PennState

Fifer: Tackling Resource Underutilization in the Serverless Era

Jashwant Raj Gunasekaran, Prashanth Thinakaran, Nachiappan Chidambaram,
Mahmut Kandemir, Chita Das

ACM/IFIP Middleware'21
Dec 10, 2020

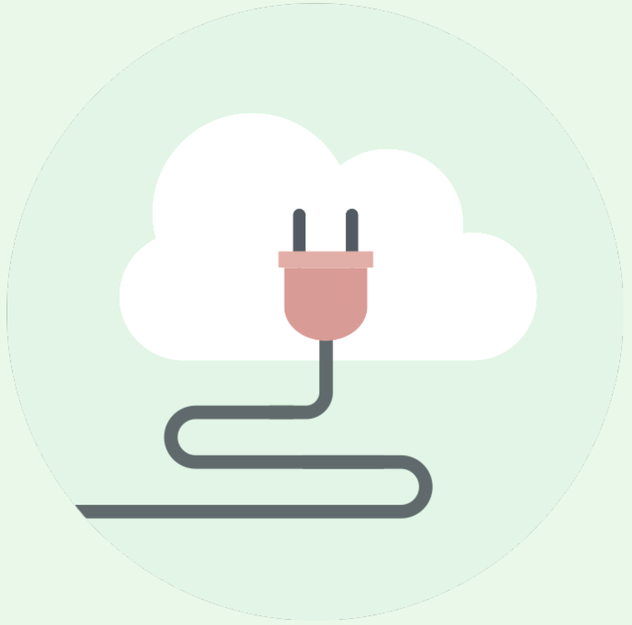
PennState
College of Engineering

PennState
High Performance
Computing Lab

Executive Summary

Tenants

