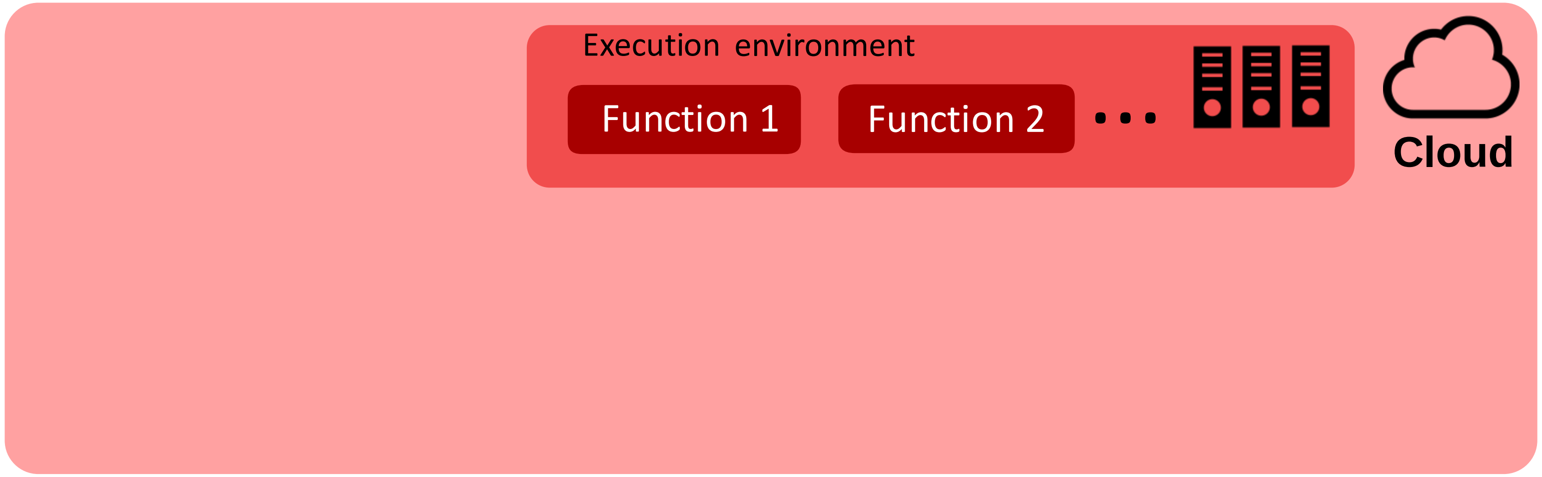
SeBS: a Serverless Benchmark Suite for Function-as-a-Service Computing

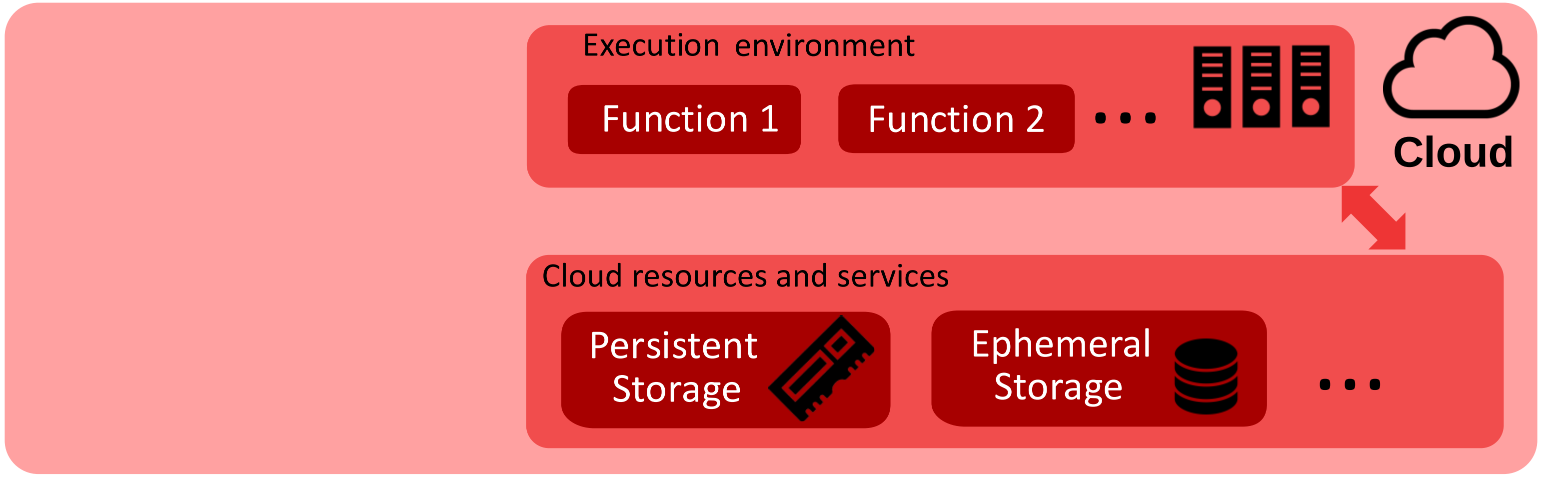
Marcin Copik, Grzegorz Kwasniewski, Maciej Besta, Michal Podstawski, Torsten Hoefler

What is serverless?

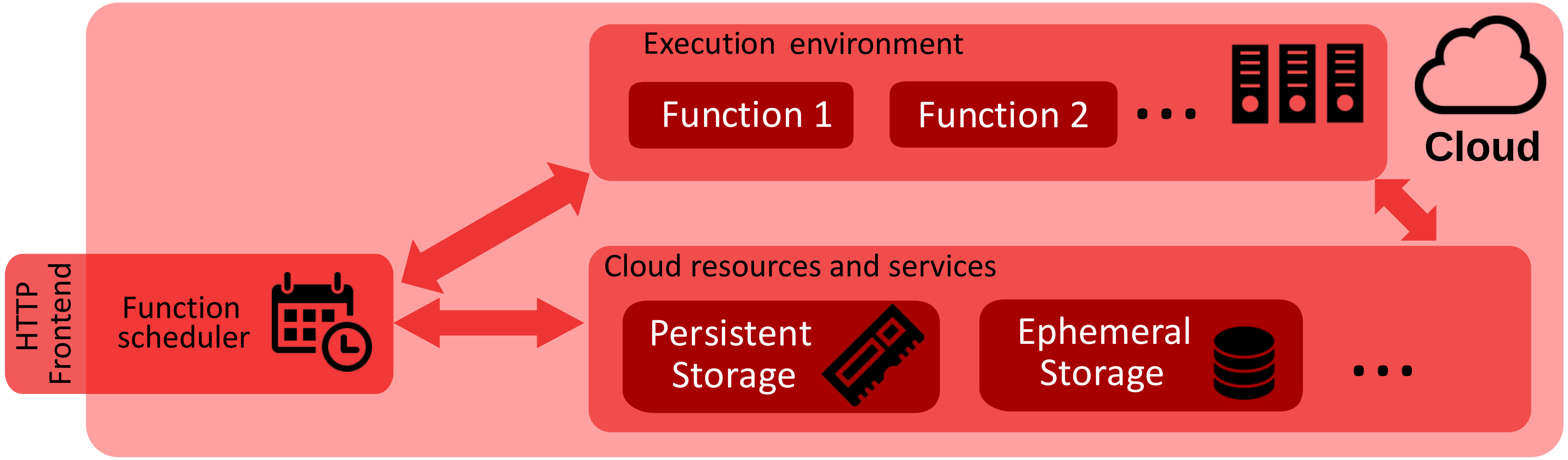
What is serverless?



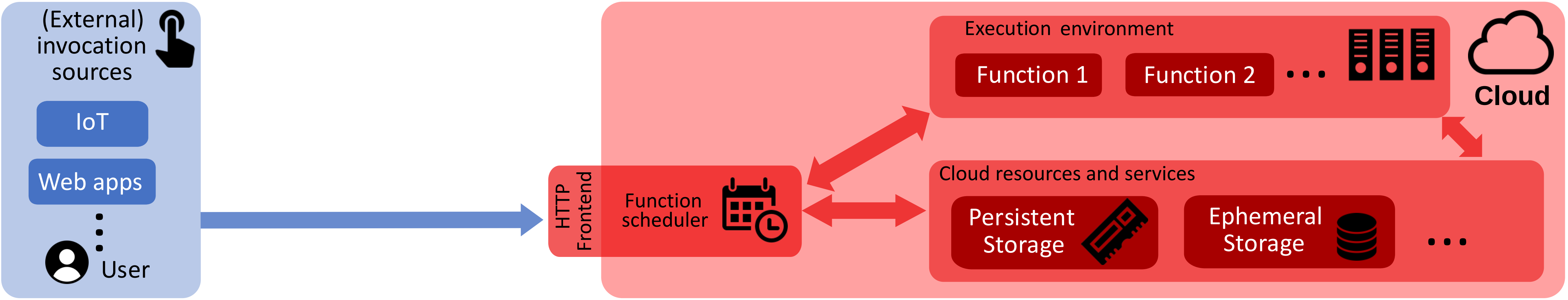
What is serverless?



What is serverless?



What is serverless?



Serverless – why & why not?

User

Cloud Provider

Serverless – why & why not?

User

- Pay-as-you-go billing
- Massive parallelism
- Simplified deployment
- Architecture agnostic

Cloud Provider

Serverless – why & why not?

User

- Pay-as-you-go billing
- Massive parallelism
- Simplified deployment
- Architecture agnostic

Cloud Provider

- Higher machine utilization
- Fine-grained scheduling

Serverless – why & why not?

User

- Pay-as-you-go billing
- Massive parallelism
- Simplified deployment
- Architecture agnostic

- High computing cost
- Variable performance
- Vendor lock-in
- Black-box platform

Cloud Provider

- Higher machine utilization
- Fine-grained scheduling

Serverless – why & why not?

User

- Pay-as-you-go billing
- Massive parallelism
- Simplified deployment
- Architecture agnostic

- High computing cost
- Variable performance
- Vendor lock-in
- Black-box platform

Cloud Provider

- Higher machine utilization
- Fine-grained scheduling

- Handling heterogeneity
- Micro-architecture effects

Commercial serverless systems

Policy	AWS Lambda	Azure Functions	Google Cloud Functions

Commercial serverless systems

Functions
Workers

Policy	AWS Lambda	Azure Functions	Google Cloud Functions
Languages (native)	Python, Node.js, C#, Java, C++, etc.	Python, TypeScript, C#, Java, etc.	Node.js, Python, Java, Go.

Commercial serverless systems

Functions
Workers

Memory
configuration

Policy	AWS Lambda	Azure Functions	Google Cloud Functions
Languages (native)	Python, Node.js, C#, Java, C++, etc.	Python, TypeScript, C#, Java, etc.	Node.js, Python, Java, Go.
Memory Allocation	Static, 128 – 3008 MB.	Dynamic, up to 1536 MB.	Static, 128, 256, 512, 1024, 2048 MB.

Commercial serverless systems

Functions Workers

Memory configuration

Policy	AWS Lambda	Azure Functions	Google Cloud Functions
Languages (native)	Python, Node.js, C#, Java, C++, etc.	Python, TypeScript, C#, Java, etc.	Node.js, Python, Java, Go.
Memory Allocation	Static, 128 – 3008 MB.	Dynamic, up to 1536 MB.	Static, 128, 256, 512, 1024, 2048 MB.
Concurrency Limit	1000 Functions	200 Function Apps.	100 Functions.

Function Model

Commercial serverless systems

Functions Workers

Memory configuration

Policy	AWS Lambda	Azure Functions	Google Cloud Functions
Languages (native)	Python, Node.js, C#, Java, C++, etc.	Python, TypeScript, C#, Java, etc.	Node.js, Python, Java, Go.
Memory Allocation	Static, 128 – 3008 MB.	Dynamic, up to 1536 MB.	Static, 128, 256, 512, 1024, 2048 MB.
Concurrency Limit	1000 Functions	200 Function Apps.	100 Functions.
CPU Allocation	Proportional to memory, 1 vCPU on 1792 MB.	Unknown.	Proportional to memory, 2.4 GHz CPU at 2048 MB.

Function Model

CPU Allocation

Commercial serverless systems

Functions Workers

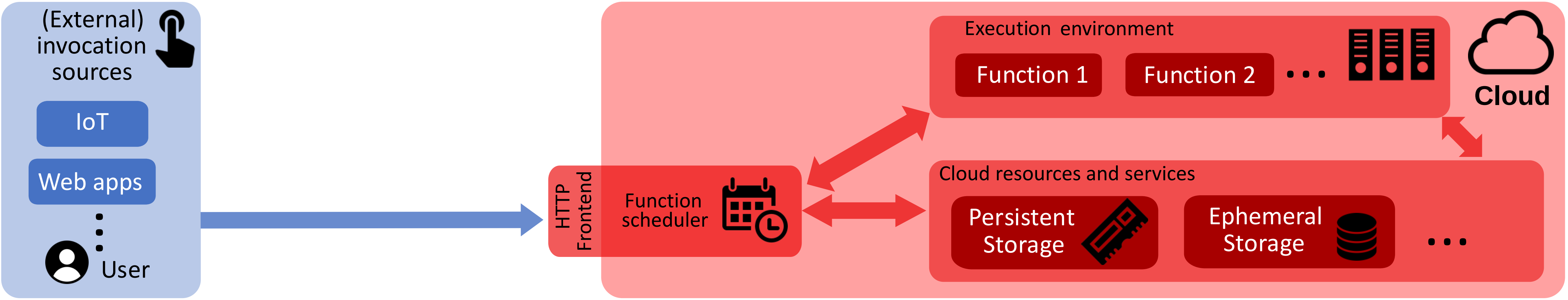
Memory configuration

Policy	AWS Lambda	Azure Functions	Google Cloud Functions
Languages (native)	Python, Node.js, C#, Java, C++, etc.	Python, TypeScript, C#, Java, etc.	Node.js, Python, Java, Go.
Memory Allocation	Static, 128 – 3008 MB.	Dynamic, up to 1536 MB.	Static, 128, 256, 512, 1024, 2048 MB.
Concurrency Limit	1000 Functions	200 Function Apps.	100 Functions.
CPU Allocation	Proportional to memory, 1 vCPU on 1792 MB.	Unknown.	Proportional to memory, 2.4 GHz CPU at 2048 MB.
Billing	Duration and declared memory.	Average memory use, duration.	Duration, declared CPU and memory.
Deployment	zip package up to 250 MB.	zip package, Docker image.	zip package, up to 100 MB.
Time Limit	15 minutes	10 min / 60 min / unlimited.	9 minutes.

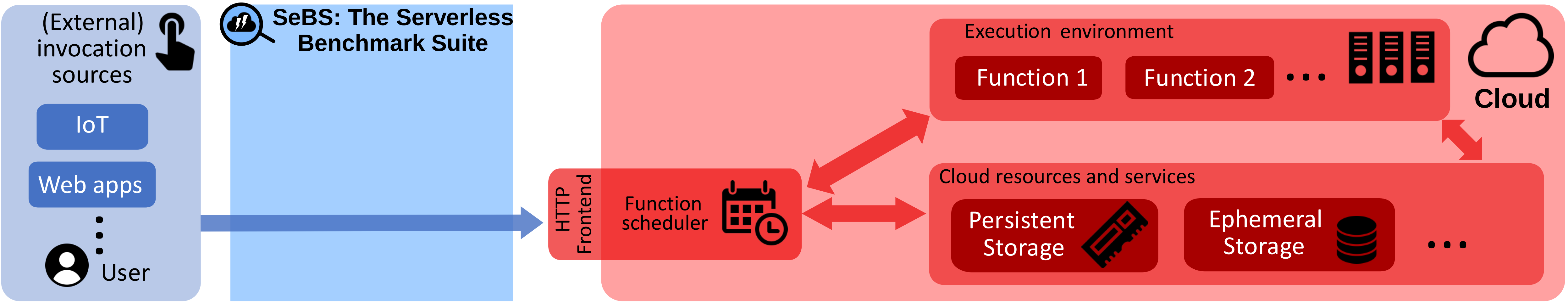
Function Model

CPU Allocation

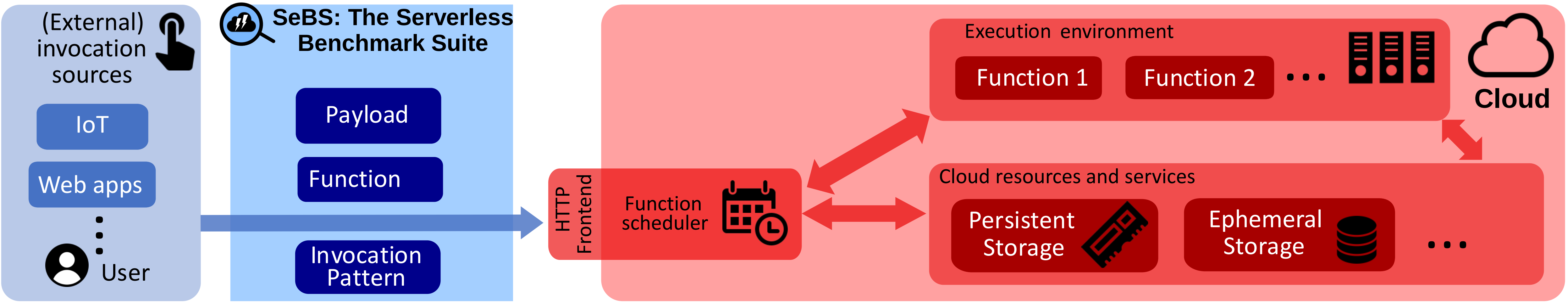
What is serverless?



What is serverless?



What is serverless?



Benchmark Applications

Type	Name	Language

Benchmark Applications

Type	Name	Language
Webapps	uploader	**Python**, Node.js

Benchmark Applications

Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++

Benchmark Applications

Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**

Benchmark Applications

Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++

Benchmark Applications

Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++
Scientific	graph-bfs	**Python**

Results and Insights

	Results, methods, and insights

Results and Insights

	Results, methods, and insights
	

Results and Insights

	Results, methods, and insights
$	

Results and Insights

	Results, methods, and insights

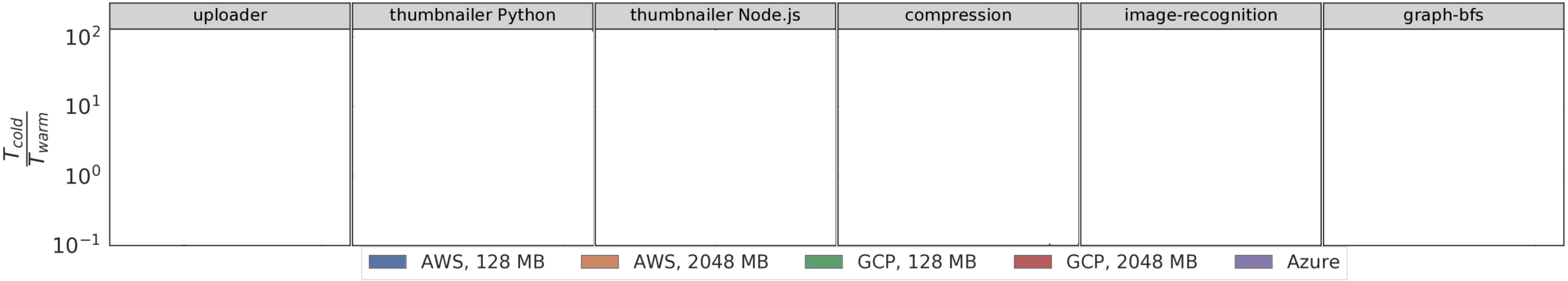
Results and Insights

	Results, methods, and insights
$	

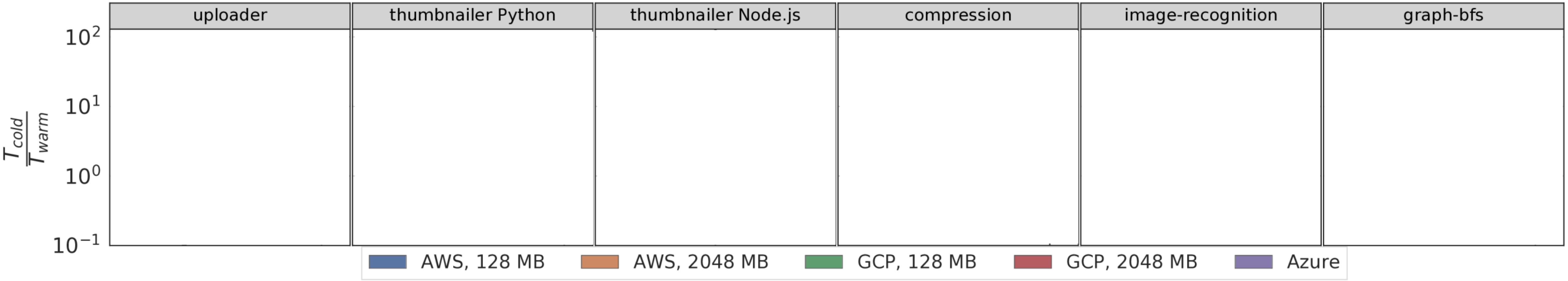
Results and Insights

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
$	

Performance Analysis: Cold Startups



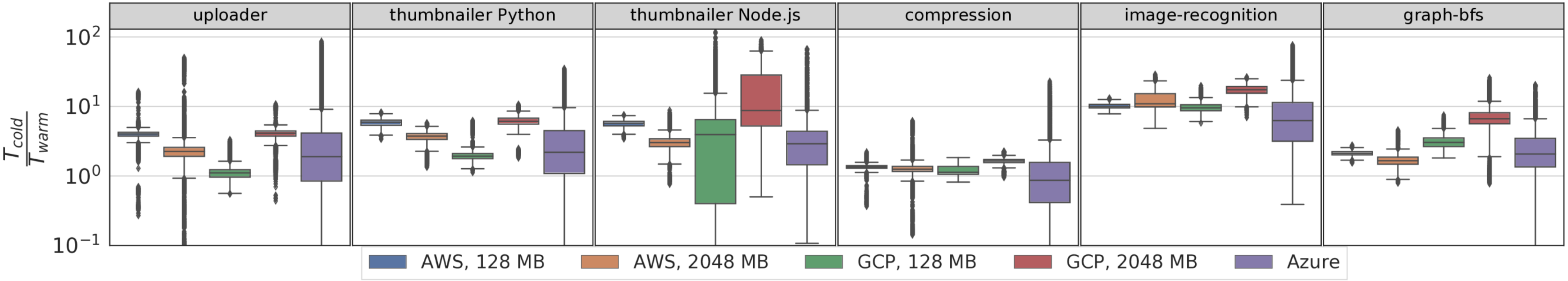
Performance Analysis: Cold Startups



Configuration:

- **Measurements:** 200 warm and 200 cold executions.
- **Estimation:** all N^2 combinations of N warm and N cold executions.
- **Azure Functions**: mixed cold and warm executions.

