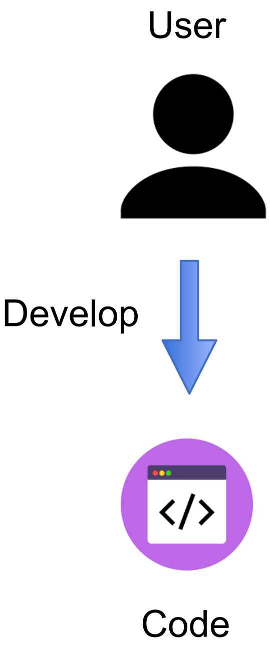
Accelerating Serverless Computing by Harvesting Idle Resources

Hanfei Yu[1], Hao Wang[1], Jian Li[2], Xu Yuan[3], Seung-Jong Park[1]

[1]Louisiana State University, [2]SUNY-Binghamton University, [3]University of Louisiana at Lafayette

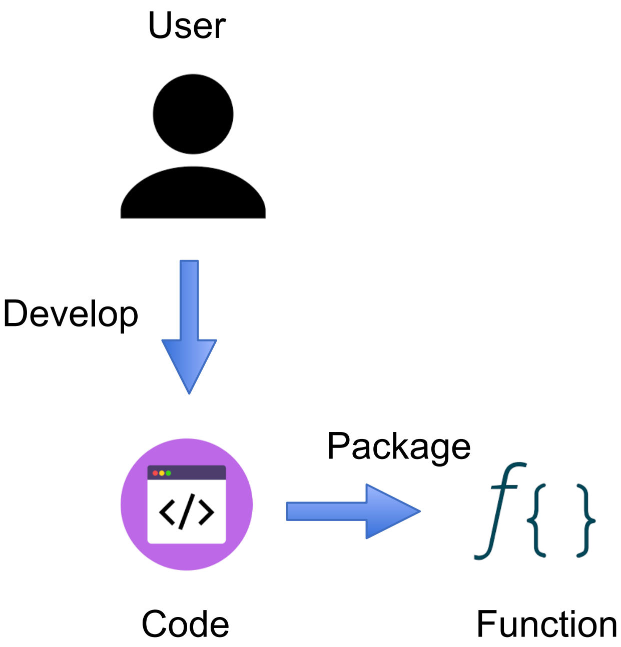
Serverless Computing


User

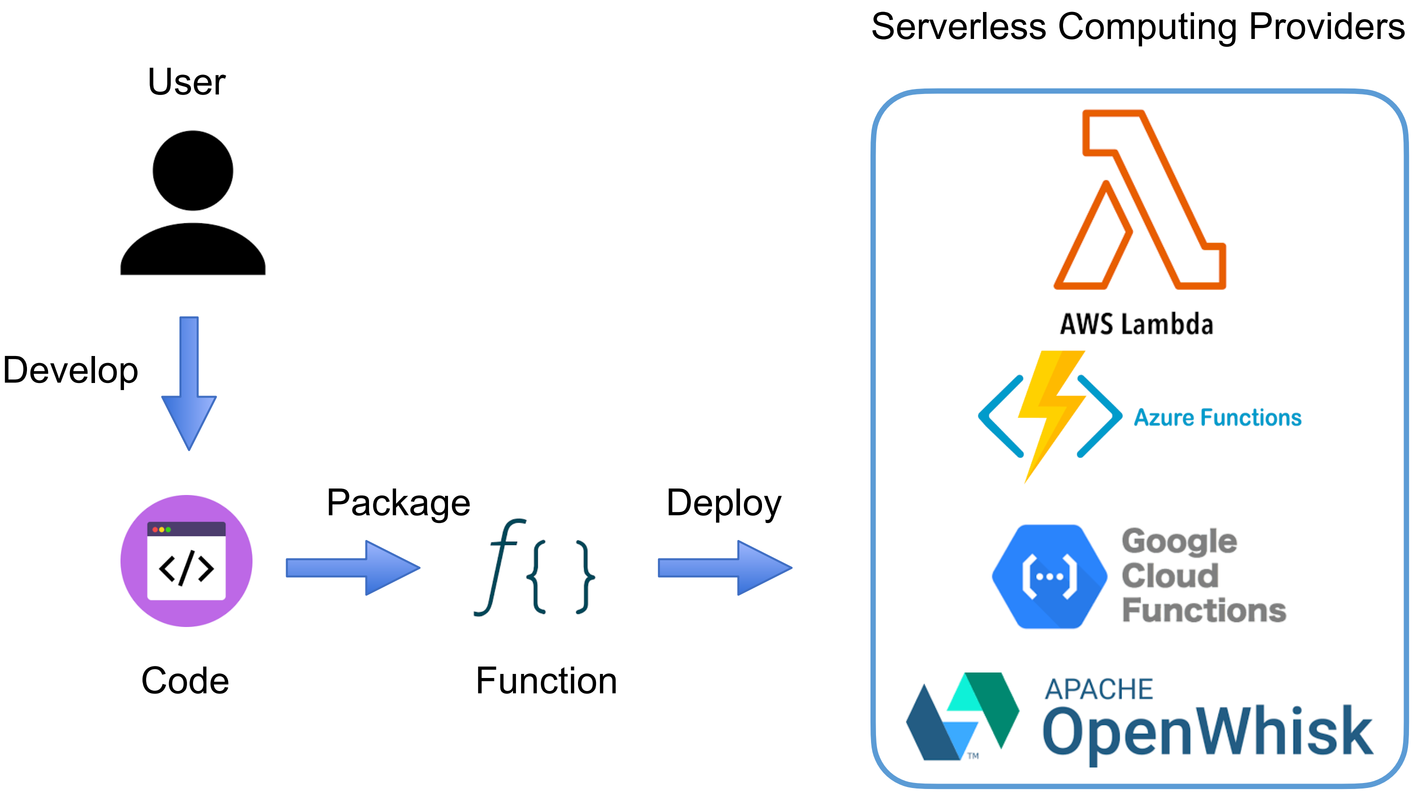
Develop

Code

Serverless Computing



Serverless Computing



Serverless Computing Providers

User

Develop

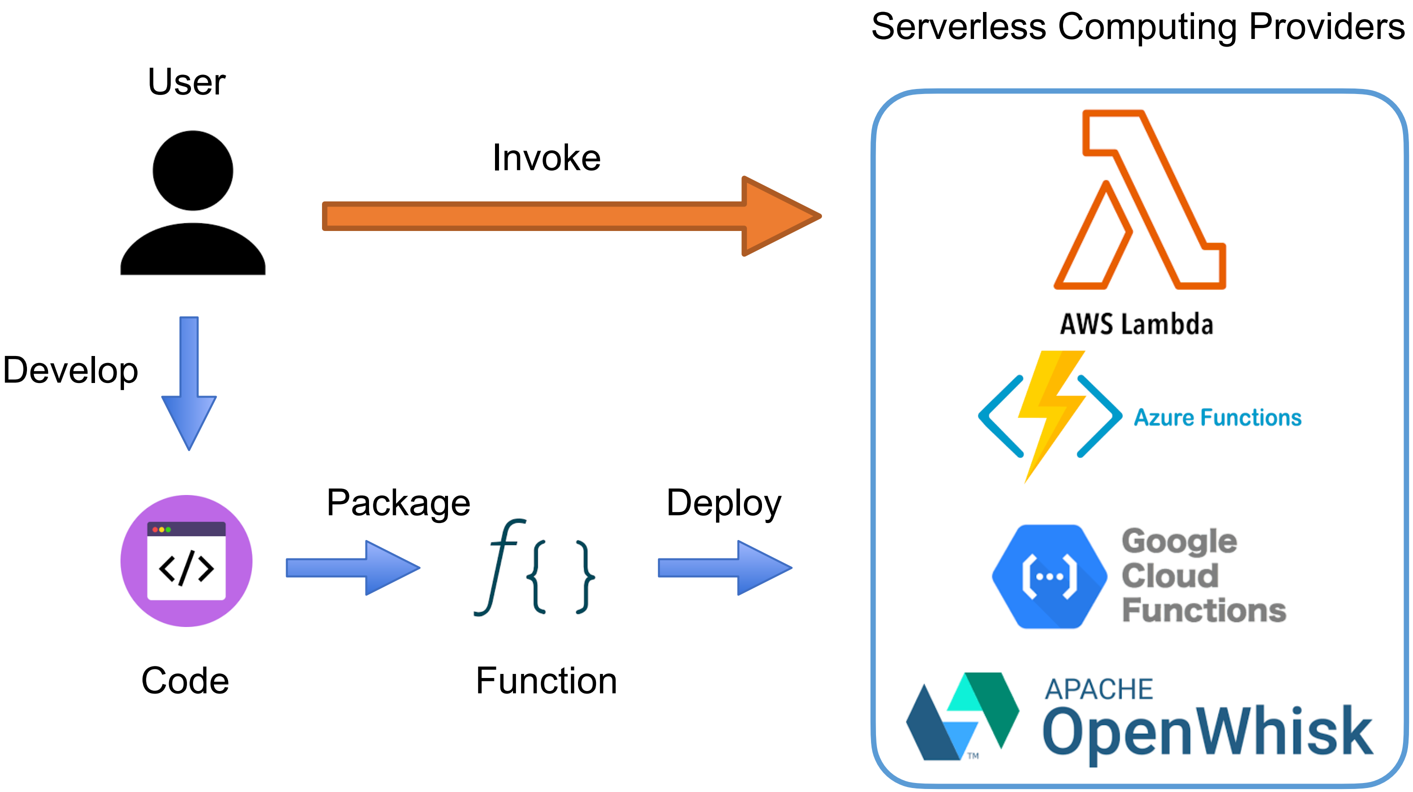
Code

Package

Function

Deploy

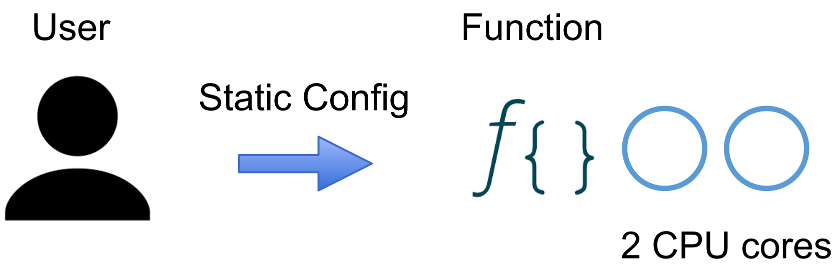
Serverless Computing



Function Static Configuration

🟢 : Busy core

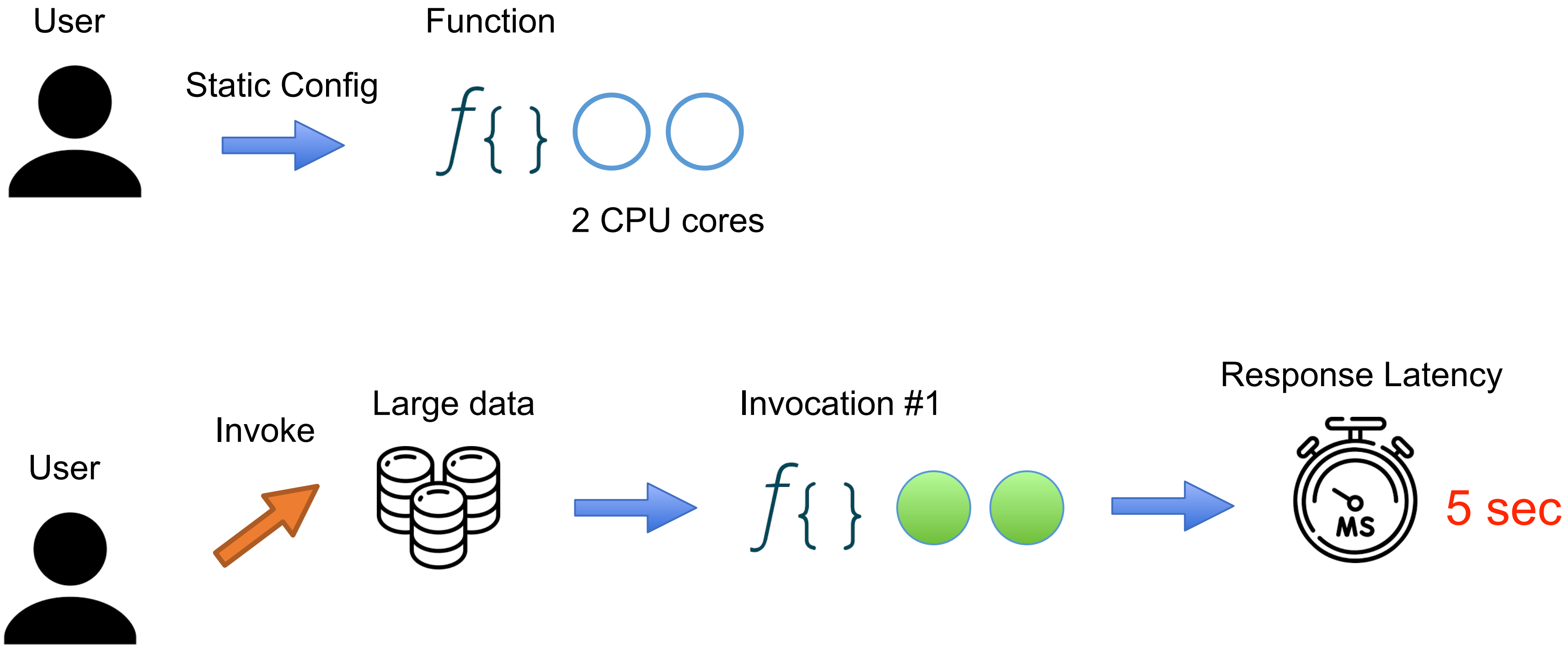
○ : Idle core



Varying Input Data Size

🟢 : Busy core

○ : Idle core



User

Static Config

Function

2 CPU cores

User

Invoke

Large data

Invocation #1

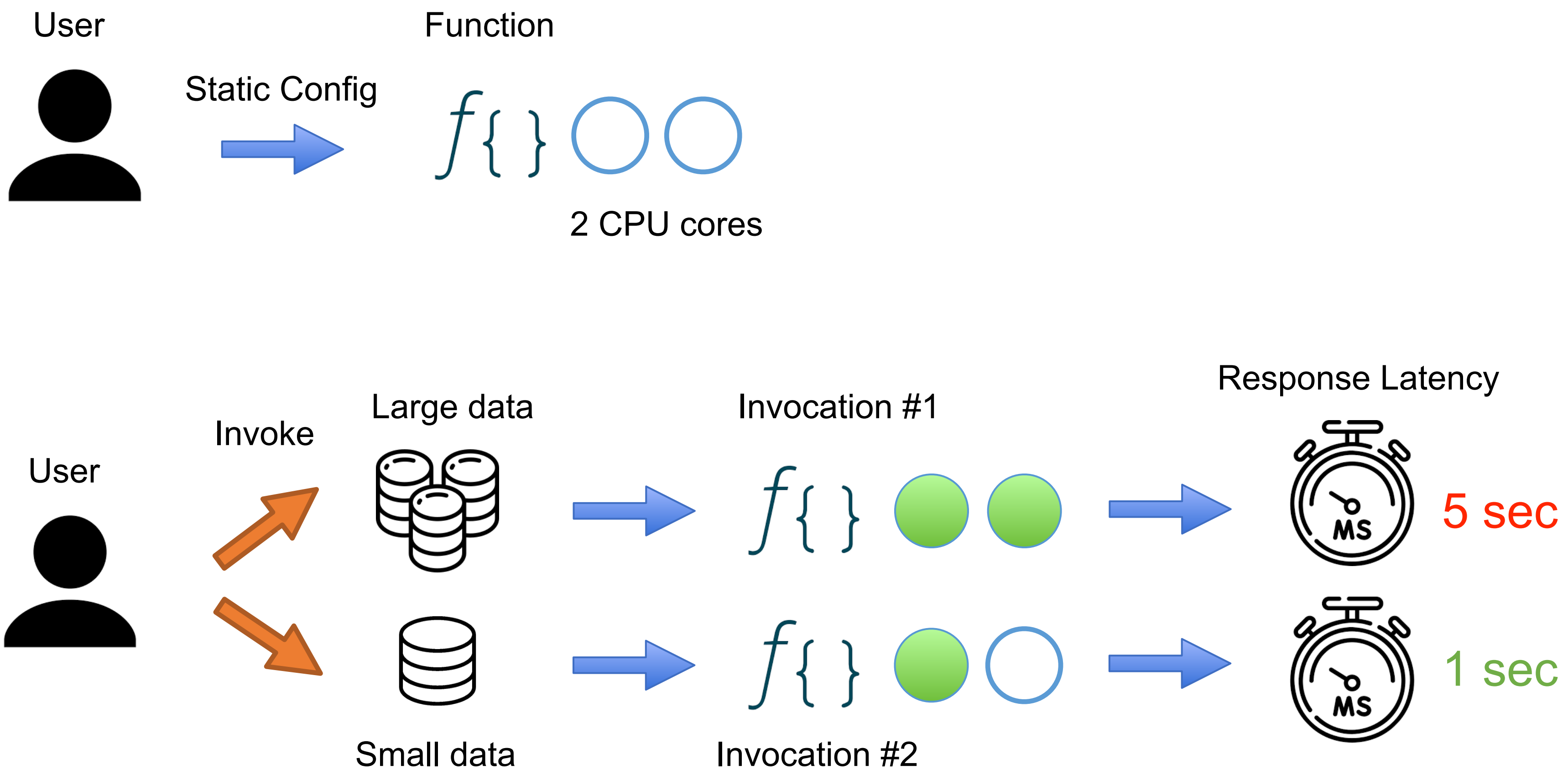
Response Latency

5 sec

Varying Input Data Size

