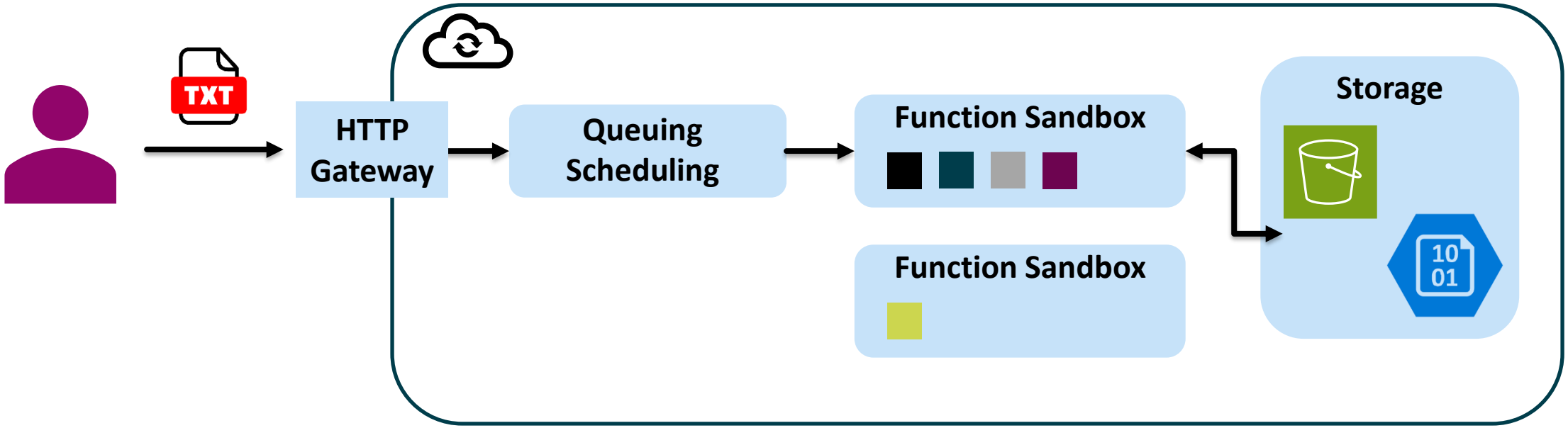


LaTeX Microservice in Serverless

Is cache up to date?
Did other function edit file of this client?



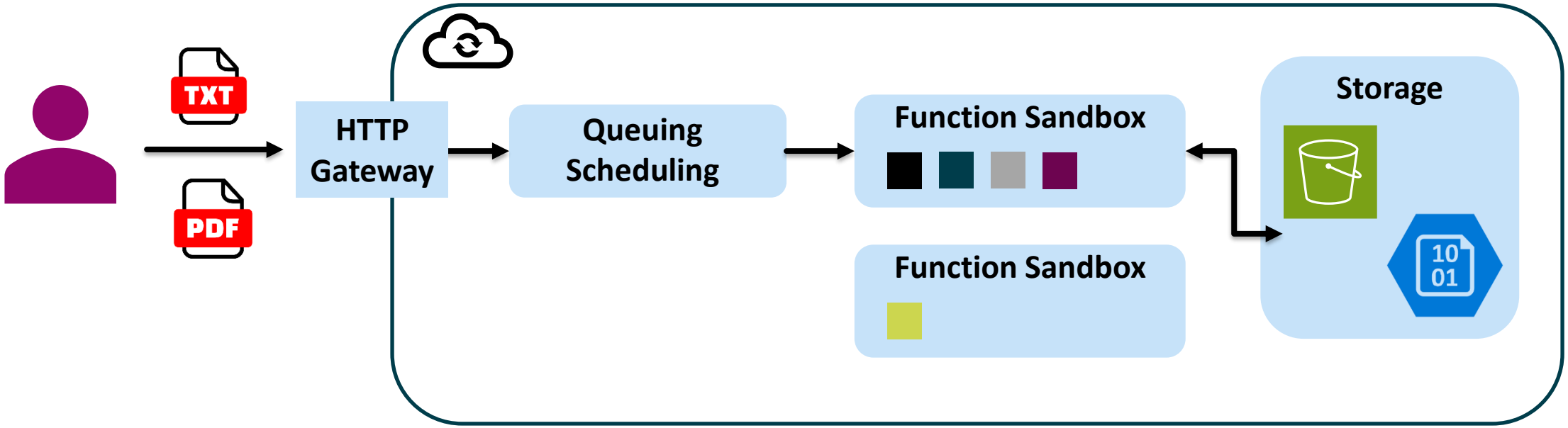
LaTeX Microservice in Serverless



Serverless Service

- Cached function state is not reliable

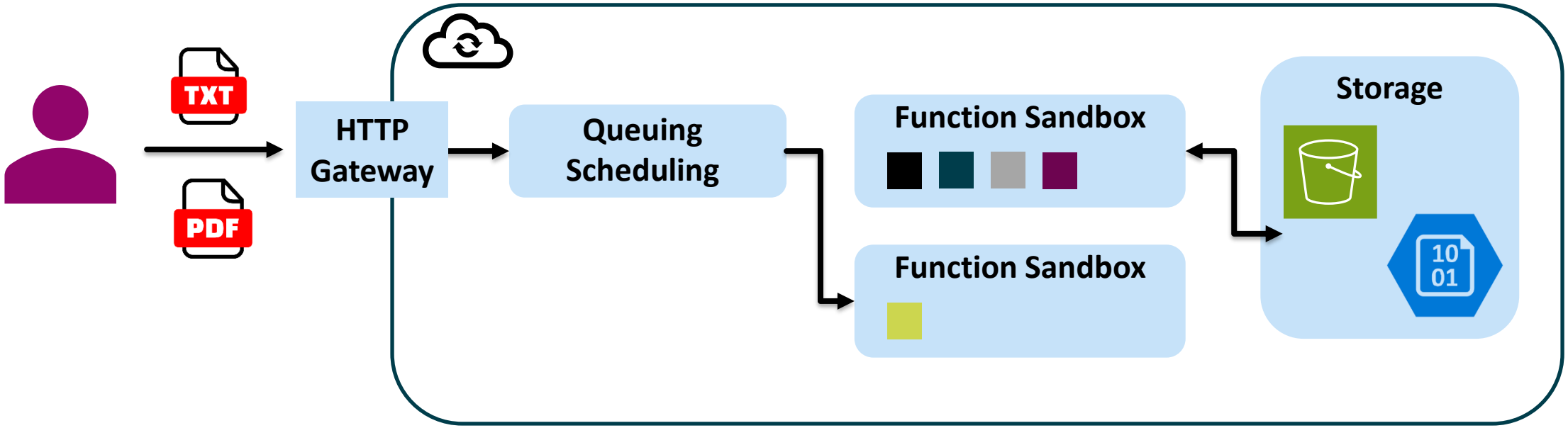
LaTeX Microservice in Serverless



Serverless Service

- Cached function state is not reliable

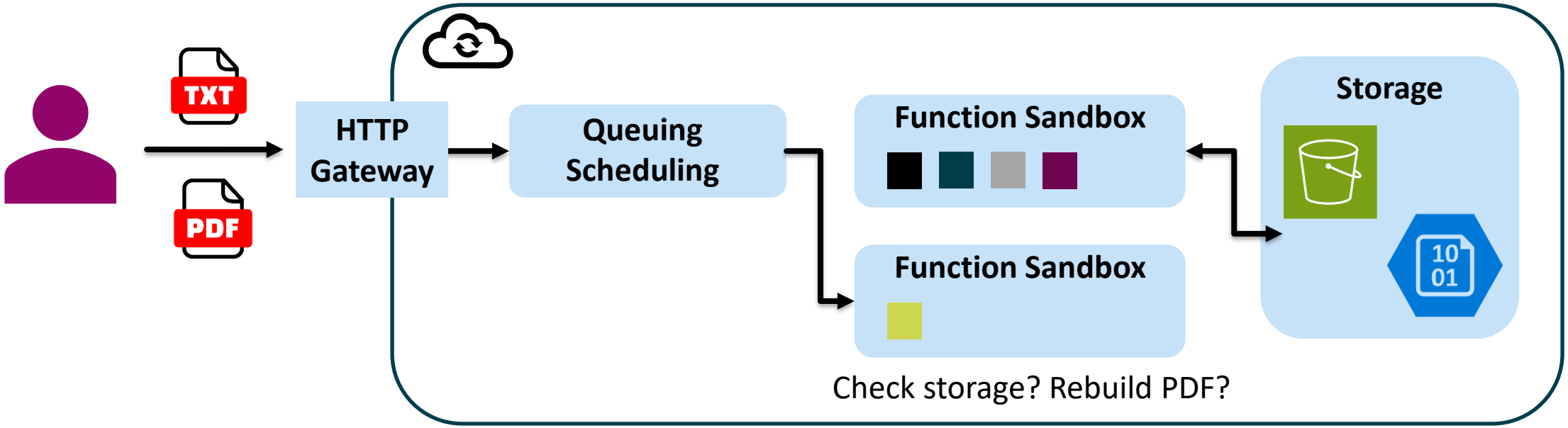
LaTeX Microservice in Serverless



Serverless Service

- Cached function state is not reliable

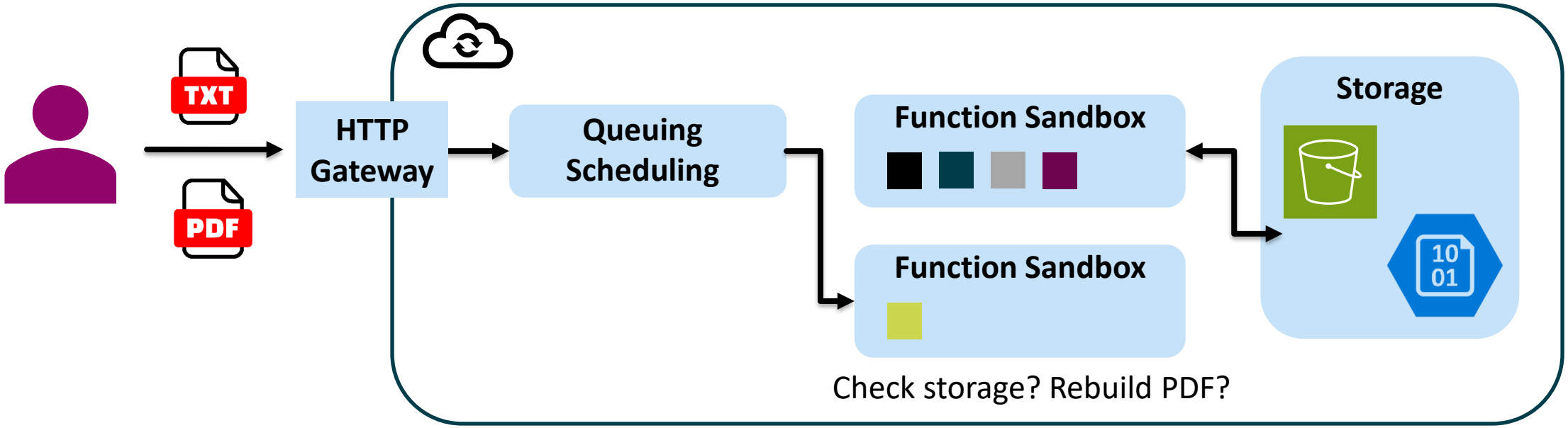
LaTeX Microservice in Serverless



Serverless Service

- Cached function state is not reliable

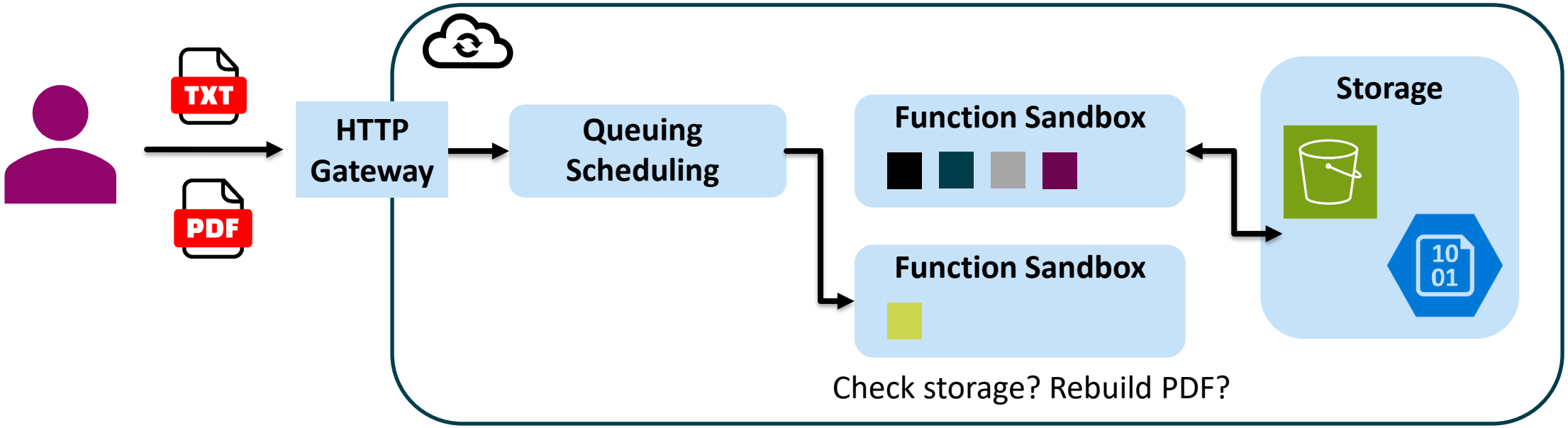
LaTeX Microservice in Serverless



Serverless Service

- ✗ Cached function state is not reliable
- ✗ Request context has no influence on scheduling

LaTeX Microservice in Serverless



Serverless Service

- Cached function state is not reliable
- Request context has no influence on scheduling

Standard Microservice

- Extensive caching
- Load balancing with sticky sessions

From Serverless Functions to Applications

From Serverless Functions to Applications

Workflows

Offered by the cloud provider.

From Serverless Functions to Applications

Workflows

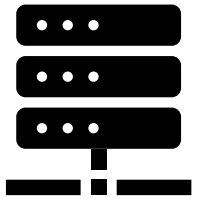
Offered by the cloud provider.

“Applications”

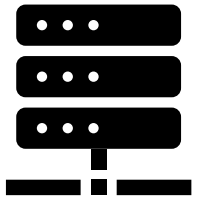
No such product in the cloud.

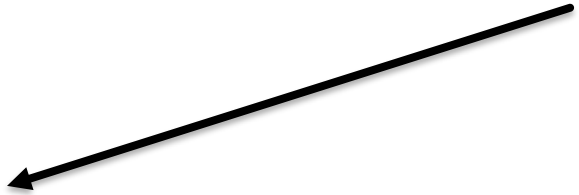
Combine functions, storage, queues, caches, containers and VMs.

Serverless Applications: Disaggregate



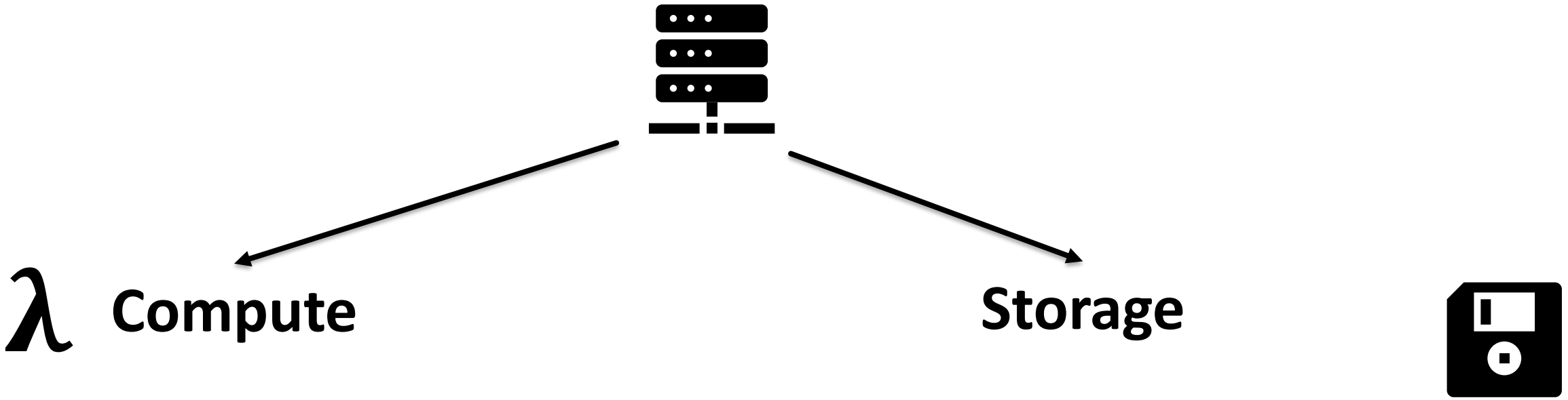
Serverless Applications: Disaggregate



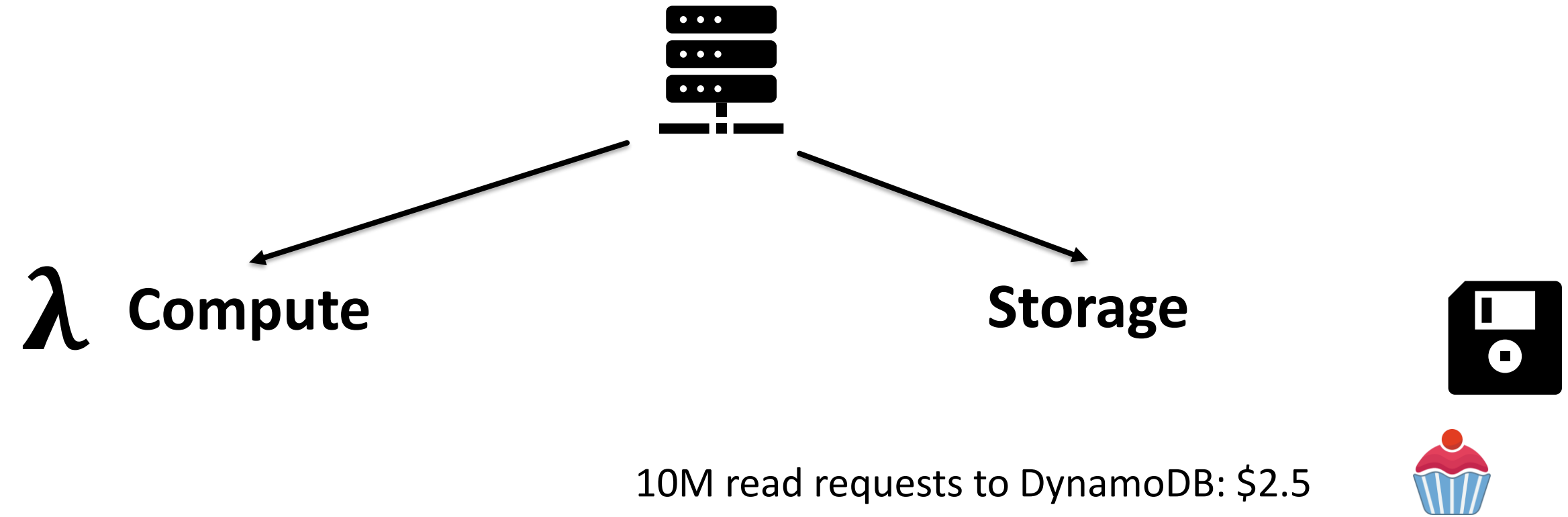


λ Compute

Serverless Applications: Disaggregate

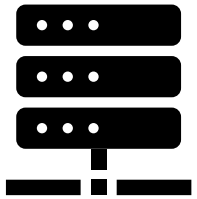


Serverless Applications: Disaggregate



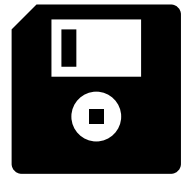
10M read requests to DynamoDB: $2.5

Serverless Applications: Disaggregate



λ Compute

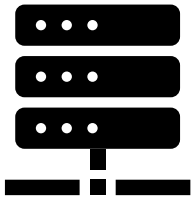
Storage



10M read requests to DynamoDB: $2.5

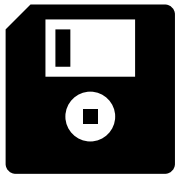
10M read requests to S3: $4

Serverless Applications: Disaggregate



λ Compute

Storage



10M read requests to DynamoDB: $2.5

10M read requests to S3: $4

10M Lambda invocations for 30 ms: $10

Serverless Applications: From Design to the Cloud

System Concept
Functions
Object Storage
Key-Value Storage
Concurrency Primitives
Queue