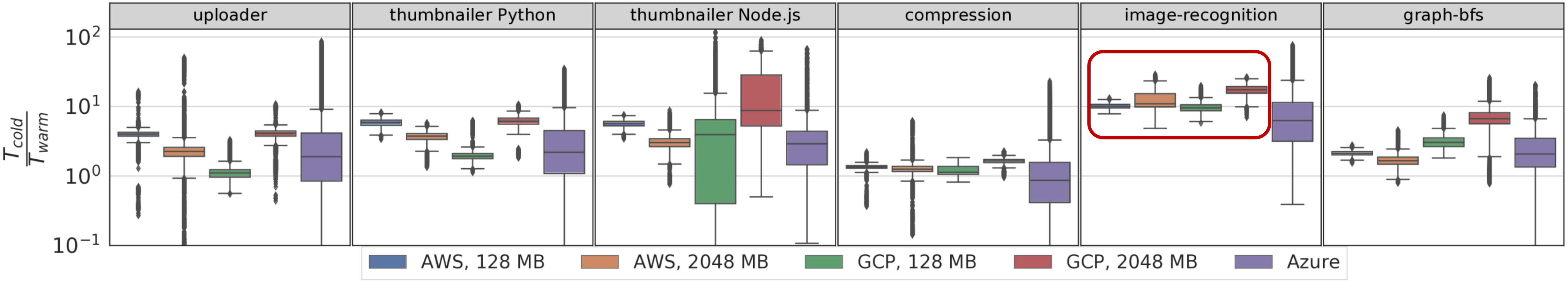
Performance Analysis: Cold Startups



Configuration:

- **Measurements:** 200 warm and 200 cold executions.
- **Estimation:** all N^2 combinations of N warm and N cold executions.
- **Azure Functions**: mixed cold and warm executions.

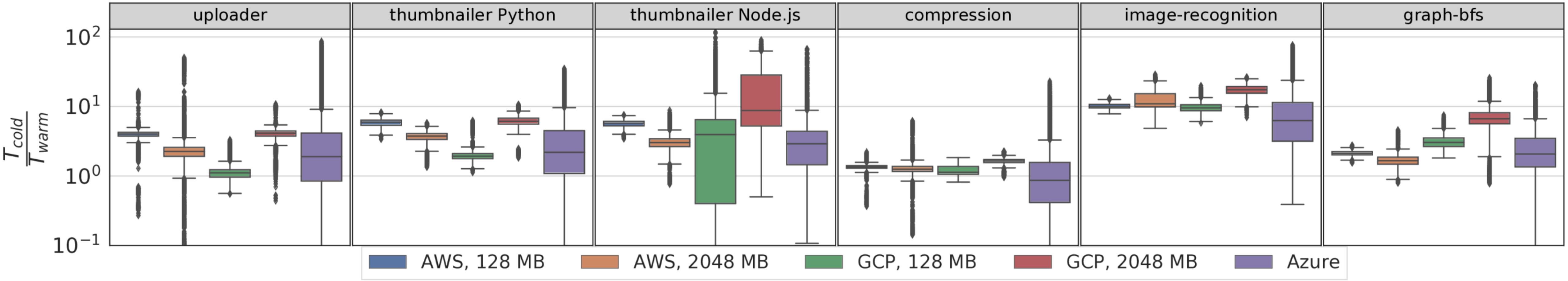
Performance Analysis: Cold Startups



Configuration:

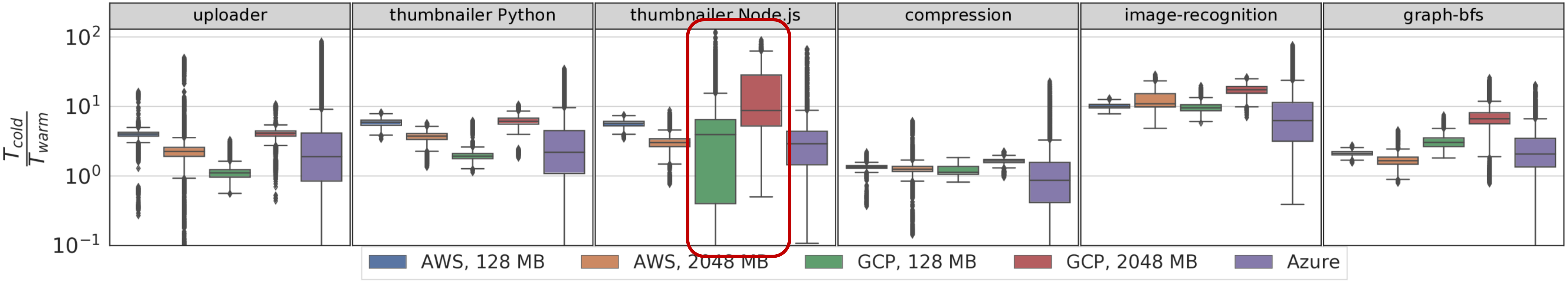
- **Measurements:** 200 warm and 200 cold executions.
- **Estimation:** all N^2 combinations of N warm and N cold executions.
- **Azure Functions**: mixed cold and warm executions.

Performance Analysis: Cold Startups



Configuration:
- **Measurements:** 200 warm and 200 cold executions.
- **Estimation:** all N^2 combinations of N warm and N cold executions.
- **Azure Functions**: mixed cold and warm executions.

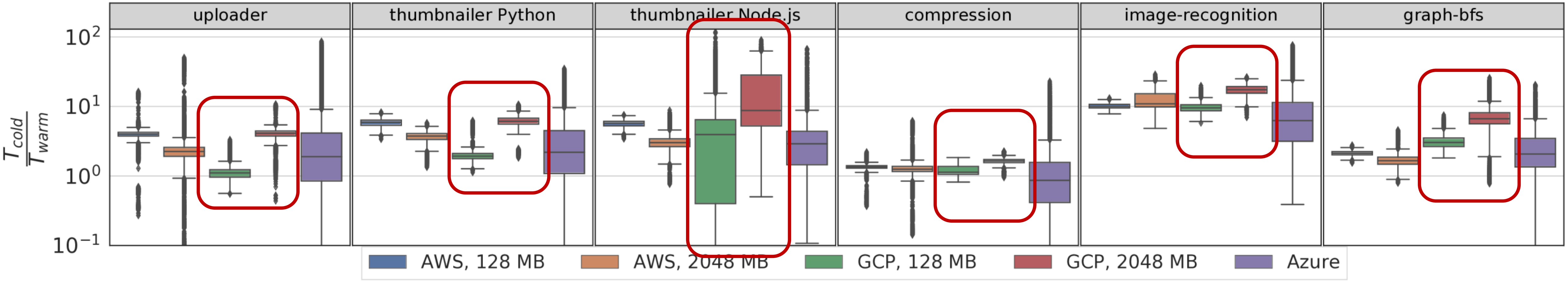
Performance Analysis: Cold Startups



Configuration:

- **Measurements:** 200 warm and 200 cold executions.
- **Estimation:** all N^2 combinations of N warm and N cold executions.
- **Azure Functions**: mixed cold and warm executions.

Performance Analysis: Cold Startups



Configuration:
- **Measurements:** 200 warm and 200 cold executions.
- **Estimation:** all N^2 combinations of N warm and N cold executions.
- **Azure Functions**: mixed cold and warm executions.

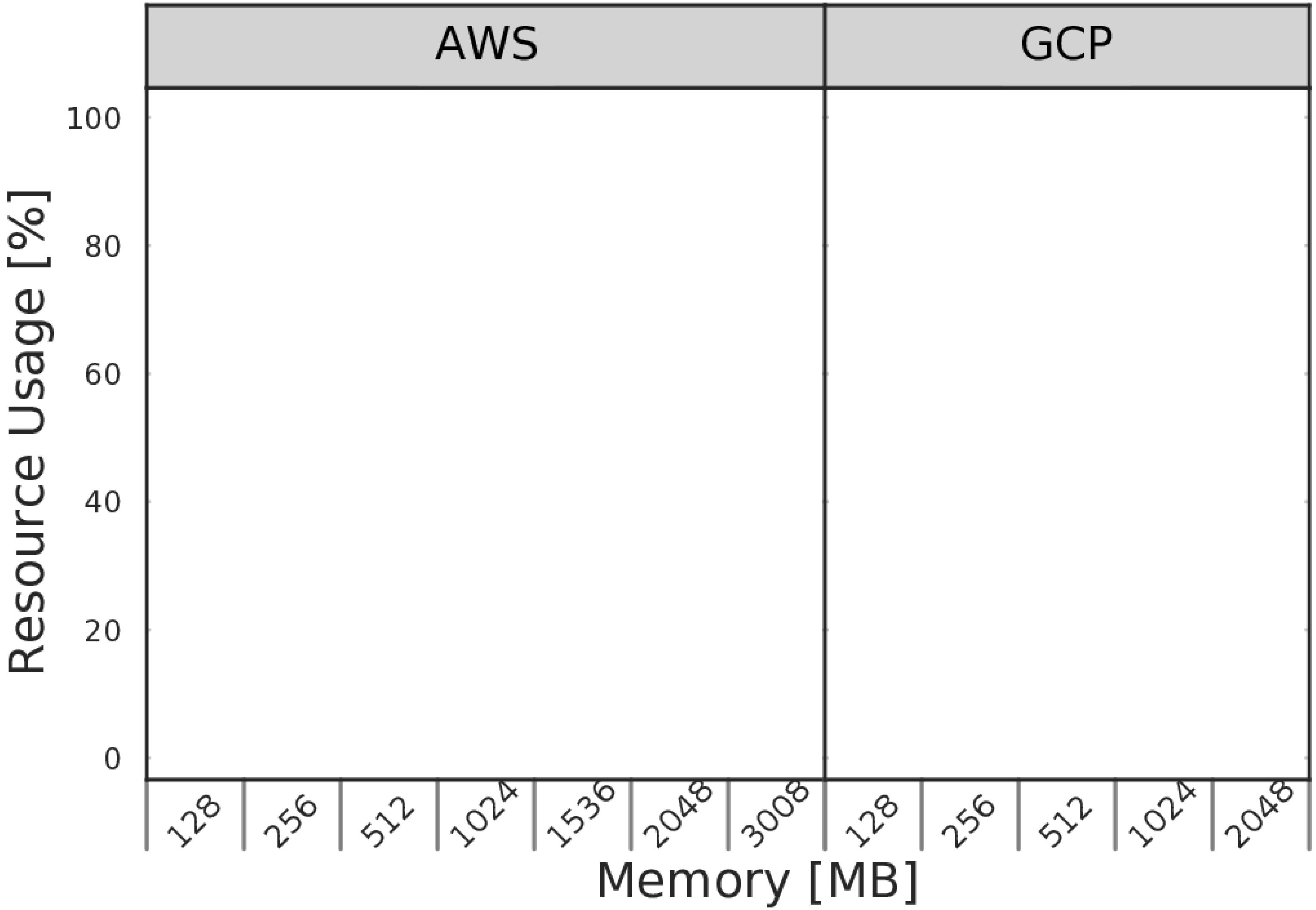
Results and Insights

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
$	

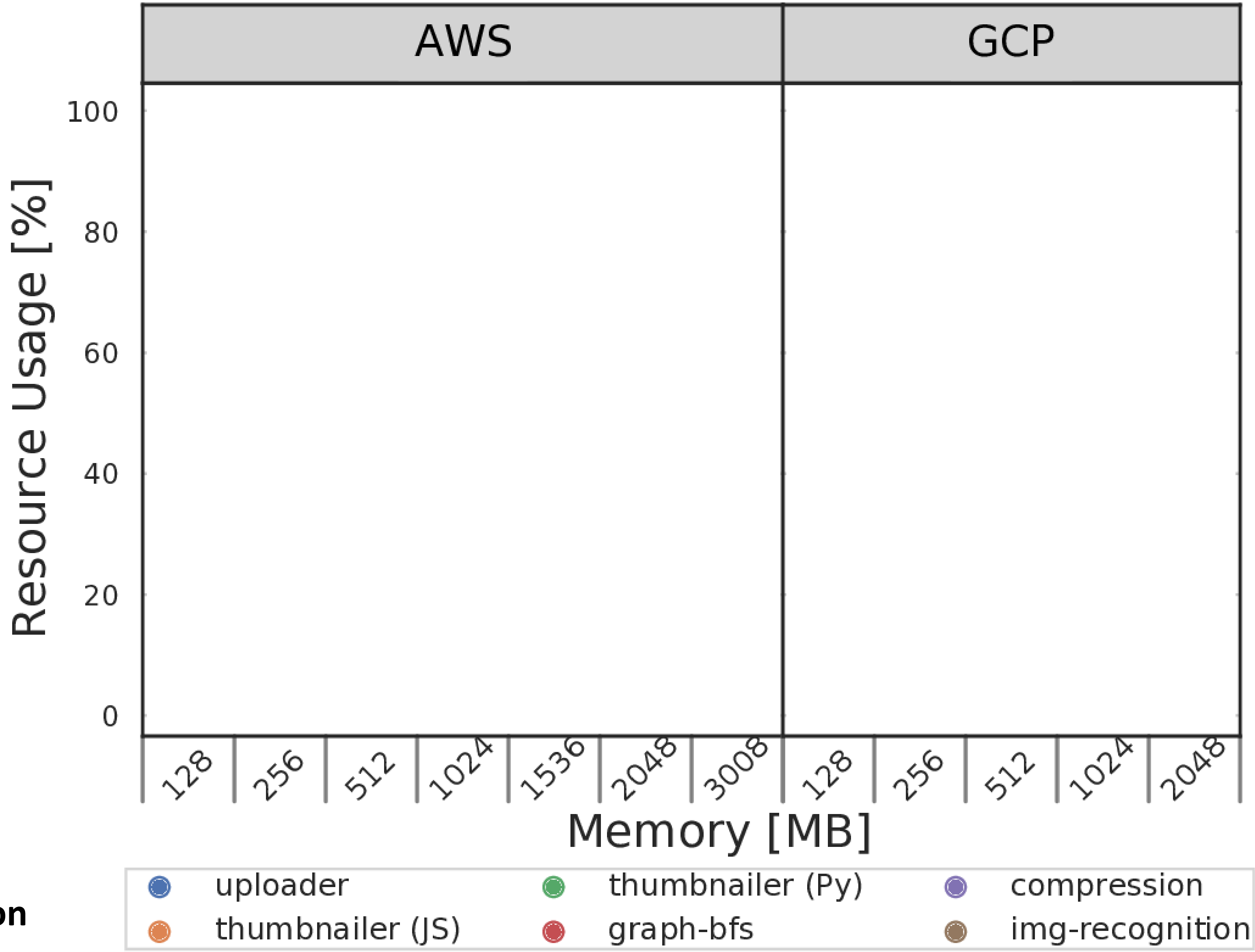
Results and Insights

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
$	Resource underutilization due to high granularity of pricing models.

Cost Analysis: Resource Usage



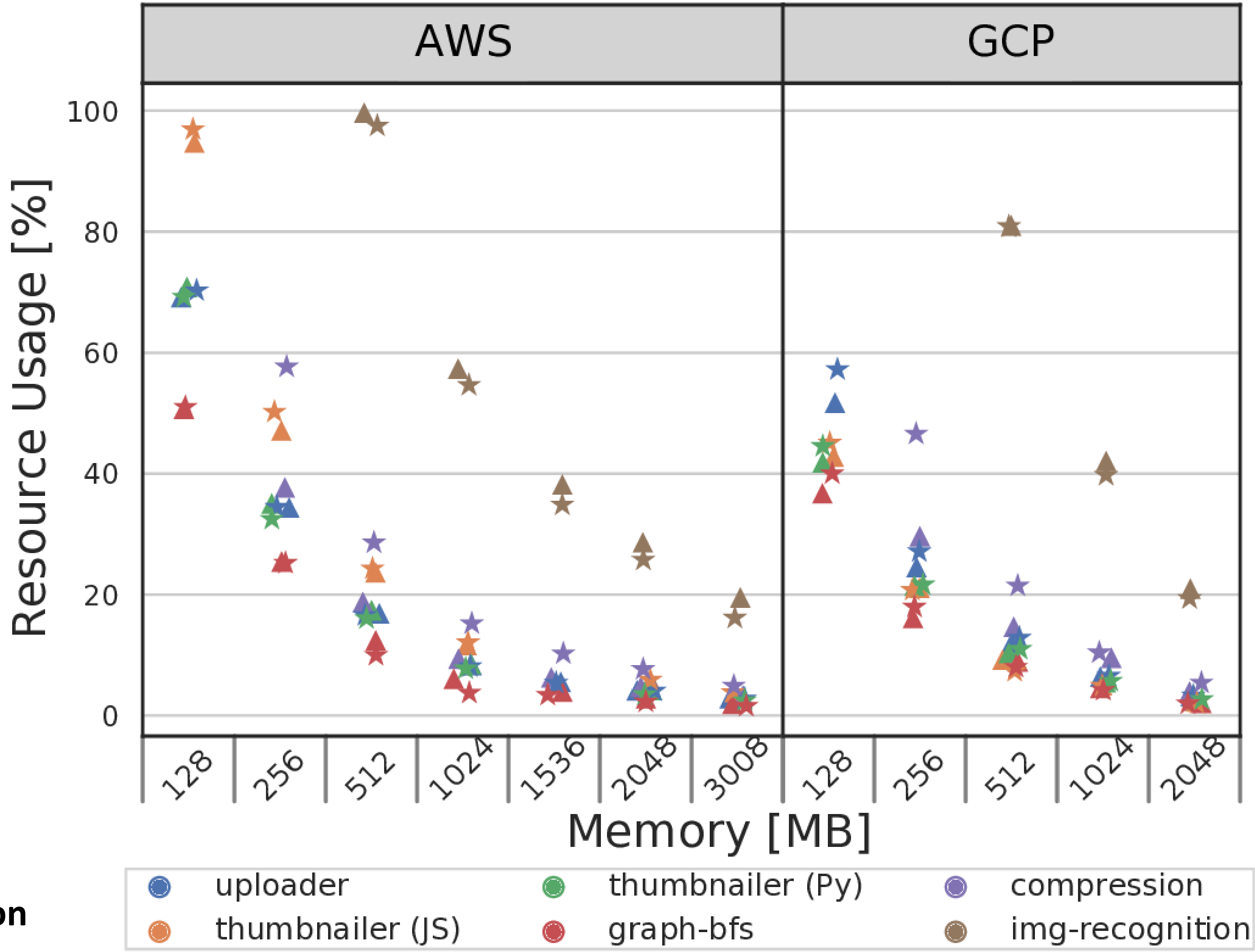
Cost Analysis: Resource Usage



Cold Execution

Warm Execution

Cost Analysis: Resource Usage



Cold Execution

Warm Execution

Results and Insights

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
$	Resource underutilization due to high granularity of pricing models.

Results and Insights

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
	Resource underutilization due to high granularity of pricing models.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.