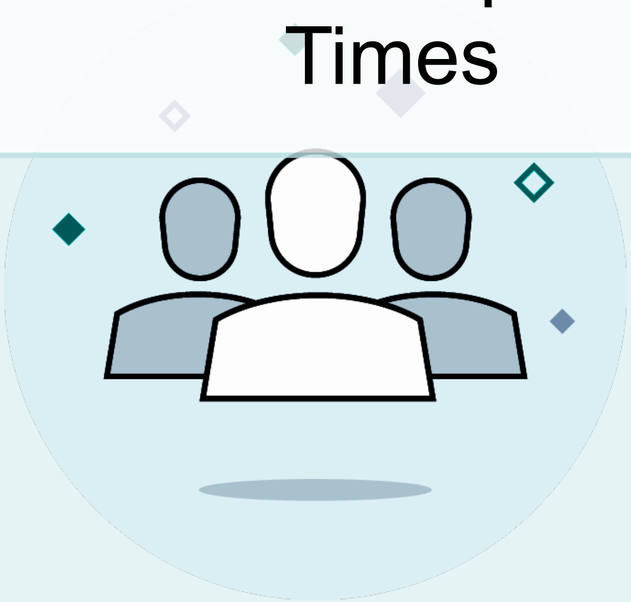
Providers



Executive Summary

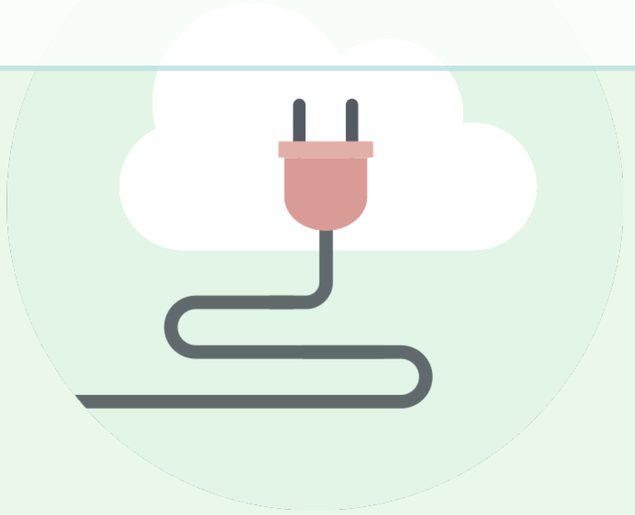
Tenants

Faster Response Times



Providers

Serverless Functions



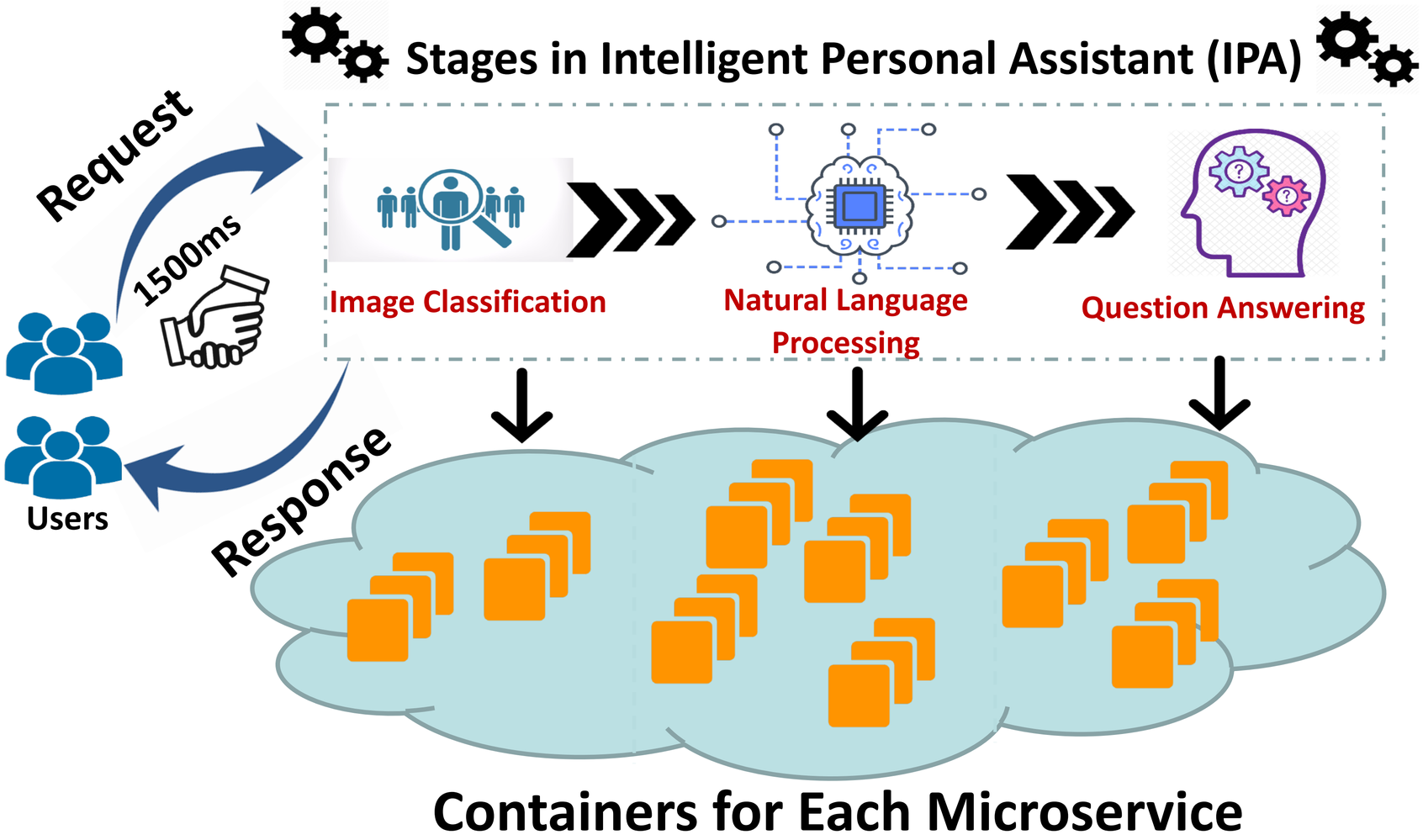
EXECUTIVE SUMMARY

TENANTS		PROVIDERS
Faster Response Times	→	Serverless Functions
SLO violations Cold-starts	↔	Over Provisioning & Underutilization