“FaaSKeeper: Learning from Building Serverless Services with ZooKeeper as an Example”, HPDC’24

Serverless Applications: From Design to the Cloud

System Concept	AWS
Functions	Lambda
Object Storage	S3
Key-Value Storage	DynamoDB
Concurrency Primitives	Update Expressions
Queue	SQS

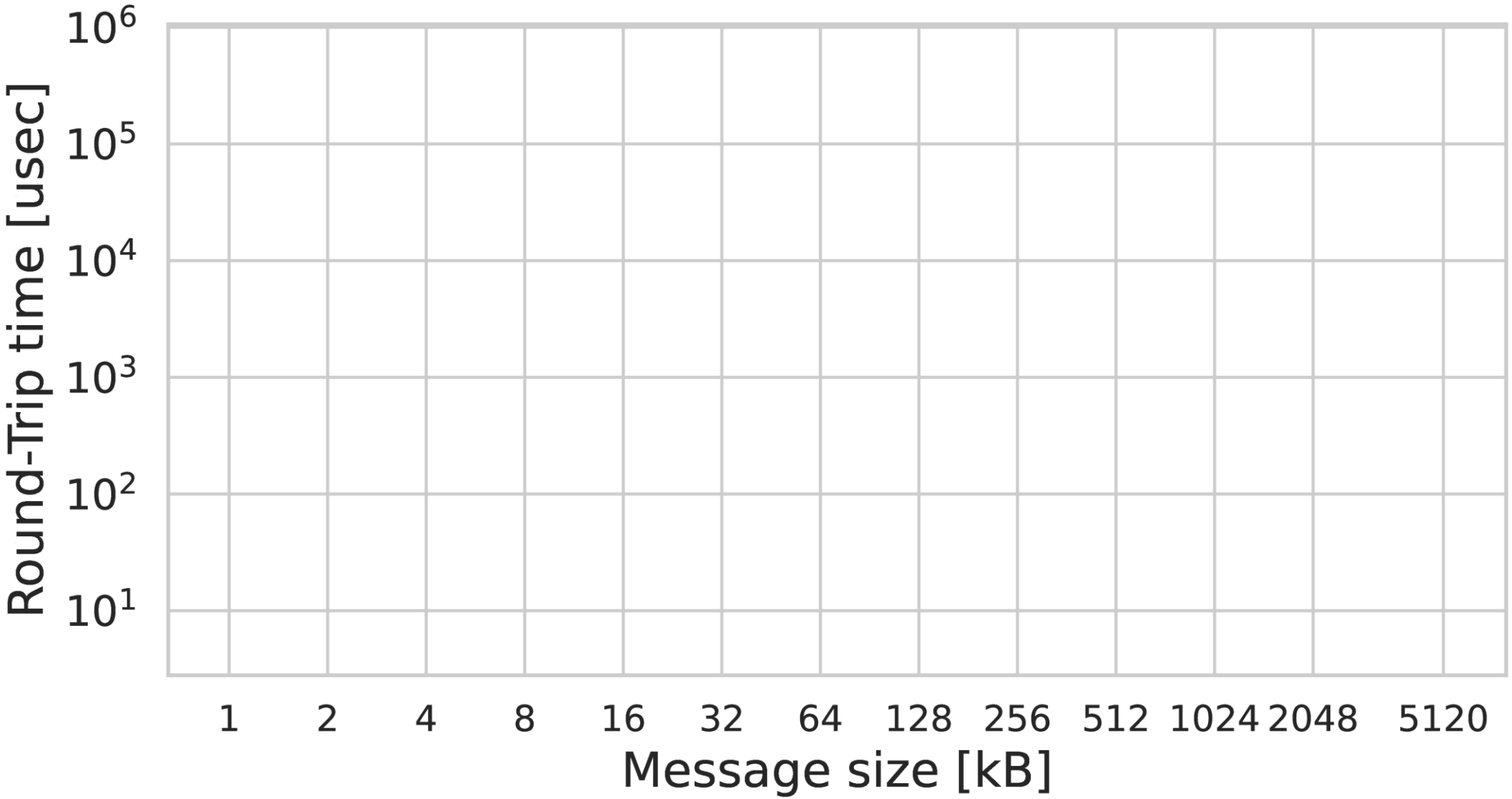
“FaaSKeeper: Learning from Building Serverless Services with ZooKeeper as an Example”, HPDC’24

Serverless Applications: From Design to the Cloud

System Concept	AWS	Google Cloud
Functions	Lambda	Cloud Function
Object Storage	S3	Storage
Key-Value Storage	DynamoDB	Datastore
Concurrency Primitives	Update Expressions	Transactions
Queue	SQS	Pub/Sub

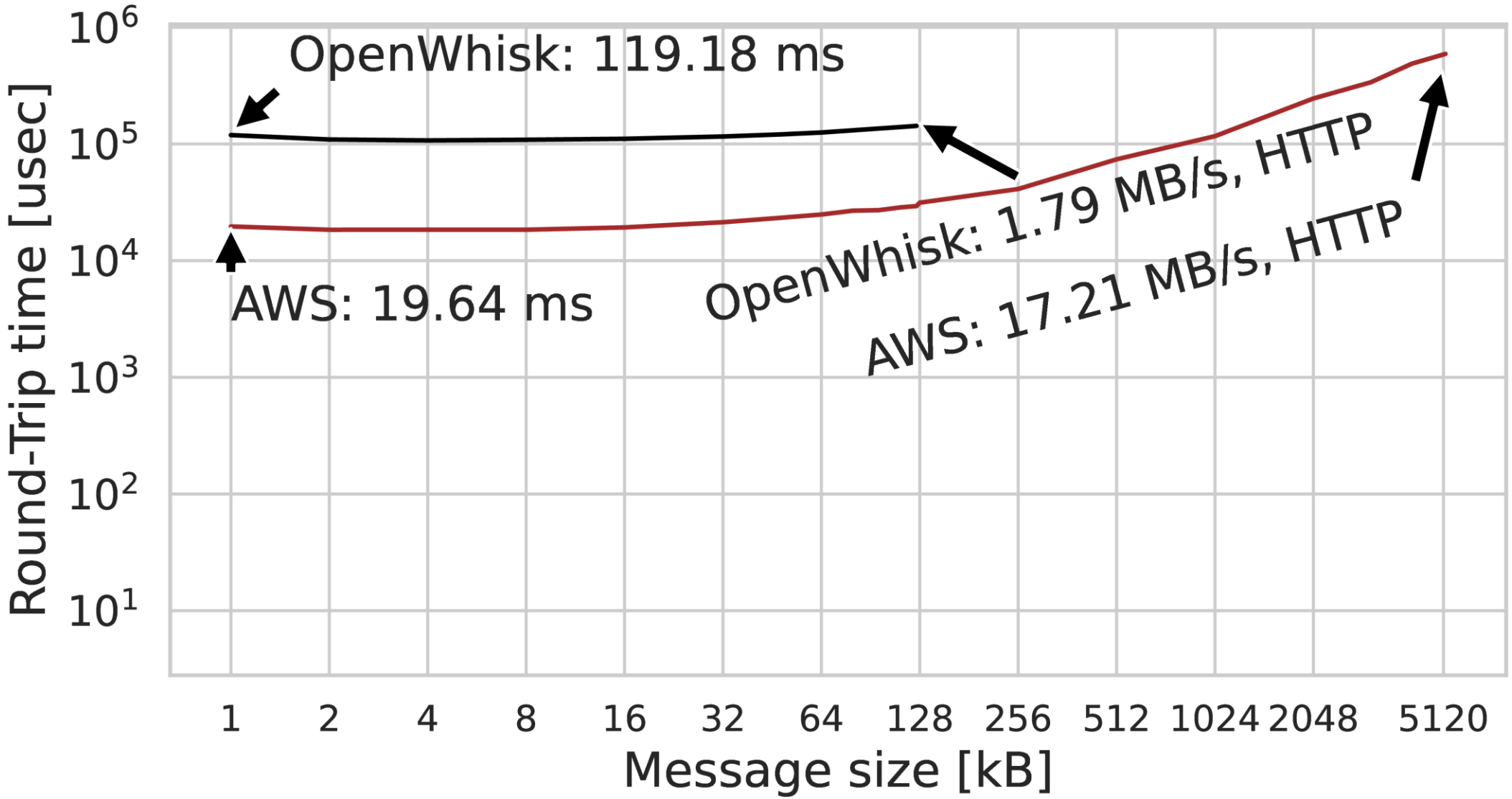
“FaaSKeeper: Learning from Building Serverless Services with ZooKeeper as an Example”, HPDC’24

Serverless Applications: Platform Overheads



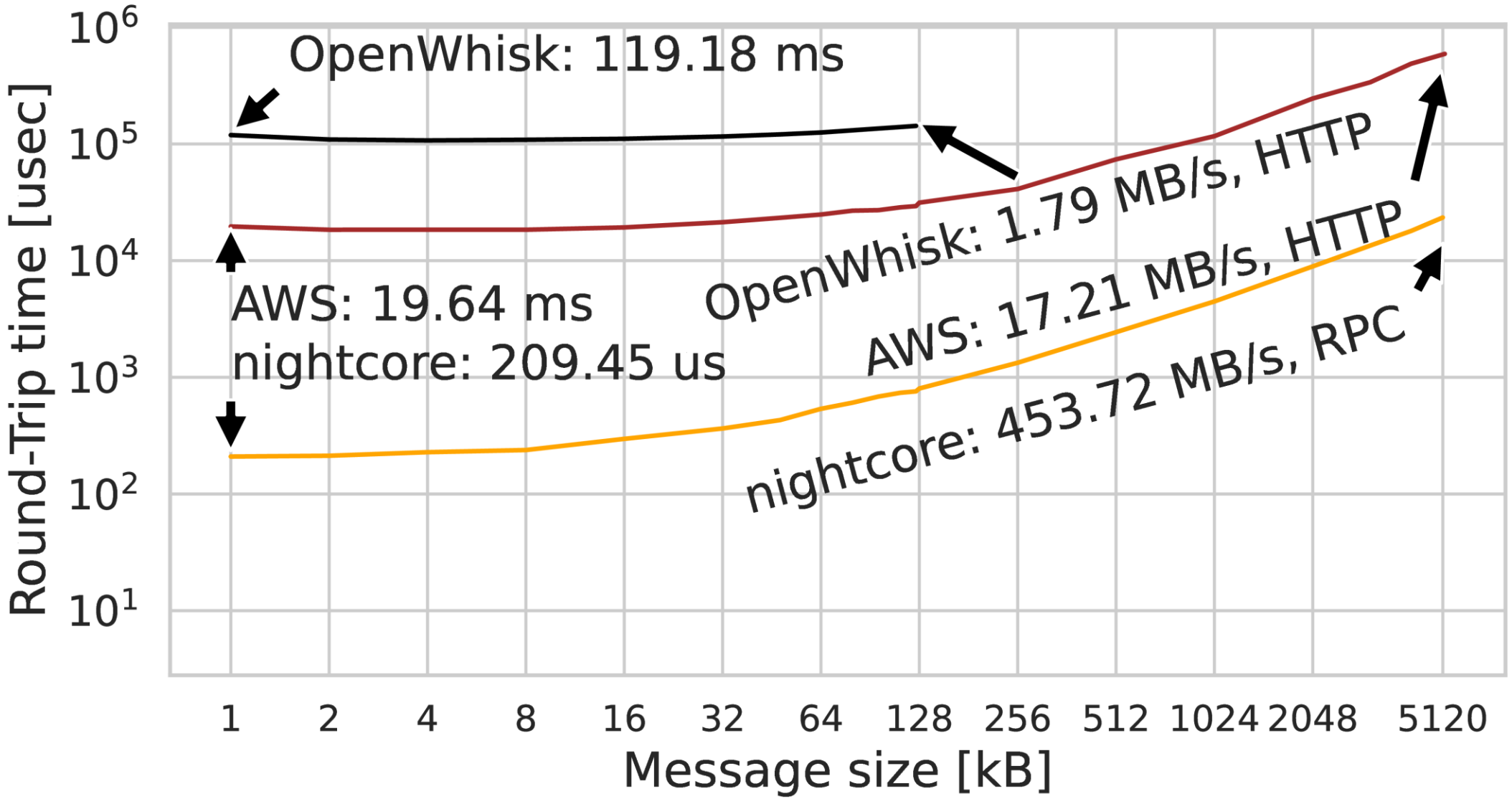
“rFaaS: Enabling High Performance Serverless with RDMA and Leases”, IPDPS’23

Serverless Applications: Platform Overheads



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

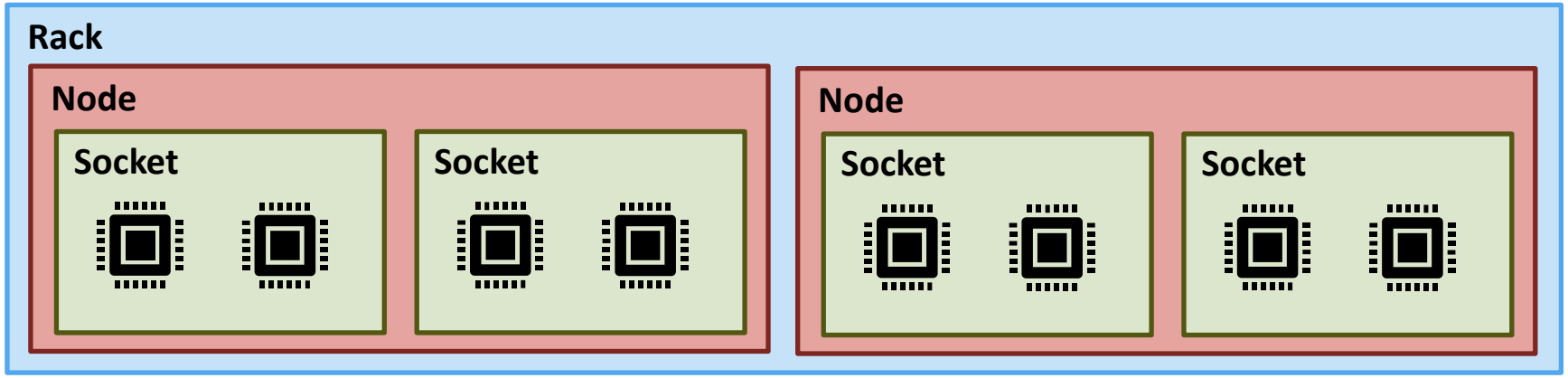
Serverless Applications: Platform Overheads