FaaS Analysis: Eviction Modeling

FaaS Analysis: Eviction Modeling

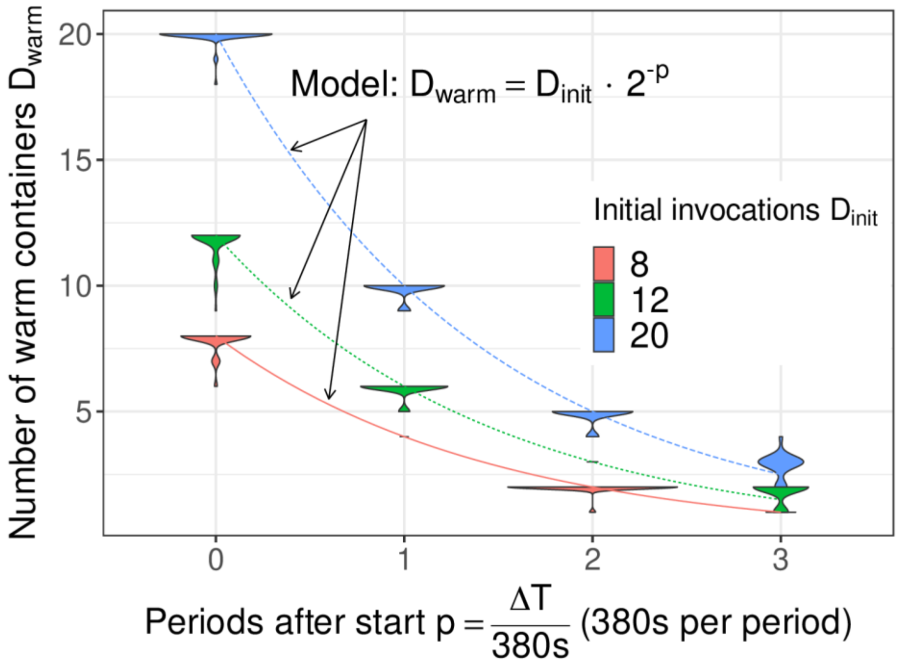
Configuration:

- **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

FaaS Analysis: Eviction Modeling

Configuration:

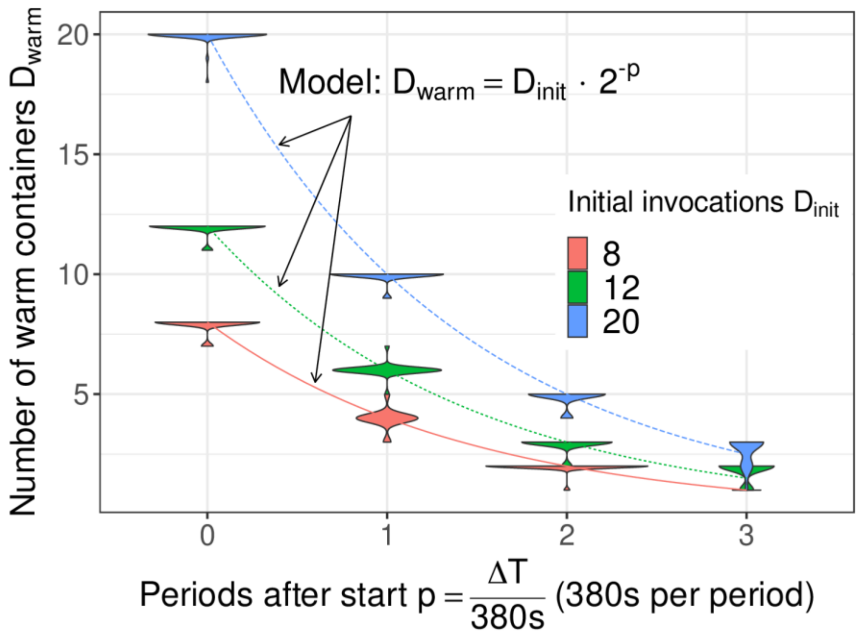
- **Time between invocations:** 1 – 1600s.
- **# of function instances:** 1 -20
- **Memory:** 128 – 1536 MB
- **Package size:** 8 kB, 250 MB
- **Duration:** 1 – 10s
- **Language:** Python, Node.js

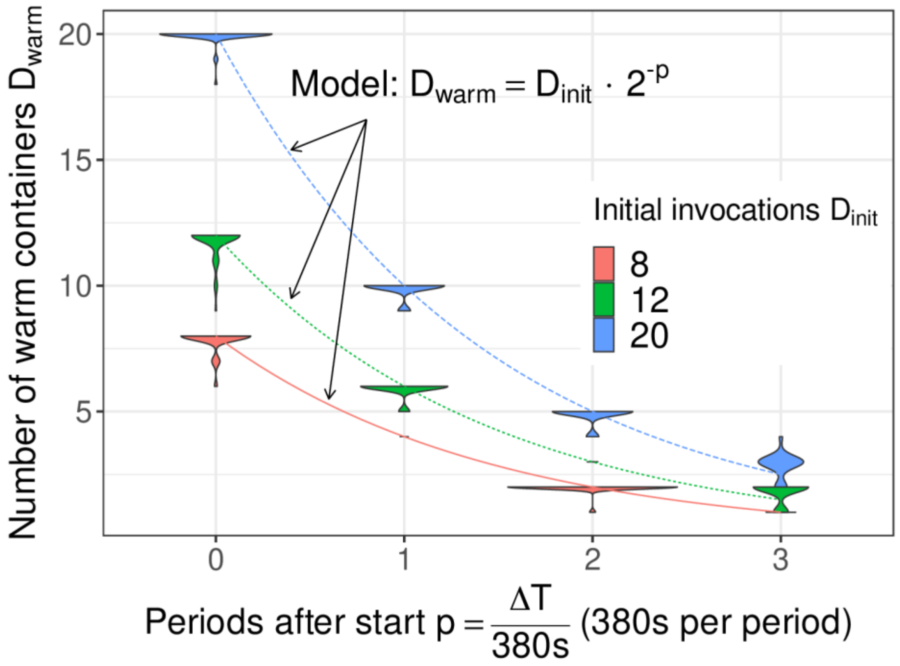


FaaS Analysis: Eviction Modeling

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

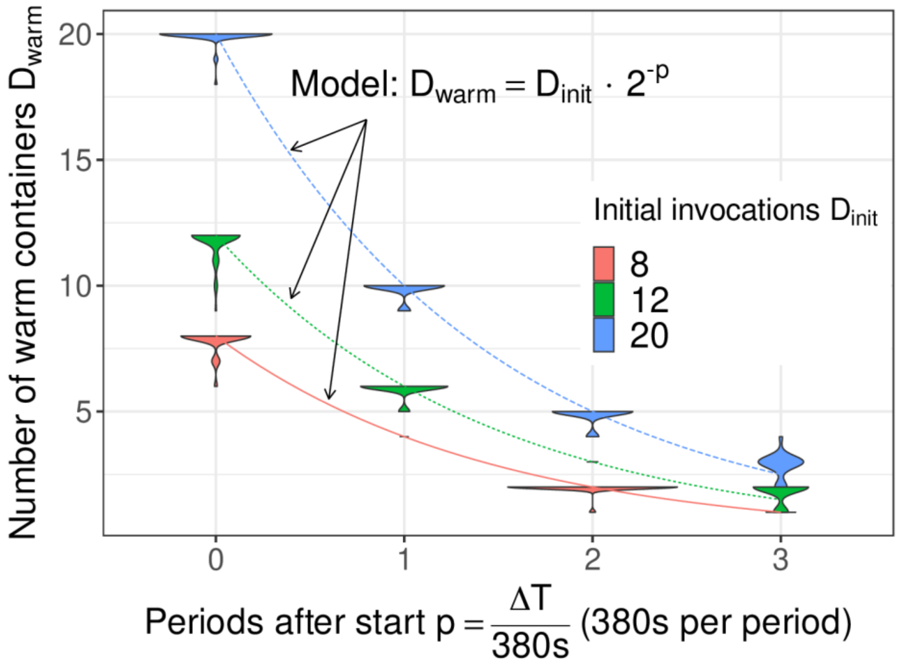


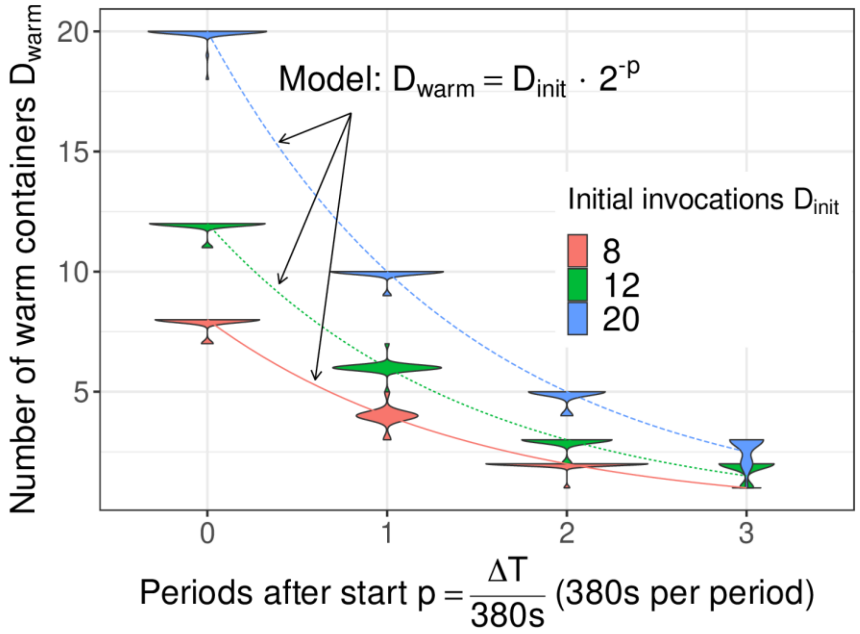


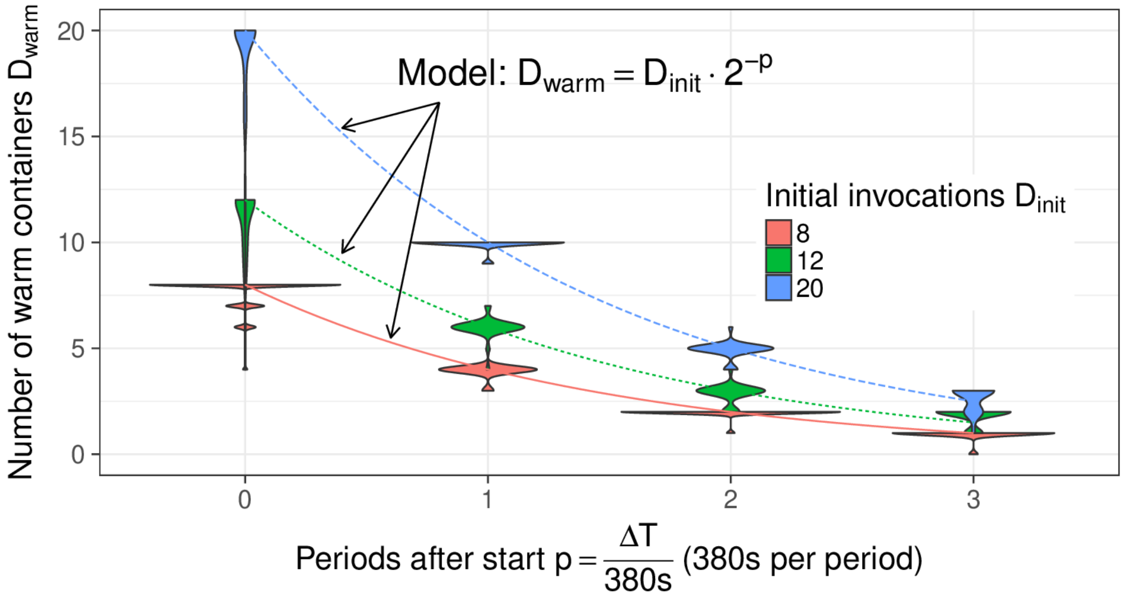
FaaS Analysis: Eviction Modeling

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







Results and Insights

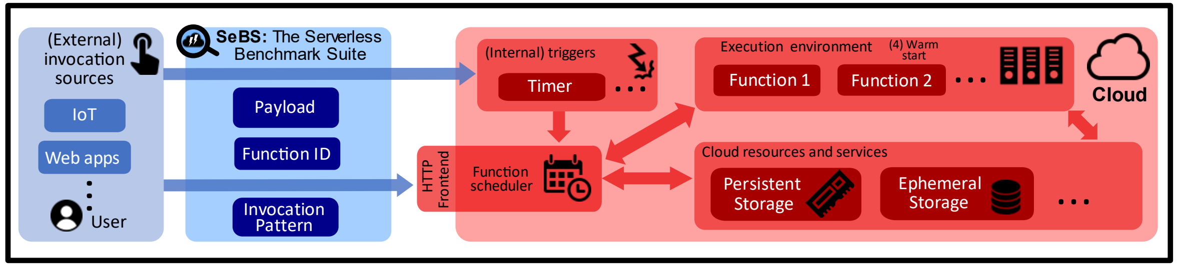
	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
$	Resource underutilization due to high granularity of pricing models.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.

Results and Insights

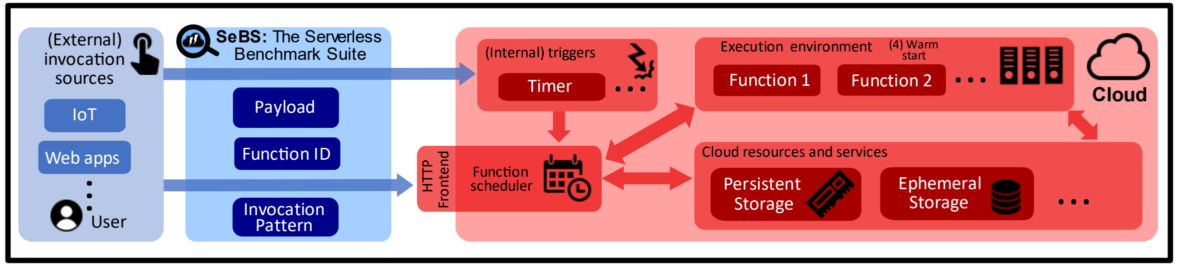
	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
	I/O-bound functions experience very high latency variations.
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
	Resource underutilization due to high granularity of pricing models.
$	Break-even analysis for IaaS and FaaS deployment.
	High costs of Azure Functions due to unconfigurable deployment.
	The function output size can be a dominating factor in pricing.
	Accurate methodology for estimation of invocation latency.
	Warm latencies are consistent and depend linearly on payload size.
	Highly variable and unpredictable cold latencies on Azure and GCP.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.

Summary

Summary

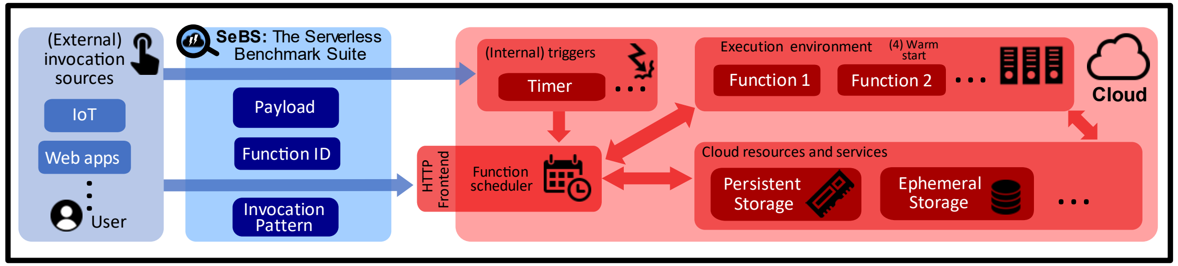

(External) invocation sources
IoT
Web apps
User
SeBS: The Serverless Benchmark Suite
Payload
Function ID
Invocation Pattern
HTTP Frontend
(Internal) triggers
Timer
Function scheduler
Execution environment
(4) Warm start
Function 1
Function 2
Cloud
Cloud resources and services
Persistent Storage
Ephemeral Storage

Summary



Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++
Scientific	graph-bfs	**Python**

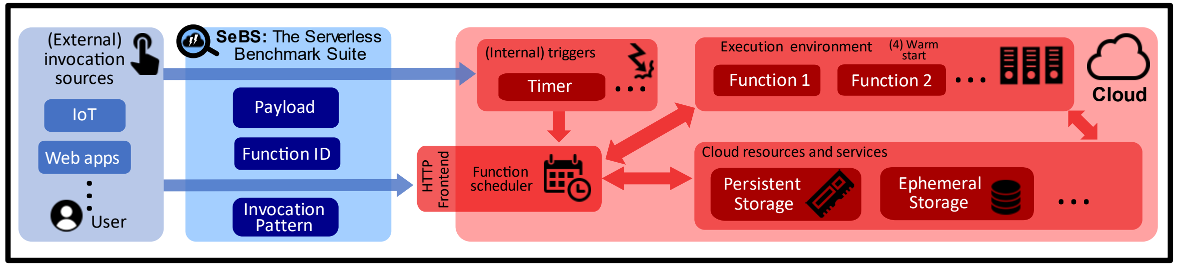
Summary



Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++
Scientific	graph-bfs	**Python**

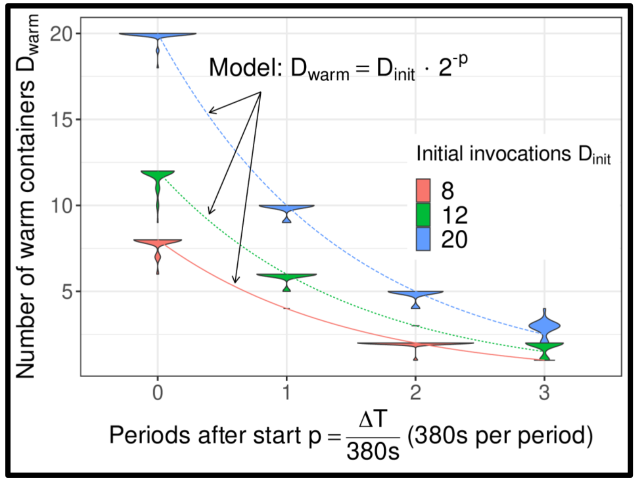
	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
	I/O-bound functions experience very high latency variations.
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	**Break-even analysis for IaaS and FaaS deployment.**
	Resource underutilization due to high granularity of pricing models.
	High costs of Azure Functions due to unconfigurable deployment.
	The function output size can be a dominating factor in pricing.
	Accurate methodology for estimation of invocation latency.
	Warm latencies are consistent and depend linearly on payload size.
	Highly variable and unpredictable cold latencies on Azure and GCP.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.

Summary

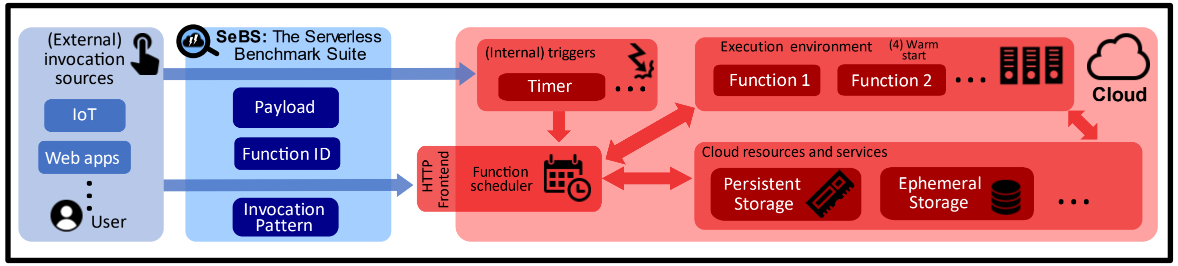


Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++
Scientific	graph-bfs	**Python**

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
	I/O-bound functions experience very high latency variations.
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	**Break-even analysis for IaaS and FaaS deployment.**
	Resource underutilization due to high granularity of pricing models.
	High costs of Azure Functions due to unconfigurable deployment.
	The function output size can be a dominating factor in pricing.
	Accurate methodology for estimation of invocation latency.
	Warm latencies are consistent and depend linearly on payload size.
	Highly variable and unpredictable cold latencies on Azure and GCP.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.

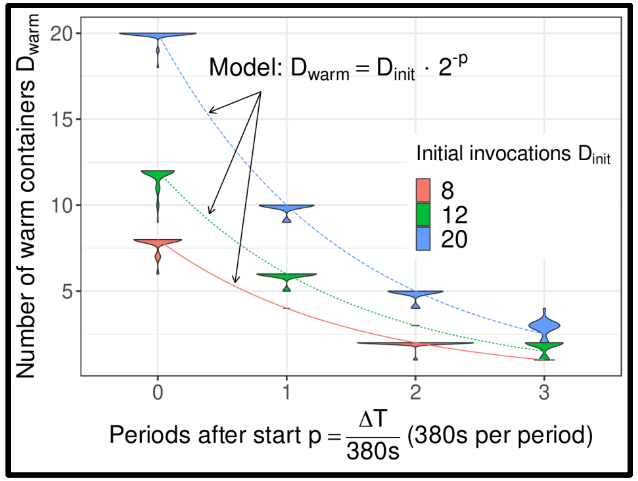


Summary



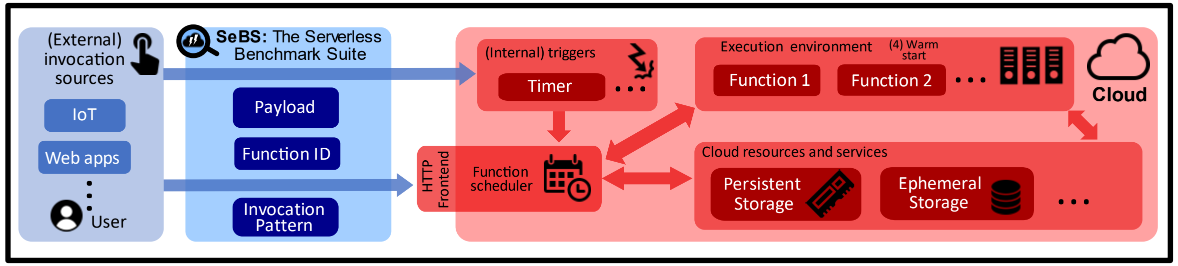
Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++
Scientific	graph-bfs	**Python**

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
	I/O-bound functions experience very high latency variations.
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	**Break-even analysis for IaaS and FaaS deployment.**
	Resource underutilization due to high granularity of pricing models.
	High costs of Azure Functions due to unconfigurable deployment.
	The function output size can be a dominating factor in pricing.
	Accurate methodology for estimation of invocation latency.
	Warm latencies are consistent and depend linearly on payload size.
	Highly variable and unpredictable cold latencies on Azure and GCP.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.