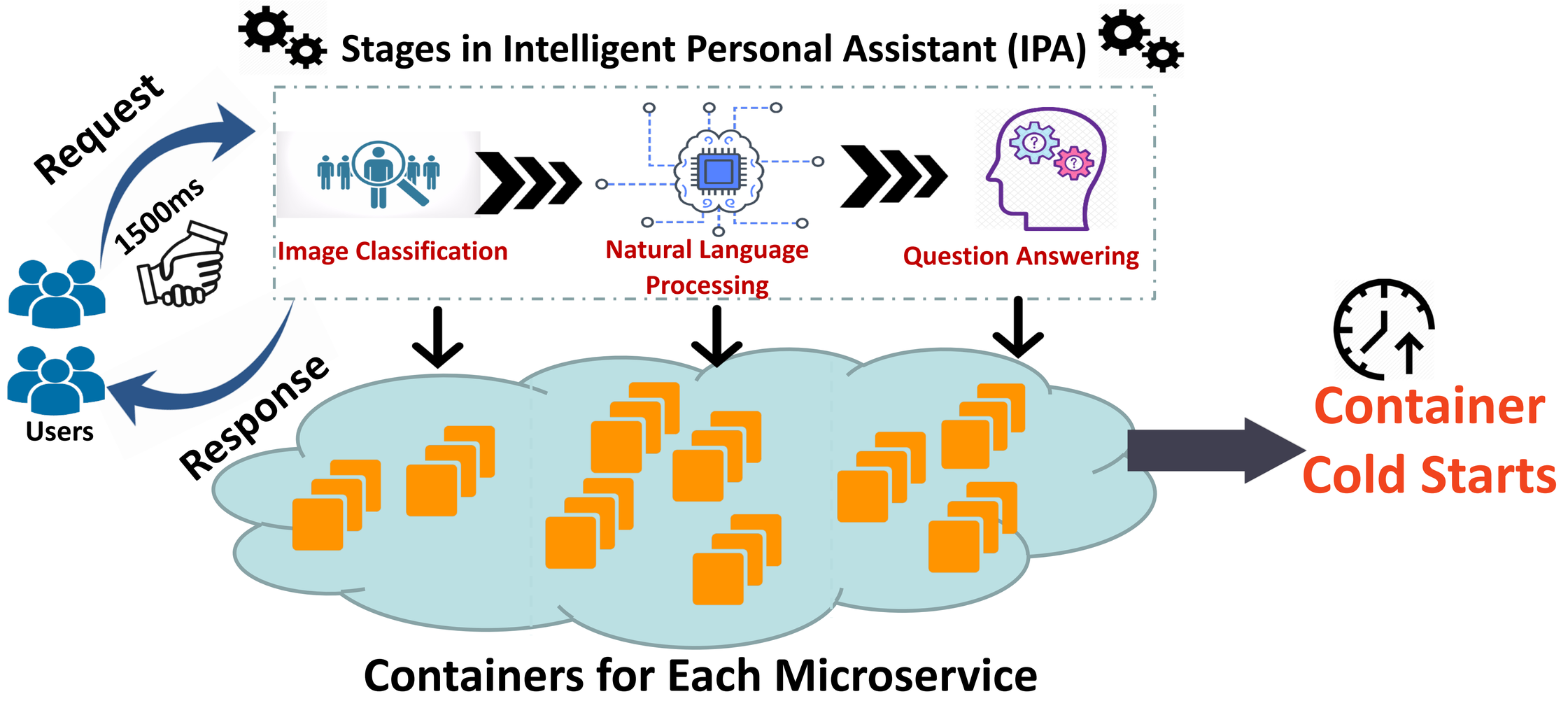
EXECUTIVE SUMMARY

TENANTS		PROVIDERS
Faster Response Times	→	Serverless Functions
SLO violations Cold-starts	↔	Over Provisioning & Underutilization
Guarantee SLOs	↔	Leverage Application Info Fully Utilize