🟢 : Busy core

○ : Idle core



User

Static Config

Function

2 CPU cores

User

Invoke

Large data

Small data

Invocation #1

Invocation #2

Response Latency

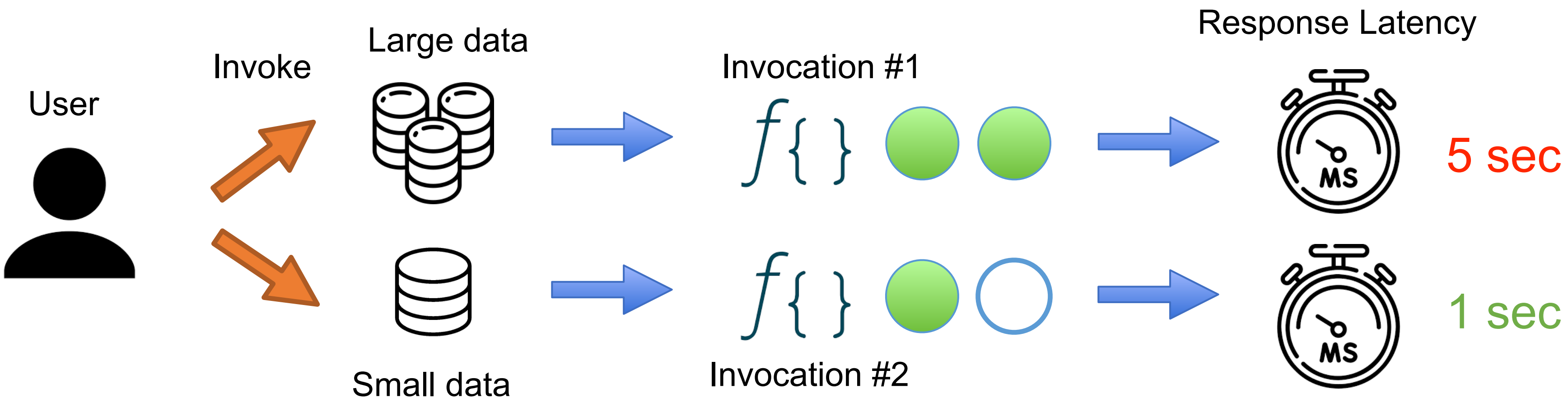
5 sec

1 sec

Harvesting & Acceleration

🟢 : Busy core

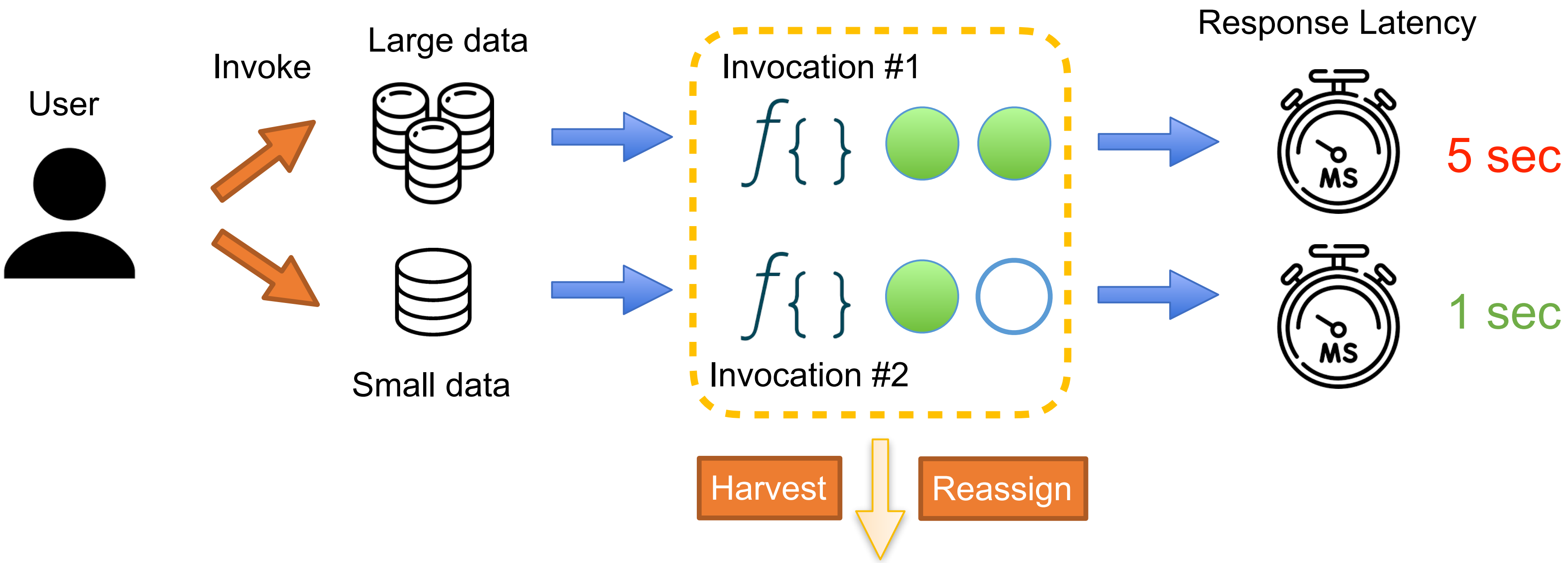
○ : Idle core



Harvesting & Acceleration

🟢 : Busy core

○ : Idle core



User

Invoke

Large data

Small data

Invocation #1

Invocation #2

Response Latency

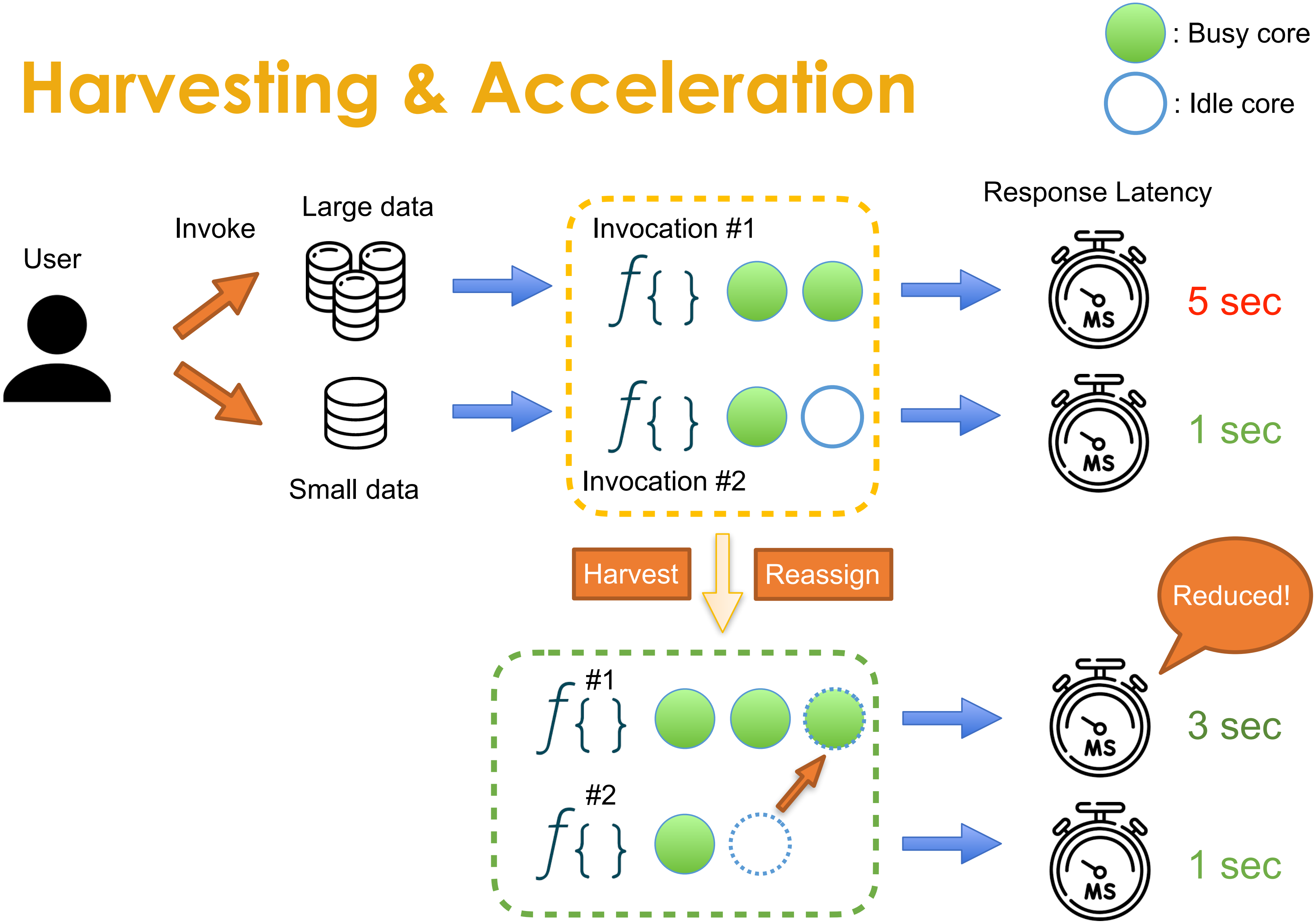
5 sec

1 sec

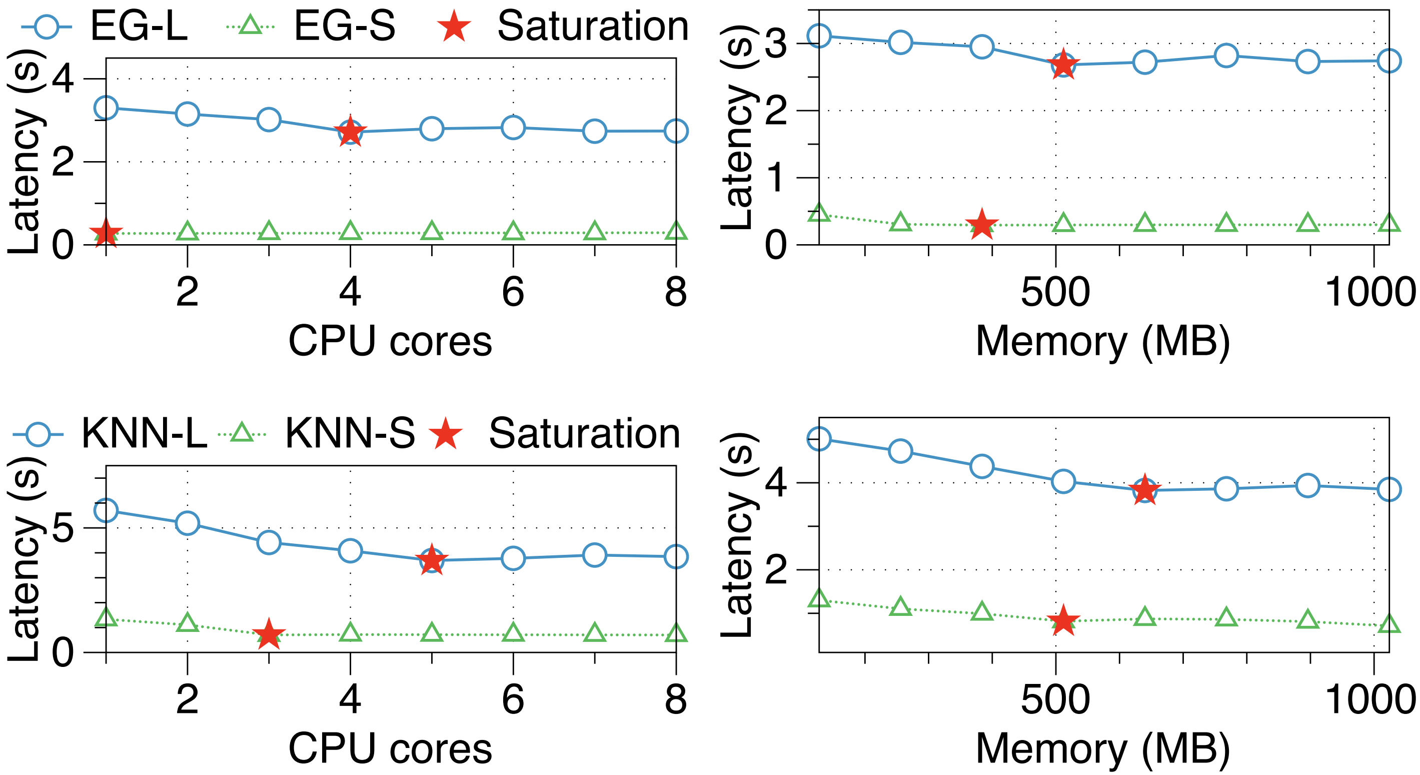
Harvest

Reassign

Harvesting & Acceleration



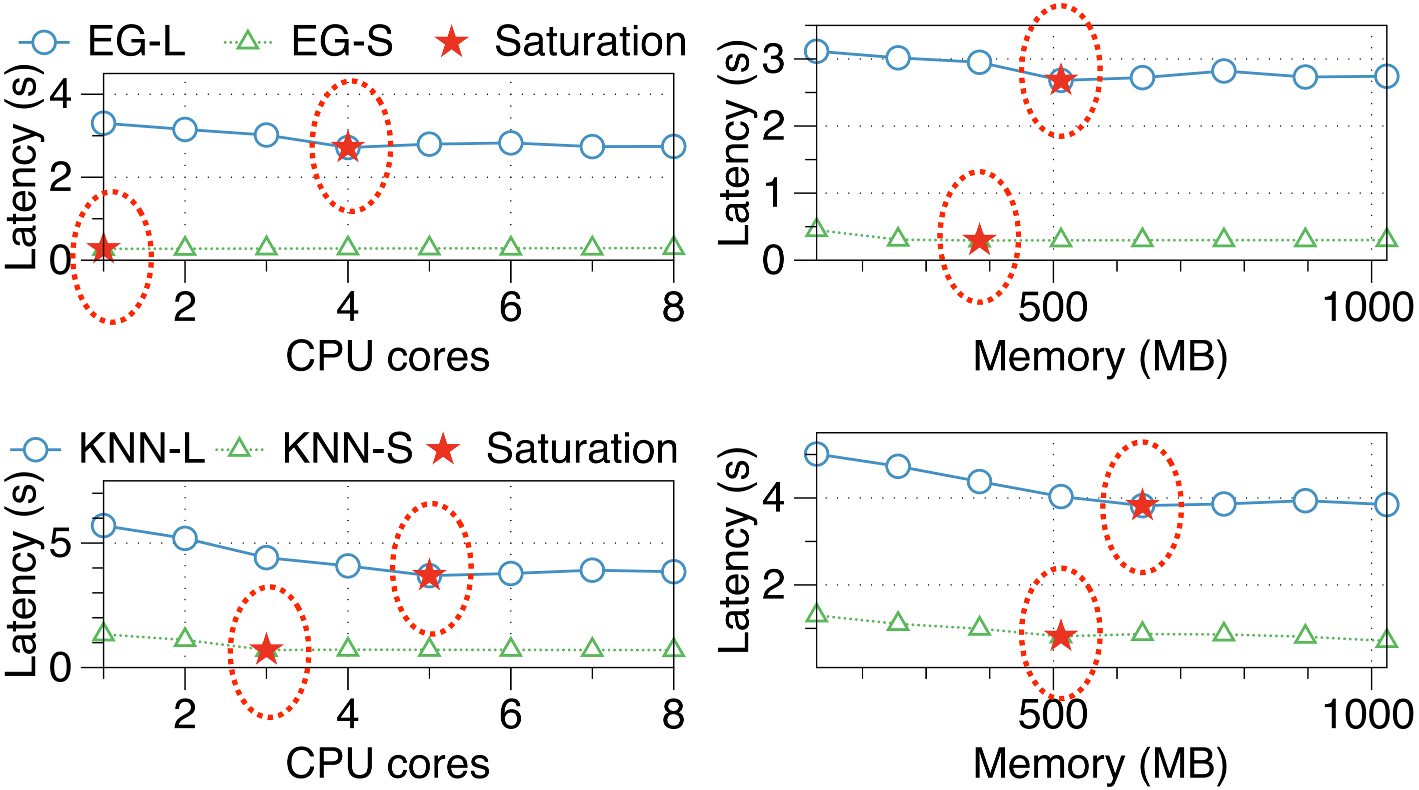
Realistic Applications



EG: email generation
KNN: K nearest neighbors

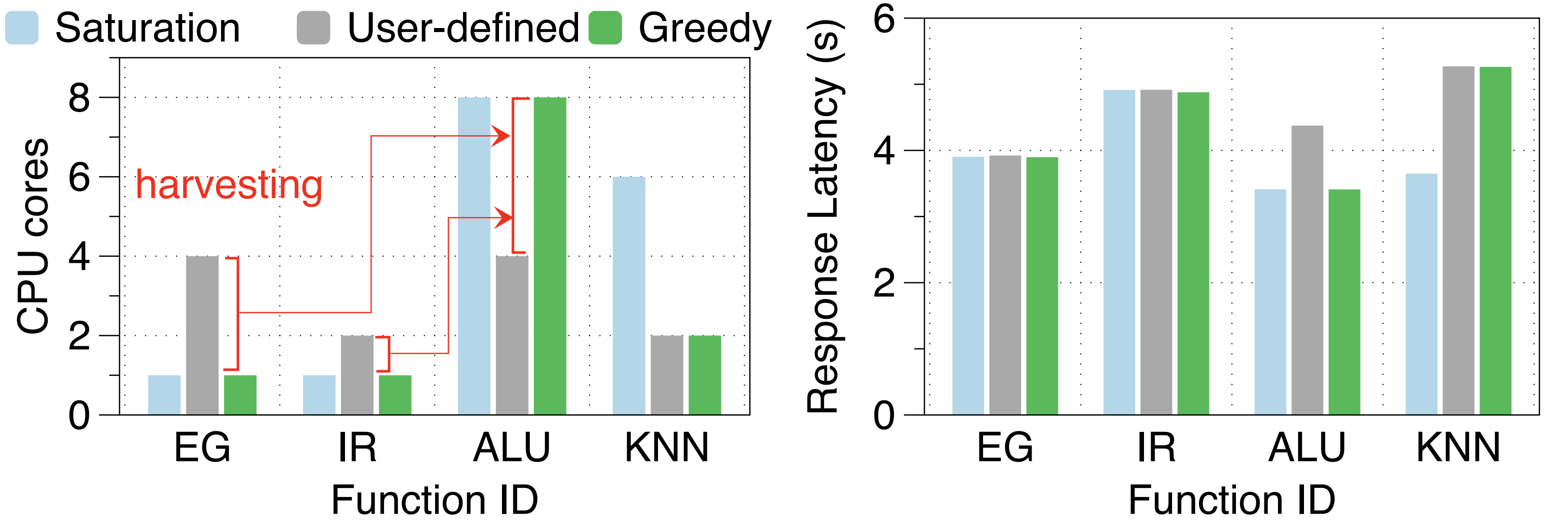
Realistic Applications

Performance stops growing when supplying more resources!



EG: email generation
KNN: K nearest neighbors

Realistic Harvesting & Acceleration



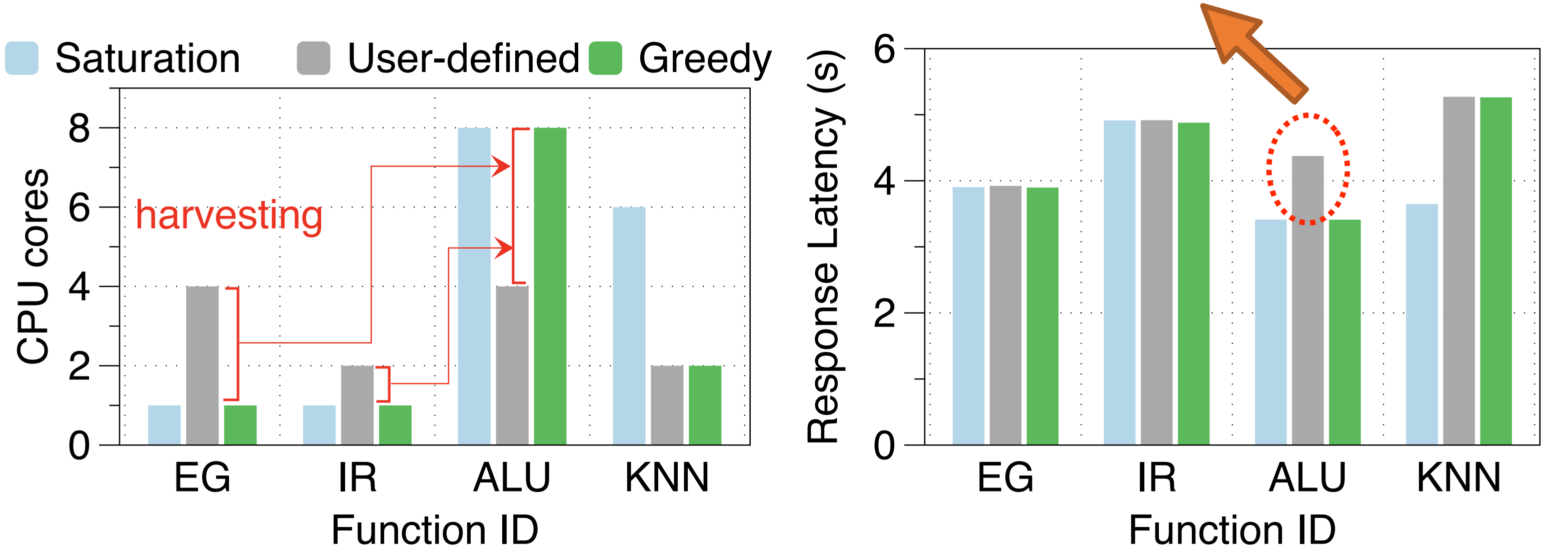