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

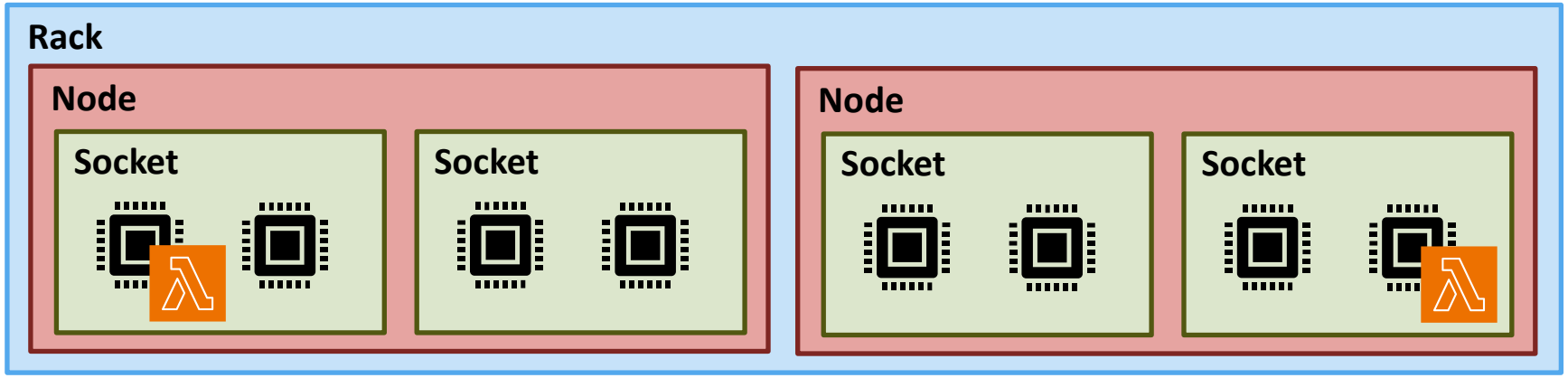
Serverless Applications: Communication

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



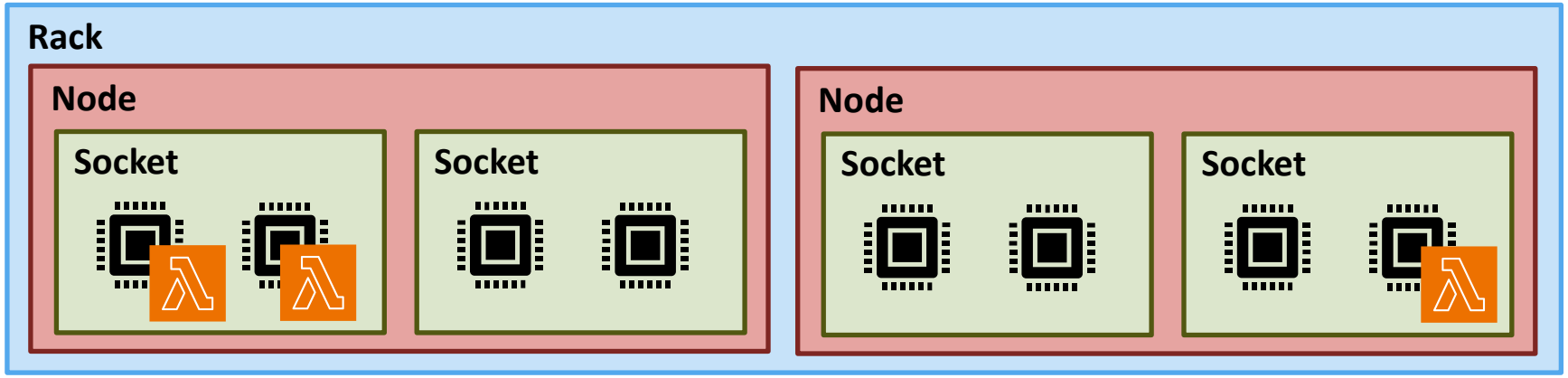
Serverless Applications: Communication

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



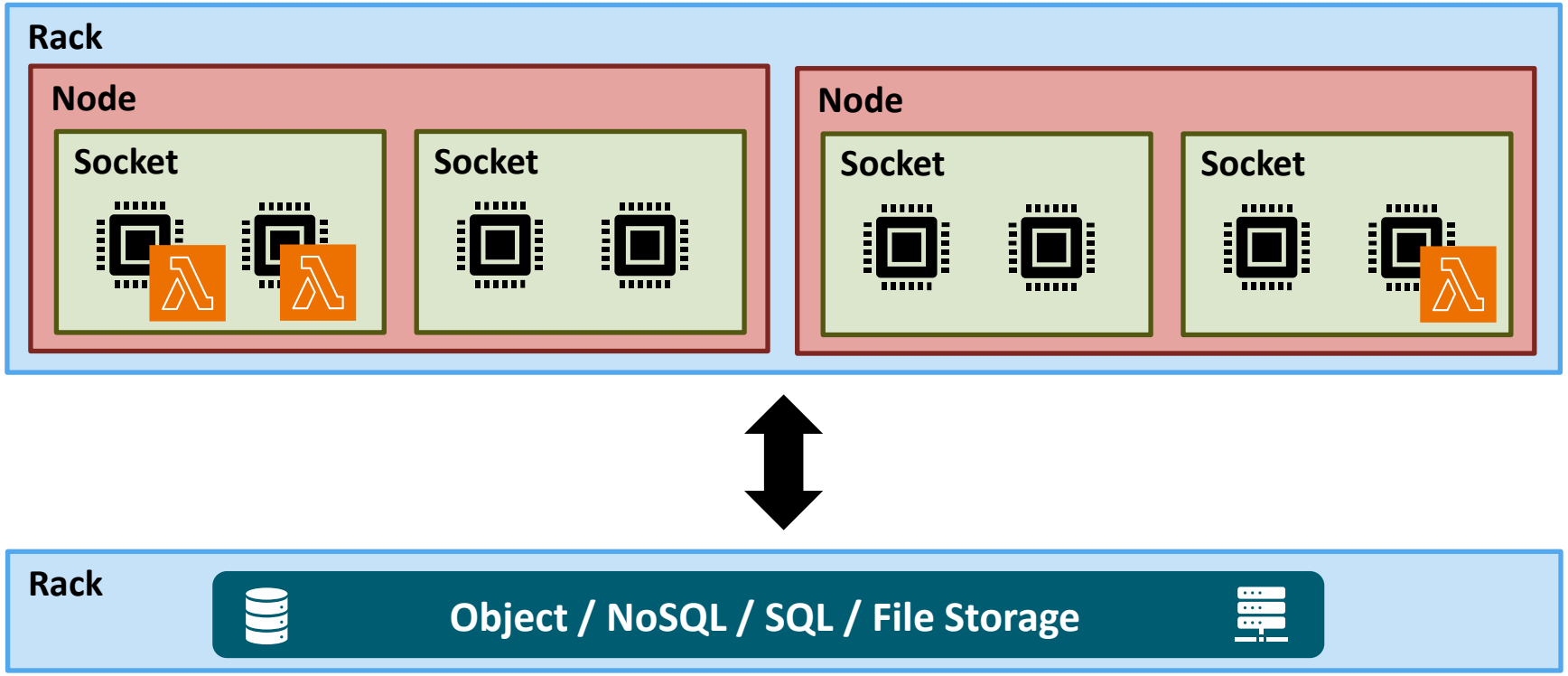
Serverless Applications: Communication

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



Serverless Applications: Communication

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



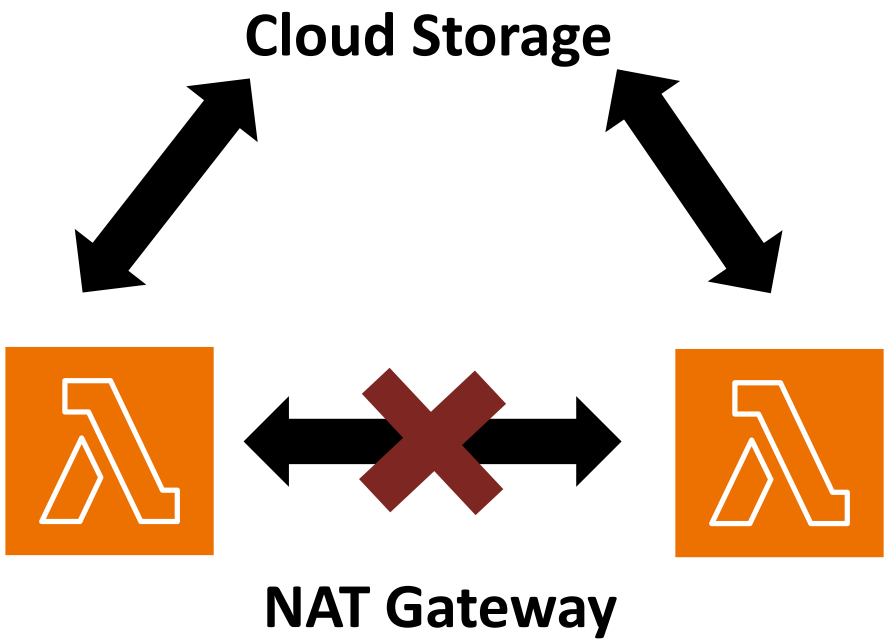
Serverless Applications: Communication

Serverless Applications: Communication

Serverless Applications: Communication

NAT Gateway

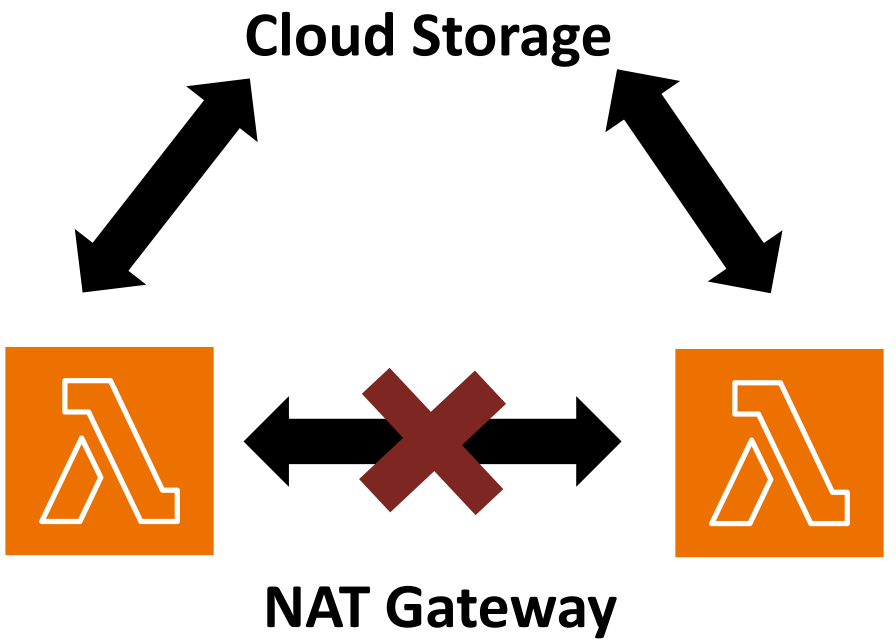
Serverless Applications: Communication



Serverless Applications: Communication

High Latency
For Small Messages

S3



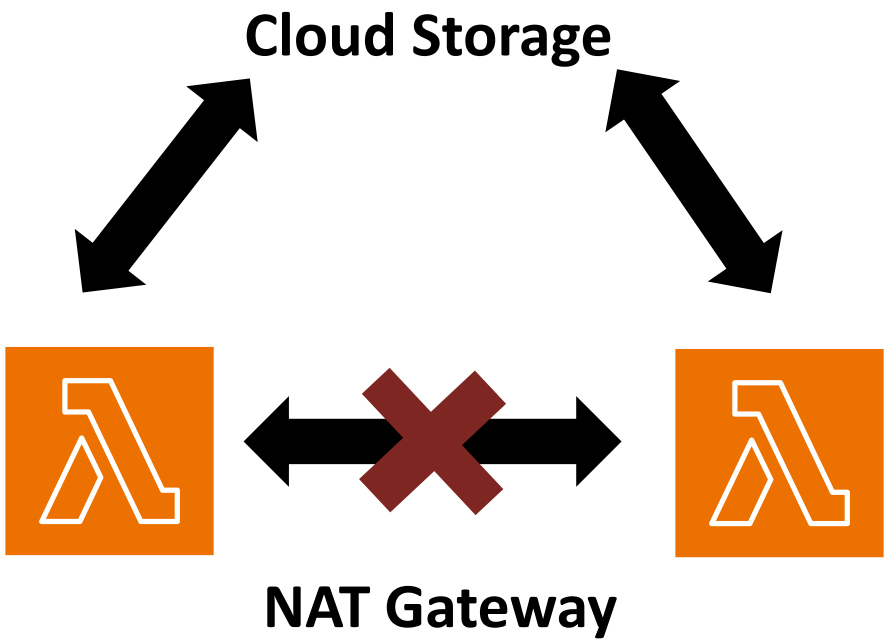
Serverless Applications: Communication

High Latency
For Small Messages

S3

Expensive for
Large Messages

DynamoDB



Serverless Applications: Communication

High Latency
For Small Messages

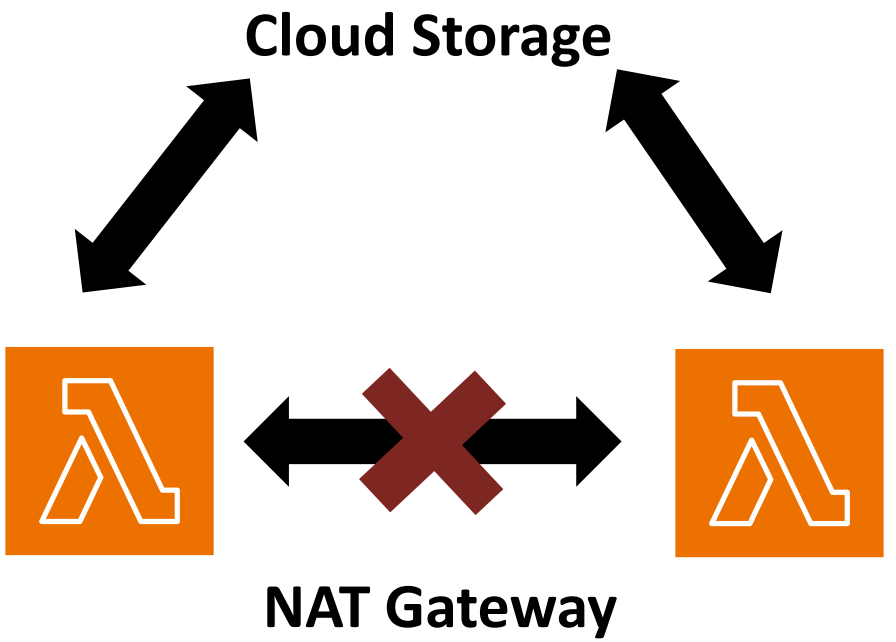
S3

Expensive for
Large Messages

DynamoDB

Not Serverless
Expensive

Redis



Serverless Applications: Communication

High Latency
For Small Messages

S3

Expensive for
Large Messages

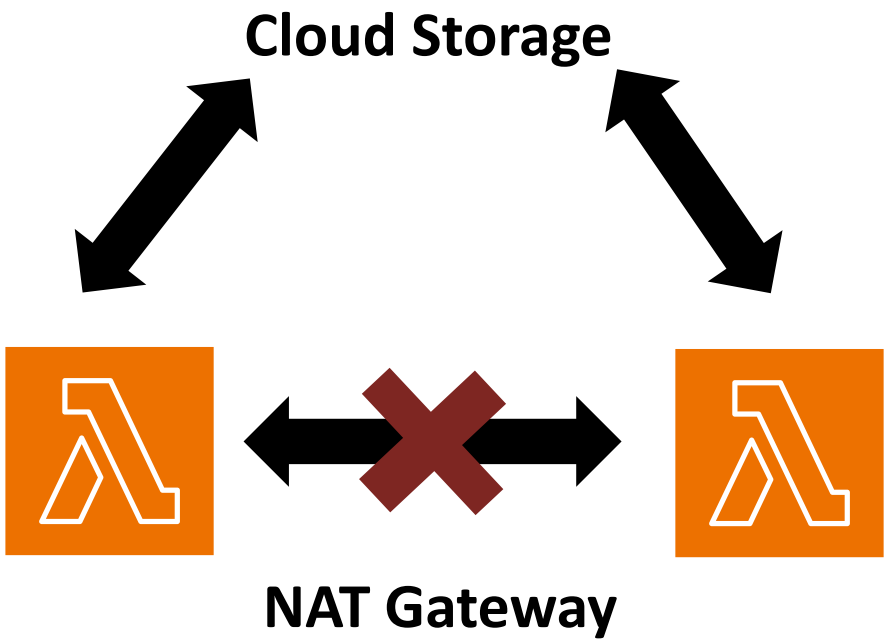
DynamoDB

Not Serverless
Expensive

Redis

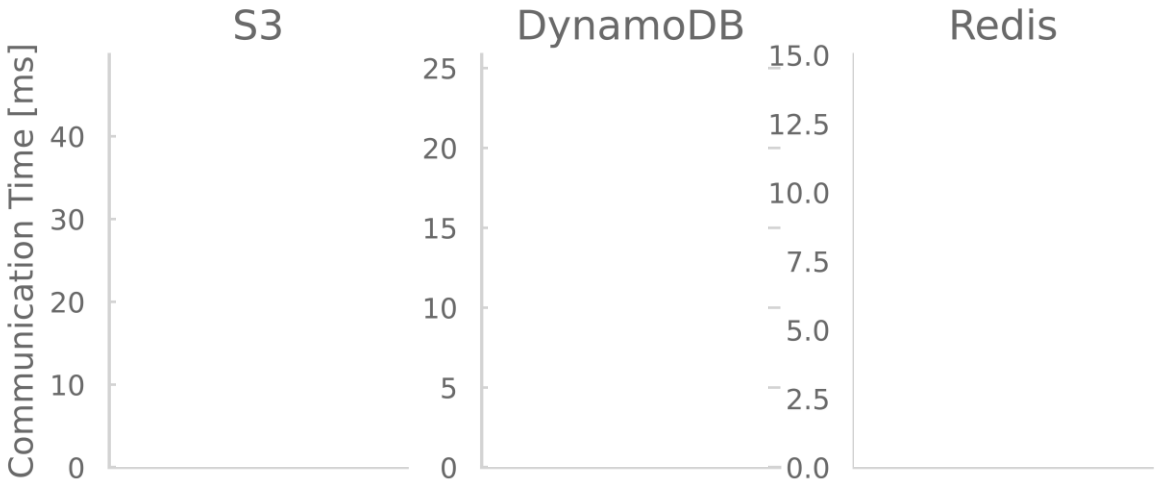
Size Limits
Good Latency