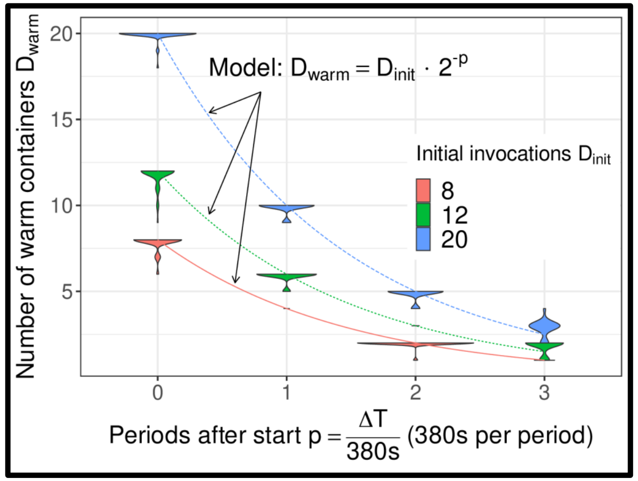
spcl/serverless-benchmarks

Summary



Type	Name	Language
Webapps	uploader	**Python**, Node.js
Multimedia	thumbnailer	**Python**, **Node.js**, C++
Utilities	compression	**Python**
Inference	image-recognition	**Python**, C++
Scientific	graph-bfs	**Python**

	Results, methods, and insights
	High-memory allocations increase cold startup overheads on GCP.
	GCP functions experience reliability and availability issues.
	I/O-bound functions experience very high latency variations.
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	**Break-even analysis for IaaS and FaaS deployment.**
	Resource underutilization due to high granularity of pricing models.
	High costs of Azure Functions due to unconfigurable deployment.
	The function output size can be a dominating factor in pricing.
	Accurate methodology for estimation of invocation latency.
	Warm latencies are consistent and depend linearly on payload size.
	Highly variable and unpredictable cold latencies on Azure and GCP.
	AWS Lambda container eviction is agnostic to function properties.
	Analytical models of AWS Lambda container eviction policy.



spcl/serverless-benchmarks

- OpenWhisk
- C++ Functions
- Serverless Workflows

Q&A

Future Work

Questions

Q&A

Future Work

Questions

spcl/serverless-benchmarks

Q&A

Future Work

- OpenWhisk
- C++ Functions
- Serverless Workflows

Questions

spcl/serverless-benchmarks

Q&A

Future Work

- OpenWhisk
- C++ Functions
- Serverless Workflows

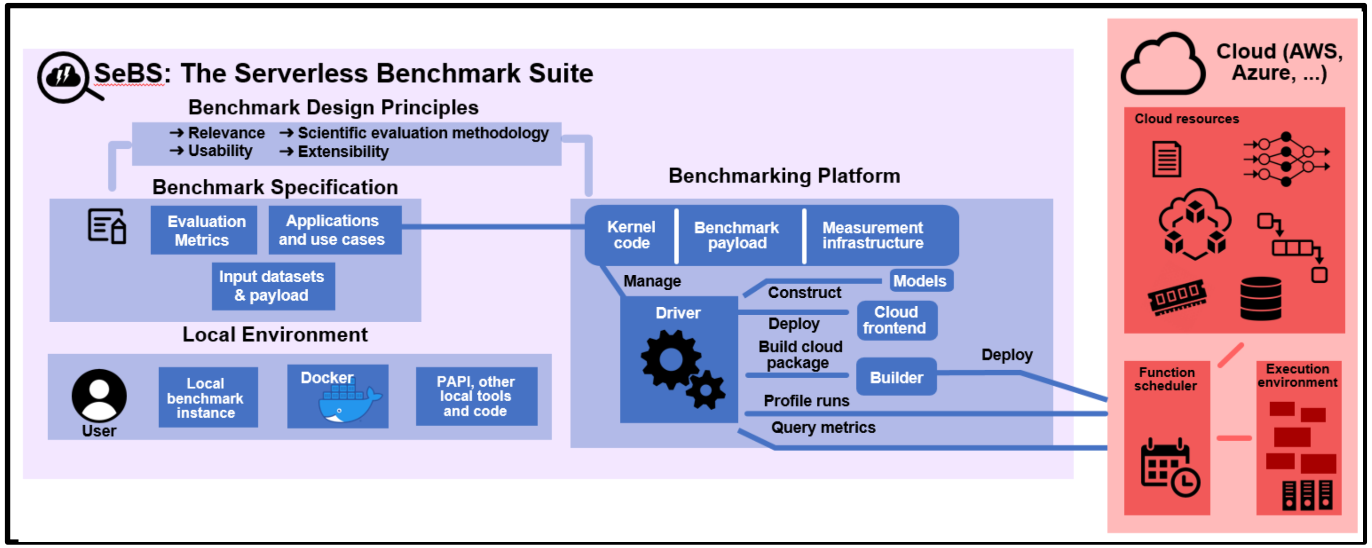
Questions

- What are the requirements for a good benchmark suite?
- How can we measure function invocation latency accurately?
- How much performance do we lose when switching from IaaS to FaaS?

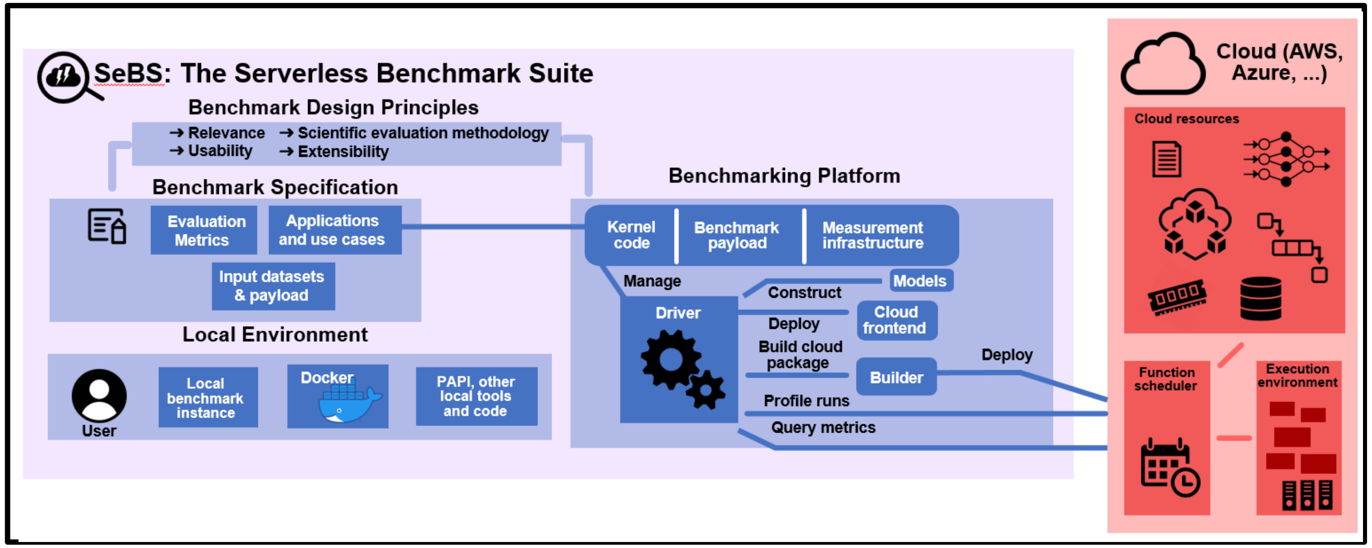
spcl/serverless-benchmarks

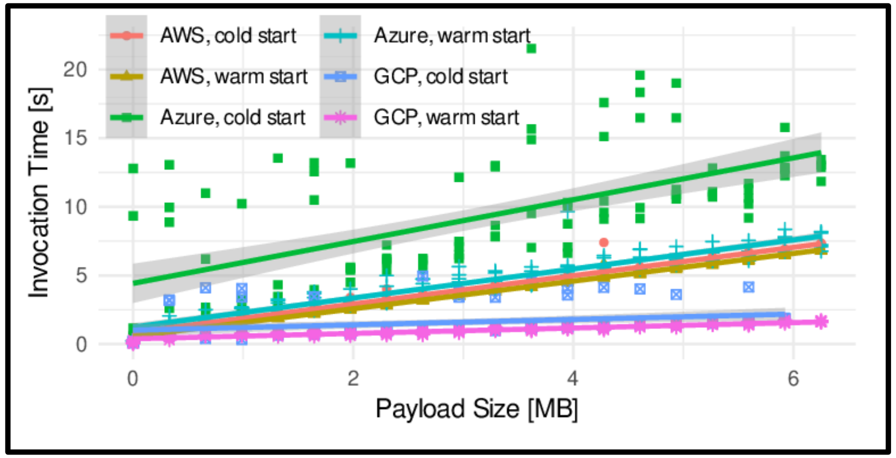
SeBS in details…

SeBS in details…

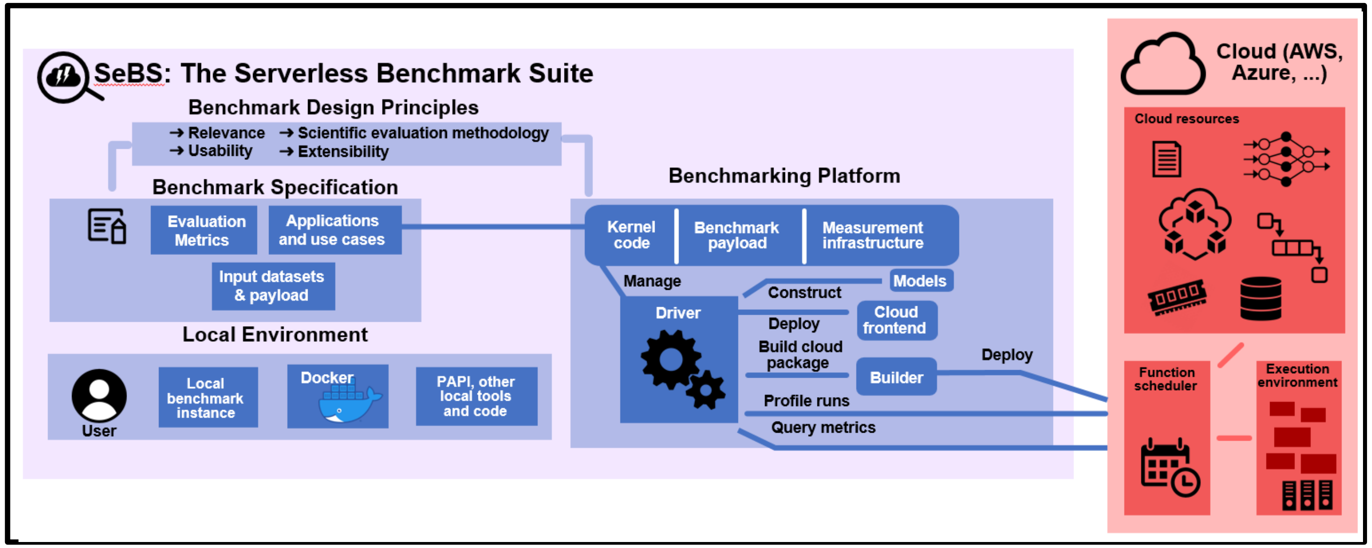


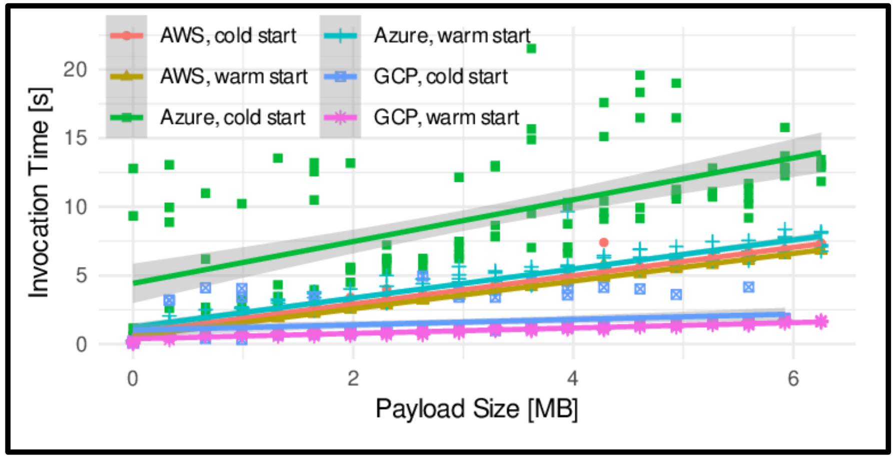
SeBS in details…

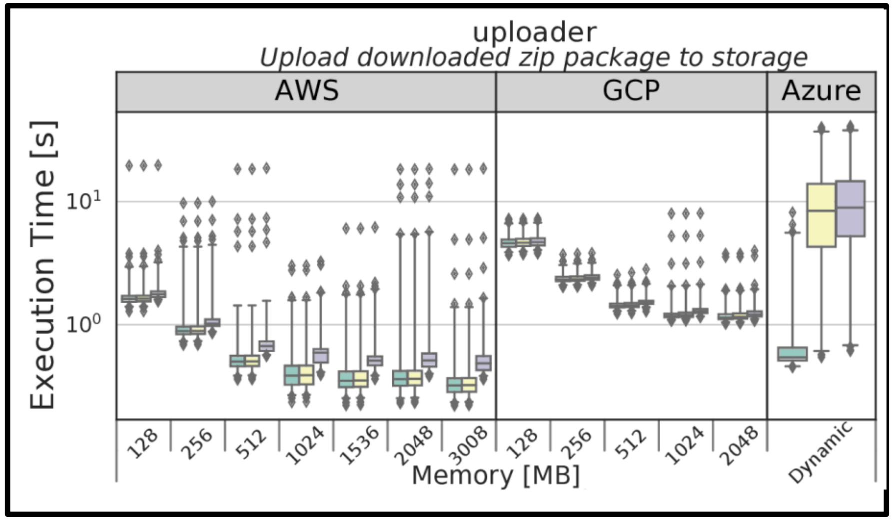




SeBS in details…

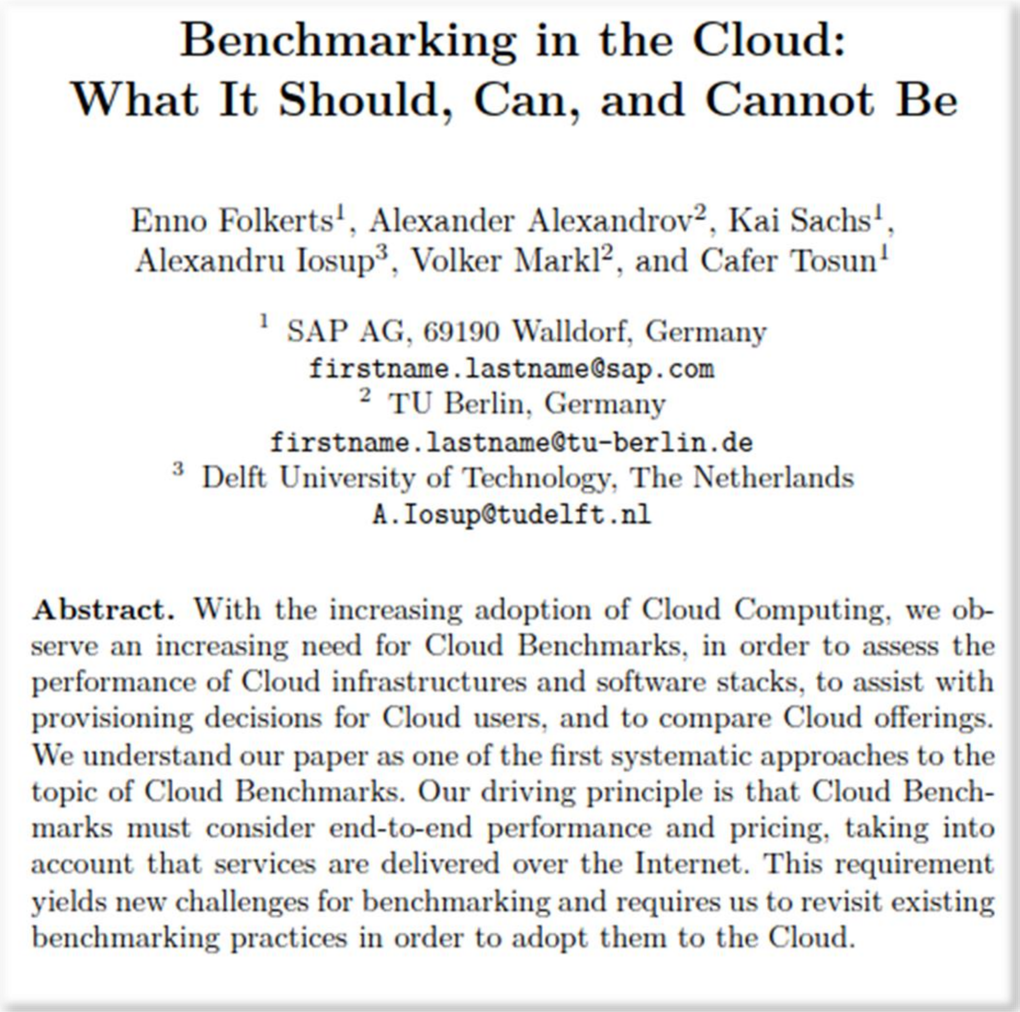






How to build a benchmark?

How to build a benchmark?



Benchmarking in the Cloud:
What It Should, Can, and Cannot Be

Enno Folkerts[1], Alexander Alexandrov[2], Kai Sachs[1],
Alexandru Iosup[3], Volker Markl[2], and Cafer Tosun[1]

[1] SAP AG, 69190 Walldorf, Germany
firstname.lastname@sap.com
[2] TU Berlin, Germany
firstname.lastname@tu-berlin.de
[3] Delft University of Technology, The Netherlands
A.Iosup@tudelft.nl

Abstract. With the increasing adoption of Cloud Computing, we observe an increasing need for Cloud Benchmarks, in order to assess the performance of Cloud infrastructures and software stacks, to assist with provisioning decisions for Cloud users, and to compare Cloud offerings. We understand our paper as one of the first systematic approaches to the topic of Cloud Benchmarks. Our driving principle is that Cloud Benchmarks must consider end-to-end performance and pricing, taking into account that services are delivered over the Internet. This requirement yields new challenges for benchmarking and requires us to revisit existing benchmarking practices in order to adopt them to the Cloud.

How to build a benchmark?

Benchmarking in the Cloud: What It Should, Can, and Cannot Be

Enno Folkerts[1], Alexander Alexandrov[2], Kai Sachs[1], Alexandru Iosup[3], Volker Markl[2], and Cafer Tosun[1]

[1] SAP AG, 69190 Walldorf, Germany
firstname.lastname@sap.com
[2] TU Berlin, Germany
firstname.lastname@tu-berlin.de
[3] Delft University of Technology, The Netherlands
A.Iosup@tudelft.nl

Abstract. With the increasing adoption of Cloud Computing, we observe an increasing need for Cloud Benchmarks, in order to assess the performance of Cloud infrastructures and software stacks, to assist with provisioning decisions for Cloud users, and to compare Cloud offerings. We understand our paper as one of the first systematic approaches to the topic of Cloud Benchmarks. Our driving principle is that Cloud Benchmarks must consider end-to-end performance and pricing, taking into account that services are delivered over the Internet. This requirement yields new challenges for benchmarking and requires us to revisit existing benchmarking practices in order to adopt them to the Cloud.

How is the Weather tomorrow? Towards a Benchmark for the Cloud

Carsten Binnig   Donald Kossmann   Tim Kraska   Simon Loesing

Systems Group, Department of Computer Science, ETH Zurich
{firstname.lastname}@inf.ethz.ch

How to build a benchmark?

Benchmarking in the Cloud:
What It Should, Can, and Cannot Be

Enno Folkerts[1], Alexander Alexandrov[2], Kai Sachs[1], Alexandru Iosup[3], Volker Markl[2], and Cafer Tosun[1]

[1] SAP AG, 69190 Walldorf, Germany
firstname.lastname@sap.com
[2] TU Berlin, Germany
firstname.lastname@tu-berlin.de
[3] Delft University of Technology, The Netherlands
A.Iosup@tudelft.nl

Abstract. With the increasing adoption of Cloud Computing, we observe an increasing need for Cloud Benchmarks, in order to assess the performance of Cloud infrastructures and software stacks, to assist with provisioning decisions for Cloud users, and to compar
We understand our paper as one of the first systematic
topic of Cloud Benchmarks. Our driving principle is t
marks must consider end-to-end performance and pr
account that services are delivered over the Internet.
yields new challenges for benchmarking and requires us
benchmarking practices in order to adopt them to the

How is the Weather tomorrow?
Towards a Benchmark for the Cloud

Carsten Binnig Donald Kossmann Tim Kraska Simon Loesing

Systems Group, Department of Computer Science, ETH Zurich
{firstname.lastname}@inf.ethz.ch

Scientific Benchmarking of Parallel Computing Systems
Twelve ways to tell the masses when reporting performance results

Torsten Hoefler
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
htor@inf.ethz.ch

Roberto Belli
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
bellir@inf.ethz.ch

How to build a benchmark?