(a) CPU allocation

(b) Function response latency

EG: email generation
IR: image recognition
ALU: arithmetic logic units
KNN: K nearest neighbors

Realistic Harvesting & Acceleration

Latency can be reduced with supplying harvested resources!

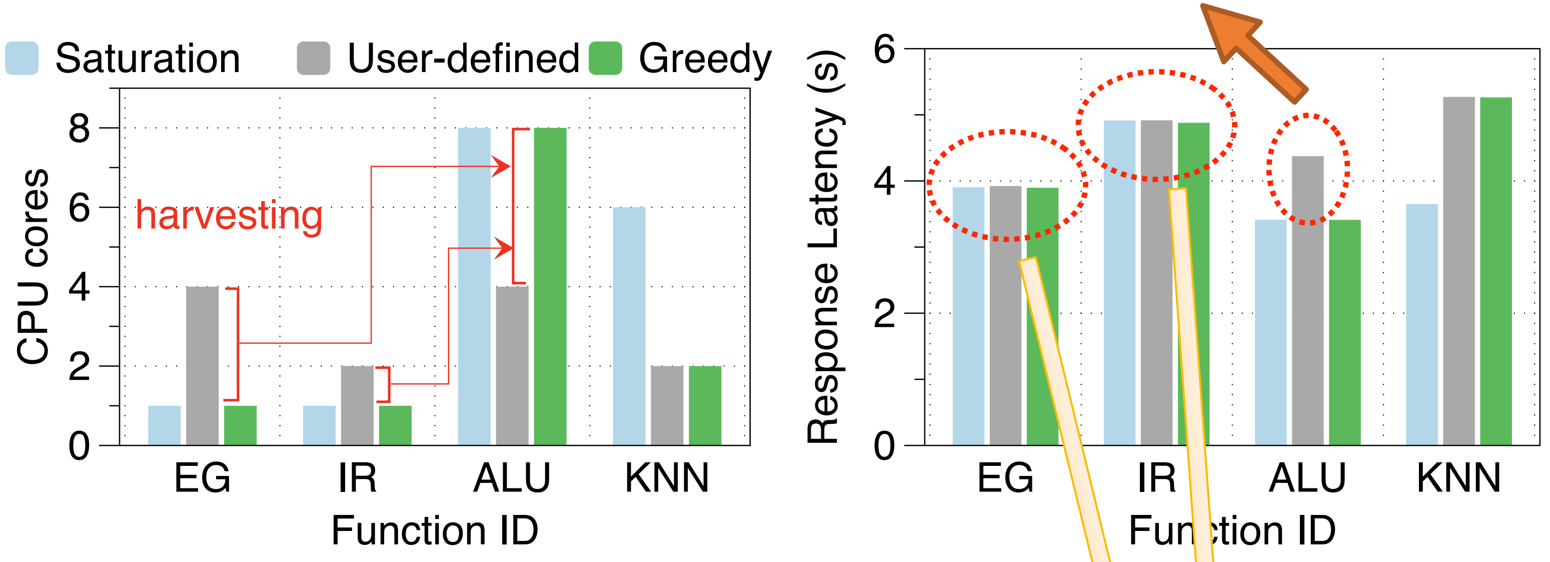


(a) CPU allocation (b) Function response latency

EG: email generation
IR: image recognition
ALU: arithmetic logic units
KNN: K nearest neighbors

Realistic Harvesting & Acceleration

Latency can be reduced with supplying harvested resources!



(a) CPU allocation

(b) Function response latency

Careful harvesting does not degrade performance

EG: email generation
IR: image recognition
ALU: arithmetic logic units
KNN: K nearest neighbors

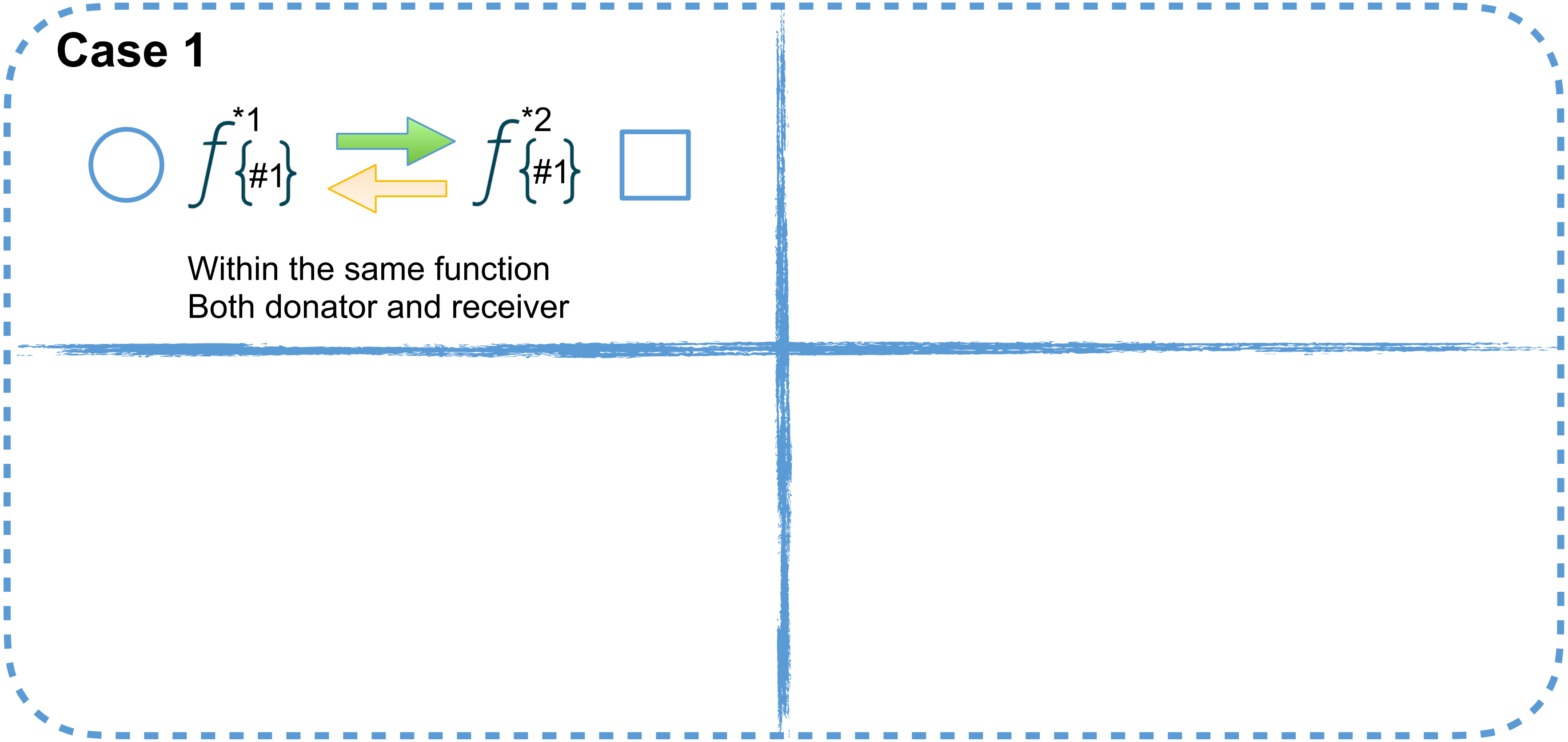
General Rebalance Cases

○ Idle core

□ Idle memory

# Function index

* Invocation index



Case 1

$$\bigcirc \quad f^{*1}_{\{\#1\}} \rightleftarrows f^{*2}_{\{\#1\}} \quad \square$$