SQS

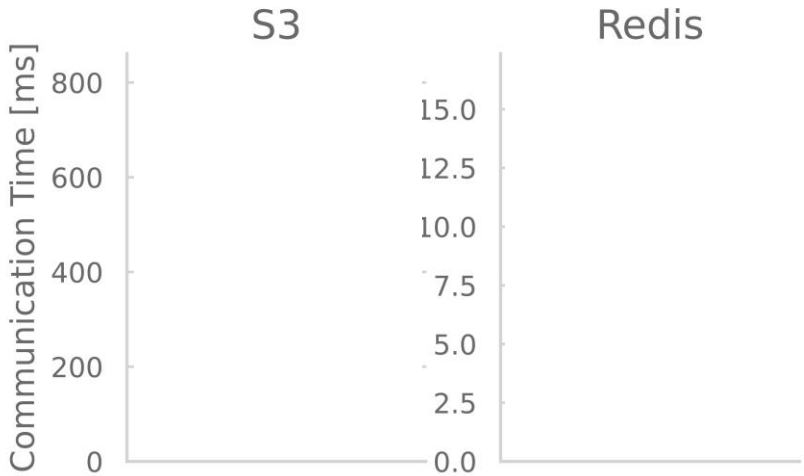


Serverless Applications: Communication

1 B



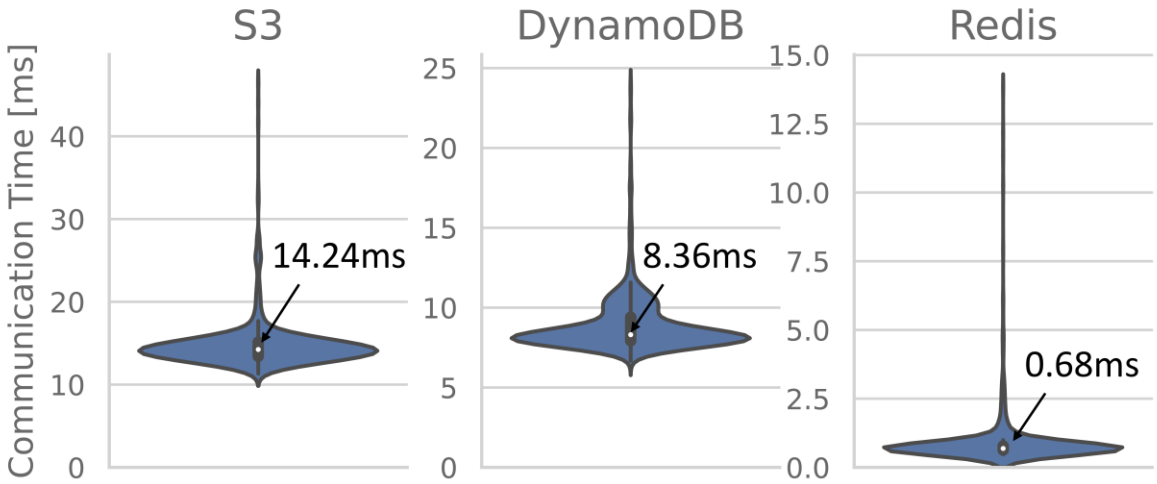
1 MB



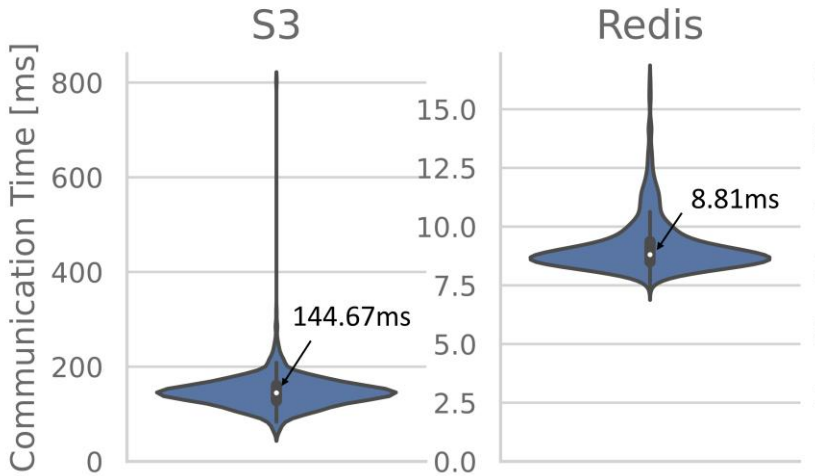
“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

Serverless Applications: Communication

1 B



1 MB



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

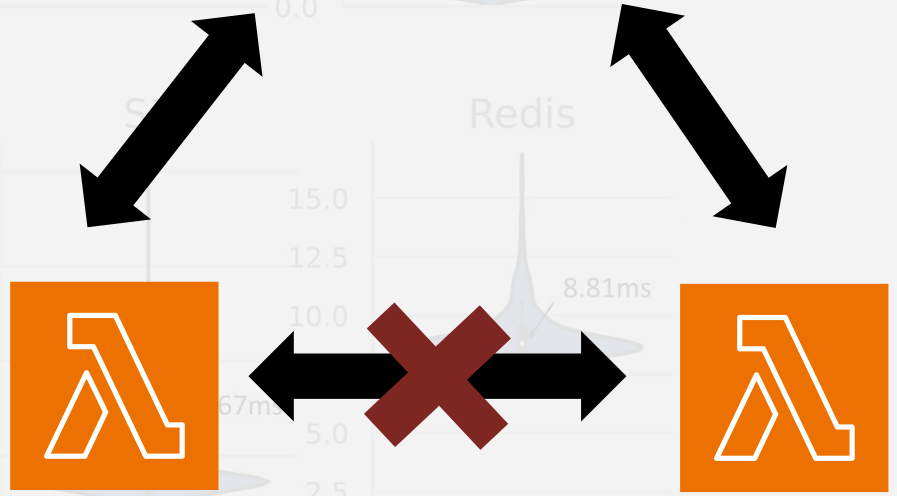
Serverless Applications: Communication

Object Storage
Example: AWS S3

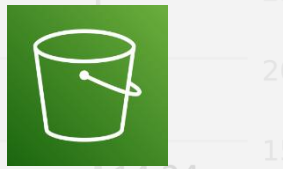
Key-Value Storage
Example: AWS DynamoDB

In-Memory Storage
Example: Redis

Queue
Example: AWS SQS



Serverless Applications: Communication

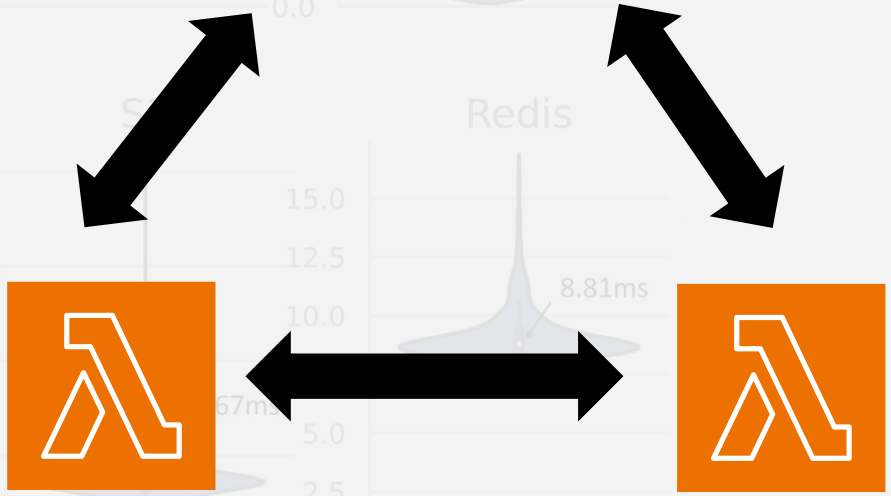


Object Storage
Example: AWS S3

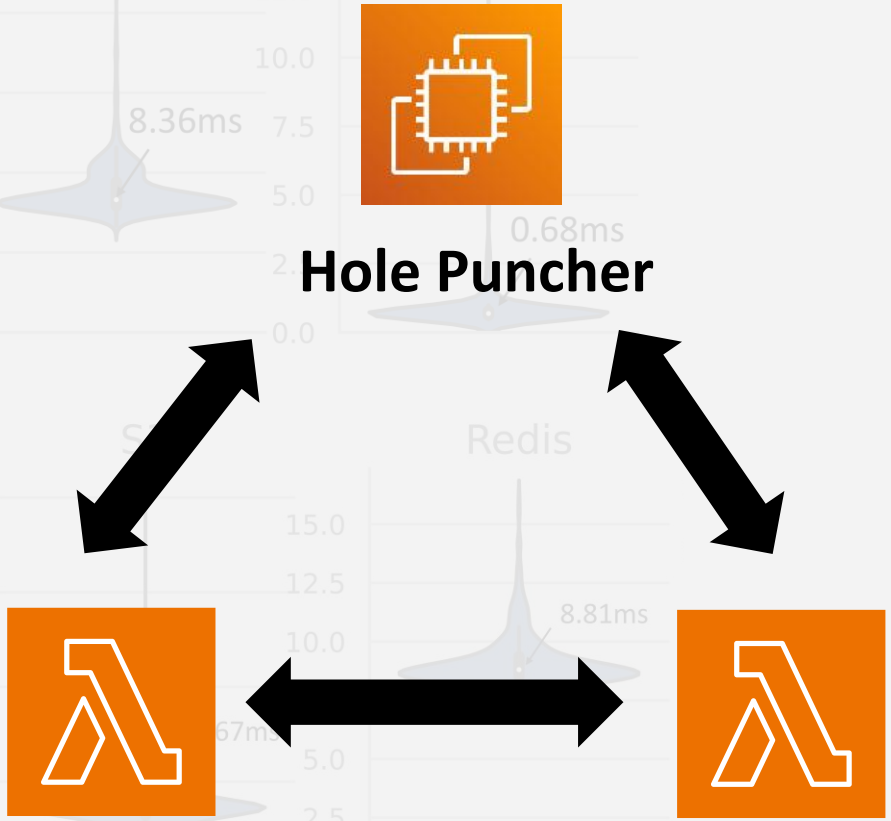
Key-Value Storage
Example: AWS DynamoDB

In-Memory Storage
Example: Redis

Queue
Example: AWS SQS



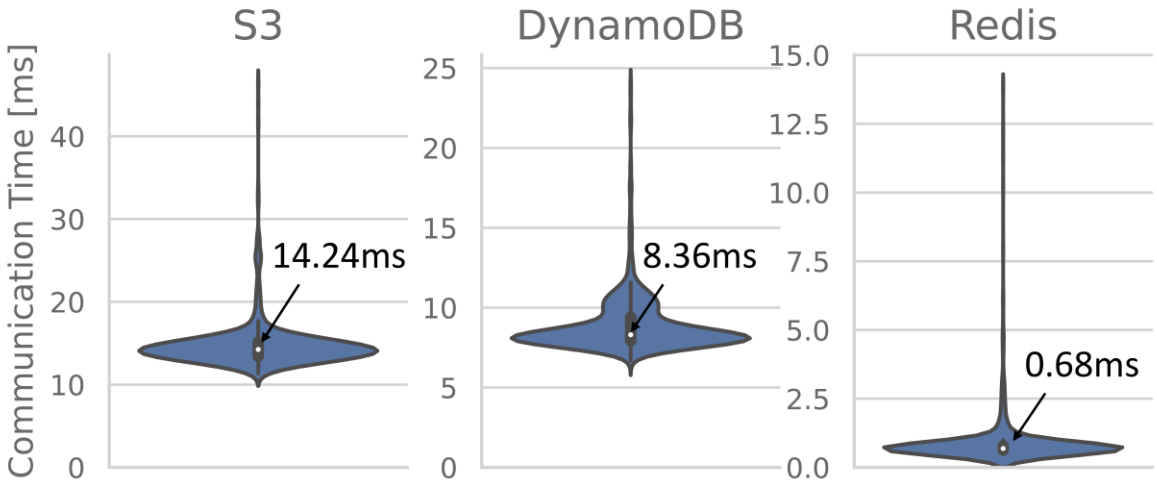
Serverless Applications: Communication



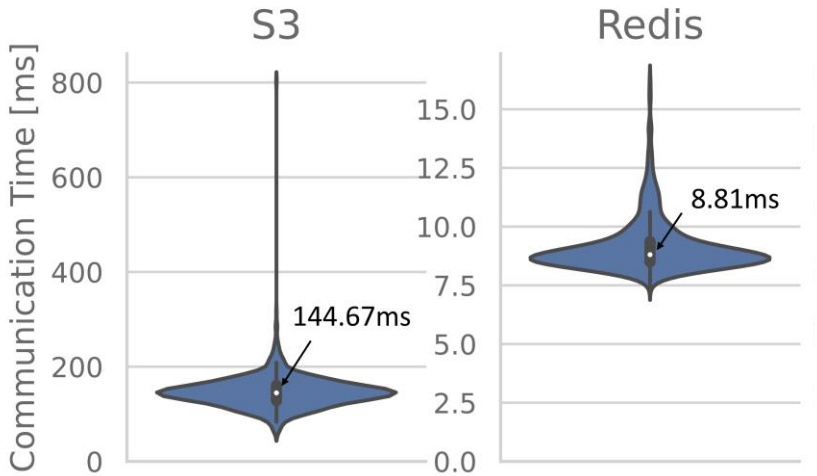
“FMI: Fast and Cheap Message Passing for Serverless Functions”, ICS’23

Serverless Applications: Communication

1 B



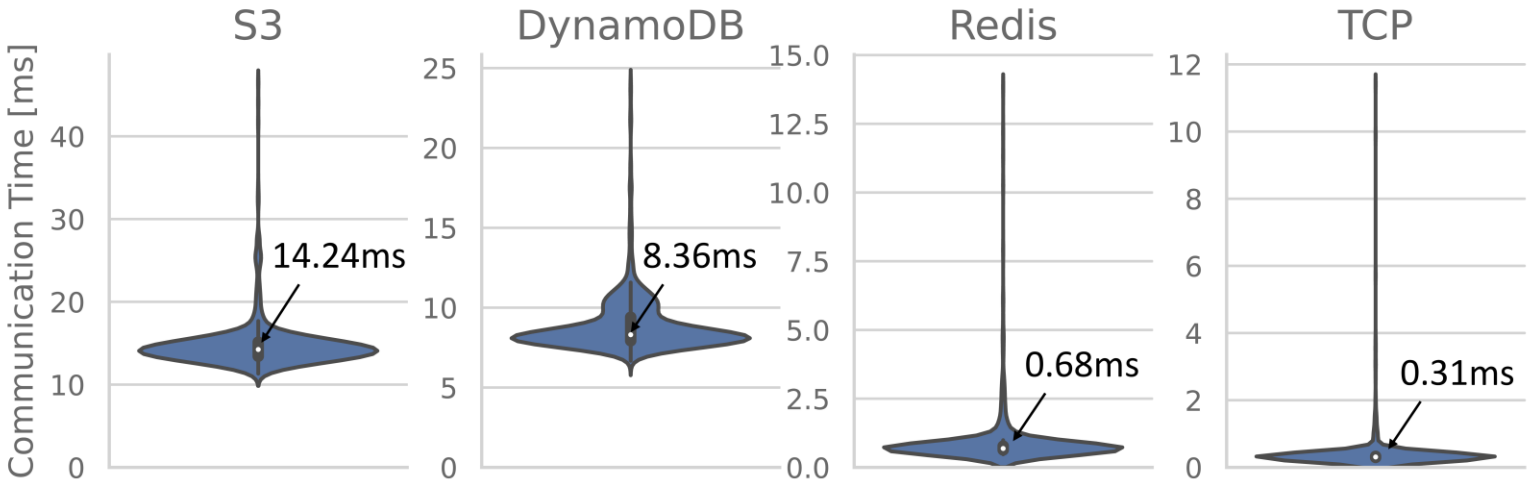
1 MB



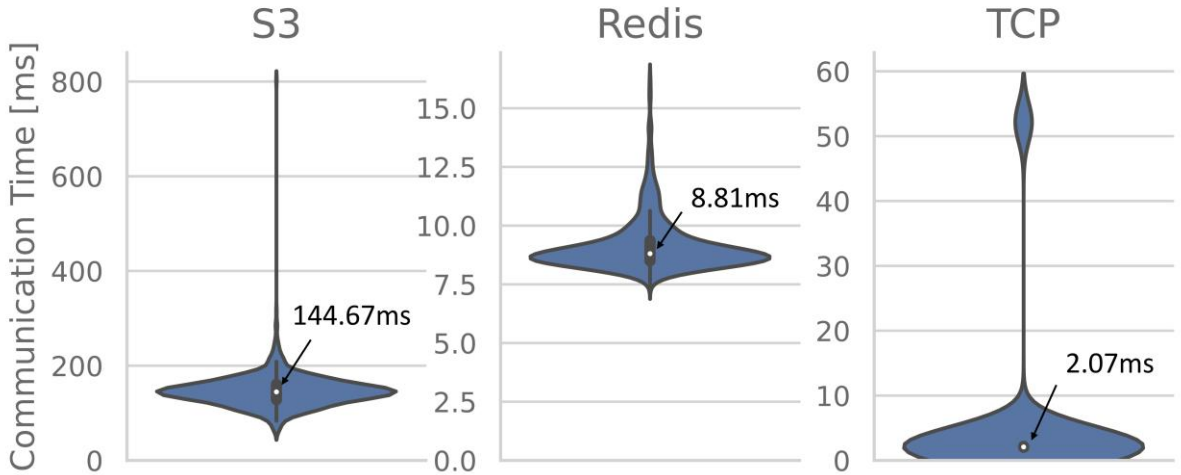
"FMI: Fast and Cheap Message Passing for Serverless Functions", ICS'23

Serverless Applications: Communication

1 B



1 MB



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

Serverless Applications: Communication

An Empirical Evaluation of Serverless Cloud Infrastructure for Large-Scale Data Processing