Benchmarking in the Cloud: What It Should, Can, and Cannot Be

Enno Folkert
Alexandru Io

[1] SA
f

firs
[3] Delft Uni

Abstract. With the
serve an increasing n
performance of Cloud
provisioning decisions
We understand our paper as one of the first systematic
topic of Cloud Benchmarks. Our driving principle is t
marks must consider end-to-end performance and pr
account that services are delivered over the Internet.
yields new challenges for benchmarking and requires us
benchmarking practices in order to adopt them to the

How is the Weather tomorrow? Towards a Benchmark for the Cloud

Tim Kraska Simon Loesing
mputer Science, ETH Zurich
e}@inf.ethz.ch

How to Build a Benchmark

Jóakim v. Kistowski
University of Würzburg
joakim.kistowski@uni-wuerzburg.de

Jeremy A. Arnold
IBM Corporation
arnoldje@us.ibm.com

Karl Huppler
karl.huppler@gmail.com

Klaus-Dieter Lange
Hewlett-Packard Company
klaus.lange@hp.com

John L. Henning
Oracle
john.henning@oracle.com

Paul Cao
Hewlett-Packard Company
paul.cao@hp.com

Scientific Benchmarking of Parallel Computing Systems

Twelve ways to tell the masses when reporting performance results

Torsten Hoefler
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
htor@inf.ethz.ch

Roberto Belli
Dept. of Computer Science
ETH Zurich
Zurich, Switzerland
bellir@inf.ethz.ch

Benchmarking Goals

Benchmarking Goals

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Benchmarking Goals

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Applications

- ✓ Realistic workloads
- ✓ Single implementation
- ✓ Varying computational characteristics

Benchmarking Goals

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Metrics

- ✓ Cloud time
- ✓ User time
- ✓ Resource utilization
- ✓ I/O

Applications

- ✓ Realistic workloads
- ✓ Single implementation
- ✓ Varying computational characteristics

Benchmarking Goals

Principles

- ✓ Usability
- ✓ Portability
- ✓ Extensibility
- ✓ Scientific

Applications

- ✓ Realistic workloads
- ✓ Single implementation
- ✓ Varying computational characteristics

Metrics

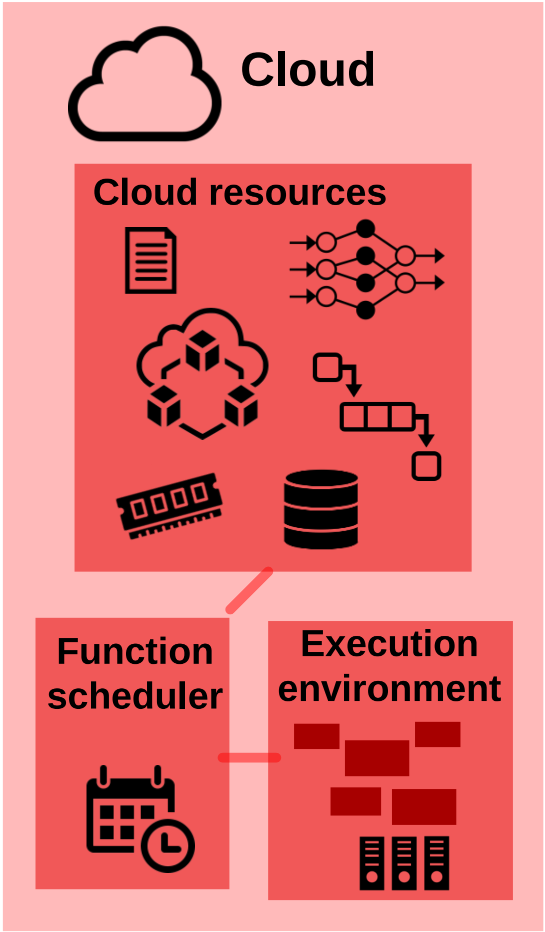
- ✓ Cloud time
- ✓ User time
- ✓ Resource utilization
- ✓ I/O

Experiments

- ✓ Performance and cost
- ✓ FaaS vs IaaS
- ✓ Invocation overhead
- ✓ Container eviction

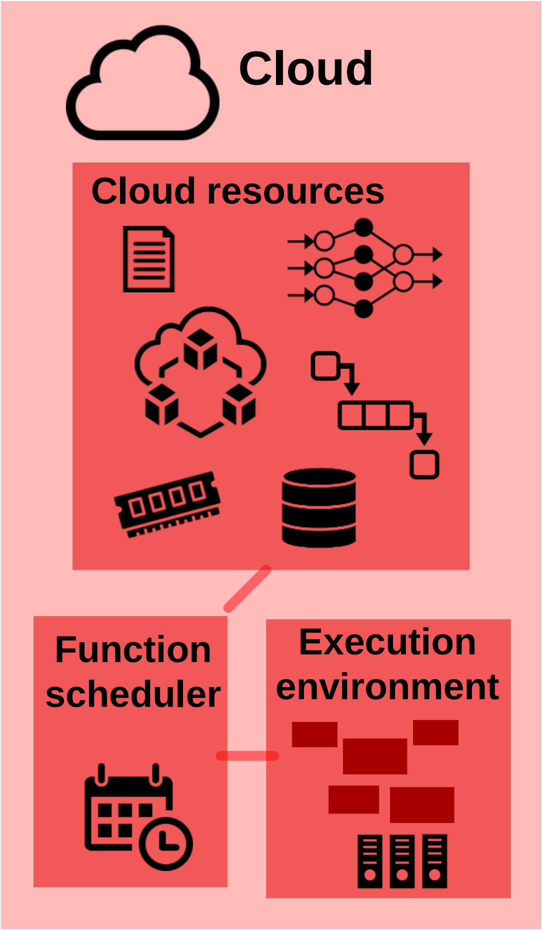
How to build a benchmark?

How to build a benchmark?



How to build a benchmark?

SeBS: The Serverless Benchmark Suite

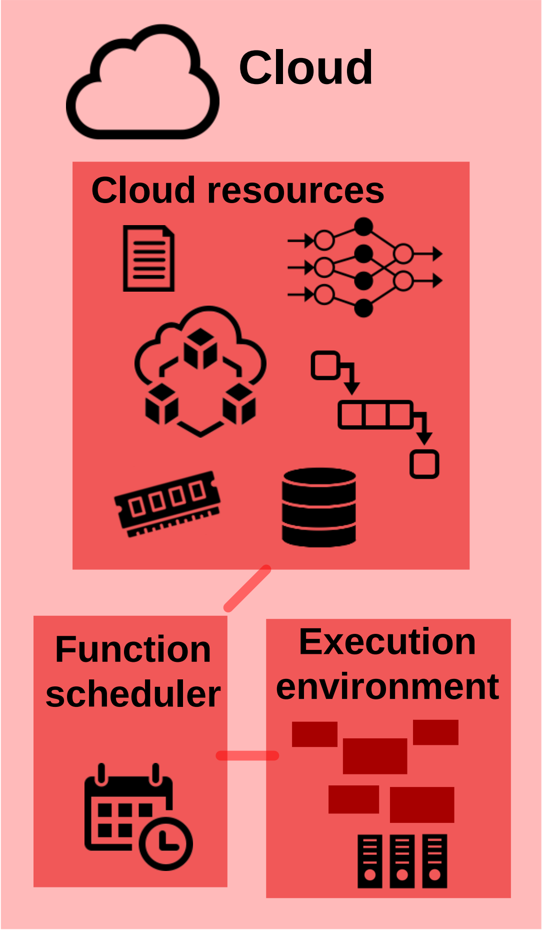


How to build a benchmark?

SeBS: The Serverless Benchmark Suite

Benchmark Design Principles

- ➜ Relevance
- ➜ Usability
- ➜ Scientific evaluation methodology
- ➜ Extensibility



How to build a benchmark?

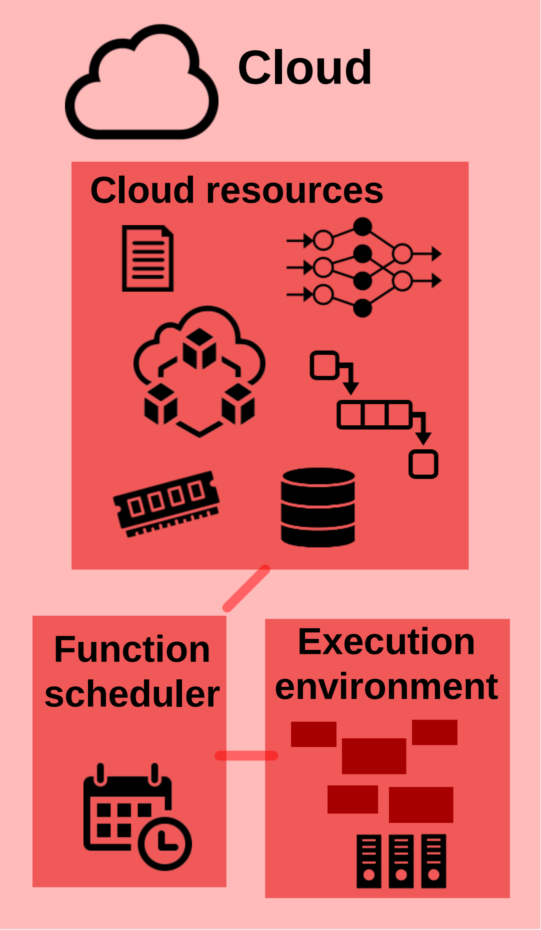
SeBS: The Serverless Benchmark Suite

Benchmark Design Principles

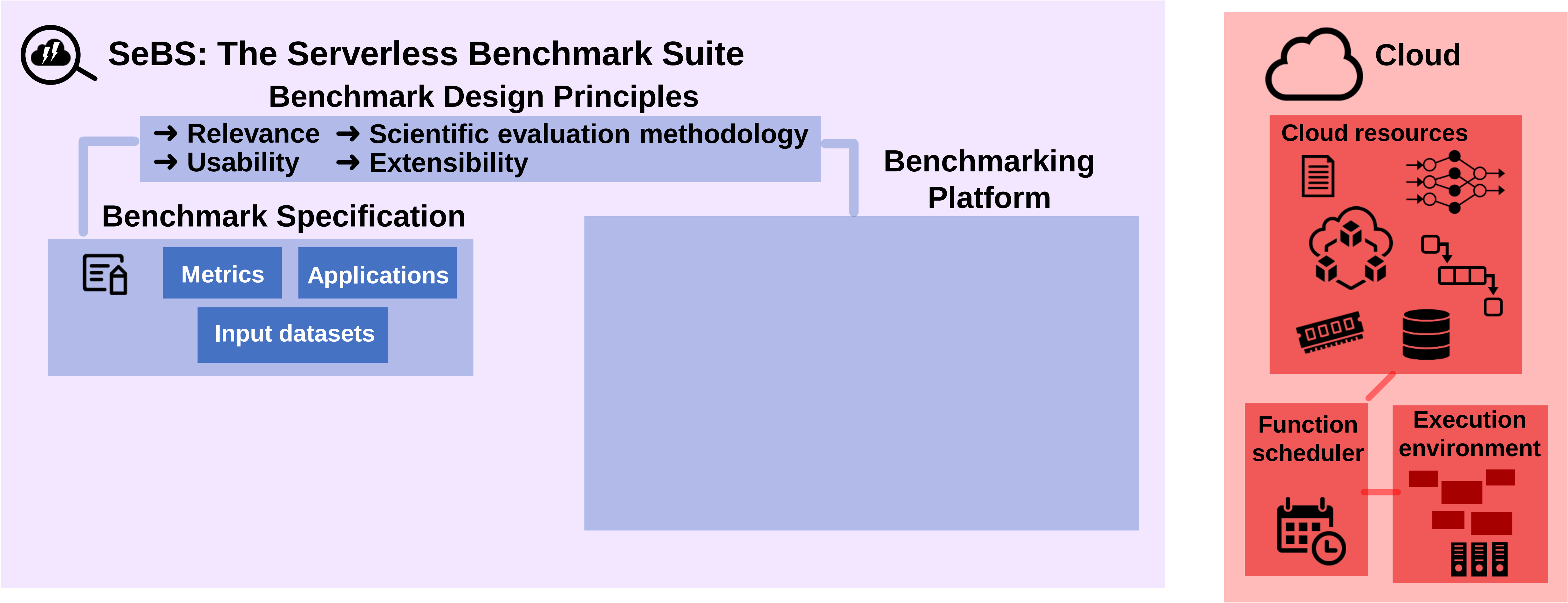
- ➜ Relevance
- ➜ Usability
- ➜ Scientific evaluation methodology
- ➜ Extensibility

Benchmark Specification

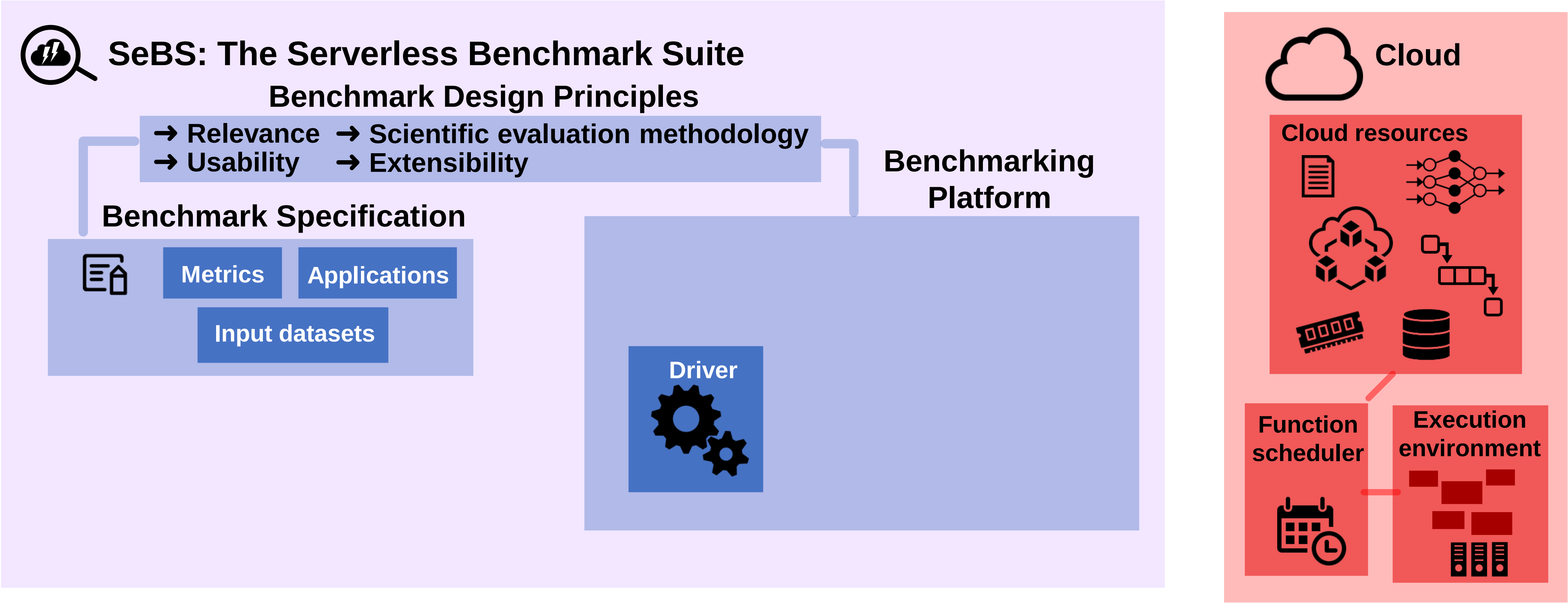
- Metrics
- Applications
- Input datasets



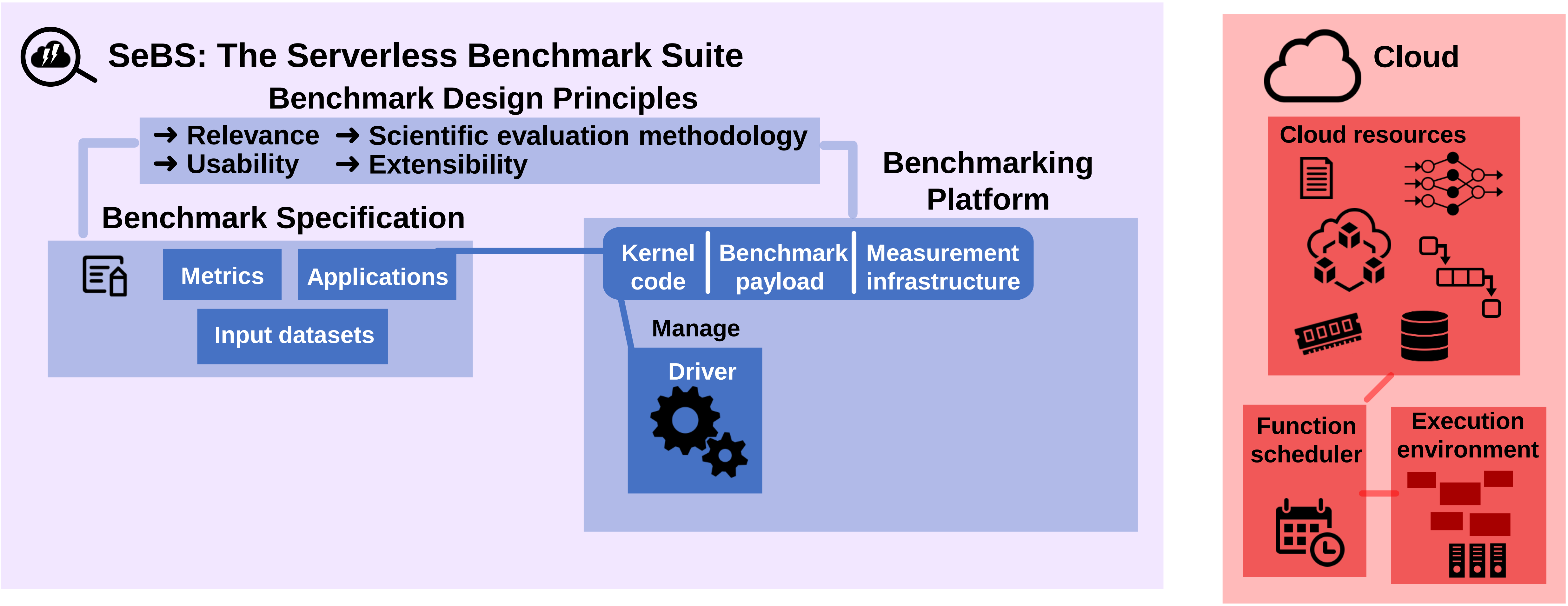
How to build a benchmark?



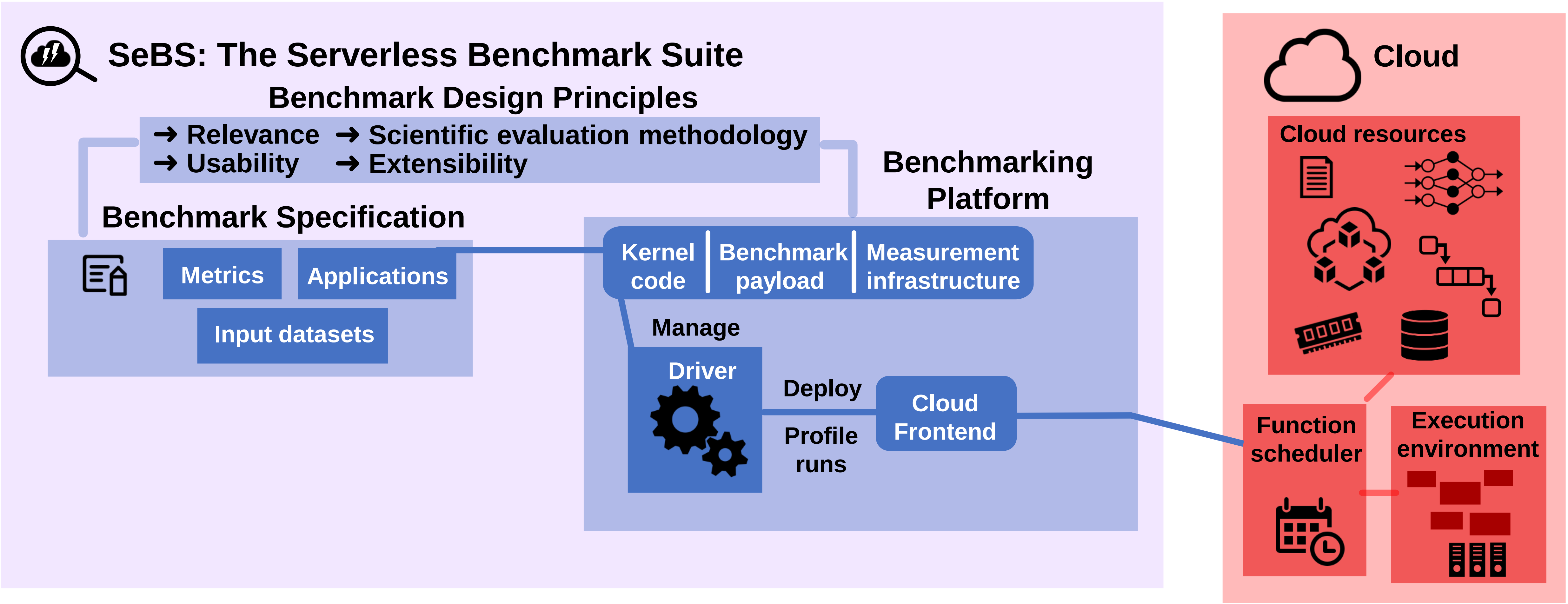
How to build a benchmark?