Within the same function
Both donator and receiver

Serverless Cluster

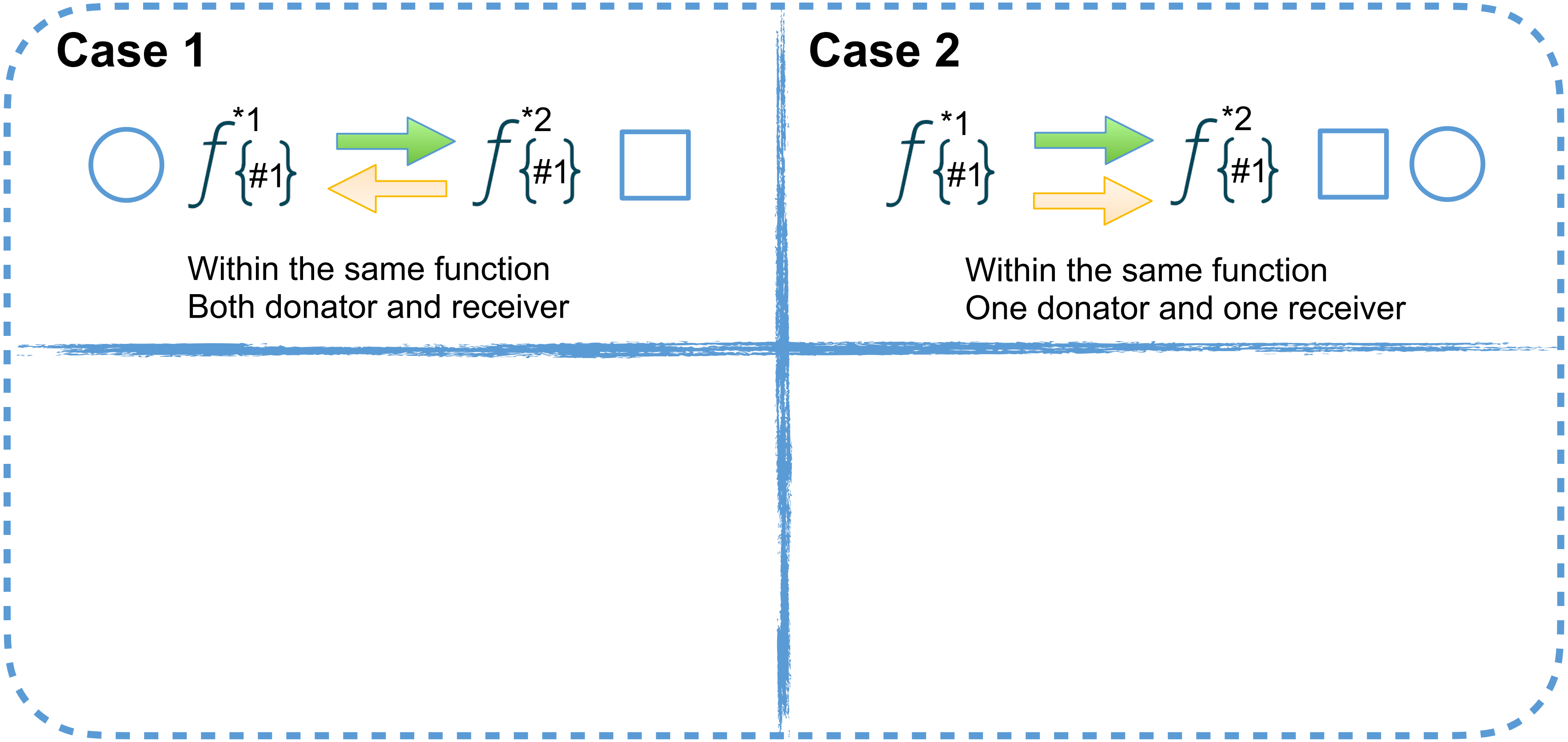
General Rebalance Cases

○ Idle core

□ Idle memory

# Function index

* Invocation index



Case 1

○ $f_{\{\#1\}}^{*1}$ ⇄ $f_{\{\#1\}}^{*2}$ □

Within the same function
Both donator and receiver

Case 2

$f_{\{\#1\}}^{*1}$ ⇉ $f_{\{\#1\}}^{*2}$ □ ○

Within the same function
One donator and one receiver

Serverless Cluster

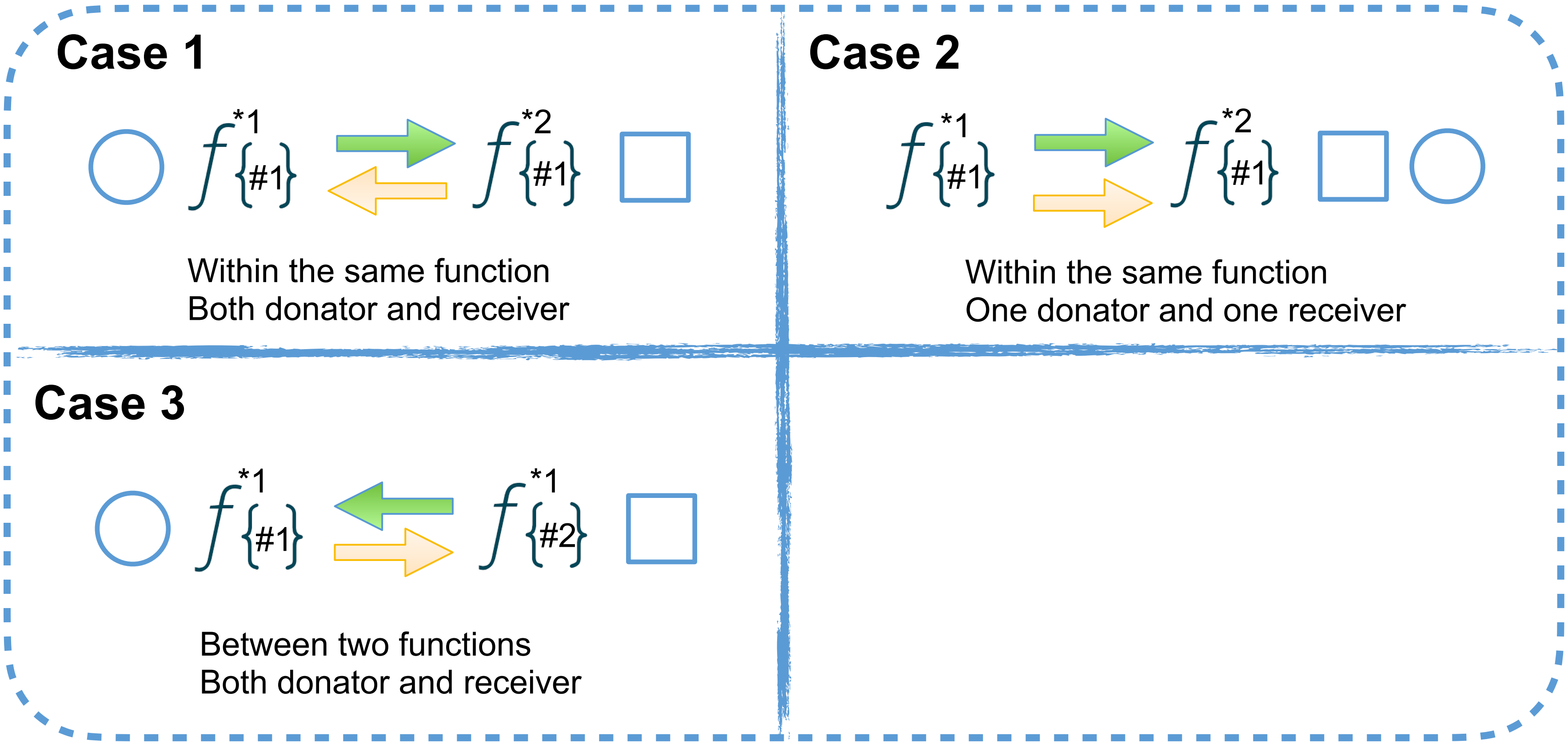
General Rebalance Cases

○ Idle core

□ Idle memory

# Function index

* Invocation index



Case 1

Within the same function
Both donator and receiver

Case 2

Within the same function
One donator and one receiver

Case 3

Between two functions
Both donator and receiver

Serverless Cluster

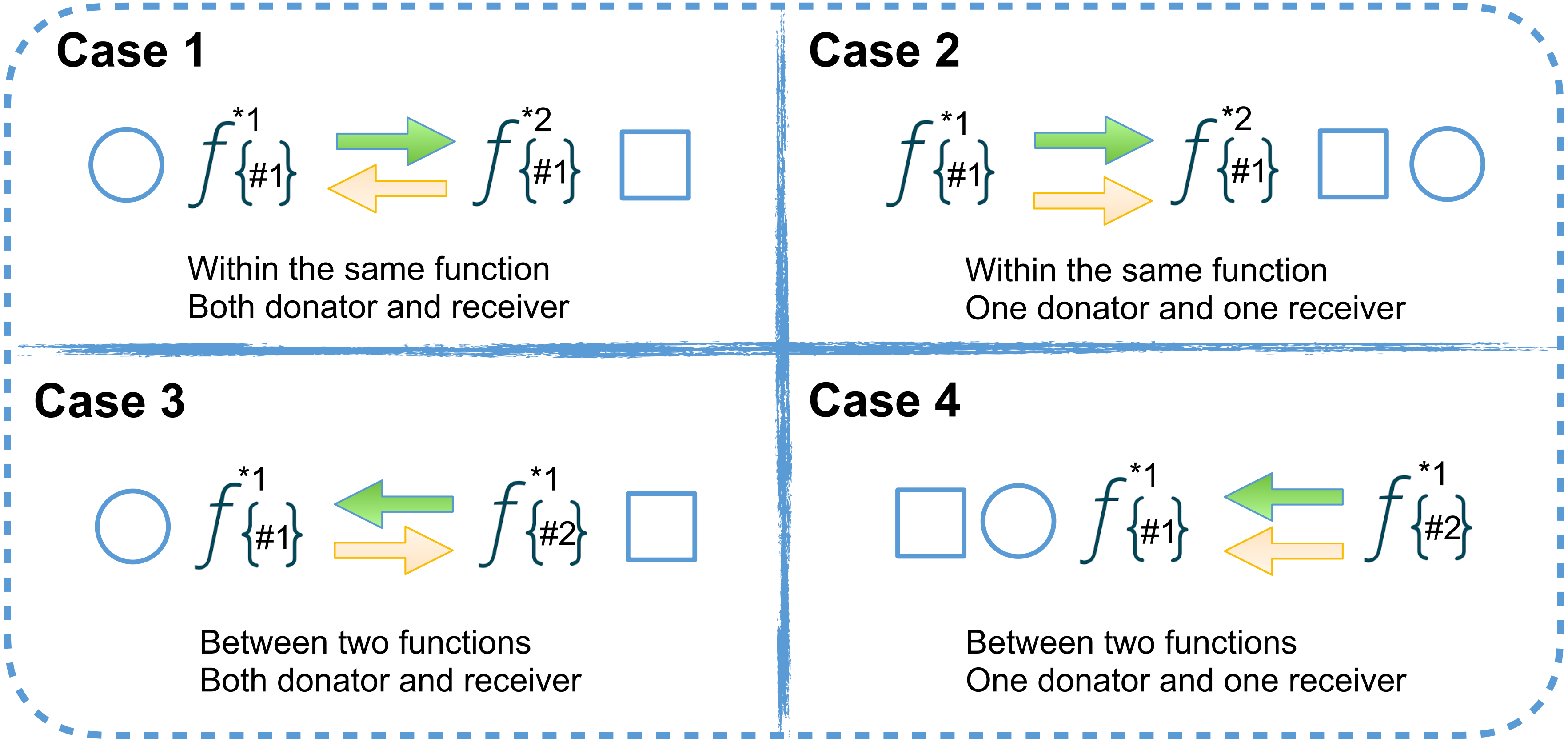
General Rebalance Cases

○ Idle core

□ Idle memory

# Function index

* Invocation index



Case 1

Within the same function
Both donator and receiver

Case 2

Within the same function
One donator and one receiver

Case 3

Between two functions
Both donator and receiver

Case 4

Between two functions
One donator and one receiver

Serverless Cluster

Dynamic Decisions

Perspective of a serverless platform:

- Varying **functions**

Dynamic Decisions

Perspective of a serverless platform:

- Varying **functions**
- Varying **invocations** per functions

Dynamic Decisions

Perspective of a serverless platform:

- Varying **functions**
- Varying **invocations** per functions
- Varying **input data** per invocation

Dynamic Decisions

Perspective of a serverless platform:

- Varying **functions**
- Varying **invocations** per functions
- Varying **input data** per invocation
- Every invocation requires an **allocation decision**

Dynamic Decisions

Perspective of a serverless platform:

- Varying **functions**
- Varying **invocations** per functions
- Varying **input data** per invocation
- Every invocation requires an **allocation decision**

A series of sequential allocation decisions

Dynamic Decisions

Perspective of a serverless platform:

- Varying **functions**
- Varying **invocations** per functions
- Varying **input data** per invocation
- Every invocation requires an **allocation decision**

A series of sequential allocation decisions

Markov Decision Process (MDP)

Deep Reinforcement Learning

Perspective of a serverless platform:

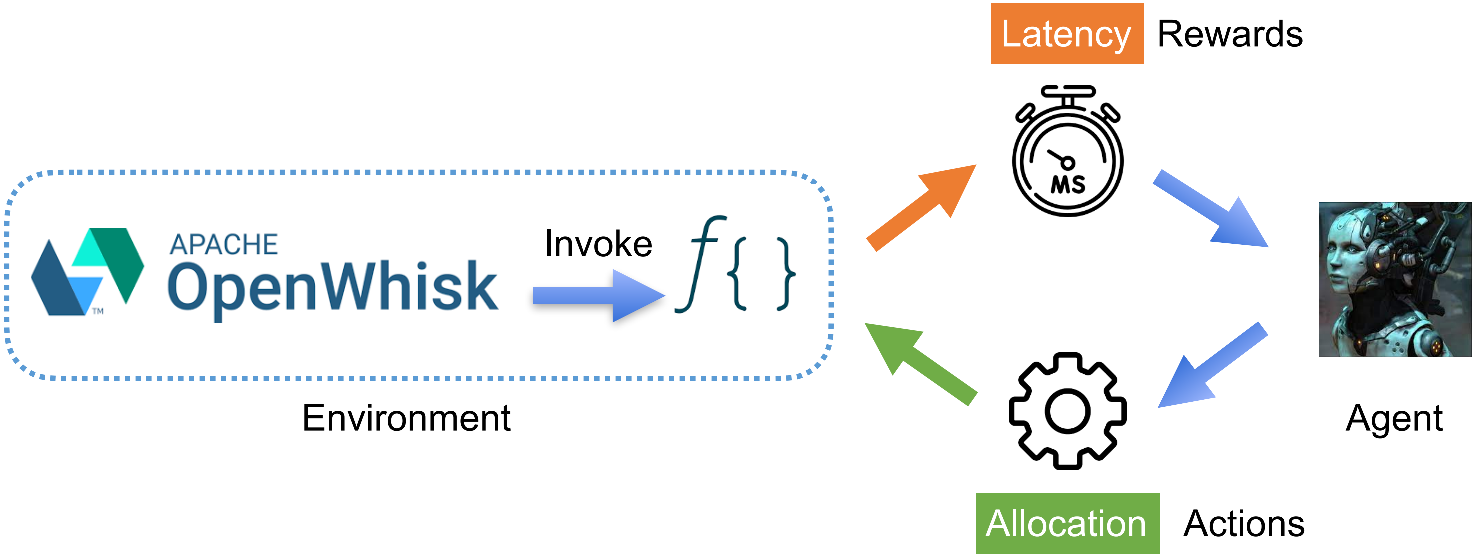
- Varying **functions**
- Varying **invocations** per functions
- Varying **input data** per invocation
- Every invocation requires an **allocation decision**