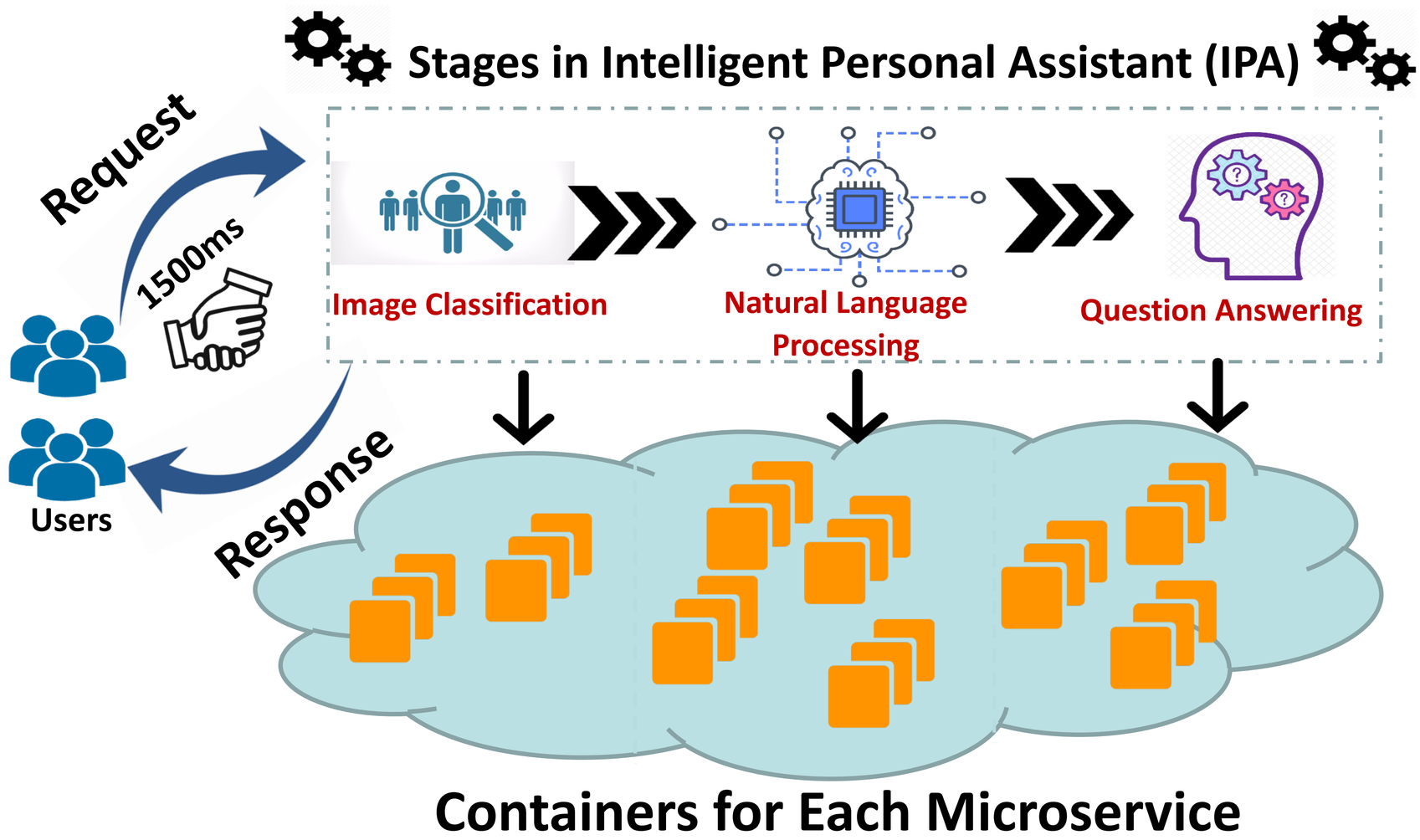
Serverless Function Chains



Serverless Function Chains



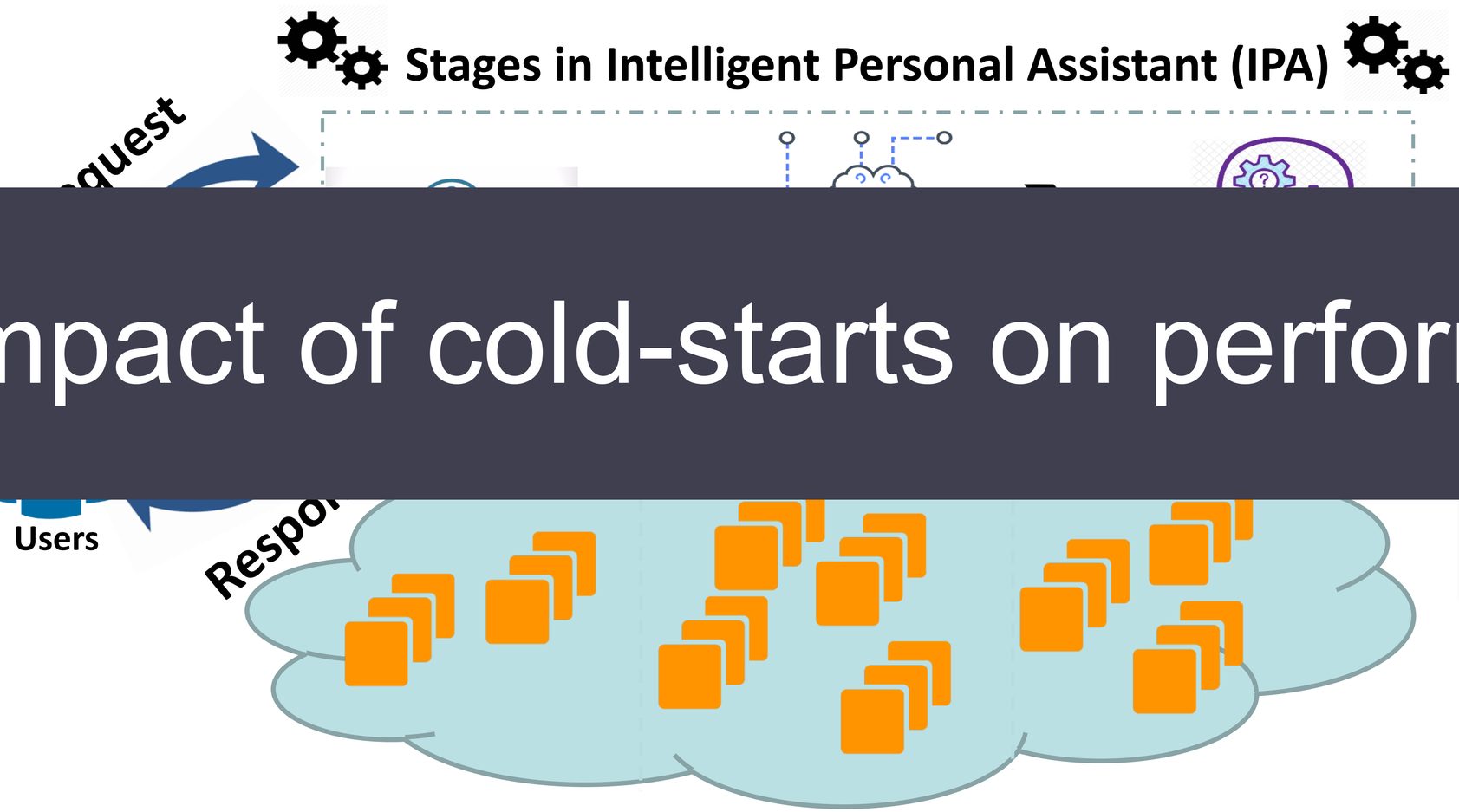
Serverless Function Chains



Container Cold Starts