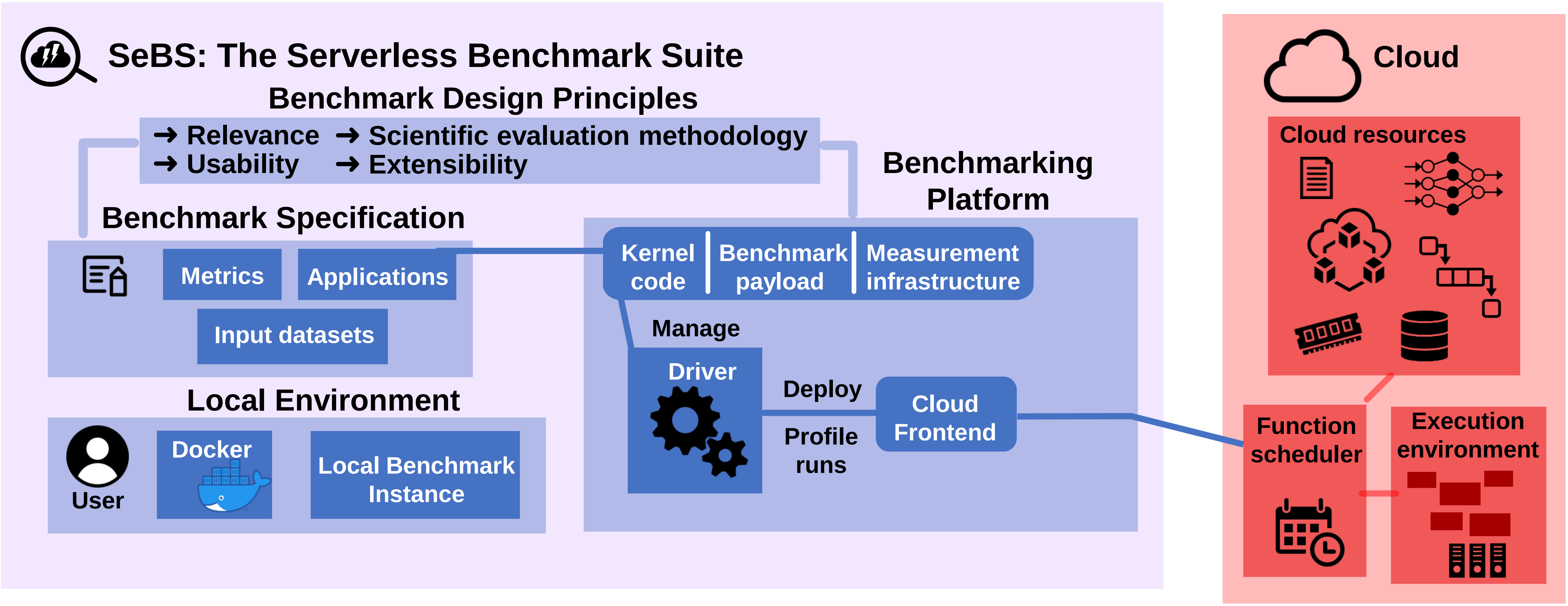
How to build a benchmark?



How to build a benchmark?



How to build a benchmark?



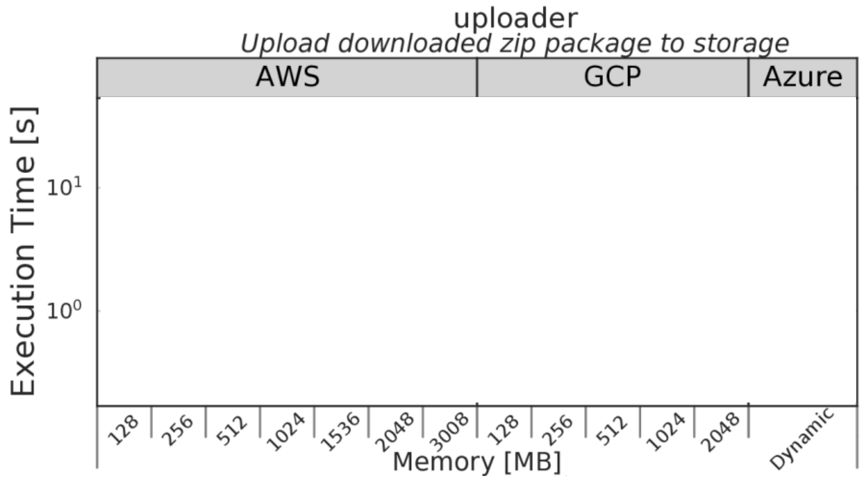
Results and Insights

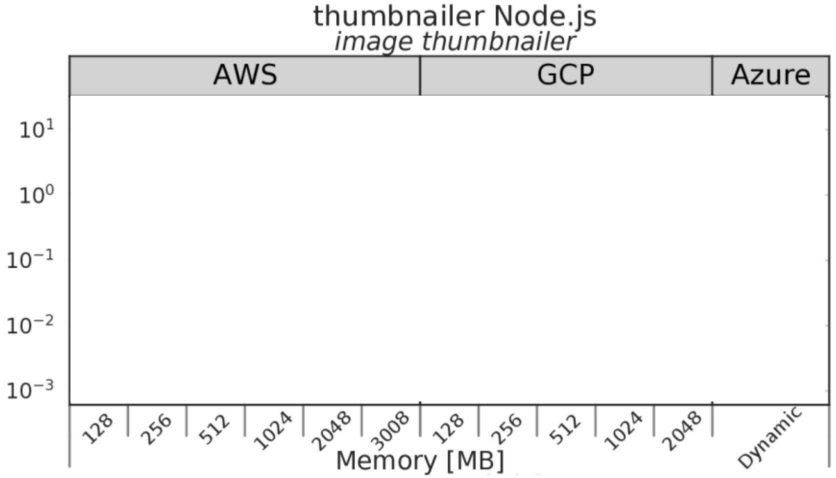
	Results, methods, and insights
$	

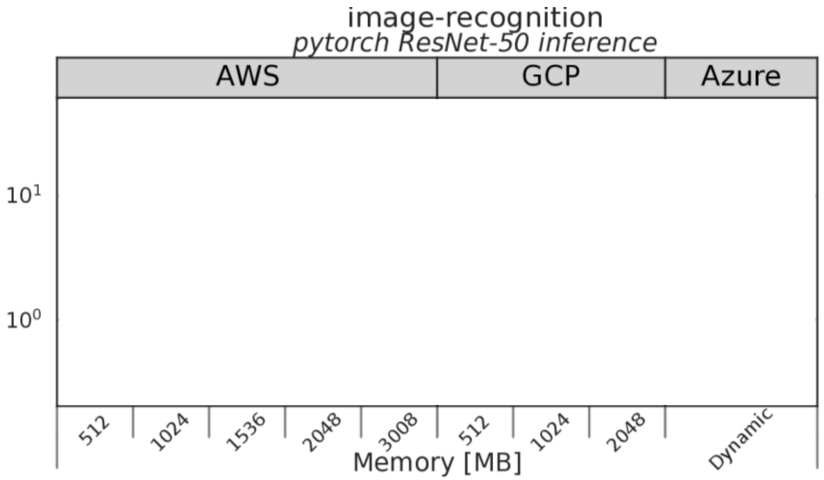
Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.