A series of sequential allocation decisions

Markov Decision Process (MDP)

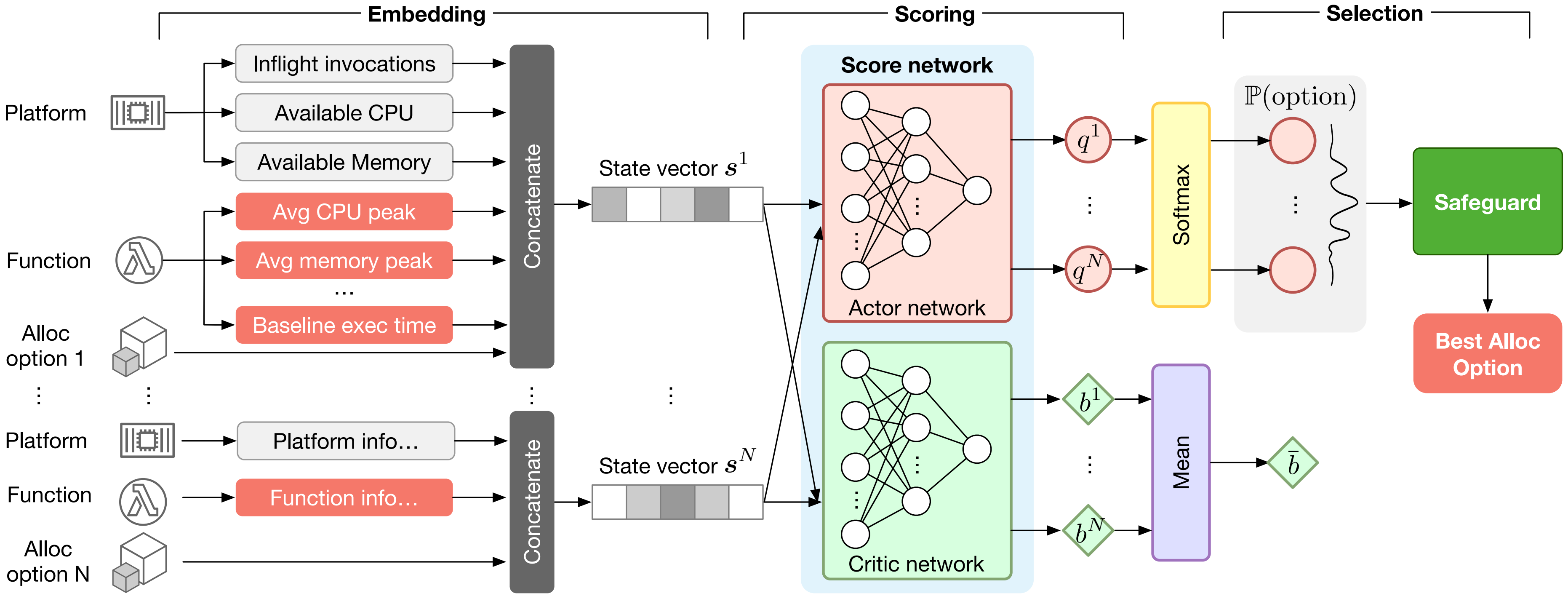
Deep Reinforcement Learning

Deep Reinforcement Learning

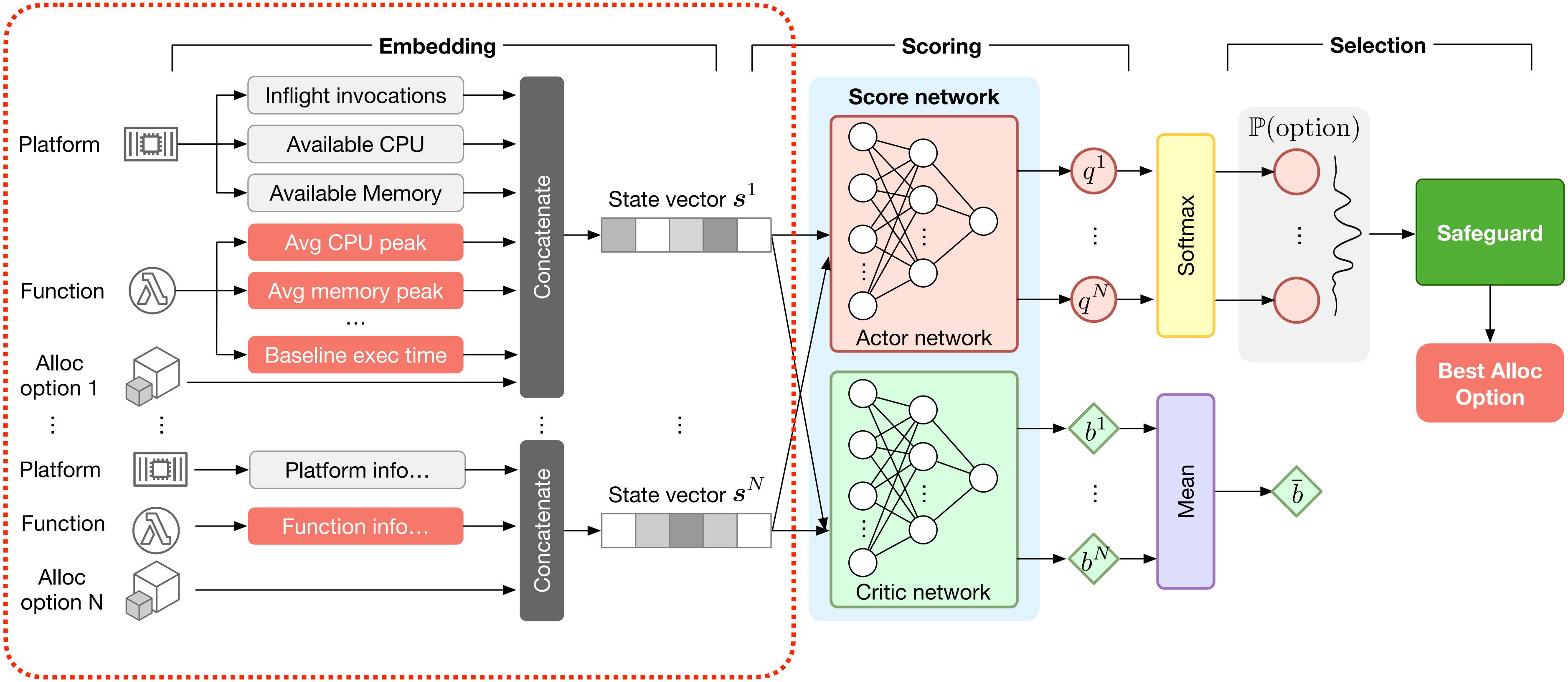


Freyr

Freyr Workflow

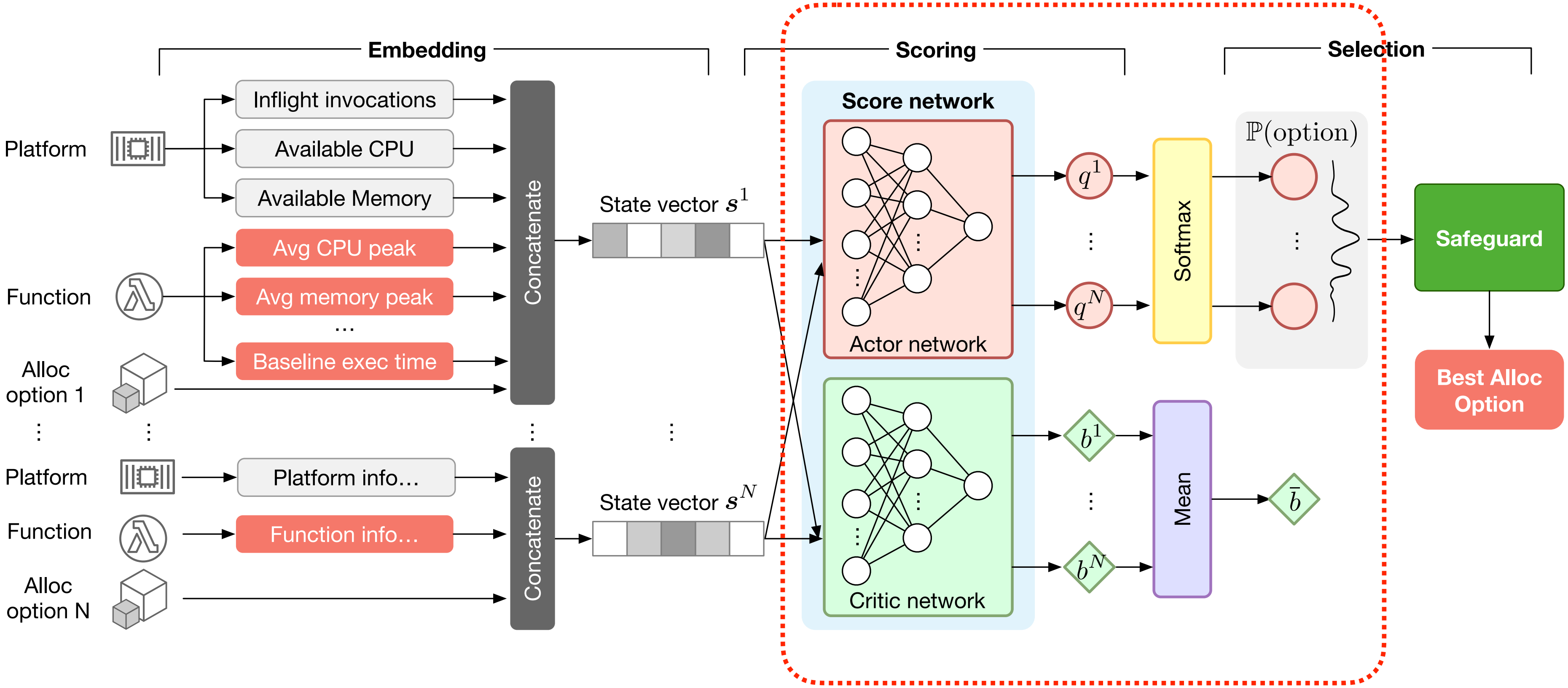


Freyr Workflow



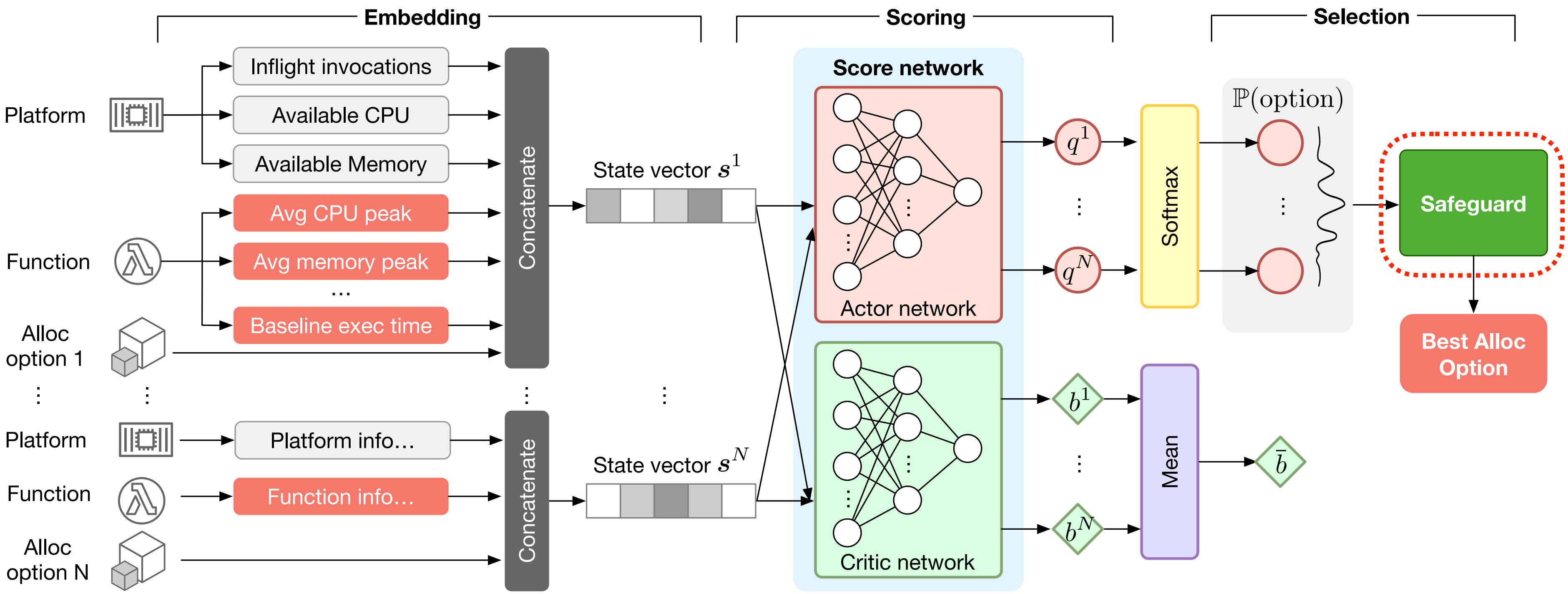
State information from the platform and the function

Freyr Workflow



Proximal Policy Optimization (PPO)