- Container creation
- Model fetch time

Serverless Function Chains



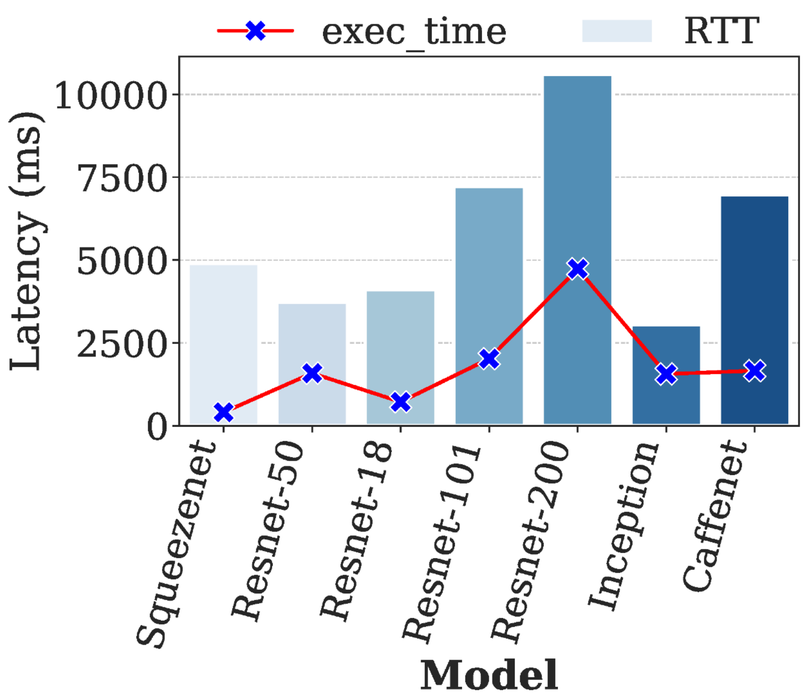
Container

- Model fetch time

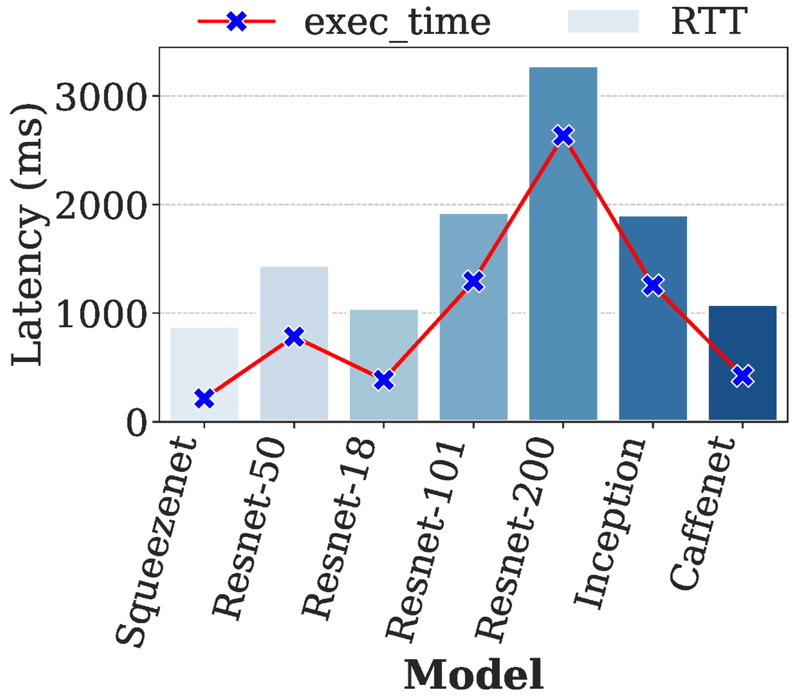