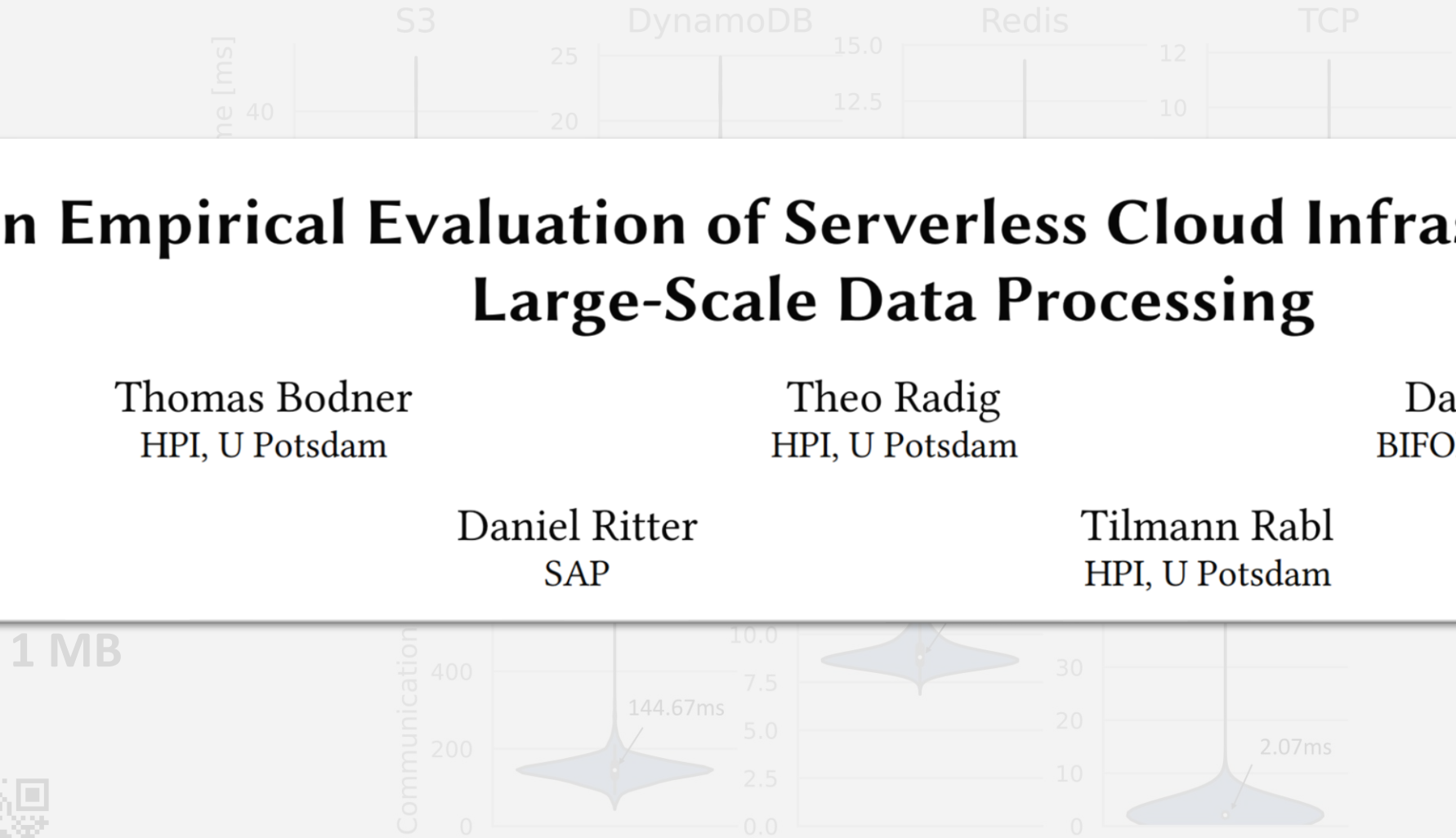
Thomas Bodner
HPI, U Potsdam

Theo Radig
HPI, U Potsdam

David Justen
BIFOLD, TU Berlin

Daniel Ritter
SAP

Tilmann Rabl
HPI, U Potsdam



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

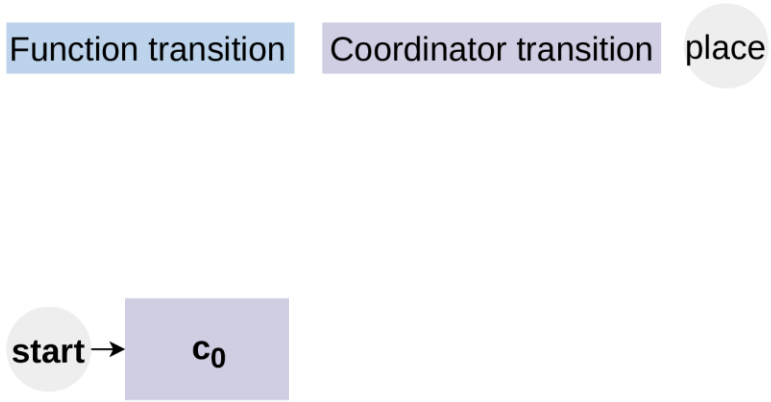
Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ **Part II**
  - ✓ Communication and Data
  - ✓ **Serverless Workflows**
  - ✓ Experiments in SeBS
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

SeBS Slack

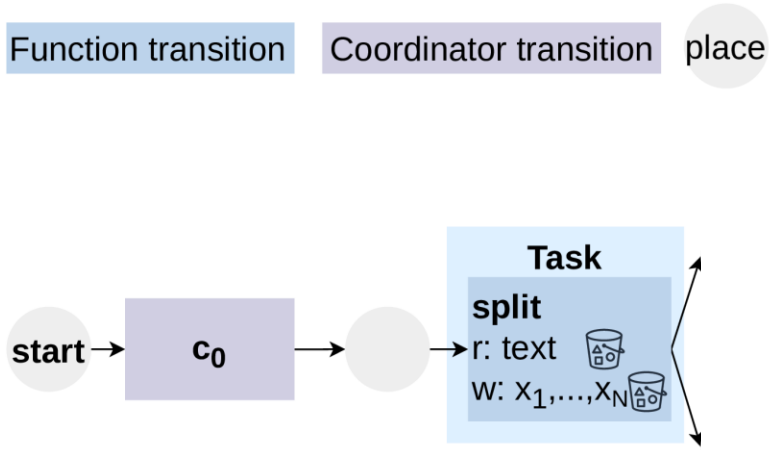
slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

SeBS-Flow: Let the Work Flow in the Cloud



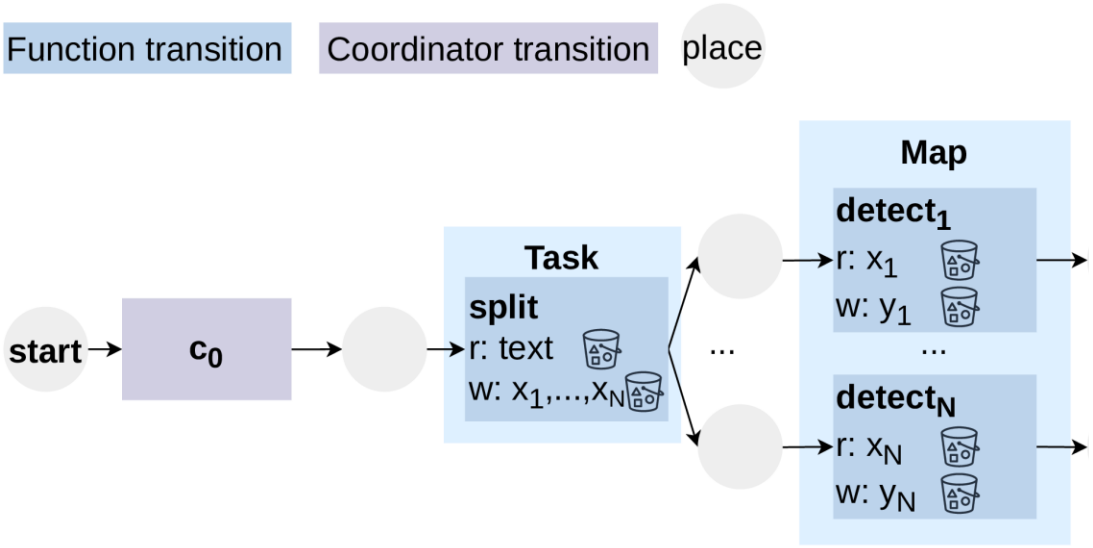
“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud



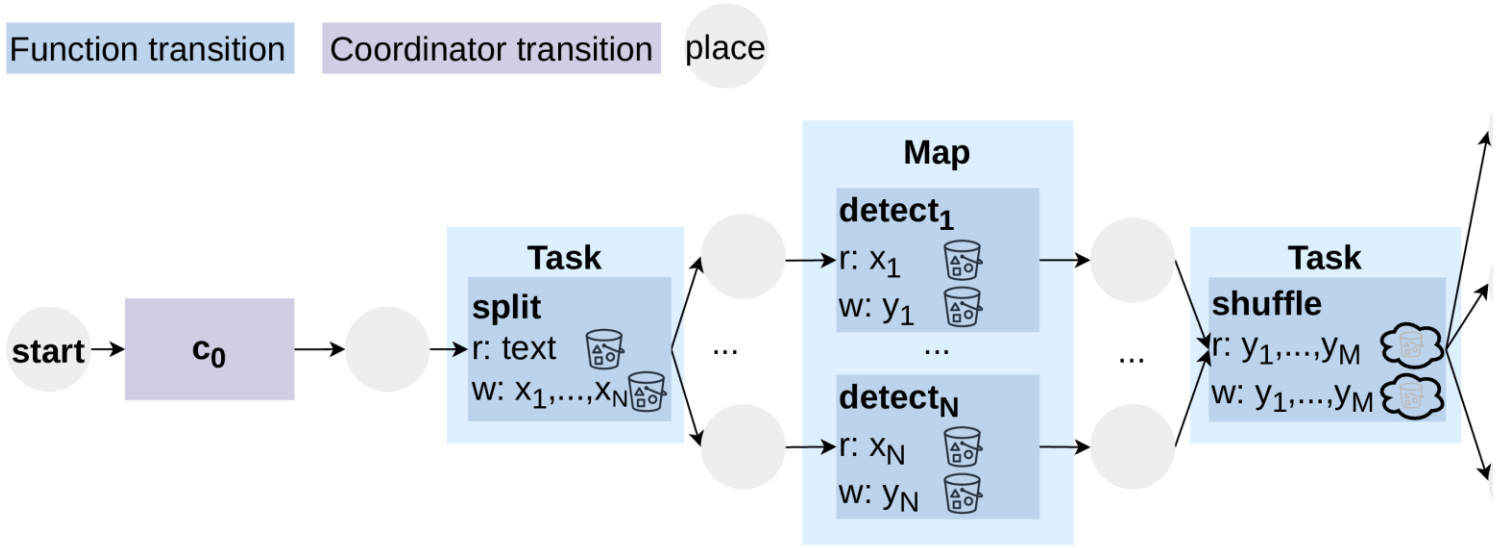
“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud



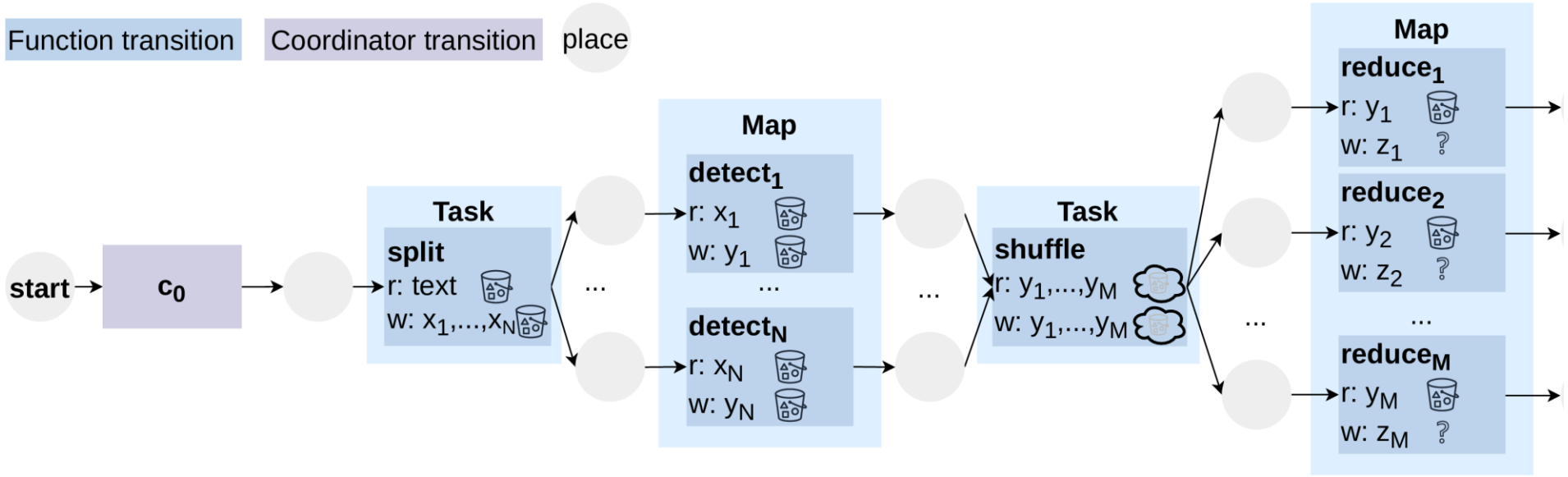
"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud



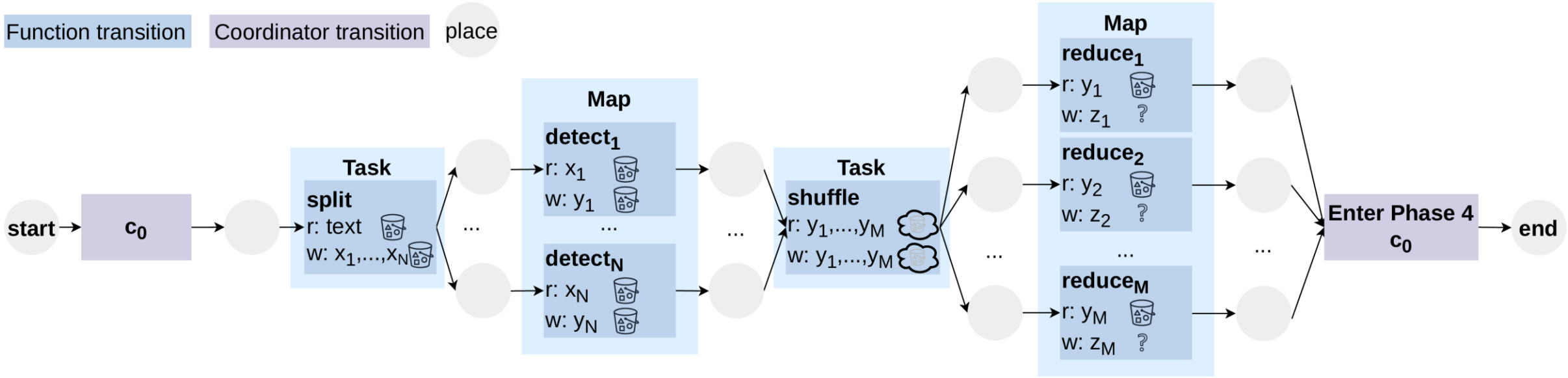
"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

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



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

SeBS-Flow: Let the Work Flow in the Cloud

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

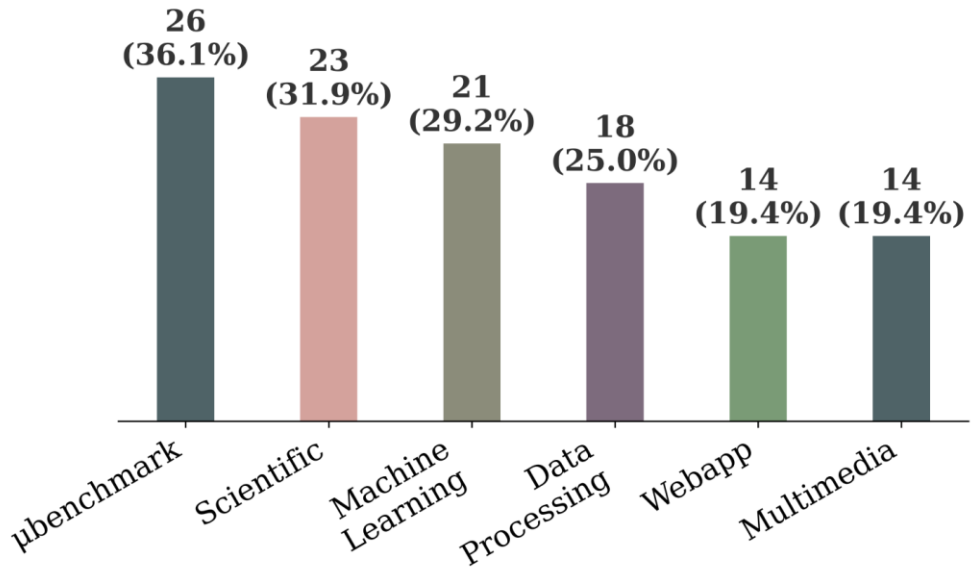
SeBS-Flow: Let the Work Flow in the Cloud

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

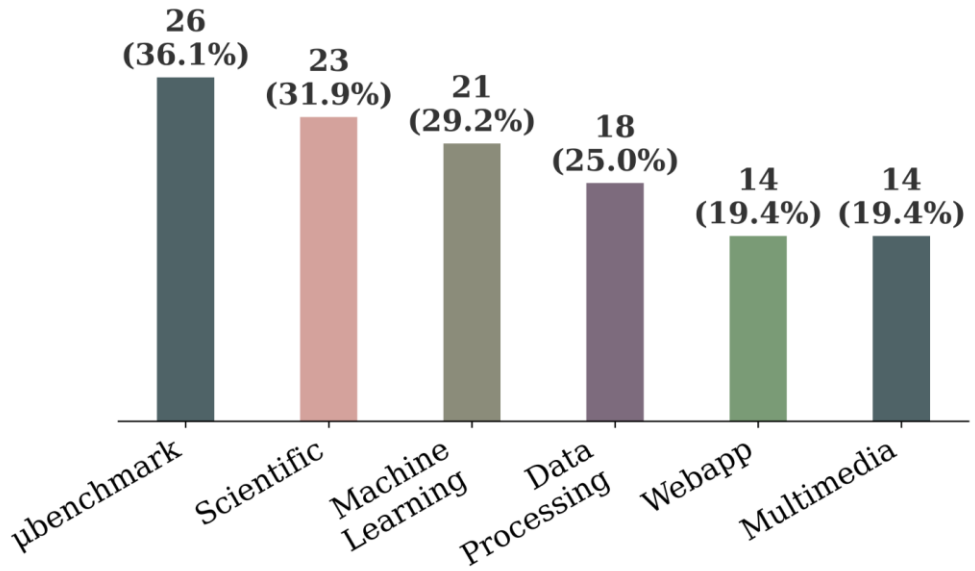
Analysis of 72 research papers: how are serverless workflows evaluated?