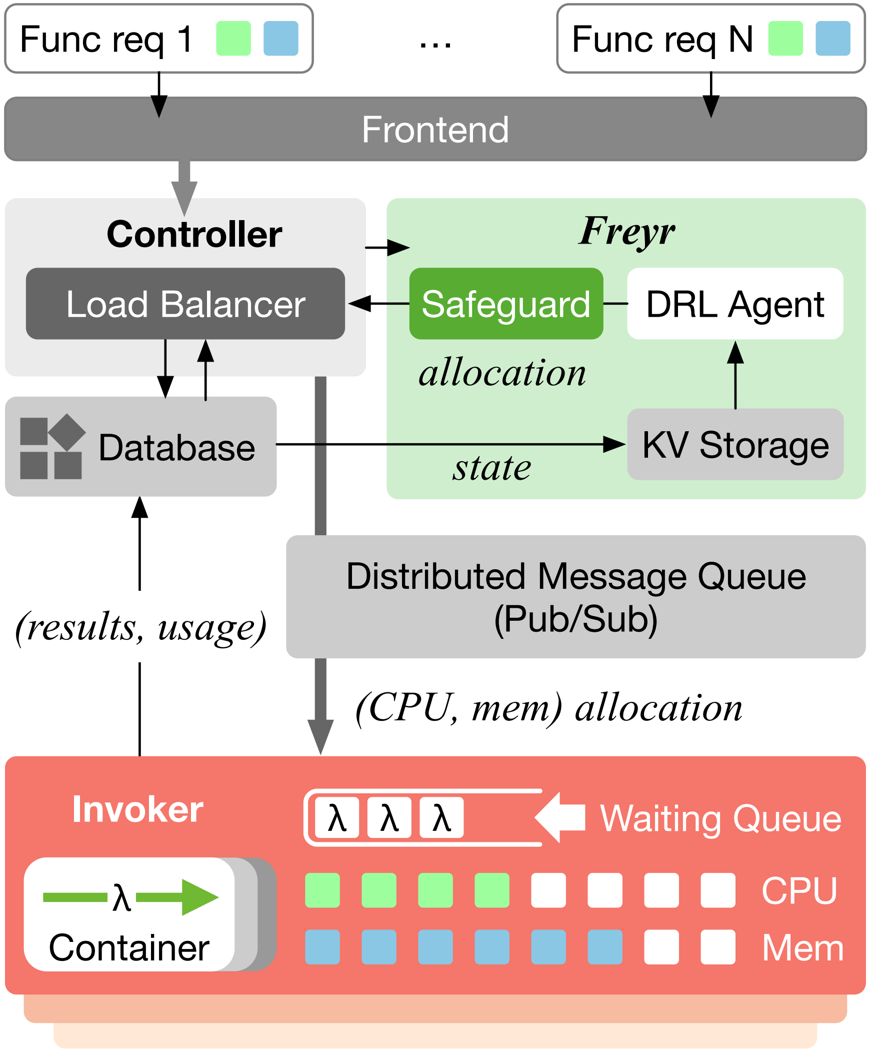
Freyr Workflow



Safeguard

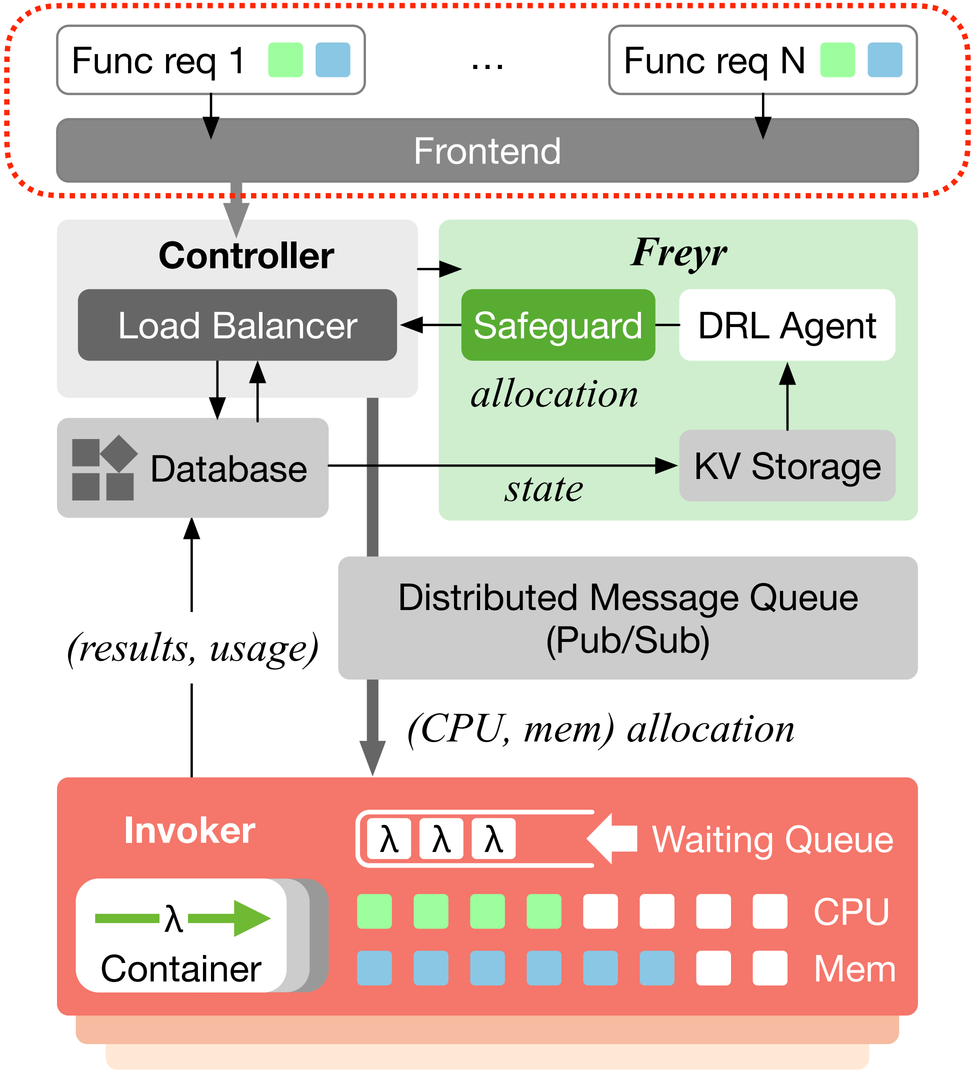
- Filter invalid allocation options
- Return resources when detecting a potential full usage

Freyr Architecture



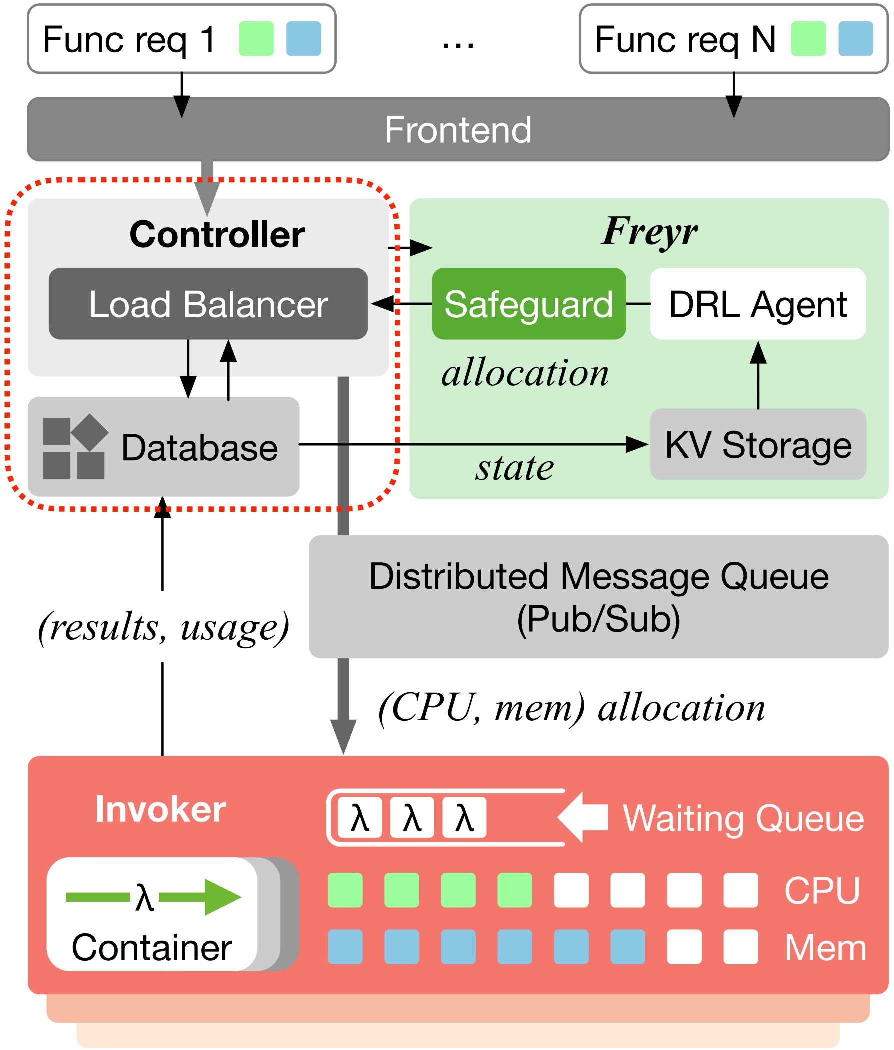
Freyr Architecture

Frontend receives function invocations from users

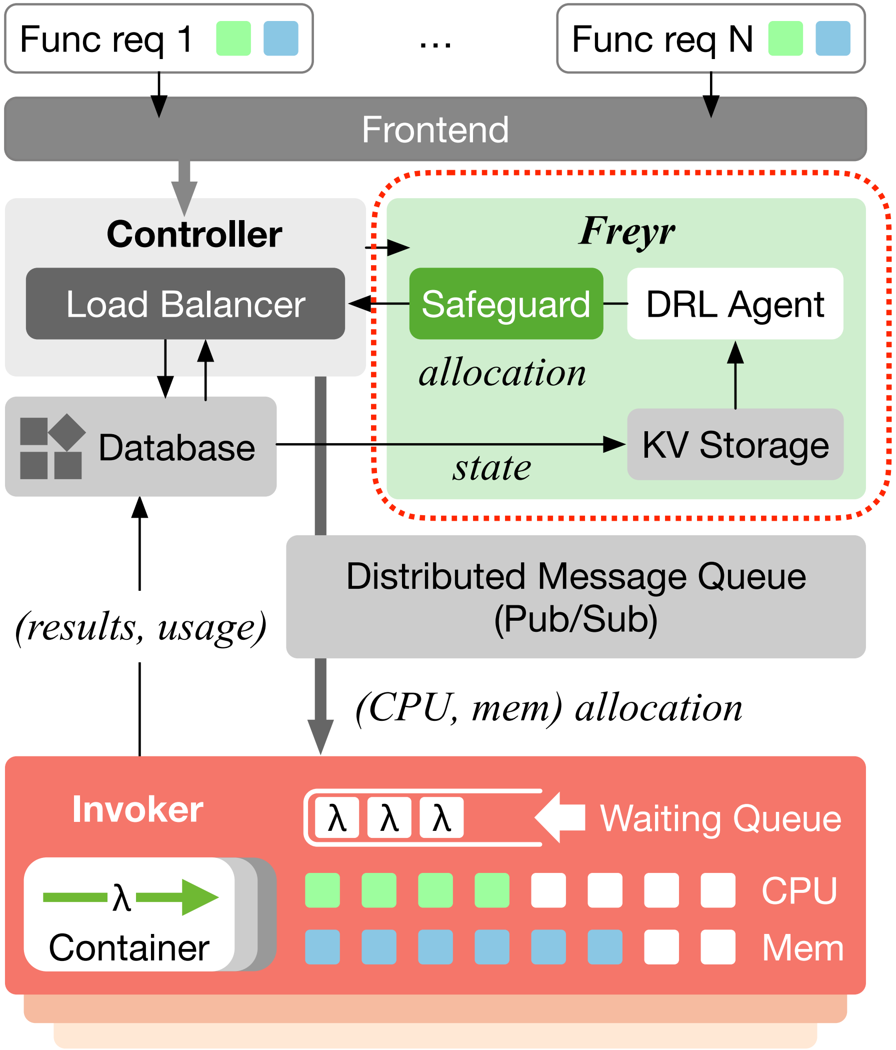


Freyr Architecture

Controller collects and sends states to the DRL agent

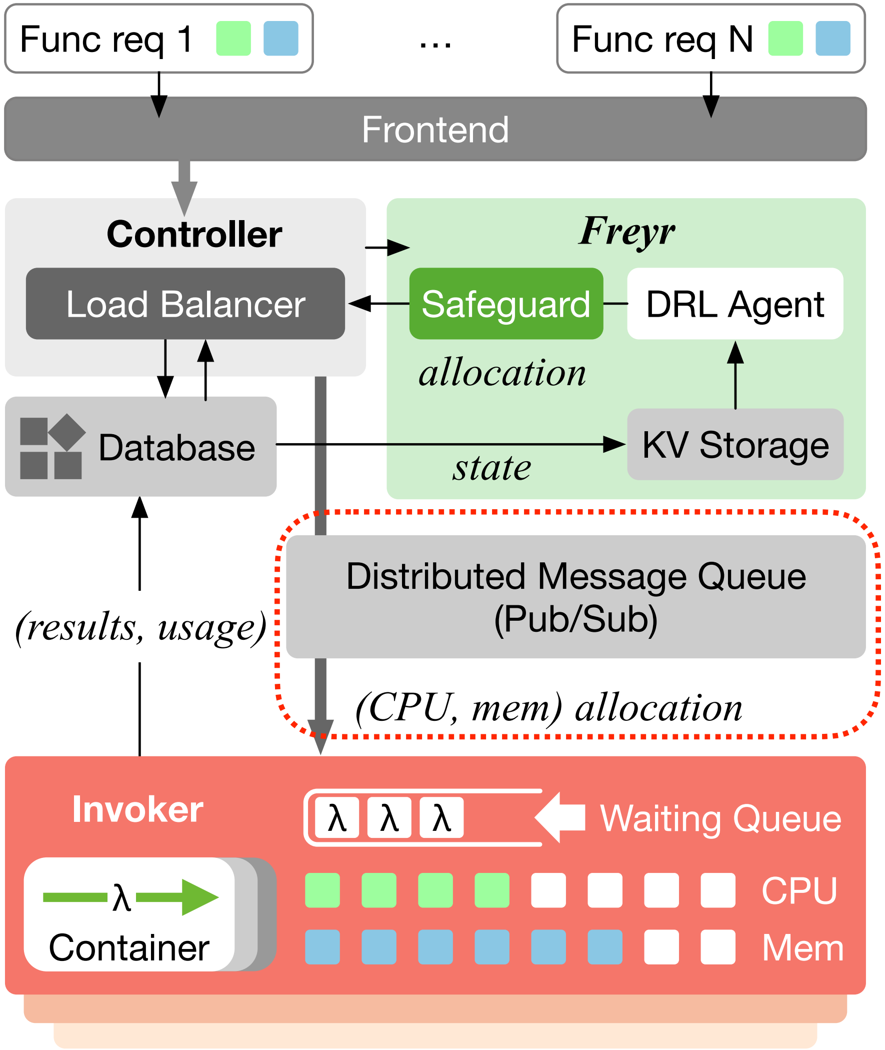


Freyr Architecture



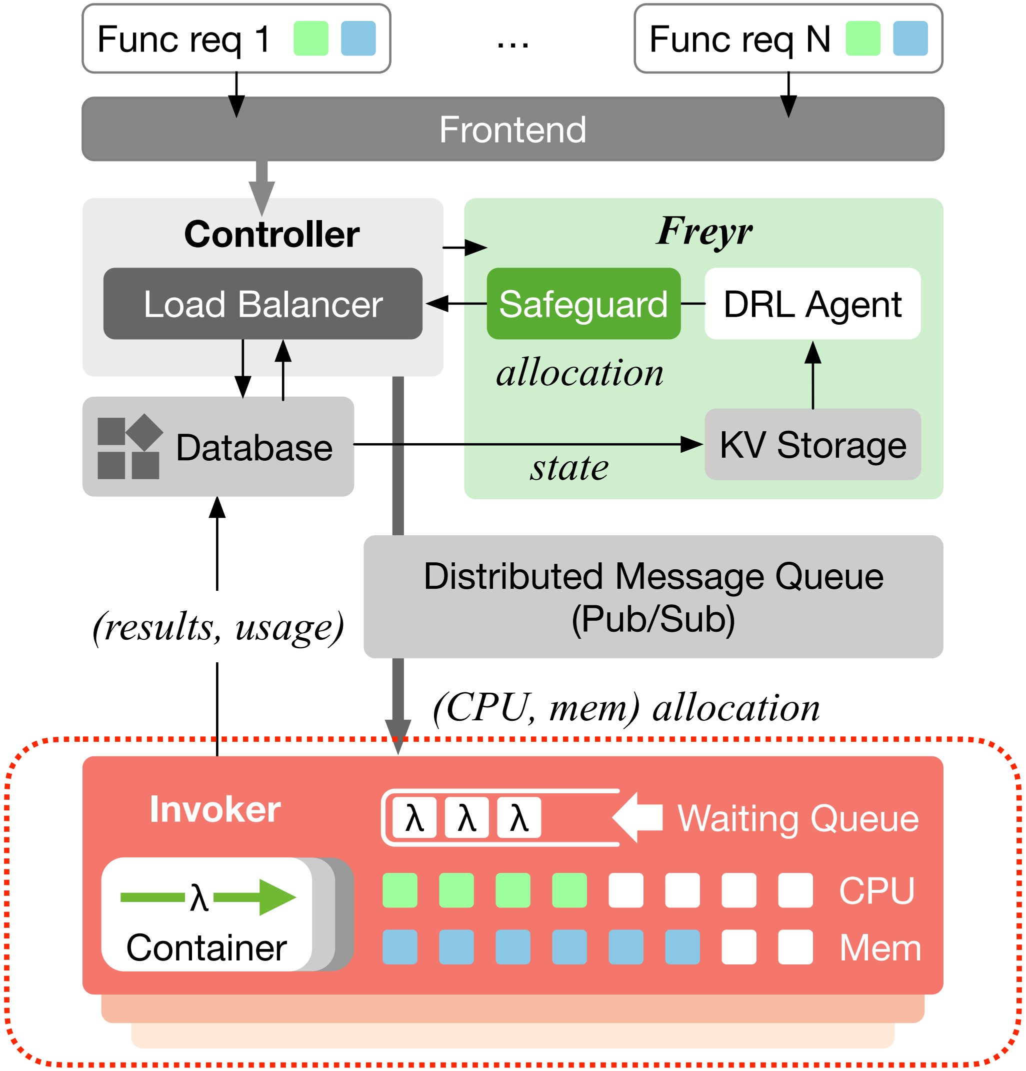
Agent predicts an allocation and sends it back to controller