Impact of cold-starts on performance?

Why Cold Starts are bad?

Cold Start (First invocation)

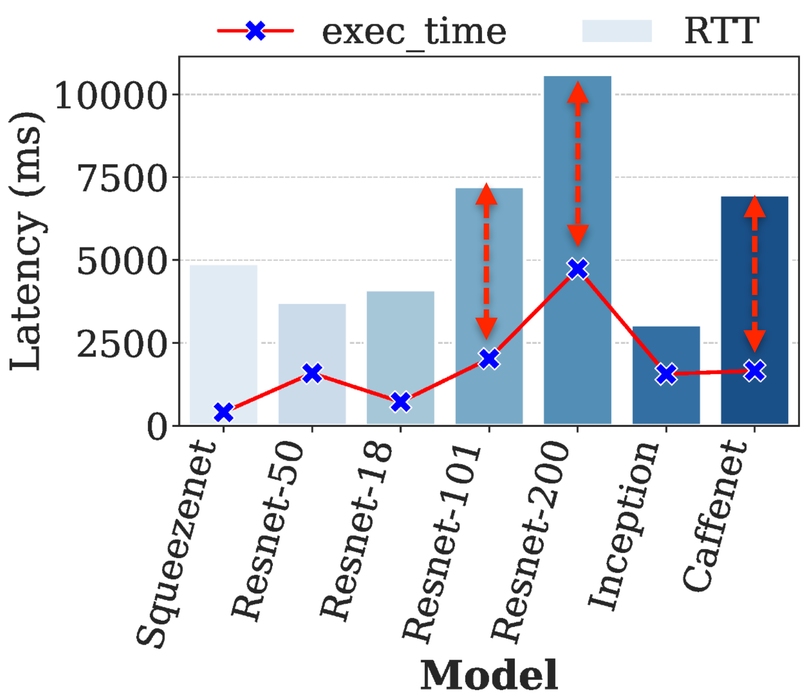


Warm Start (Concurrent Invocations)