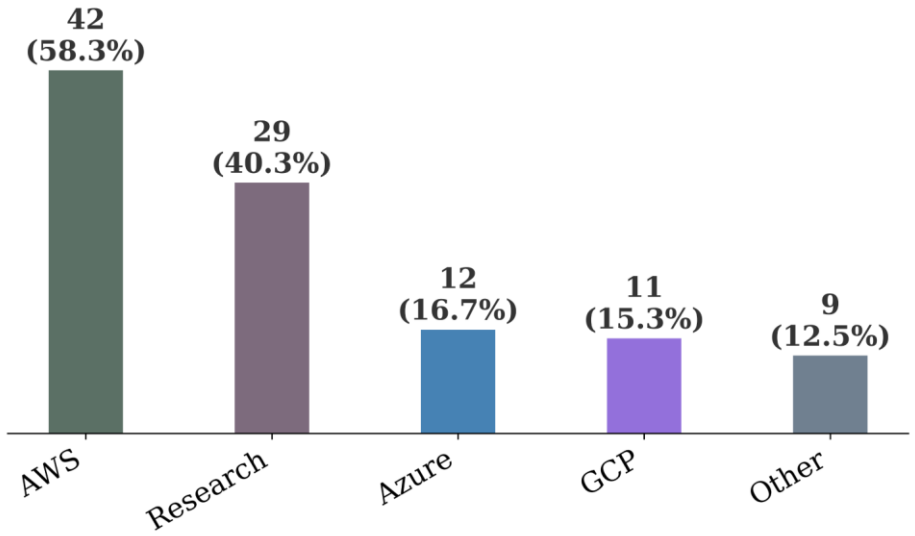


"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

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

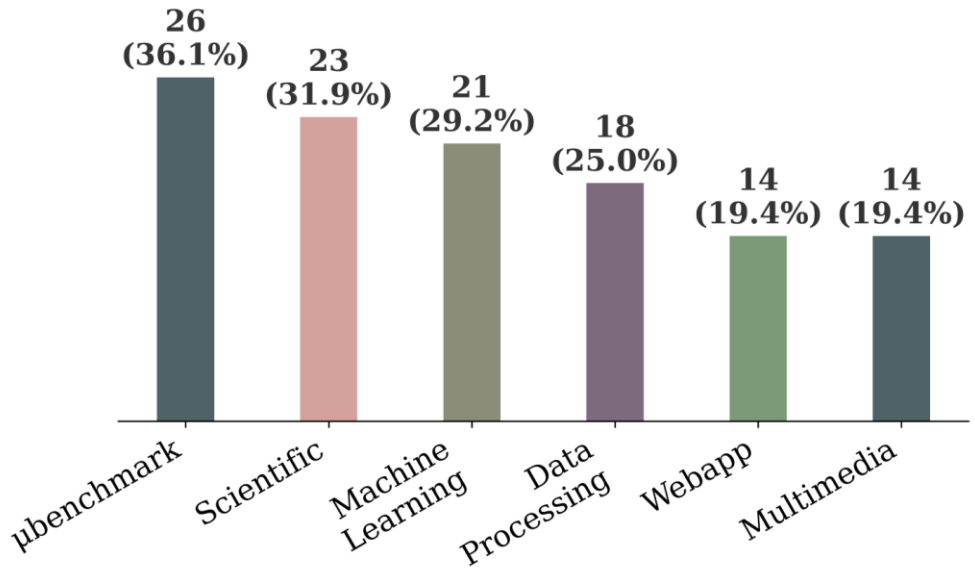


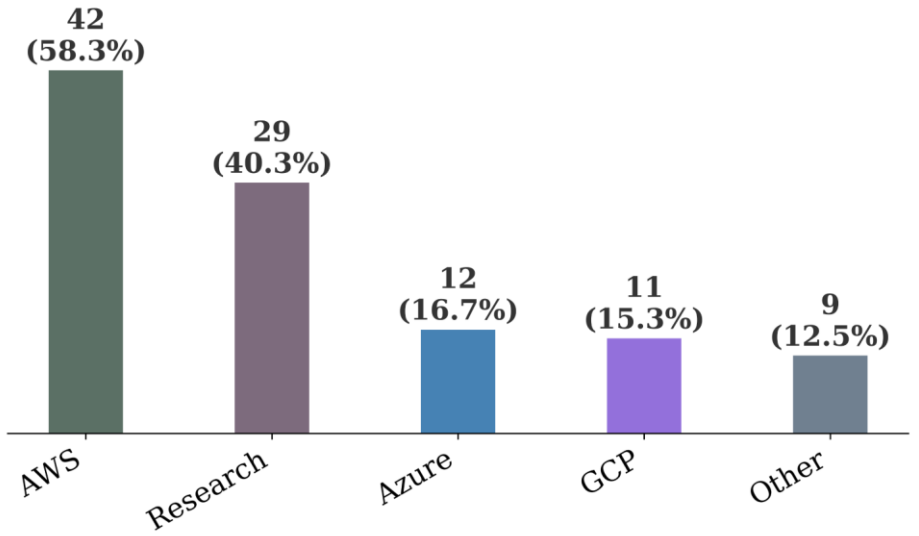


"SeBS-Flow: Benchmarking Serverless Cloud Function Workflows", Schmid et al., EuroSys 2025

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

SeBS-Flow: Let the Work Flow in the Cloud

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

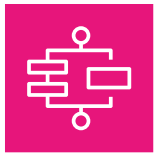
SeBS-Flow: Let the Work Flow in the Cloud

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

SeBS-Flow: Let the Work Flow in the Cloud

Workflow in Unified State Model

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

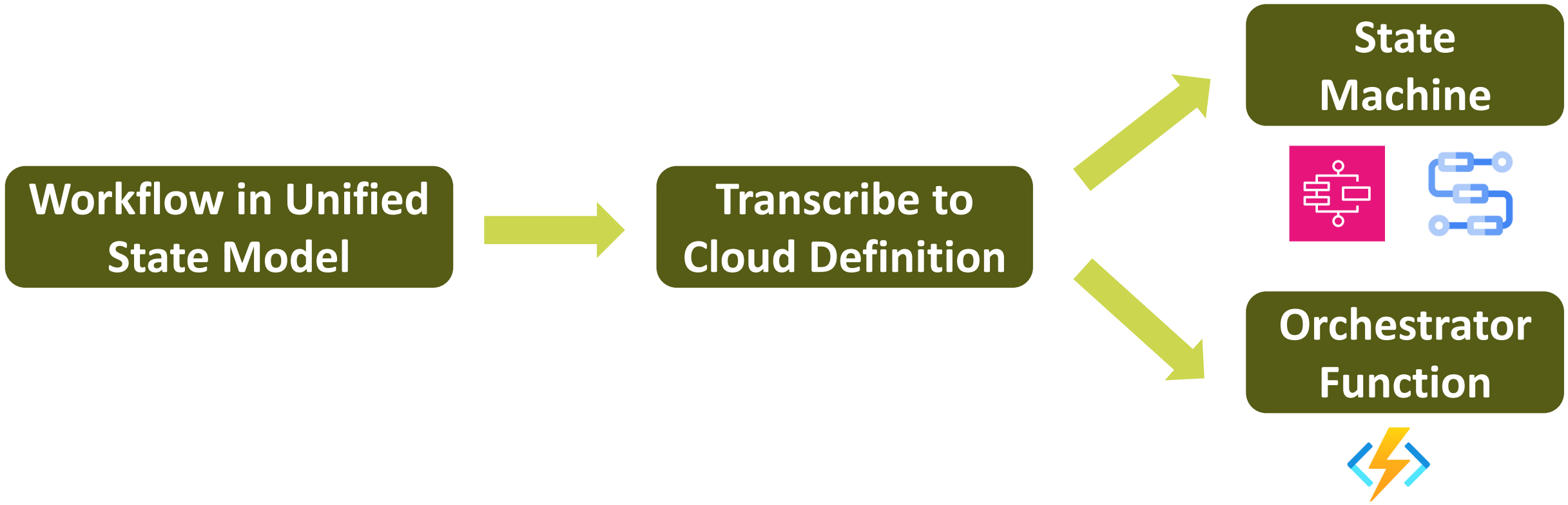
SeBS-Flow: Let the Work Flow in the Cloud

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

SeBS-Flow: Let the Work Flow in the Cloud

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

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

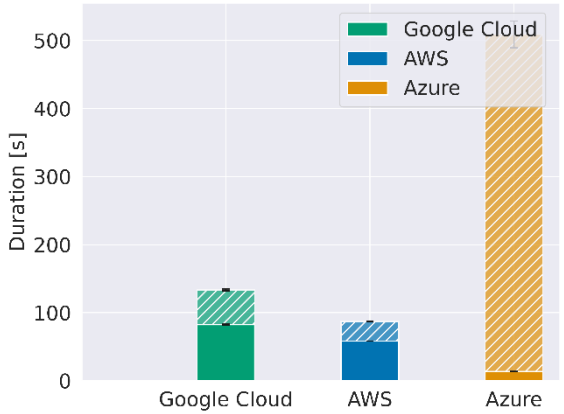
ExCamera

Trip Booking

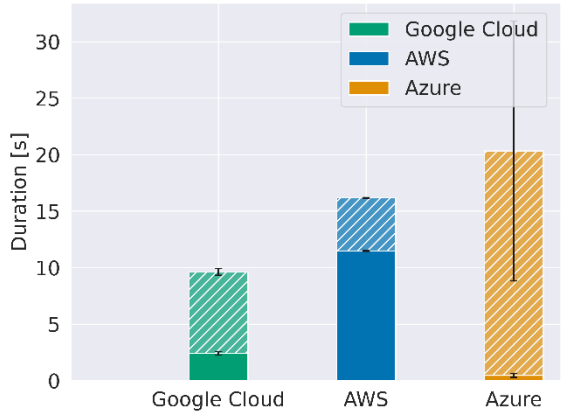
“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

SeBS-Flow: Let the Work Flow in the Cloud

ExCamera



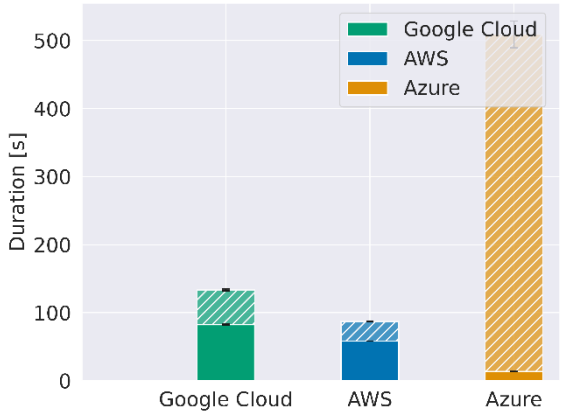
Trip Booking

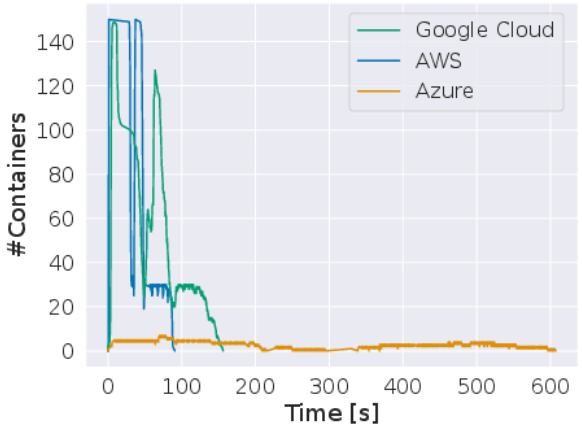


“SeBS-Flow: Benchmarking Serverless Cloud Function Workflows”, Schmid et al., EuroSys 2025

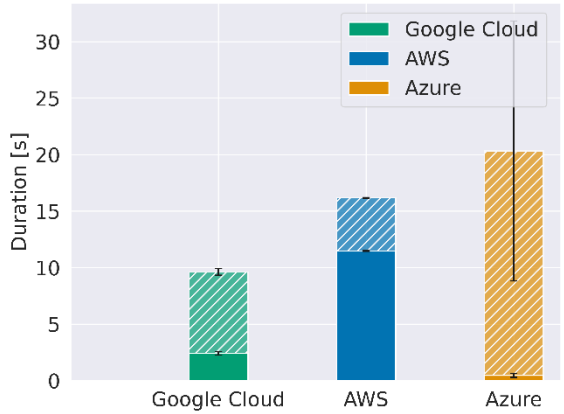
SeBS-Flow: Let the Work Flow in the Cloud

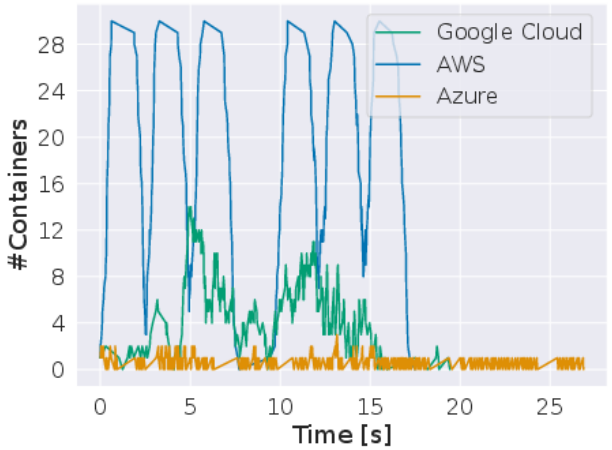
ExCamera





Trip Booking





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

SeBS-Flow: Let the Work Flow in the Cloud

spcl/serverless-benchmarks

Branch: feature/workflows

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

SeBS-Flow: Let the Work Flow in the Cloud

spcl/serverless-benchmarks

Branch: feature/workflows

Requires Redis for performance metrics.

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

SeBS-Flow: Let the Work Flow in the Cloud

spcl/serverless-benchmarks

Branch: feature/workflows

Requires Redis for performance metrics.

Works well with the Perf-Cost experiment.

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

From Functions to Applications

Workflows

“Applications”

From Functions to Applications

Workflows

“Applications”

- ❖ Native support.
- ❖ Easy transition from functions.
- ❖ “Serverless” deployment.

- ❖ Static design.
- ❖ Can be slow.
- ❖ Can be expensive.

From Functions to Applications

Workflows

- Native support.
- Easy transition from functions.
- “Serverless” deployment.

- Static design.
- Can be slow.
- Can be expensive.

“Applications”

- Dynamic behavior.
- Can be cheaper than workflows.
- Can be faster.

- Performance trade-offs unclear.
- Manual implementation.
- Different cloud semantics.

Agenda

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ **Part II**
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ **Experiments in SeBS**
- ✓ Hands-on II: FaaS Platforms & Experiments
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

SeBS Slack

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

Eviction Modeling

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

Eviction Modeling

Configuration:

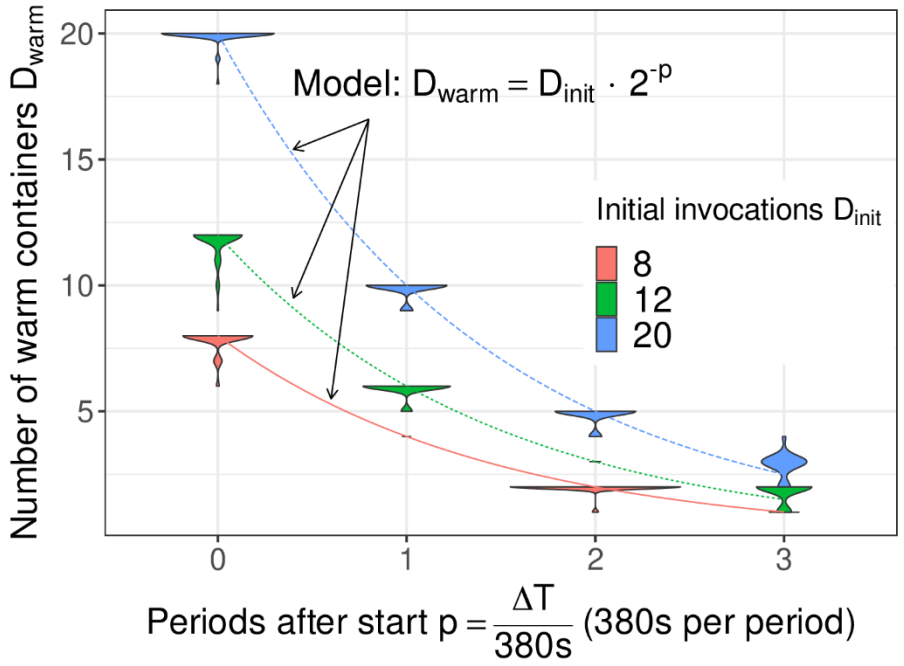
- **Time between invocations:** 1 – 1600s.
- **# of function instances:** 1 -20
- **Memory:** 128 – 1536 MB
- **Package size:** 8 kB, 250 MB
- **Duration:** 1 – 10s
- **Language:** Python, Node.js

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

Eviction Modeling

Configuration:

- **Time between invocations:** 1 – 1600s.
- **# of function instances:** 1 -20
- **Memory:** 128 – 1536 MB
- **Package size:** 8 kB, 250 MB
- **Duration:** 1 – 10s
- **Language:** Python, Node.js