Freyr Architecture



Controller then forwards the function invocation with its decision to an Invoker

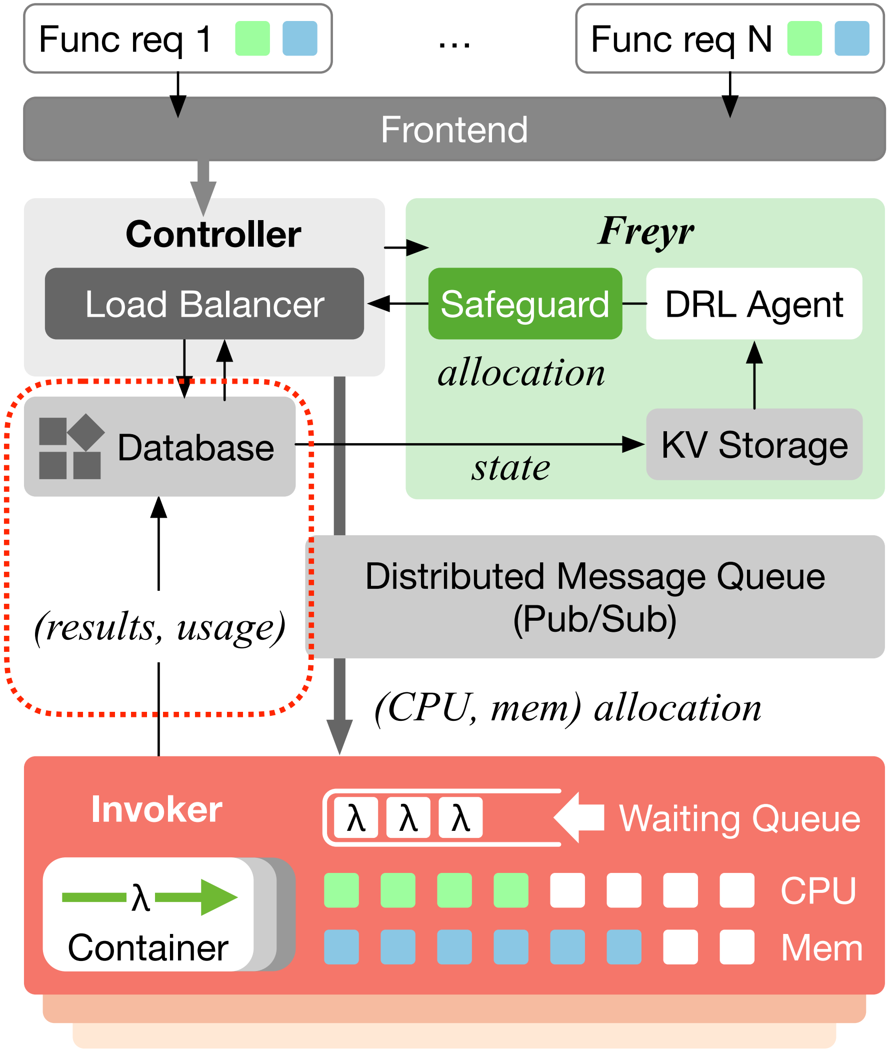
Freyr Architecture



Invoker executes the function

Freyr Architecture

Invoker submits the results and usage to database for further predictions



Experiment

Setup

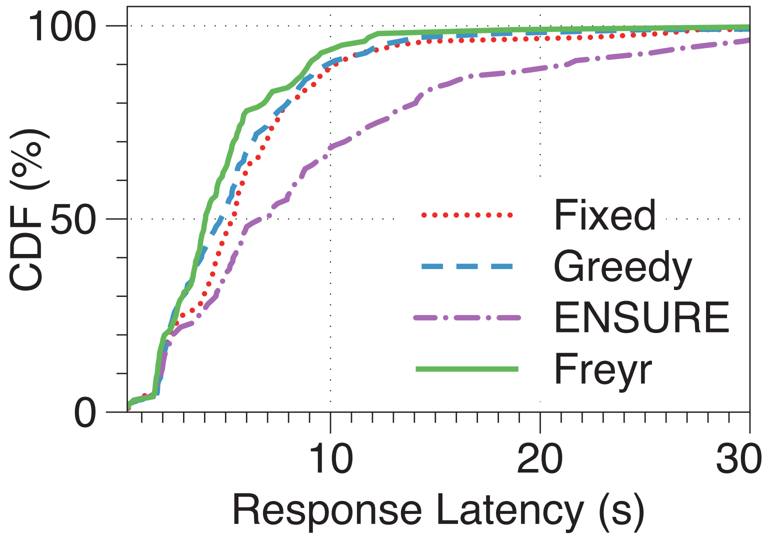
- 13 VMs, each with 8 CPUs and 32 GB memory
- One user client, one frontend, one controller
- 10 Worker nodes

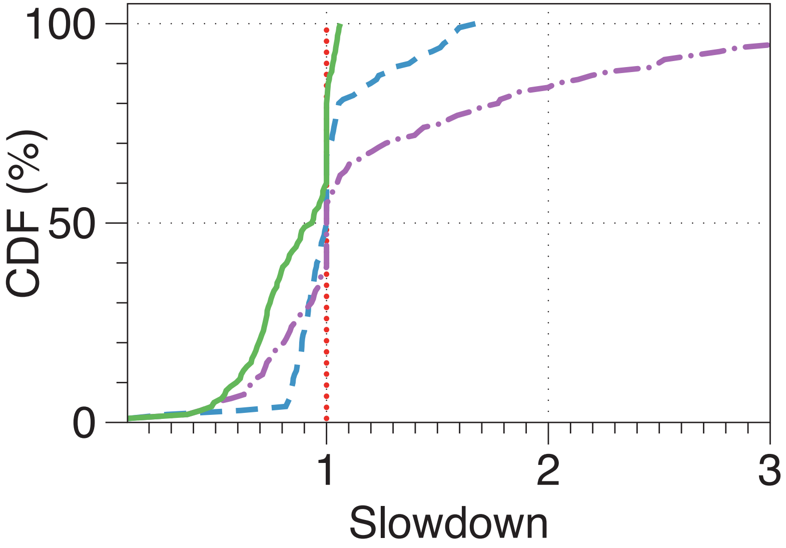
Baselines

- Fixed RM
- Greedy RM
- ENSURE

Fixed RM: default OpenWhisk as well as in existing serverless platforms
Greedy RM: heuristic
ENSURE: Suresh, Amoghavarsha, et al.
"Ensure: Efficient scheduling and autonomous resource management in serverless environments." (*ACSOS 2020)*

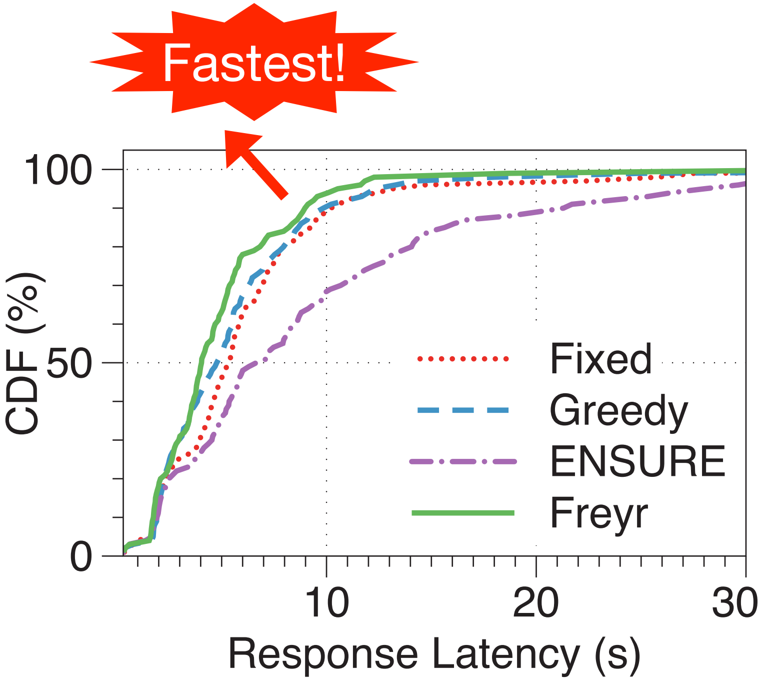
Function Execution Speedup

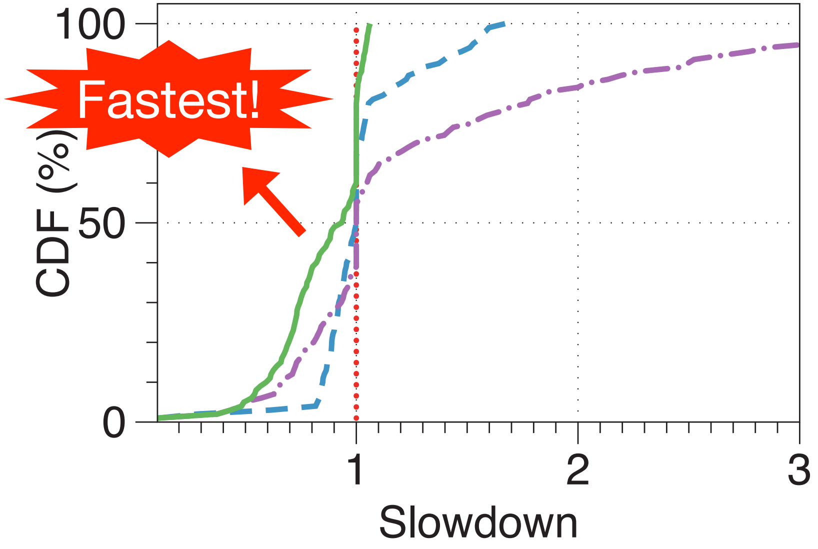




Response latency: function invocation end-to-end latency
Slowdown: relative performance compared to user-defined resources.
Larger than1.0 means degradation, less than 1.0 means speedup

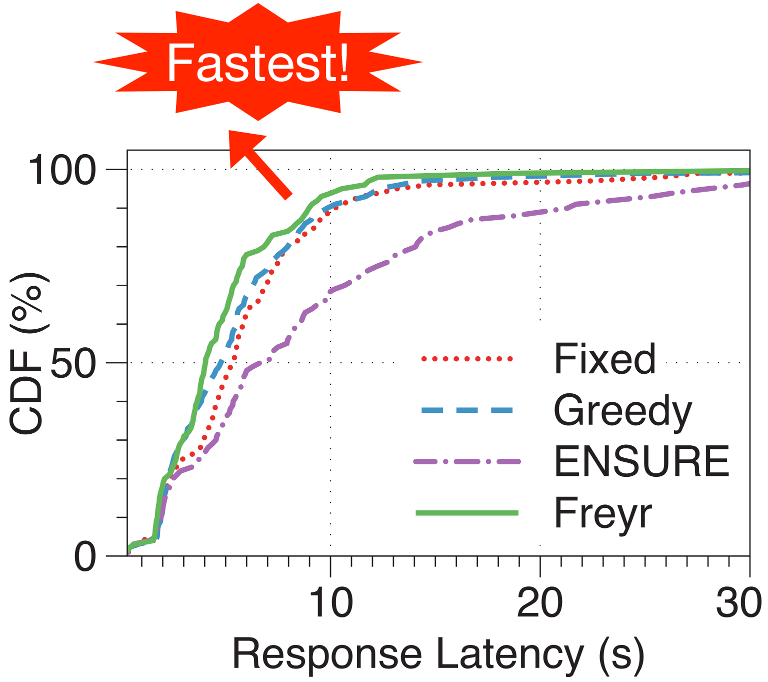
Function Execution Speedup


Fastest!

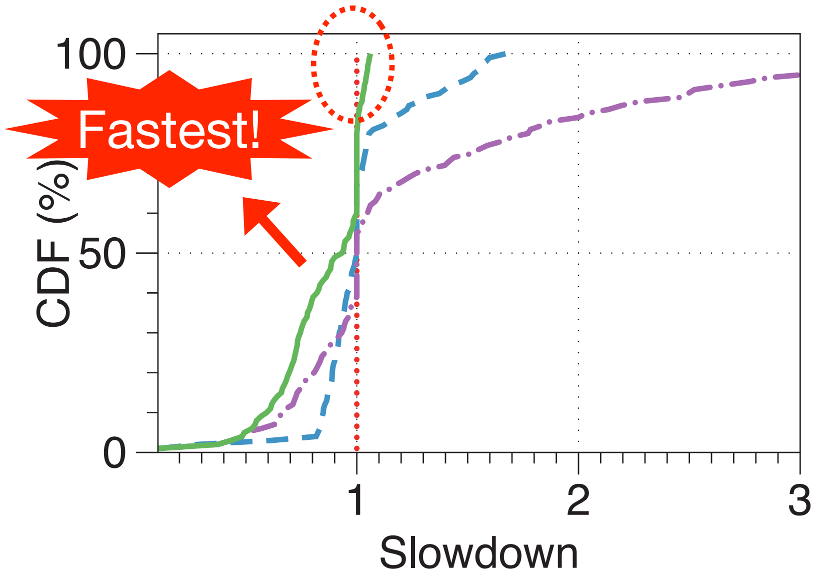

Fastest!