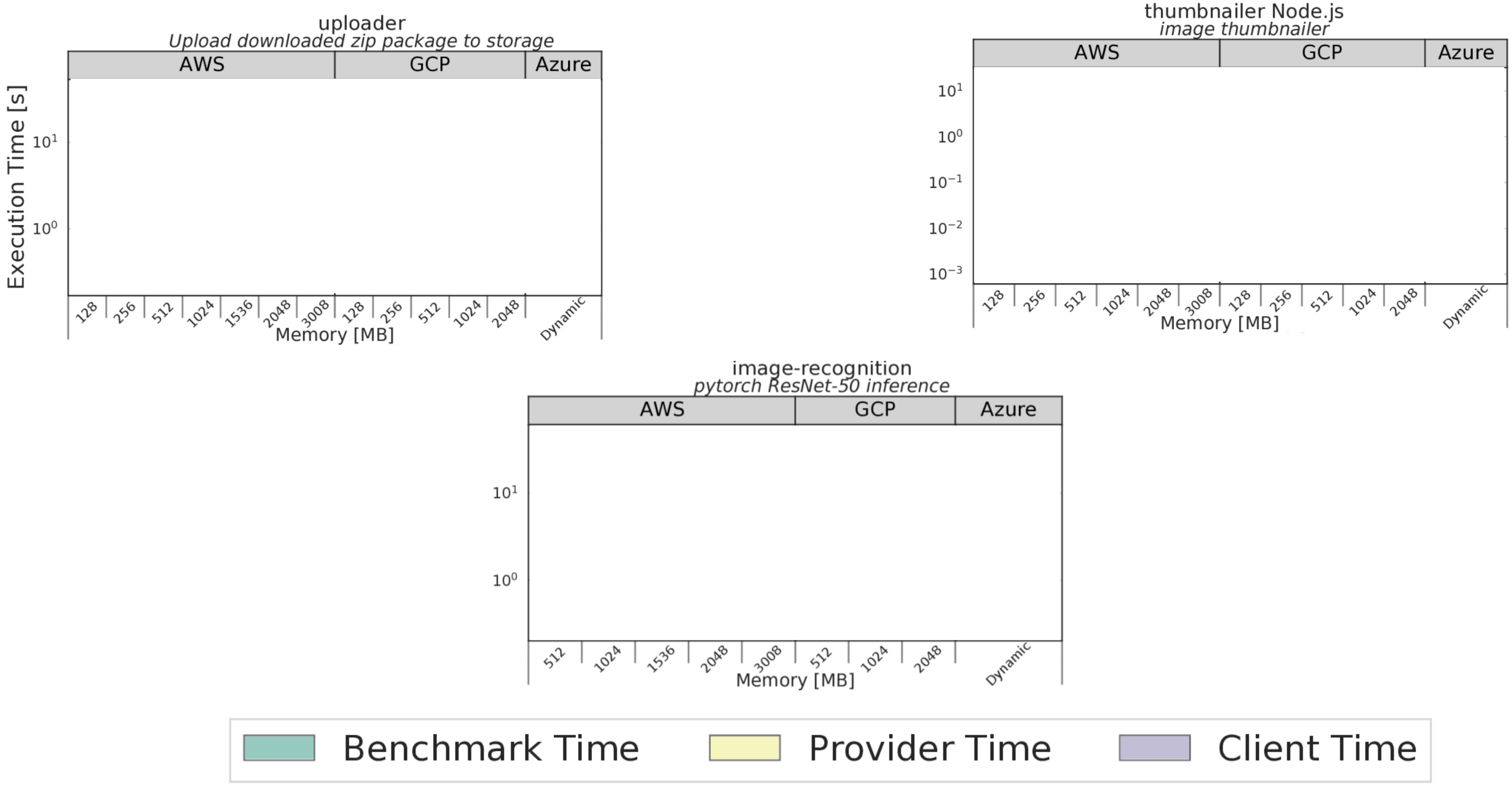
Performance Analysis: Warm Startups



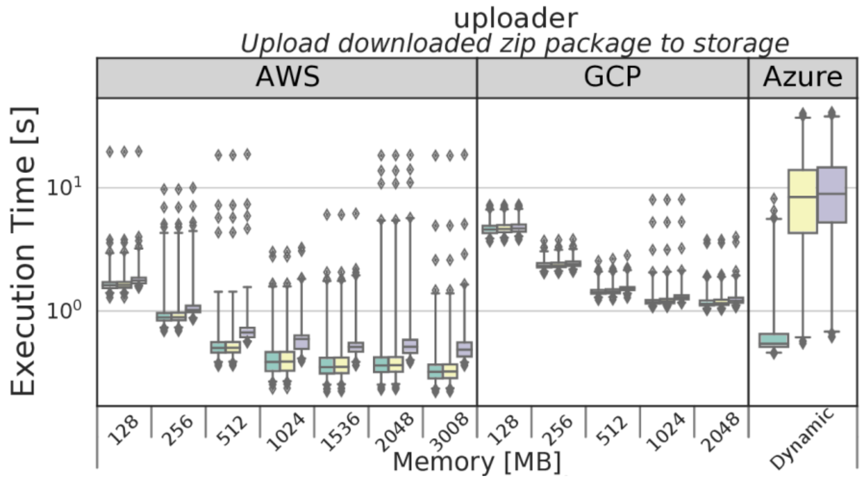


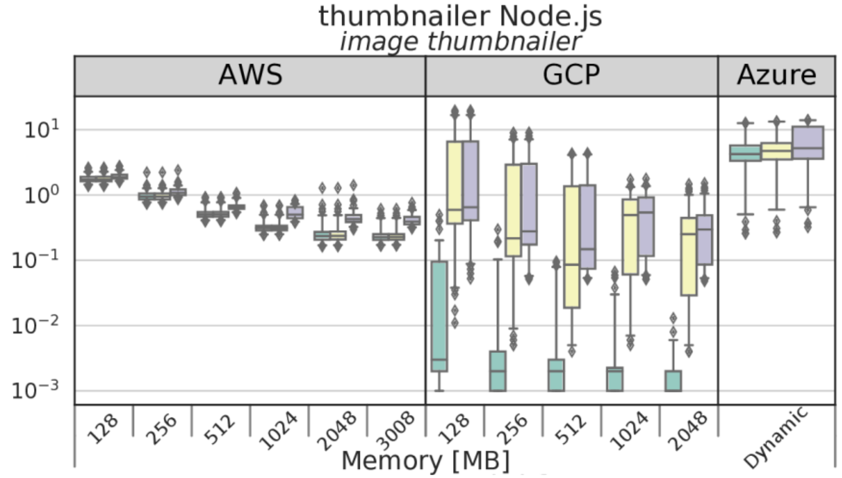


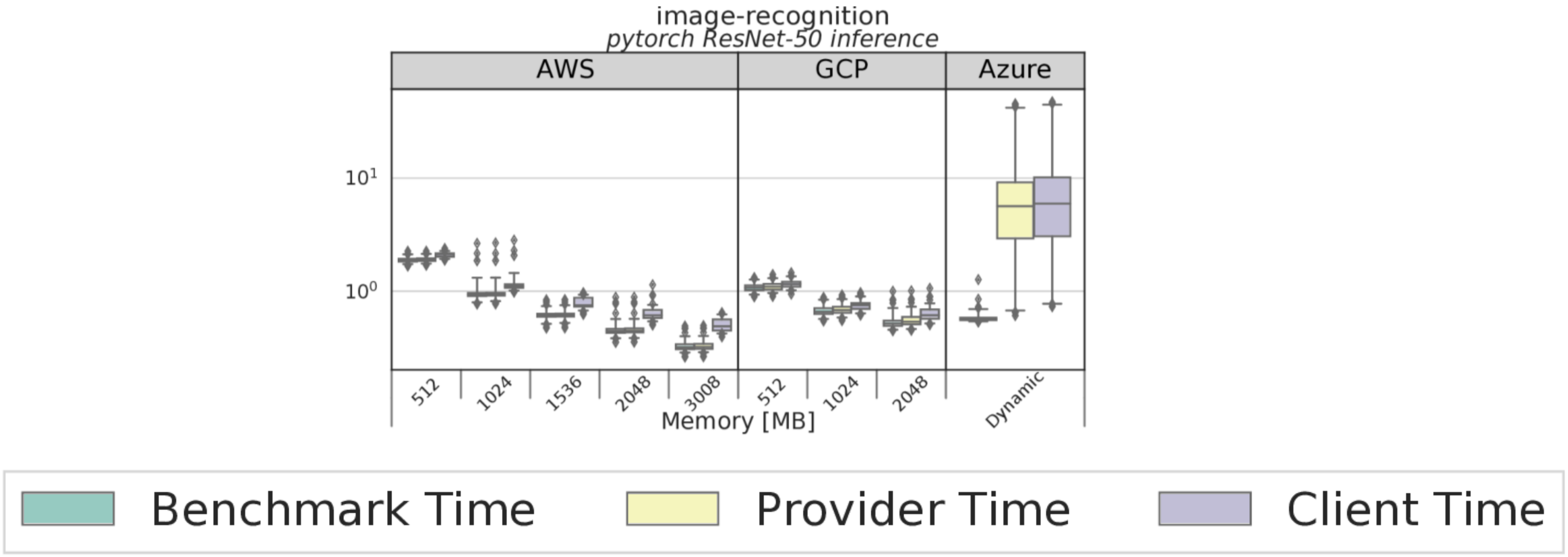
Performance Analysis: Warm Startups



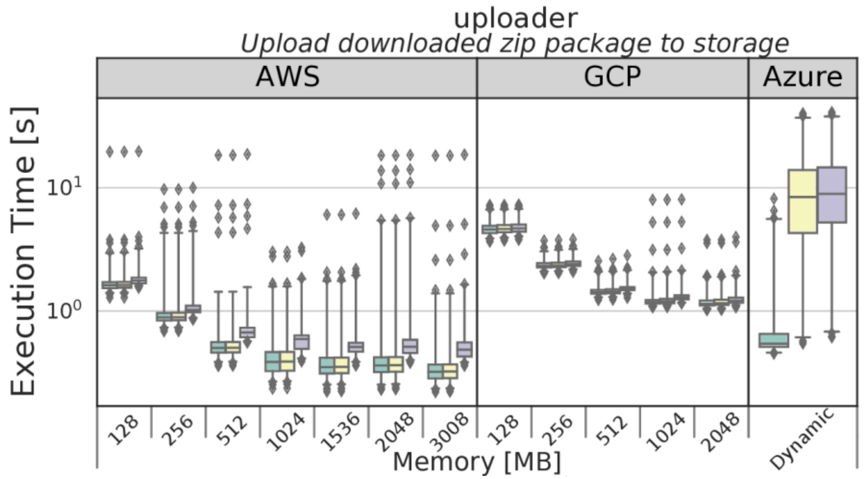
Performance Analysis: Warm Startups

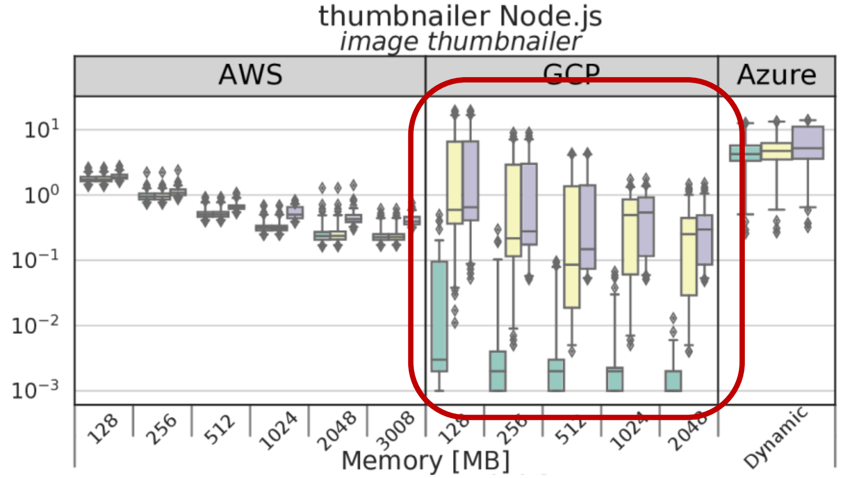


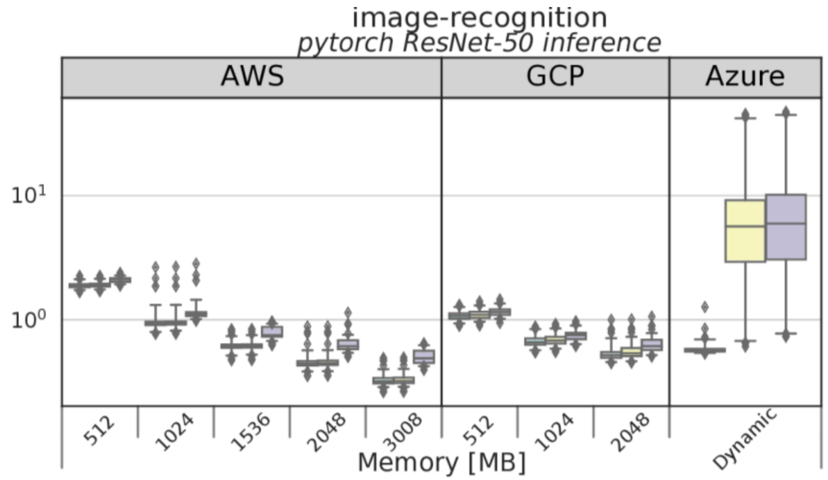




Performance Analysis: Warm Startups

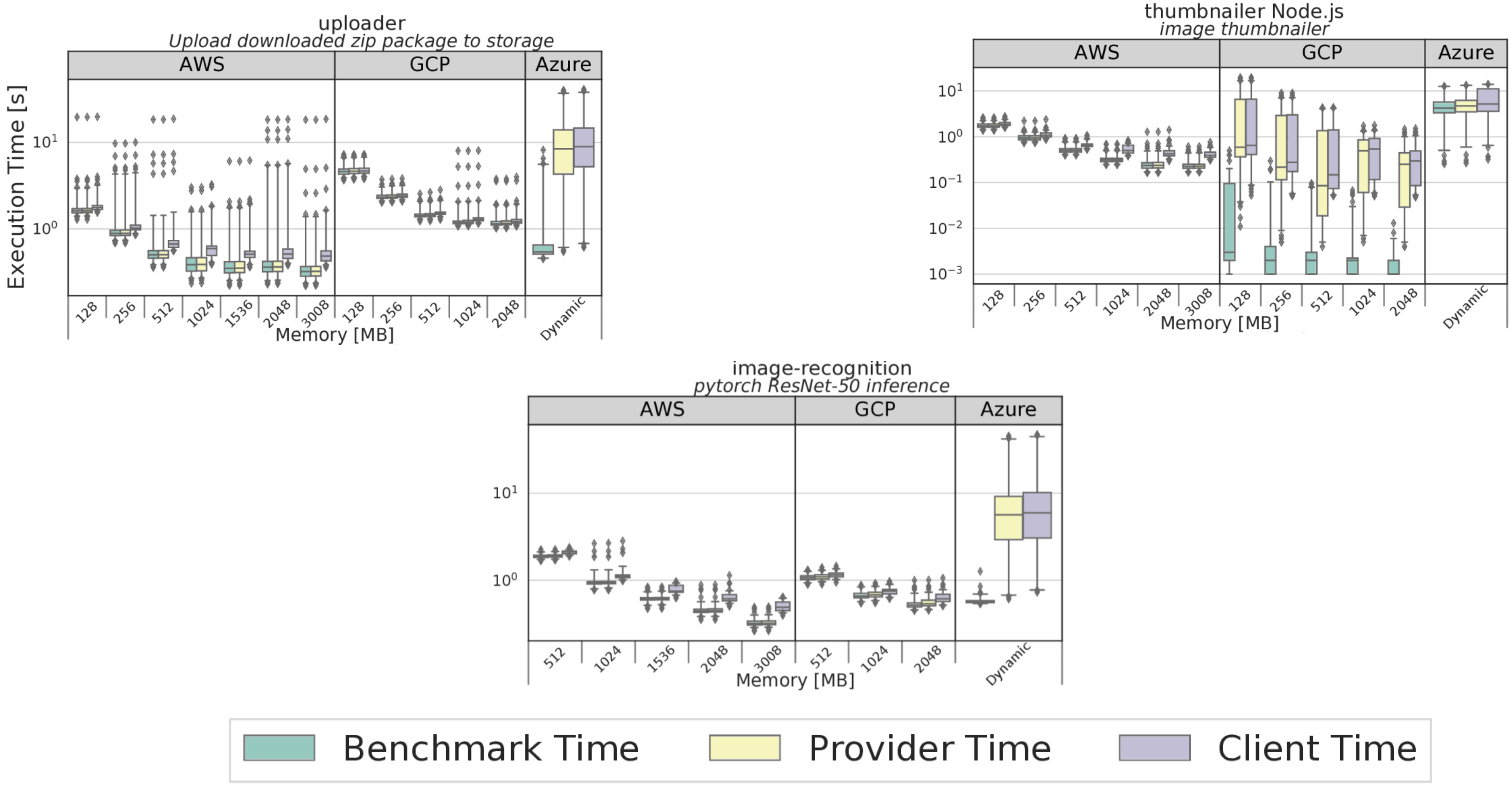




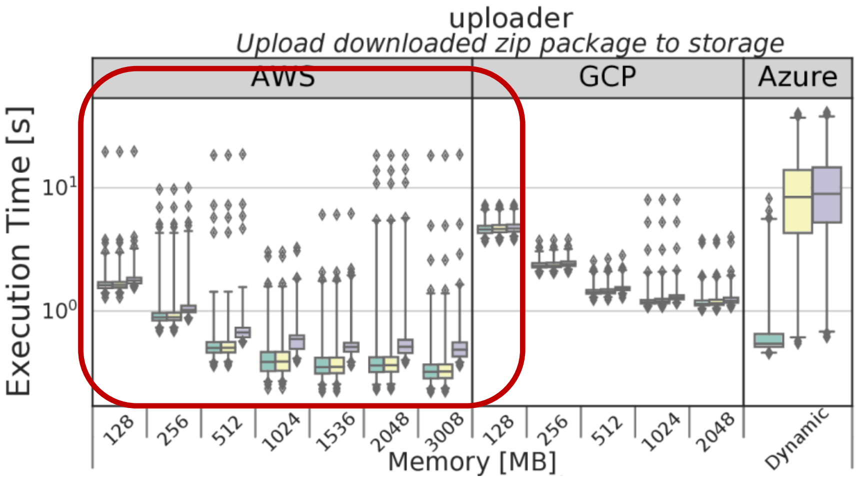


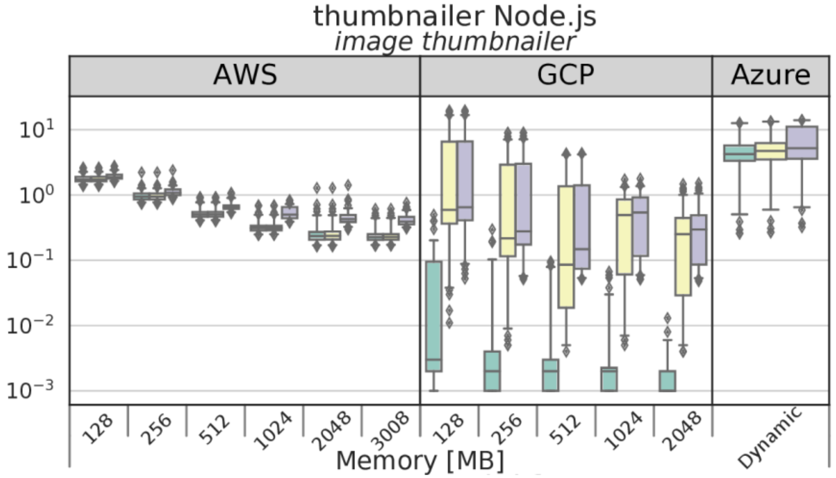
Benchmark Time Provider Time Client Time

Performance Analysis: Warm Startups



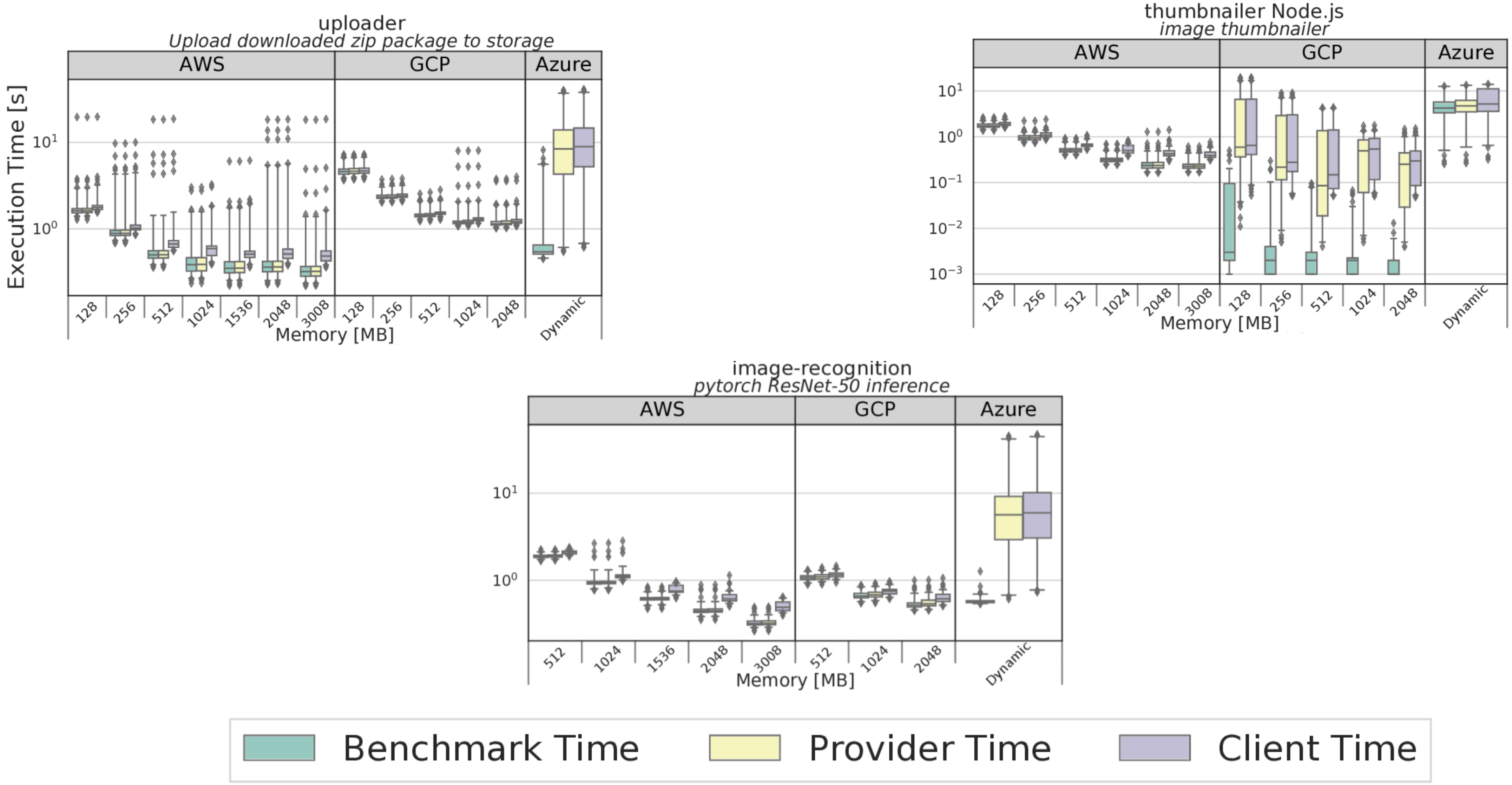
Performance Analysis: Warm Startups



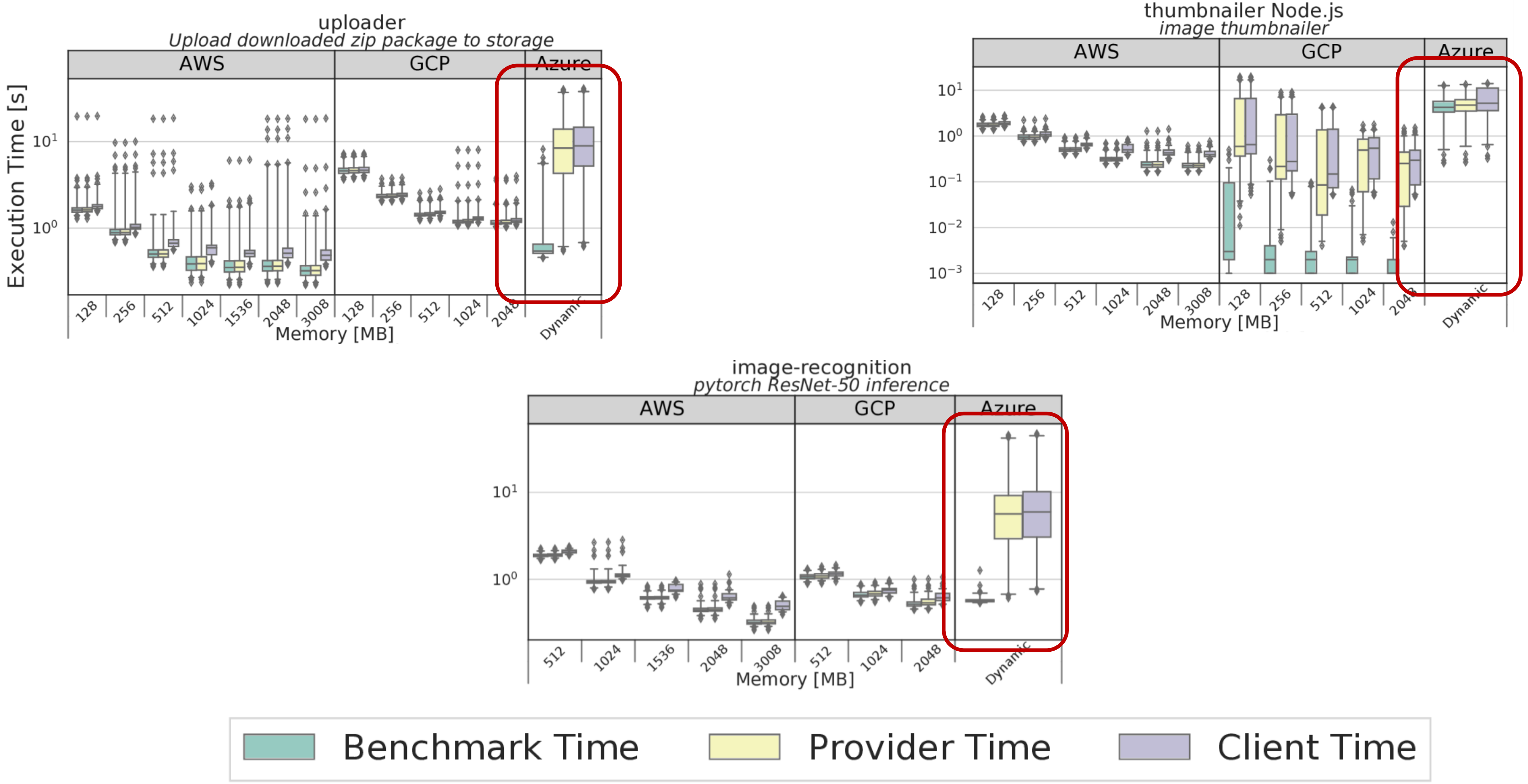


Benchmark Time Provider Time Client Time

Performance Analysis: Warm Startups



Performance Analysis: Warm Startups



Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.
$	

Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS [s]						
FaaS [s]						
Overhead						
Memory [MB]						

Configuration:
- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.
- **Measurements:** 200 warm executions.

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS [s]	0.316	0.13	0.191	2.803	0.235	0.03
FaaS [s]	0.389	0.188	0.253	2.949	0.321	0.075
Overhead	1.23x	1.43x	1.24x	1.05x	1.37x	2.4x
Memory [MB]	1024	1024	2048	1024	3008	1536

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.
- **Measurements:** 200 warm executions.

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS [s]	0.316	0.13	0.191	2.803	0.235	0.03
FaaS [s]	0.389	0.188	0.253	2.949	0.321	0.075
Overhead	1.23x	1.43x	1.24x	1.05x	1.37x	2.4x
Memory [MB]	1024	1024	2048	1024	3008	1536

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.
- **Measurements:** 200 warm executions.

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS [s]	0.316	0.13	0.191	2.803	0.235	0.03
FaaS [s]	0.389	0.188	0.253	2.949	0.321	0.075
Overhead	1.23x	1.43x	1.24x	1.05x	1.37x	$\boxed{2.4x}$
Memory [MB]	1024	1024	2048	1024	3008	1536

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.
- **Measurements:** 200 warm executions.

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS [s]	0.316	0.13	0.191	2.803	0.235	0.03
FaaS [s]	0.389	0.188	0.253	2.949	0.321	0.075
Overhead	1.23x	1.43x	1.24x	1.05x	1.37x	2.4x
Memory [MB]	1024	1024	2048	1024	3008	1536

Configuration:
- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.
- **Measurements:** 200 warm executions.

Performance Analysis: IaaS vs FaaS

	Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS [s]	0.316	0.13	0.191	2.803	0.235	0.03
FaaS [s]	0.389	0.188	0.253	2.949	0.321	0.075
Overhead	1.23x	1.43x	1.24x	1.05x	1.37x	2.4x
Memory [MB]	1024	1024	2048	1024	3008	1536

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.
- **Measurements:** 200 warm executions.

Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	

Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	Break-even analysis for IaaS and FaaS deployment.

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS							
FaaS							

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS							
FaaS							

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]						
FaaS							

Configuration:
- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS							

Configuration:
- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS	Eco 1M [$]						
	Eco Break-Even						
	Perf 1M [$]						
	Perf Break-Even						

Configuration:
- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS	Eco 1M [$]	3.54	2.29	3.75	32.1	15.8	2.08
	Eco Break-Even	3275	5062	3093	362	733	5568
	Perf 1M [$]	6.67	3.34	10	50	19.58	2.5
	Perf Break-Even	1740	3480	1160	232	592	4640

Configuration:

- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

Cost Analysis: FaaS vs IaaS

		Uploader	Thumbnailer Python	Thumbnailer Node.js	Compression	Image Recognition	Breadth-First Search
IaaS	Cloud Storage [req/h]	11371	27503	18819	1284	15312	117153
FaaS	Eco 1M [$]	3.54	2.29	3.75	32.1	15.8	2.08
	Eco Break-Even	3275	5062	3093	362	733	5568
	Perf 1M [$]	6.67	3.34	10	50	19.58	2.5
	Perf Break-Even	1740	3480	1160	232	592	4640

Configuration:
- **IaaS:** AWS t2.micro instance with 1 vCPU and 1 GB RAM, 100% utilization, $0.0116/h.
- **FaaS**: AWS Lambda.
- **Local storage**: Minio as Docker container.

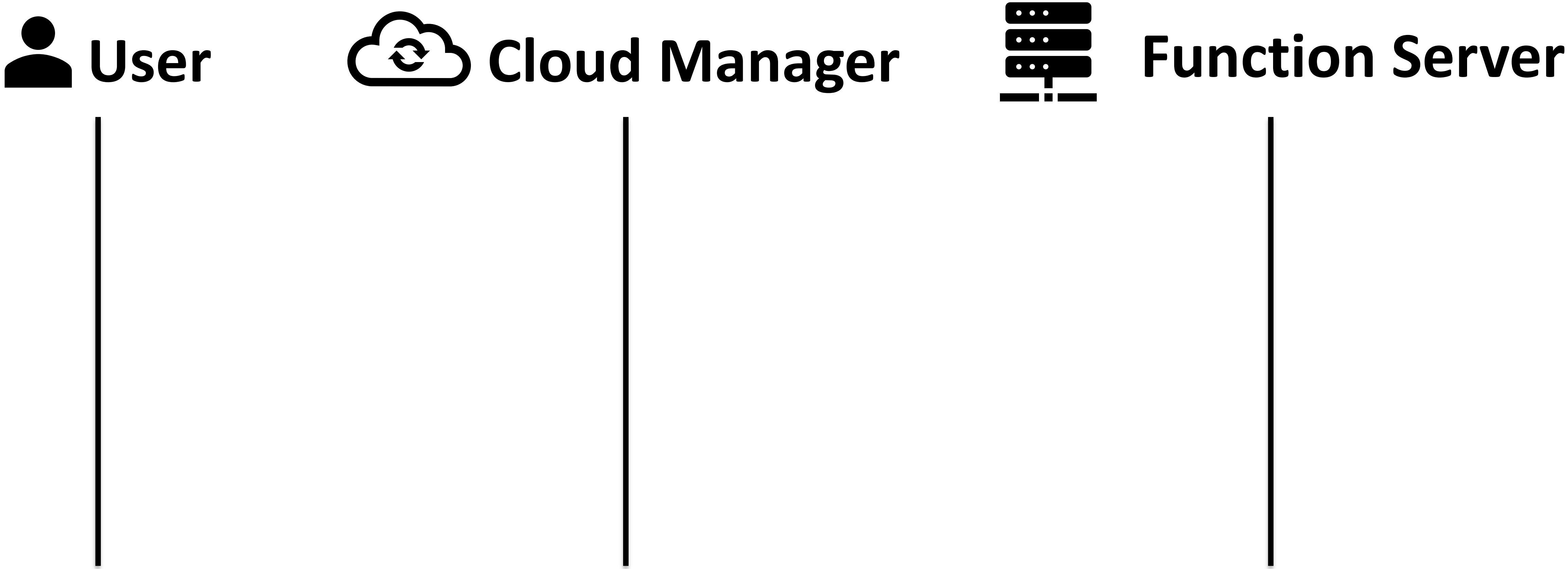
Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	
	Break-even analysis for IaaS and FaaS deployment.