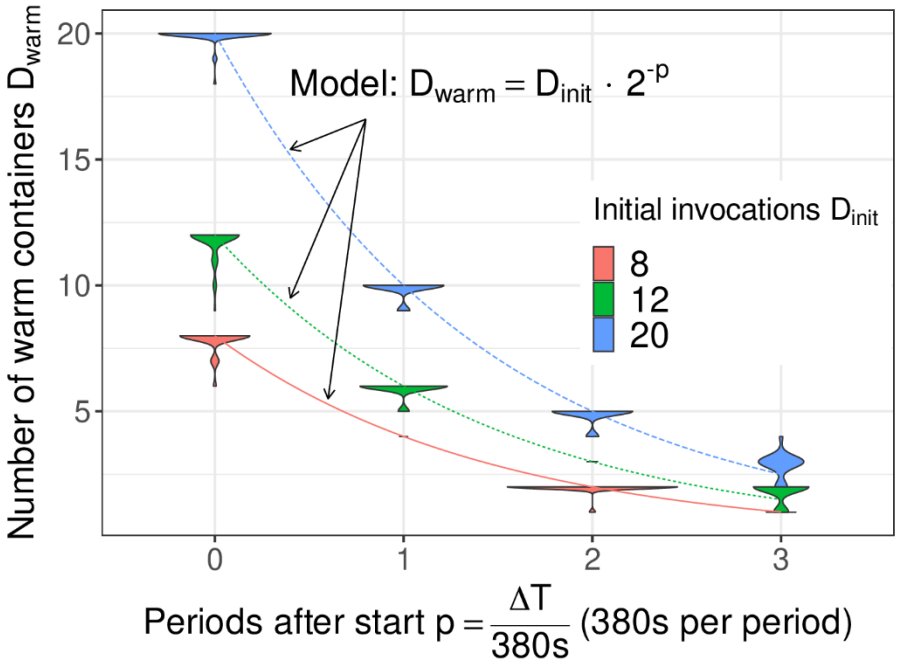


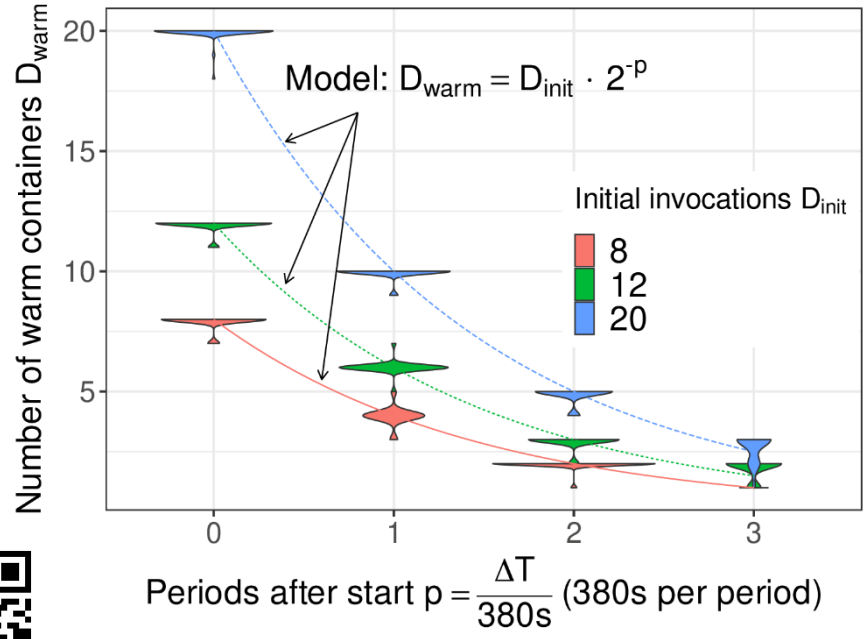
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

Eviction Modeling

Configuration:

- **Time between invocations:** 1 – 1600s.
- **# of function instances:** 1 -20
- **Memory:** 128 – 1536 MB
- **Package size:** 8 kB, 250 MB
- **Duration:** 1 – 10s
- **Language:** Python, Node.js



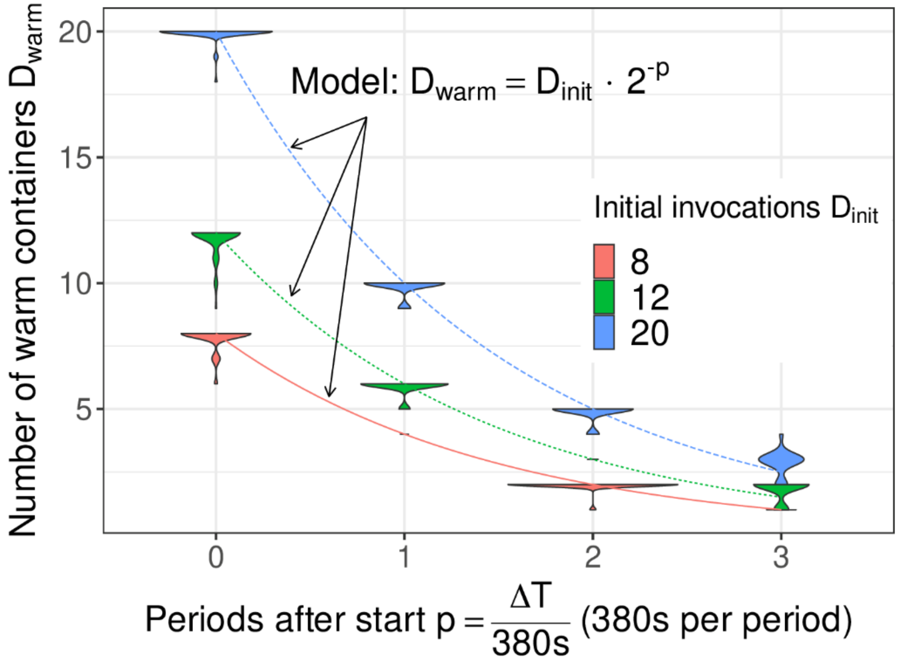


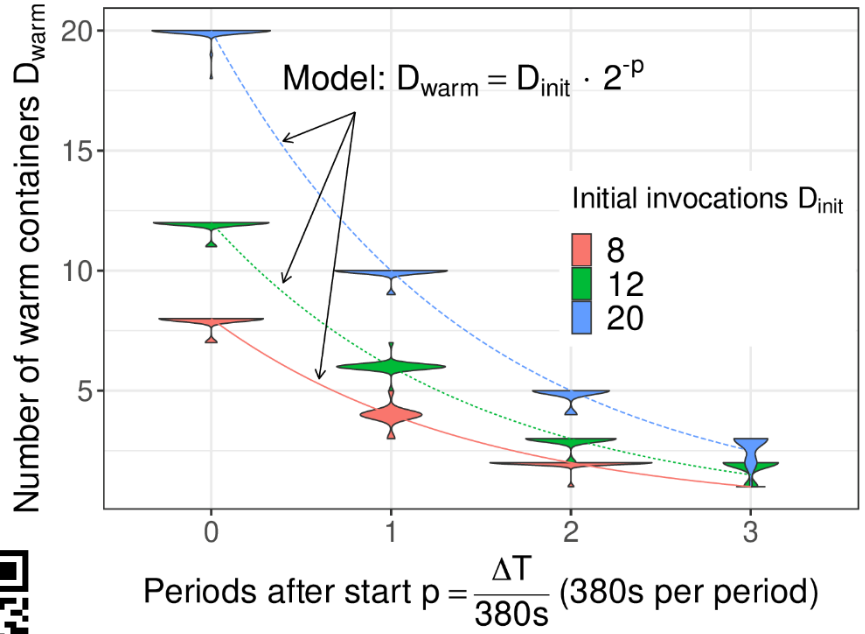
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

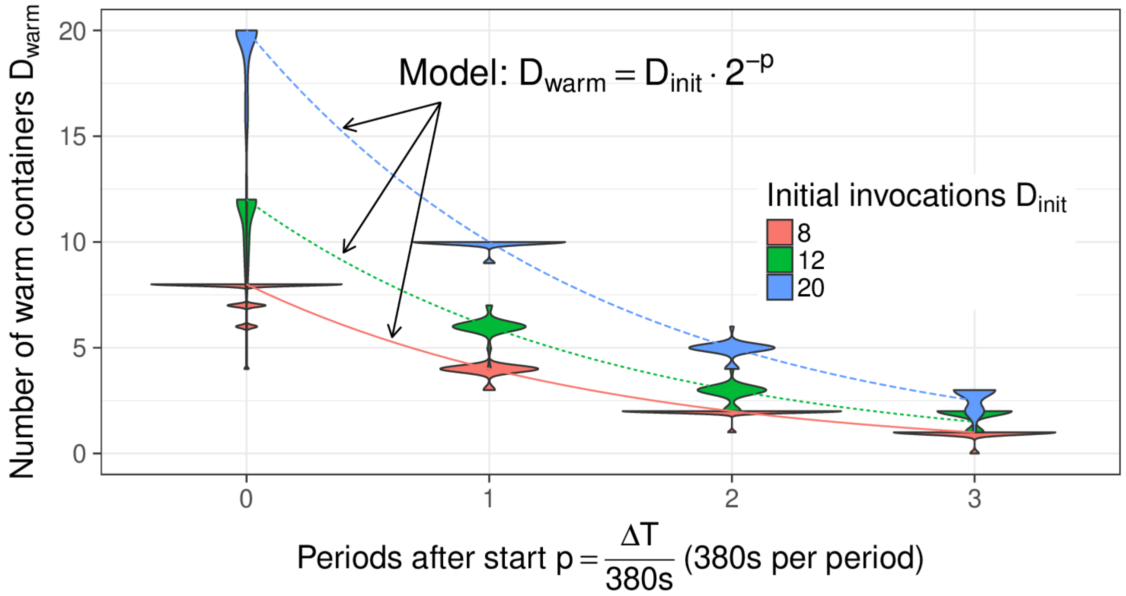
Eviction Modeling

Configuration:

- **Time between invocations:** 1 – 1600s.
- **# of function instances:** 1 -20
- **Memory:** 128 – 1536 MB
- **Package size:** 8 kB, 250 MB
- **Duration:** 1 – 10s
- **Language:** Python, Node.js

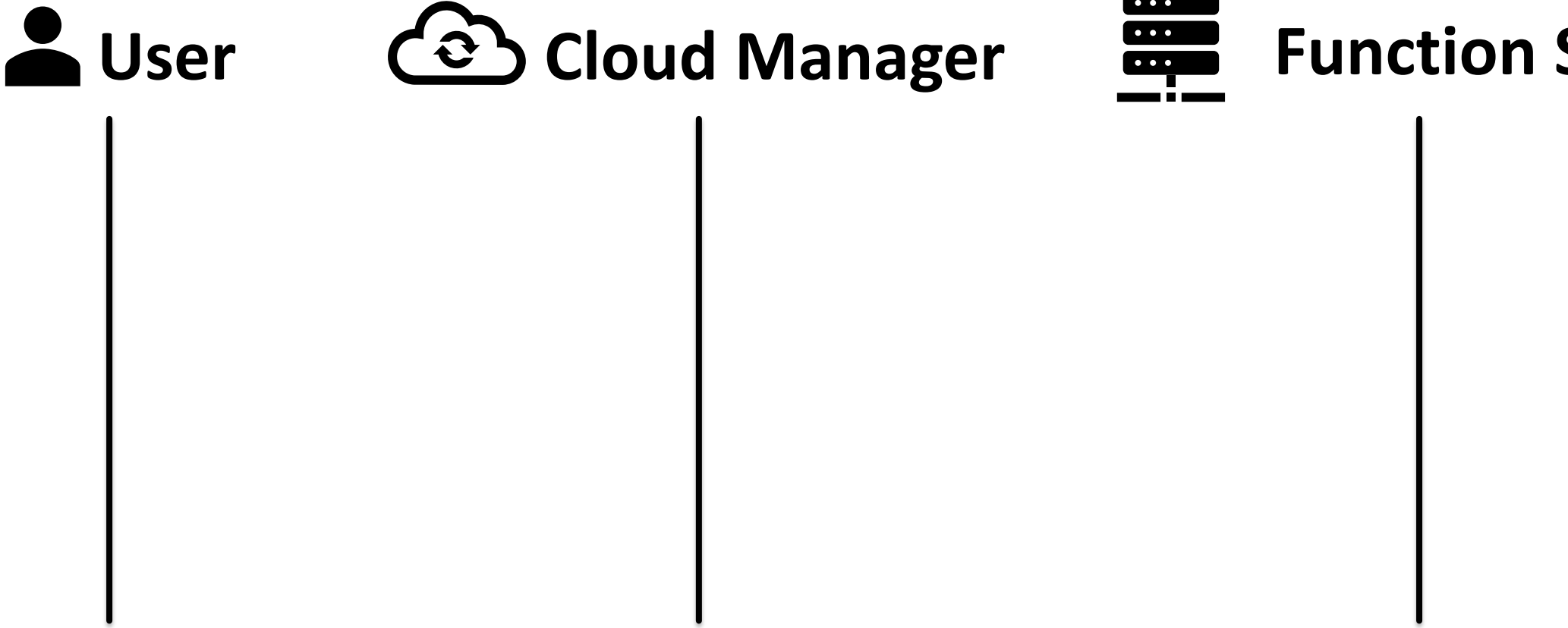






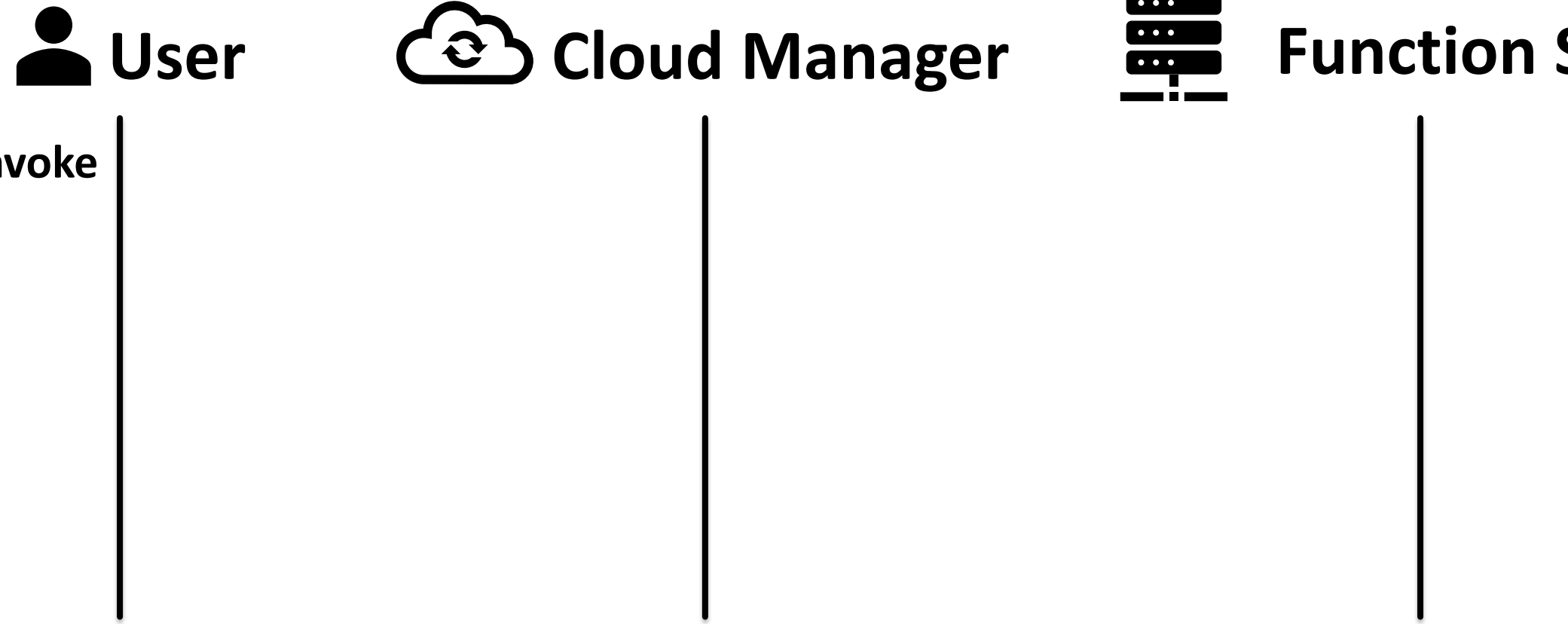
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

FaaS Analysis: Invocation Overhead



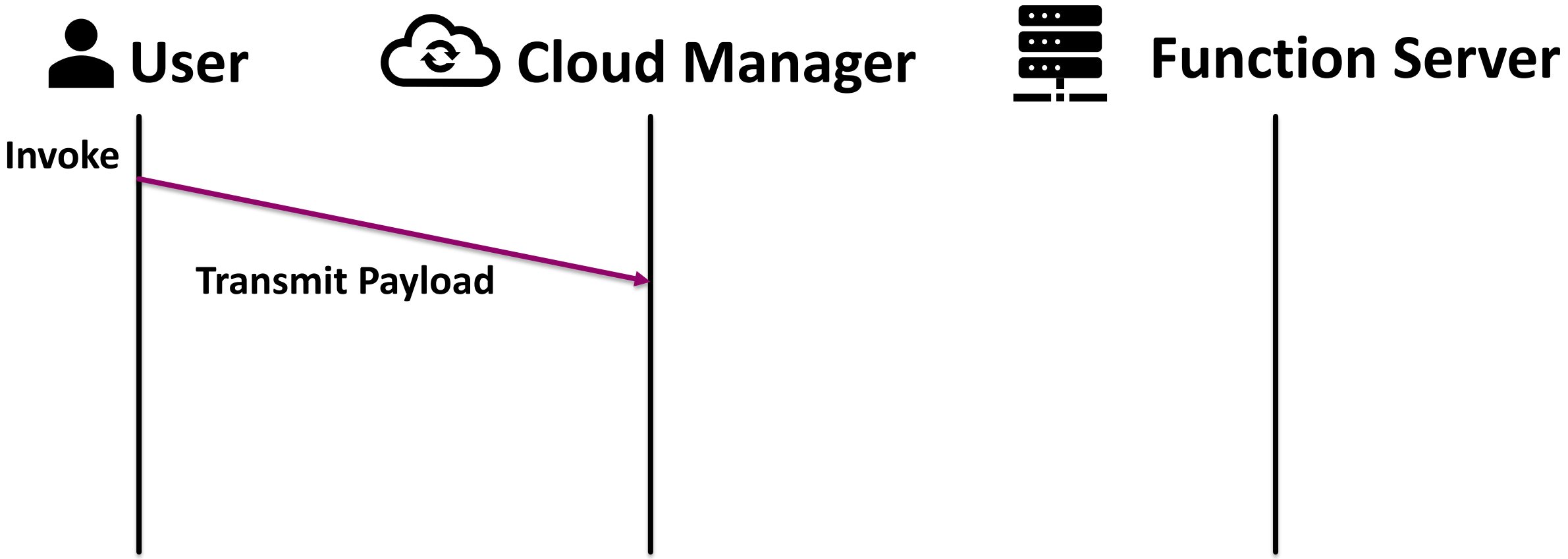
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