Response latency: function invocation end-to-end latency
Slowdown: relative performance compared to user-defined resources.
Larger than1.0 means degradation, less than 1.0 means speedup

Function Execution Speedup

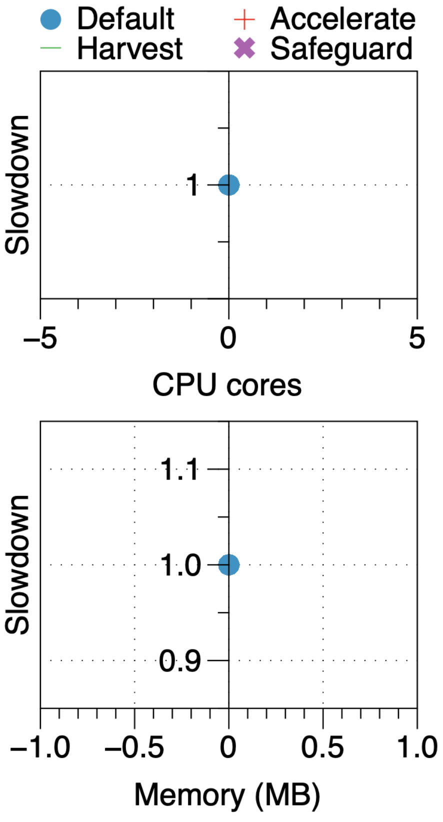


Negligible performance impact



Response latency: function invocation end-to-end latency
Slowdown: relative performance compared to user-defined resources. Larger than1.0 means degradation, less than 1.0 means speedup.

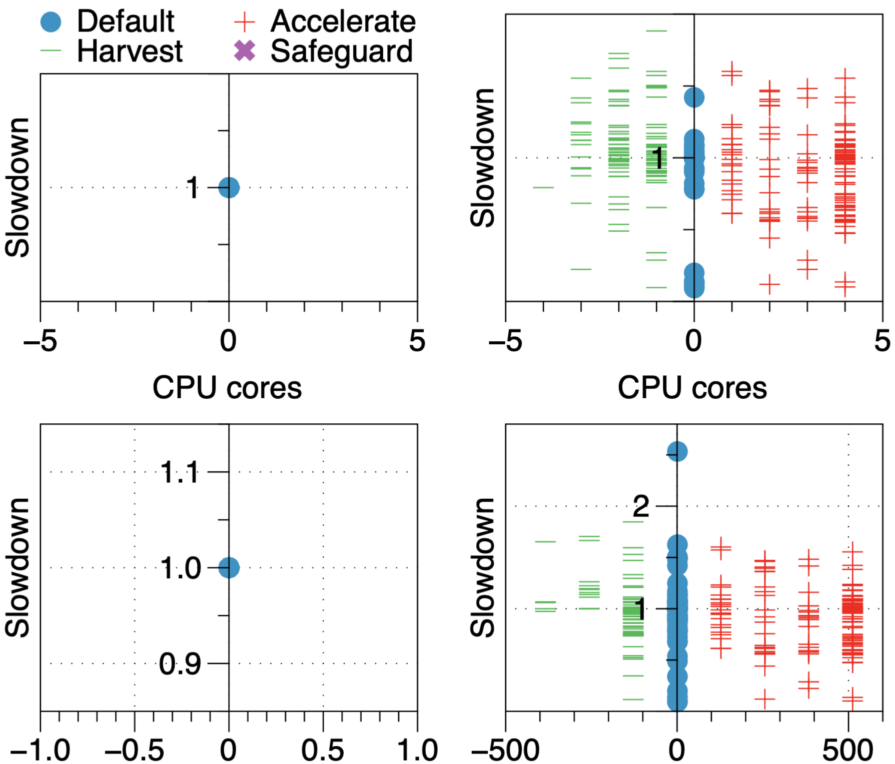
Resource Allocation



(a) Fixed RM

Slowdown: relative performance compared to user-defined resources. Larger than1.0 means degradation, less than 1.0 means speedup.

Resource Allocation

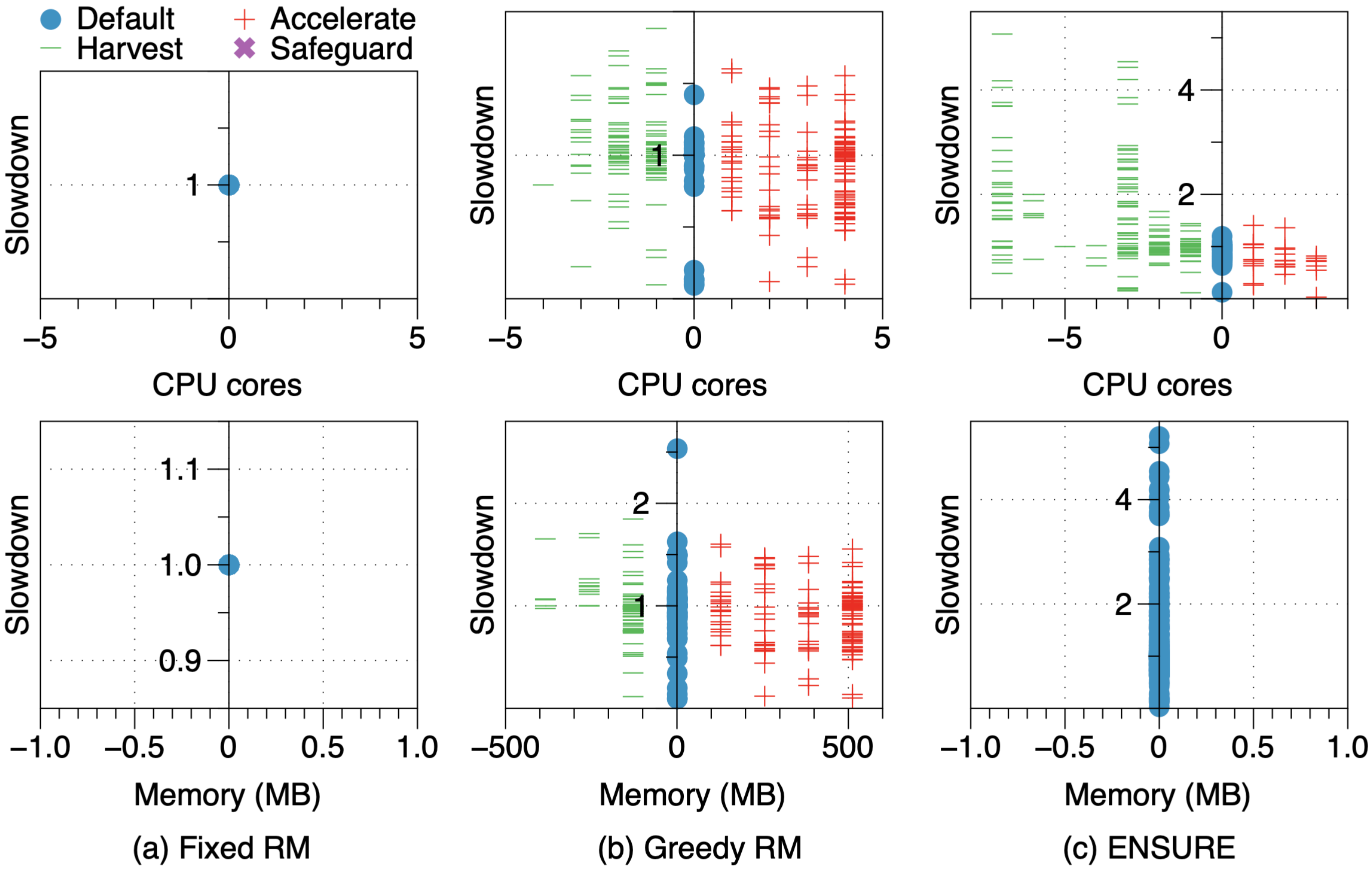


(a) Fixed RM

(b) Greedy RM

Slowdown: relative performance compared to user-defined resources. Larger than1.0 means degradation, less than 1.0 means speedup.

Resource Allocation



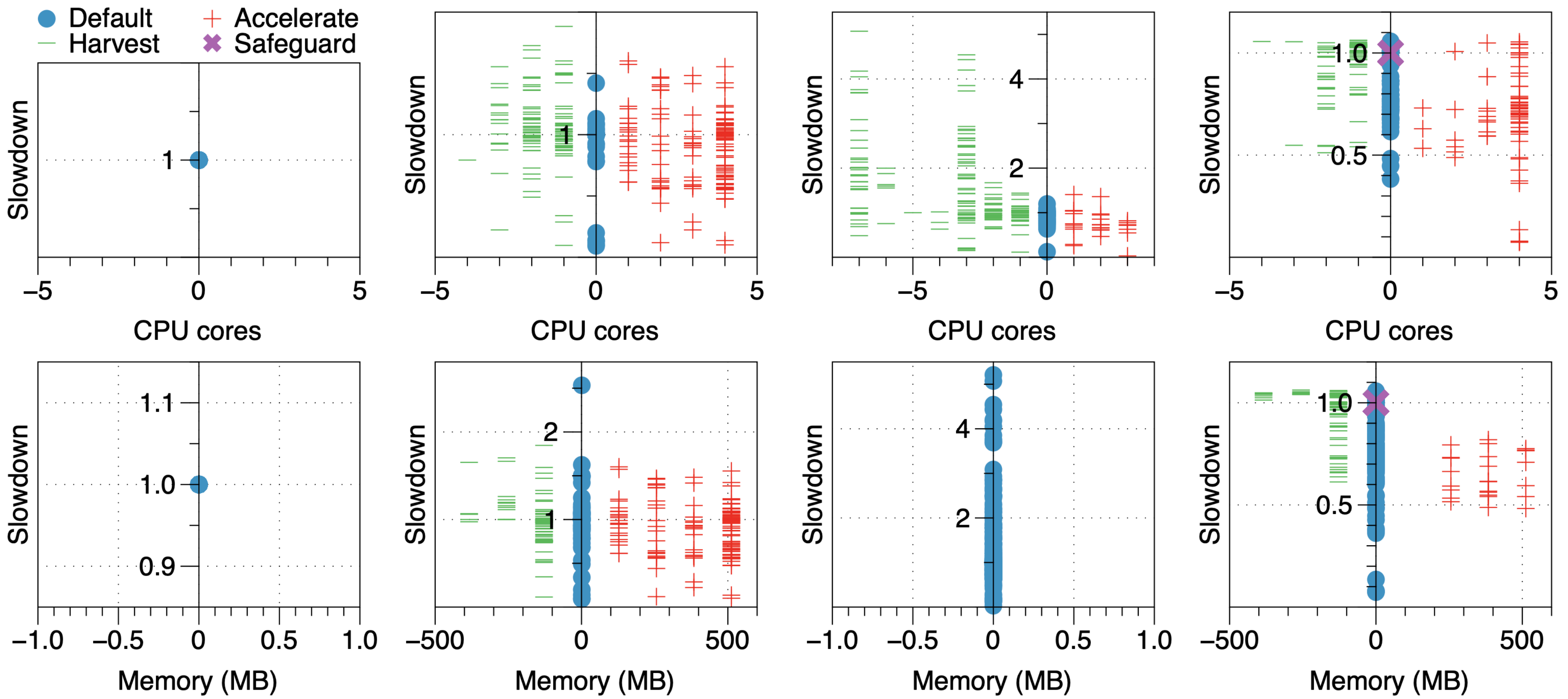
(a) Fixed RM

(b) Greedy RM

(c) ENSURE

Slowdown: relative performance compared to user-defined resources. Larger than1.0 means degradation, less than 1.0 means speedup.

Resource Allocation



(a) Fixed RM

(b) Greedy RM

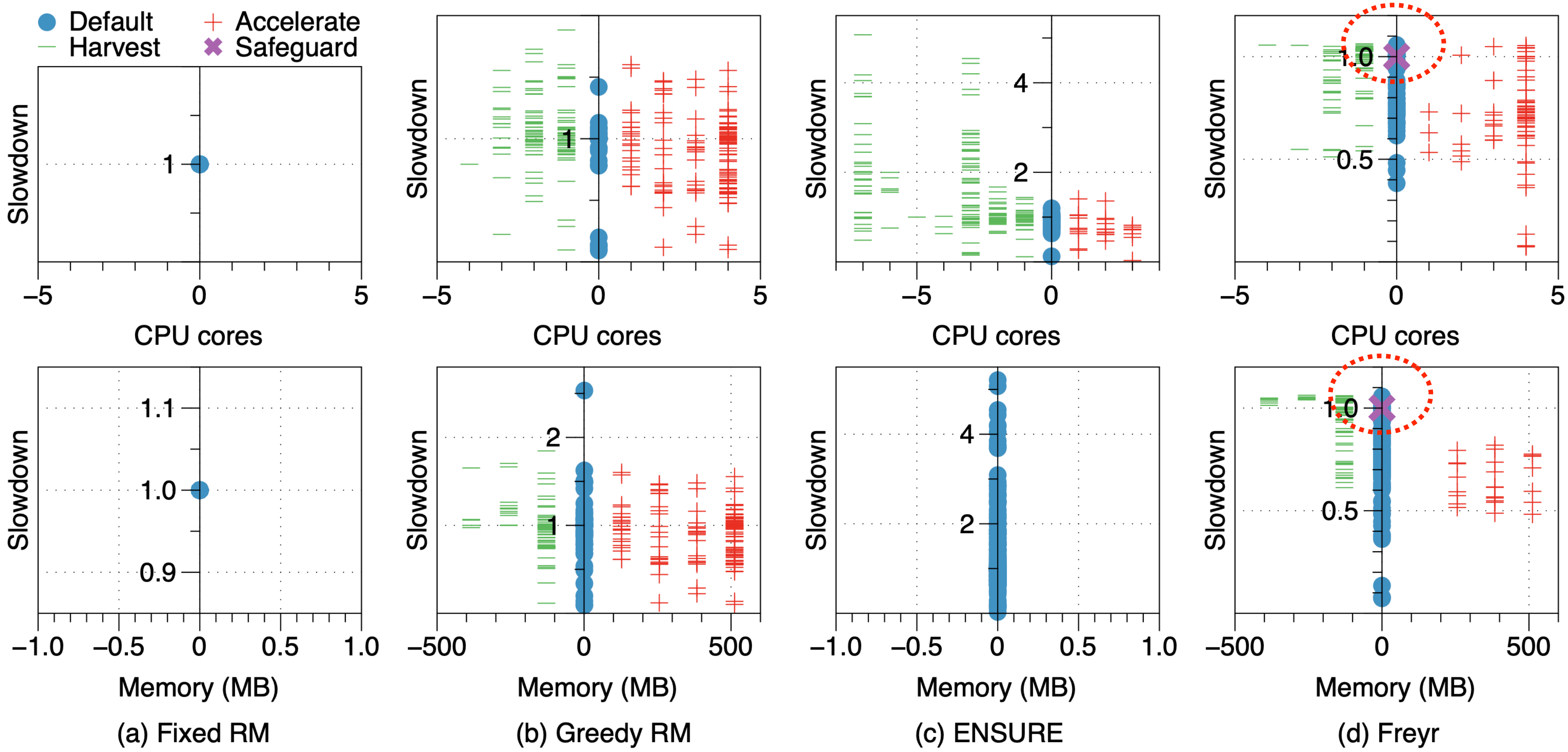
(c) ENSURE

(d) Freyr

Slowdown: relative performance compared to user-defined resources. Larger than1.0 means degradation, less than 1.0 means speedup.

Resource Allocation

Safeguard guarantees SLOs of harvested function invocations!



(a) Fixed RM

(b) Greedy RM

(c) ENSURE

(d) Freyr

Slowdown: relative performance compared to user-defined resources. Larger than1.0 means degradation, less than 1.0 means speedup.

Thank You

Hanfei Yu: hyu25@lsu.edu

IntelliSys Lab: https://intellisys.haow.ca

LSU

BINGHAMTON UNIVERSITY
State University of New York

UNIVERSITY of LOUISIANA LAFAYETTE