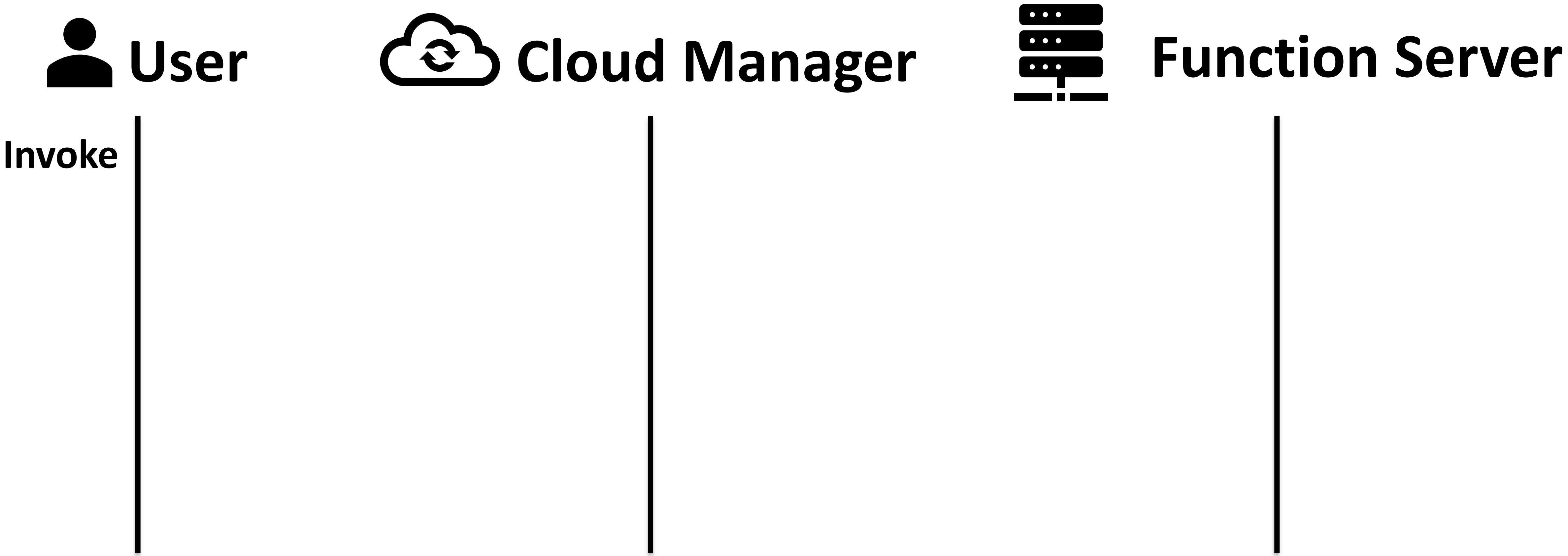
Results and Insights

	Results, methods, and insights
	AWS Lambda achieves the best performance on all workloads.
	Irregular performance of concurrent Azure Function executions.
	I/O-bound functions experience very high latency variations.
	AWS Lambda performance is not competitive against VMs assuming comparable resources.
$	
	Break-even analysis for IaaS and FaaS deployment.
	Accurate methodology for estimation of invocation latency.
	Warm latencies are consistent and depend linearly on payload size.
	Highly variable and unpredictable cold latencies on Azure and GCP.

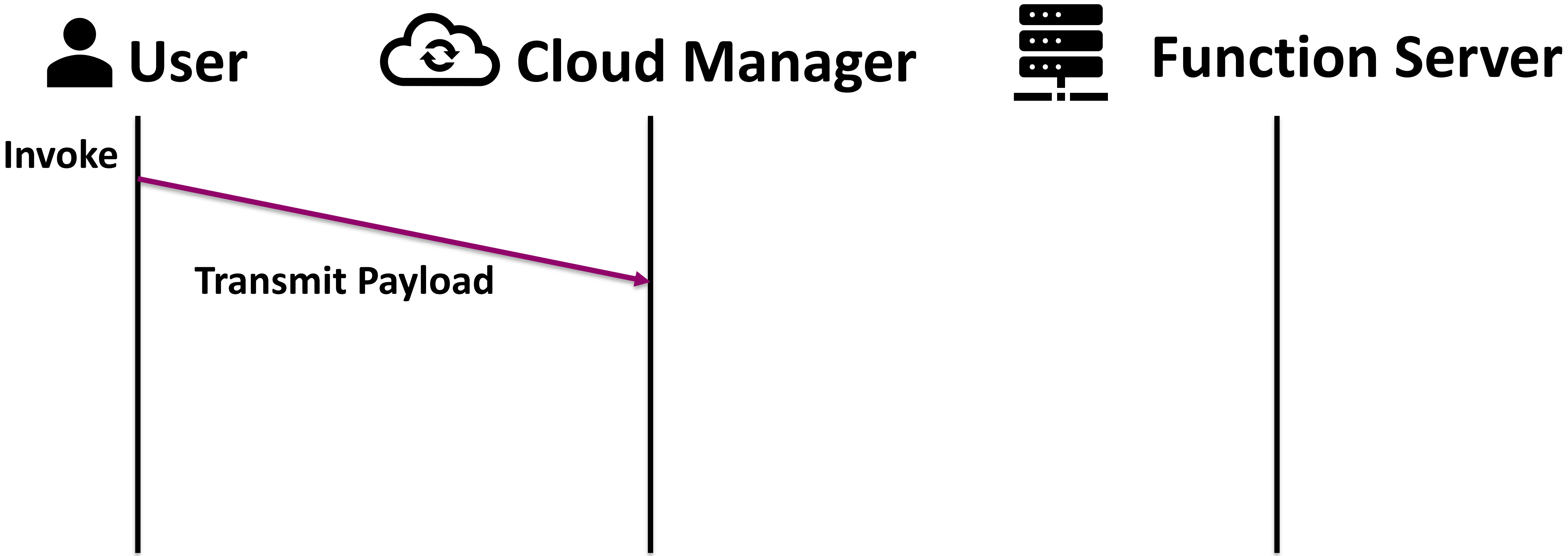
FaaS Analysis: Invocation Overhead



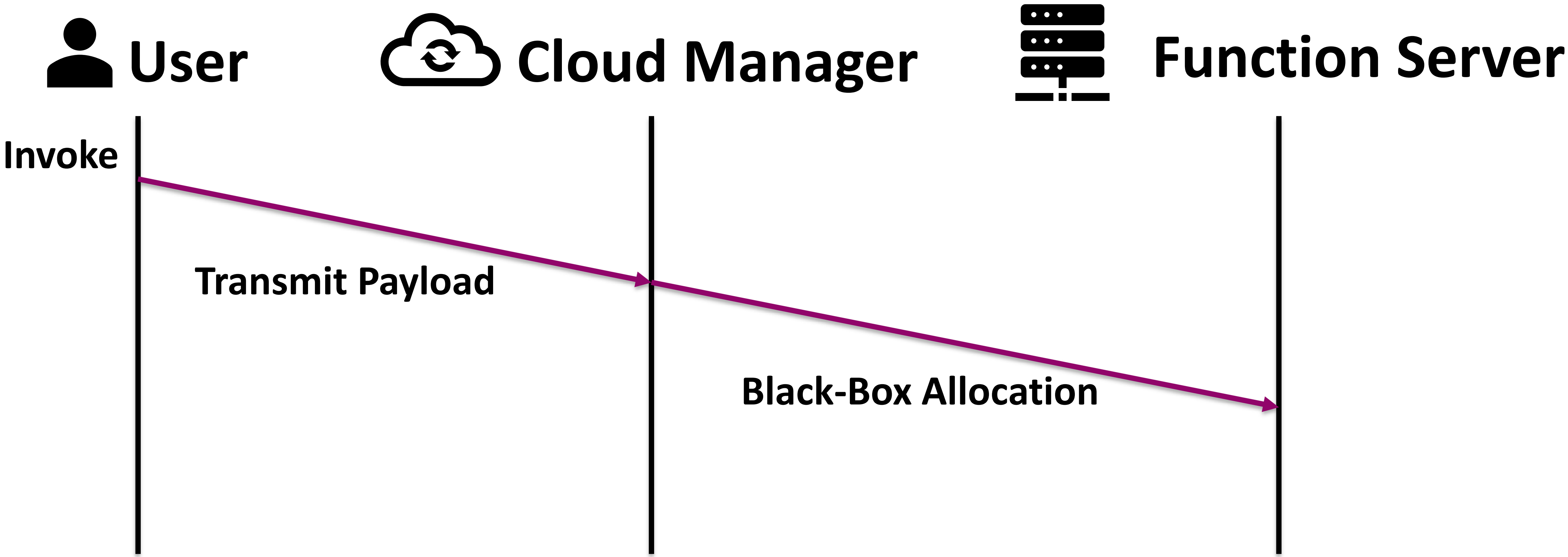
FaaS Analysis: Invocation Overhead



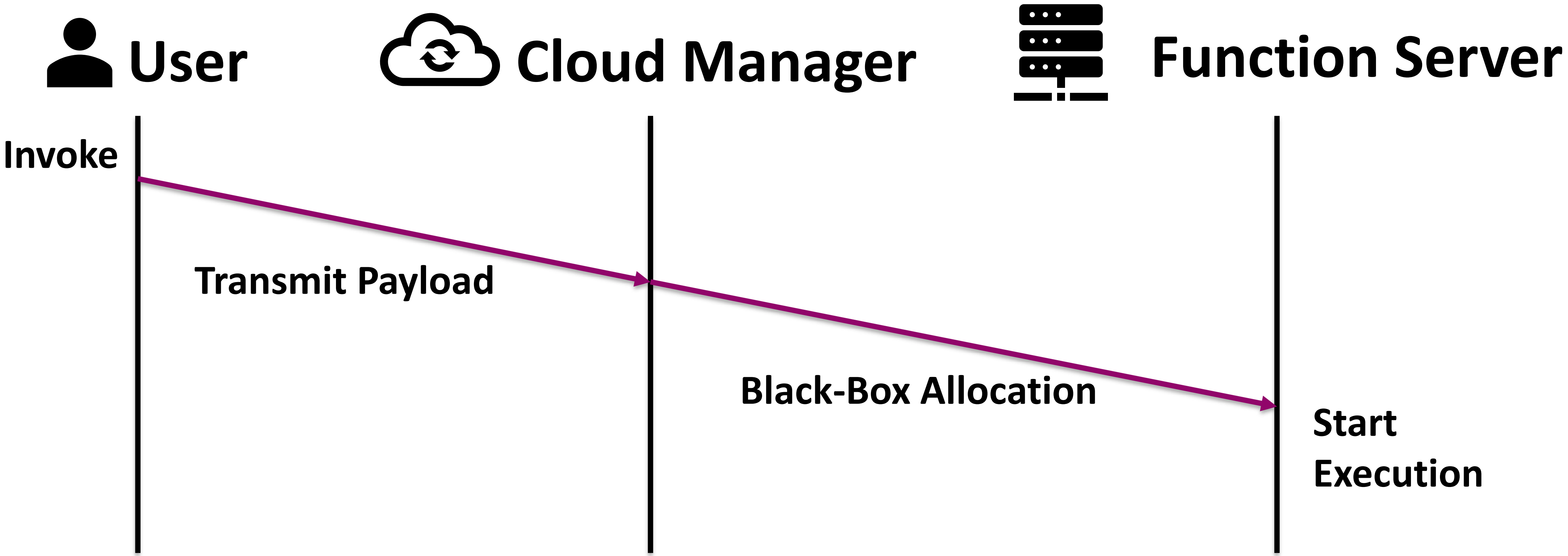
FaaS Analysis: Invocation Overhead



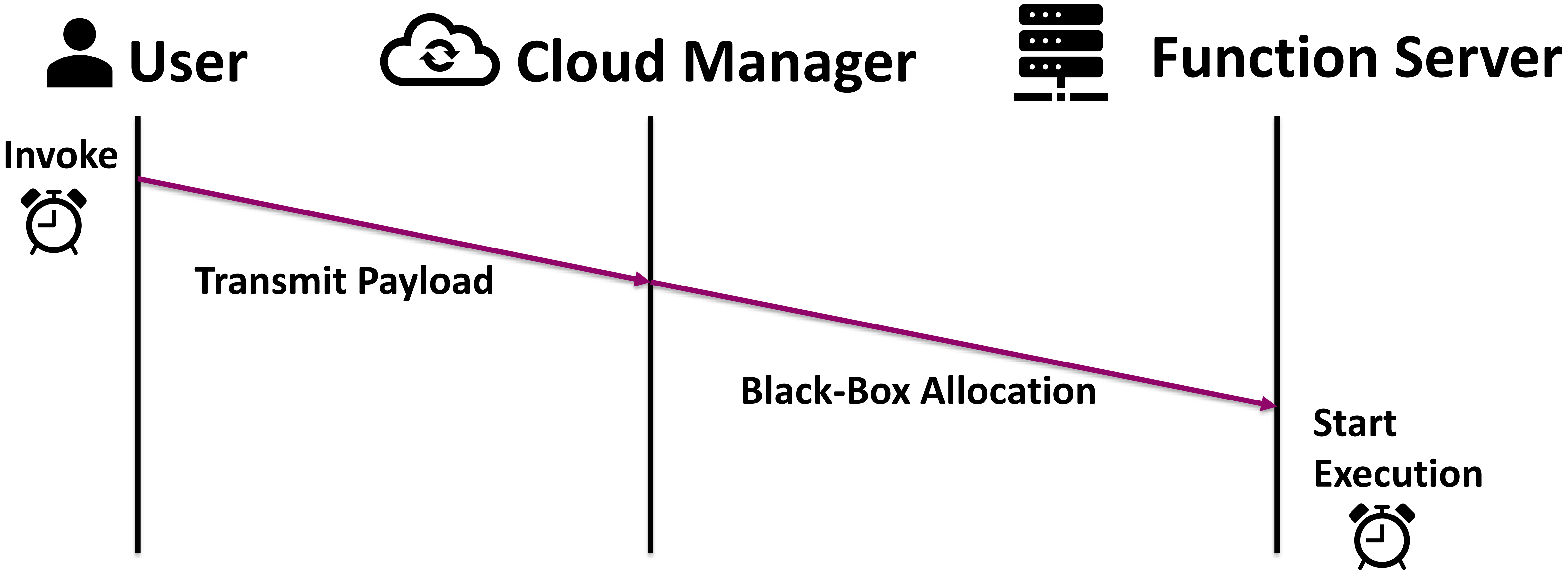
FaaS Analysis: Invocation Overhead



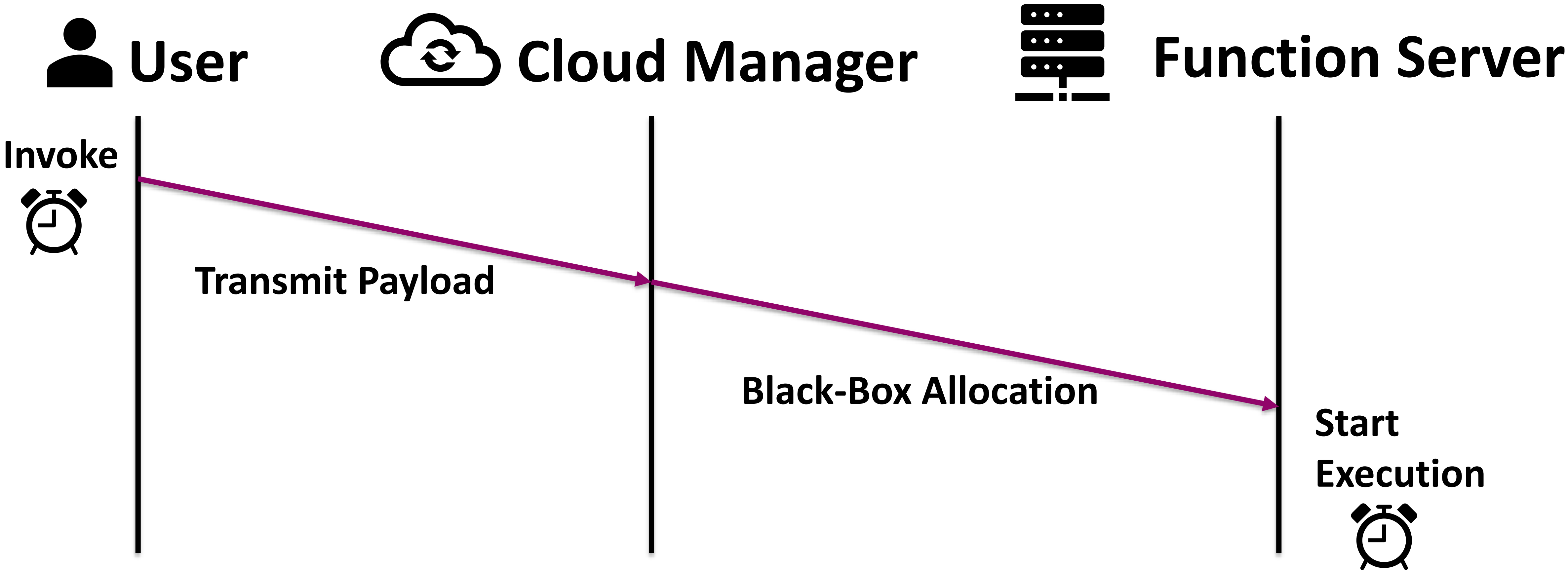
FaaS Analysis: Invocation Overhead



FaaS Analysis: Invocation Overhead



FaaS Analysis: Invocation Overhead



Solution: apply clock-drift estimation protocols!

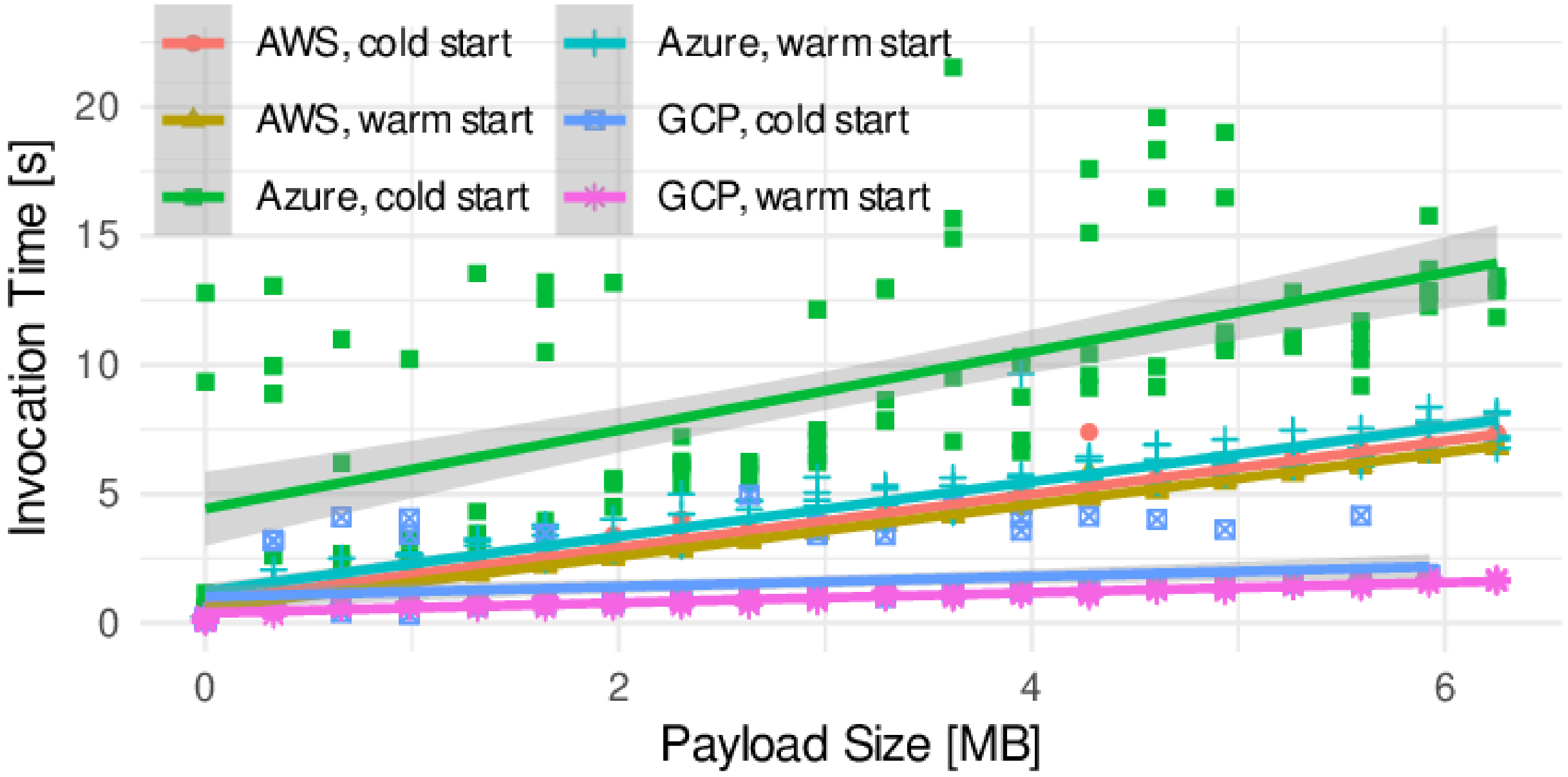
FaaS Analysis: Invocation Overhead

FaaS Analysis: Invocation Overhead

Configuration:

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

FaaS Analysis: Invocation Overhead



Configuration:

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

Summary

Summary

spcl/serverless-benchmarks