FaaS Analysis: Invocation Overhead



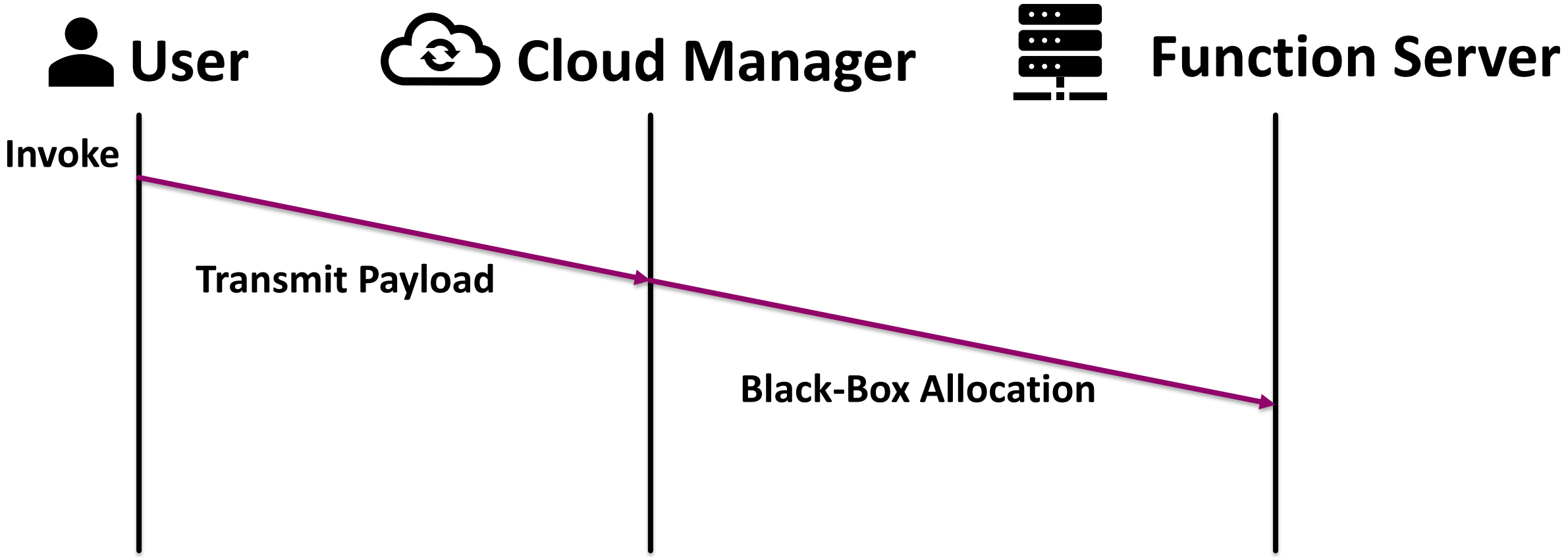
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

FaaS Analysis: Invocation Overhead



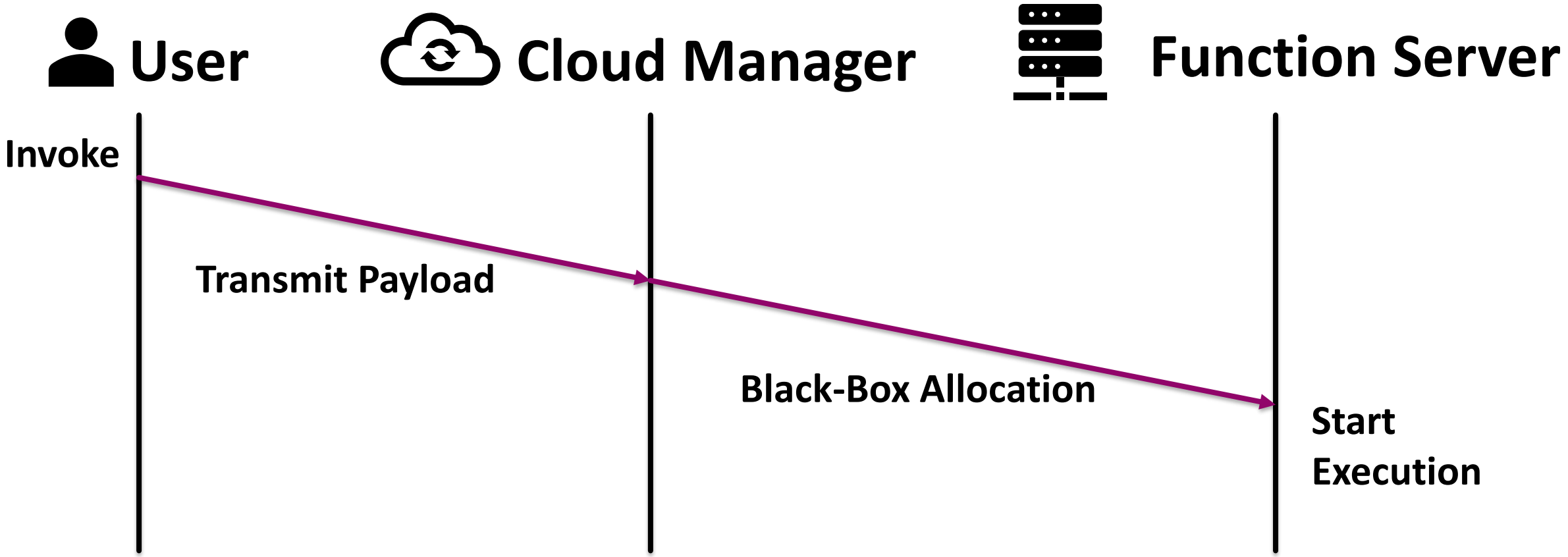
"SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing", Middleware 2021

FaaS Analysis: Invocation Overhead



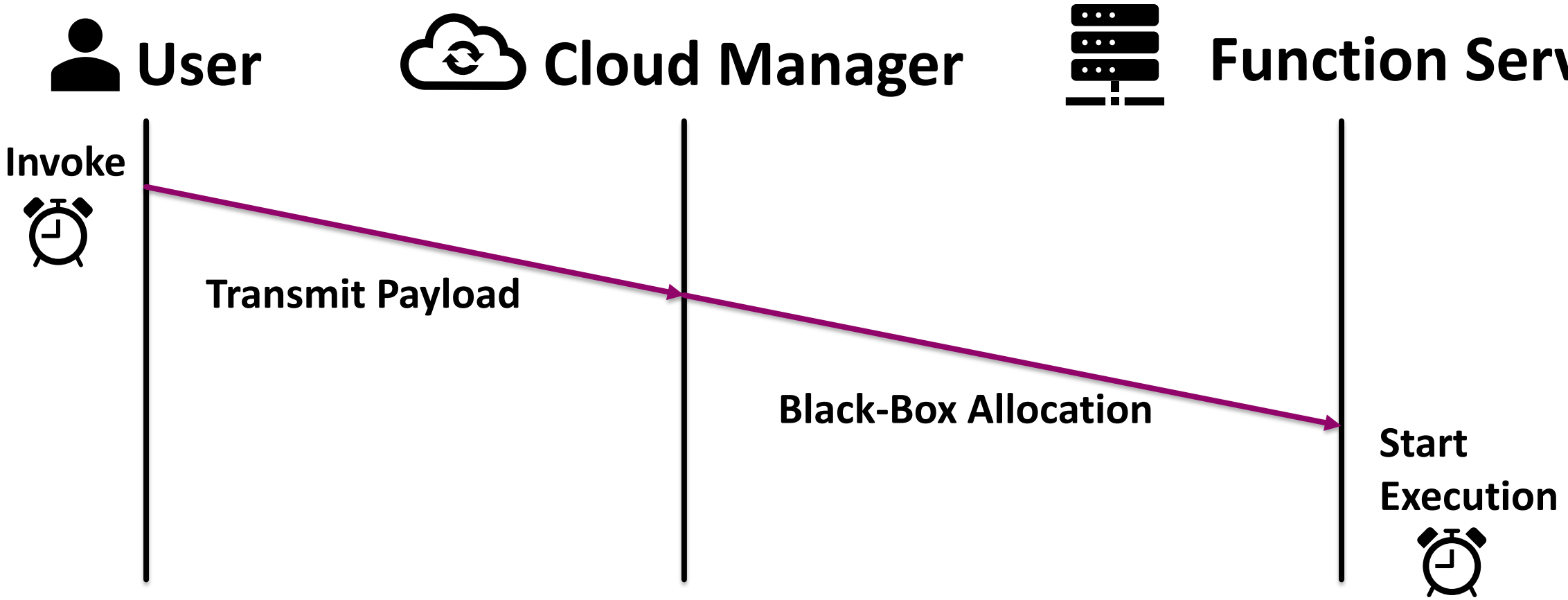
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

FaaS Analysis: Invocation Overhead



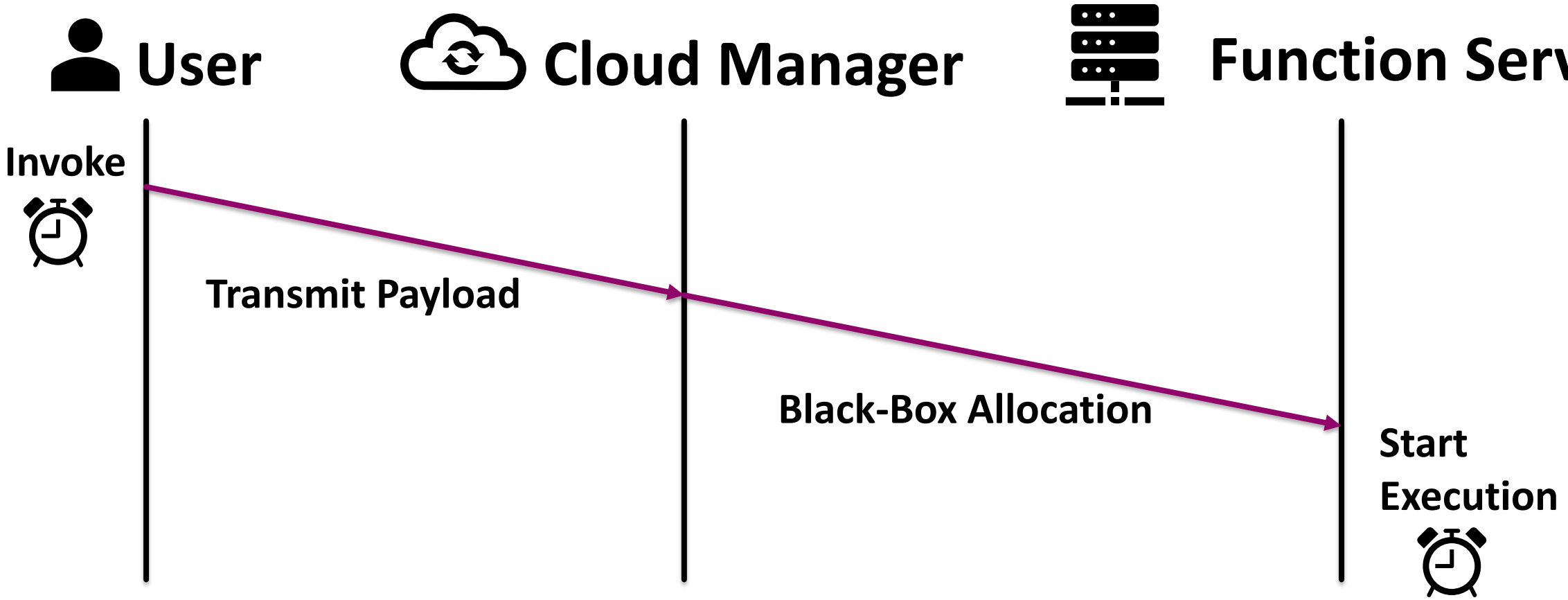
“SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing”, Middleware 2021

FaaS Analysis: Invocation Overhead



"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

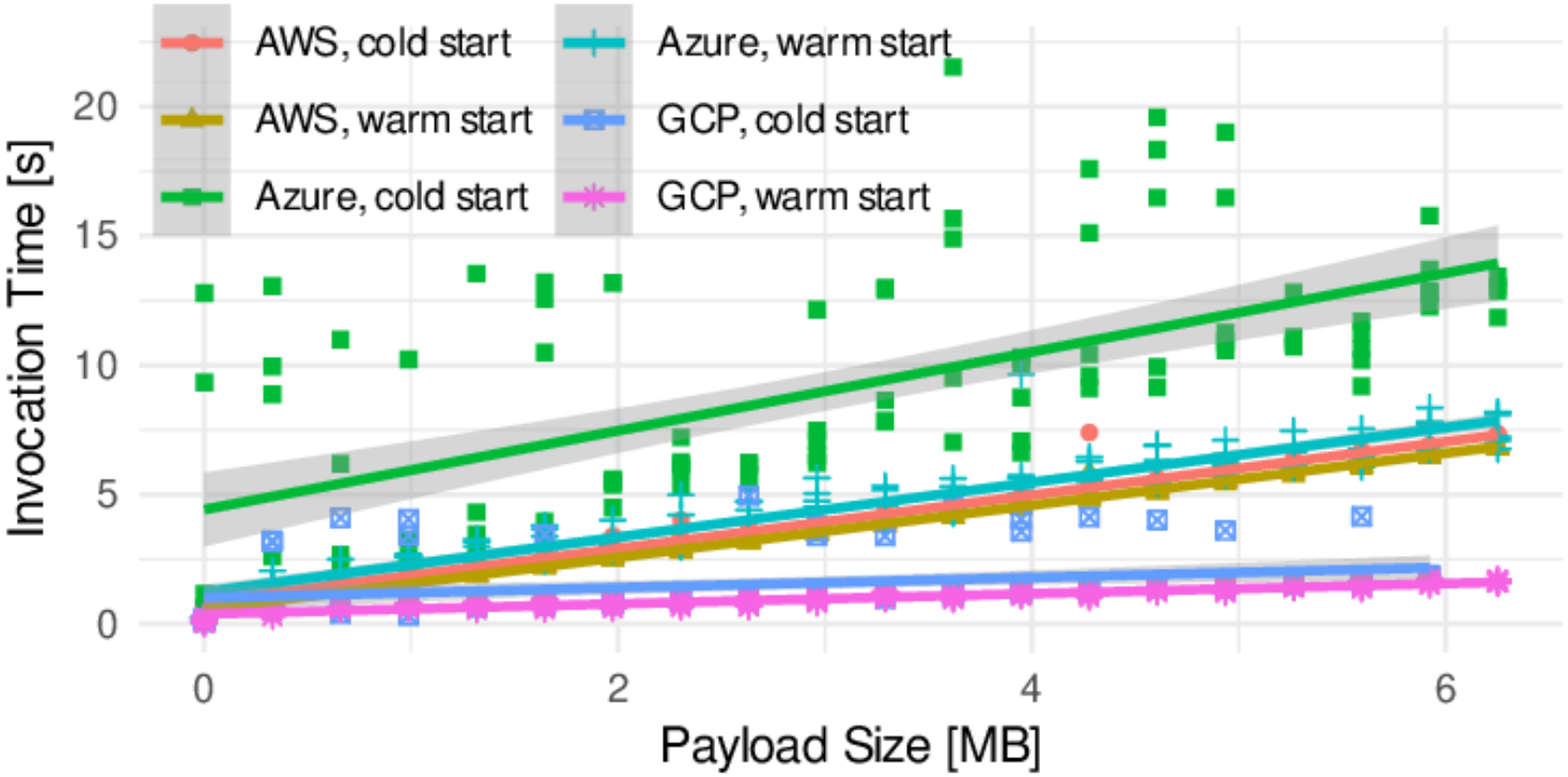
FaaS Analysis: Invocation Overhead

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

Agenda

SeBS Slack

- ✓ Part I
  - ✓ What is Serverless?
  - ✓ Benchmarking Suite SeBS
  - ✓ Working with SeBS
- ✓ Hands-on I: Local Deployment & Storage
- ✓ Part II
  - ✓ Communication and Data
  - ✓ Serverless Workflows
  - ✓ Experiments in SeBS
- ✓ **Hands-on II: FaaS Platforms & Experiments**
- ✓ Part III
  - ✓ Research Directions in Serverless
  - ✓ Development of SeBS

slack **Join SeBS Slack channel: https://serverlessbenchmark.slack.com**

SeBS Modularity

spcl/serverless-benchmarks

SeBS Modularity

spcl/serverless-benchmarks

Add new function?

- ❖ Define dependencies (pip, npm, C++ binaries, etc.).
- ❖ Define input generation and data initialization (object, storage NoSQL).
- ❖ Add code!
- ❖ Optional: custom build steps for non-language dependencies.

SeBS Modularity

spcl/serverless-benchmarks

Add new function?

- ❖ Define dependencies (pip, npm, C++ binaries, etc.).
- ❖ Define input generation and data initialization (object, storage NoSQL).
- ❖ Add code!
- ❖ Optional: custom build steps for non-language dependencies.

Add new platform?

- ❖ Define packaging function.
- ❖ Add API calls to create/update/delete function.
- ❖ Add function triggers (e.g. retrieve URL)
- ❖ Optional: download metrics, enforce cold starts.

SeBS Modularity

spcl/serverless-benchmarks

Add new function?

- Define dependencies (pip, npm, C++ binaries, etc.).
- Define input generation and data initialization (object, storage NoSQL).
- Add code!
- Optional: custom build steps for non-language dependencies.

Add new platform?

- Define packaging function.
- Add API calls to create/update/delete function.
- Add function triggers (e.g. retrieve URL)
- Optional: download metrics, enforce cold starts.

Add new experiment?

- Functions and resources automatically deployed.
- Just add your own logic.

SeBS Modularity

spcl/serverless-benchmarks

Add new function?

- ❖ Define dependencies (pip, npm, C++ binaries, etc.).
- ❖ Define input generation and data initialization (object, storage NoSQL).
- ❖ Add code!
- ❖ Optional: custom build steps for non-language dependencies.

Add new platform?

- ❖ Define packaging function.
- ❖ Add API calls to create/update/delete function.
- ❖ Add function triggers (e.g. retrieve URL)
- ❖ Optional: download metrics, enforce cold starts.

Add new experiment?

- ❖ Functions and resources automatically deployed.
- ❖ Just add your own logic.

SeBS in Practice: Part II

SeBS Slack

spcl/sebs-tutorial

git clone https://github.com/spcl/sebs-tutorial.git

slack Join SeBS Slack channel: https://serverlessbenchmark.slack.com