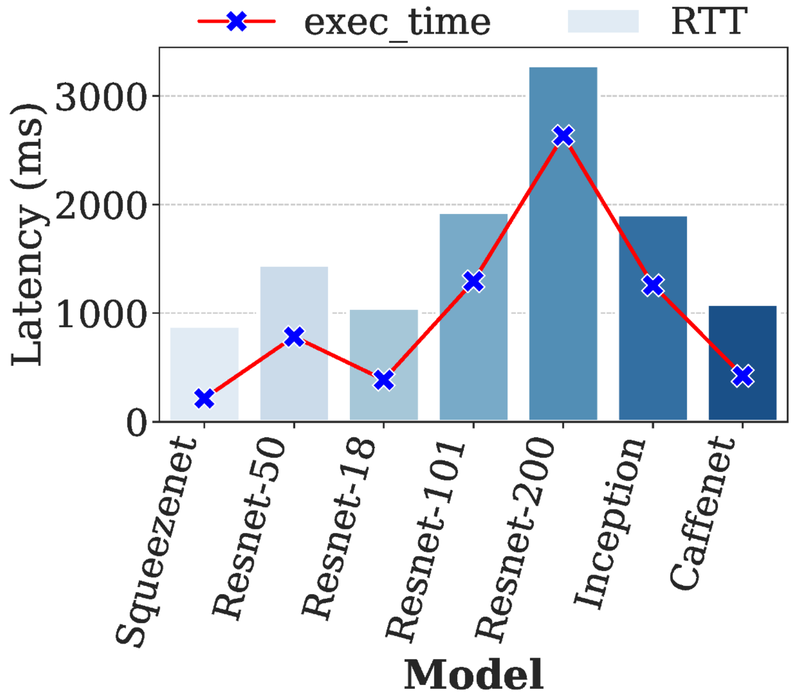


Why Cold Starts are bad?

Cold Start (First invocation)

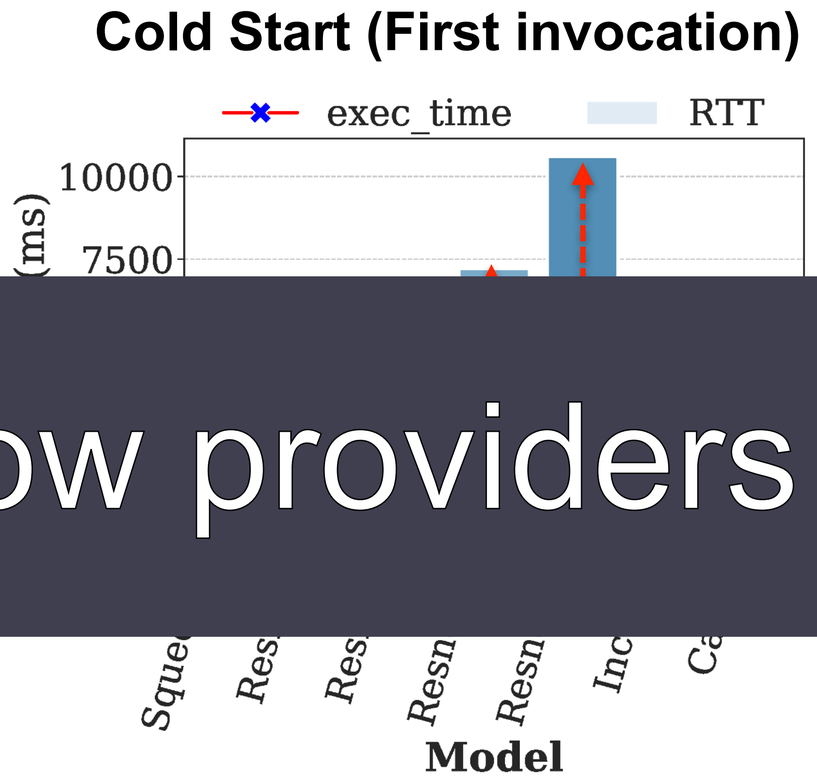


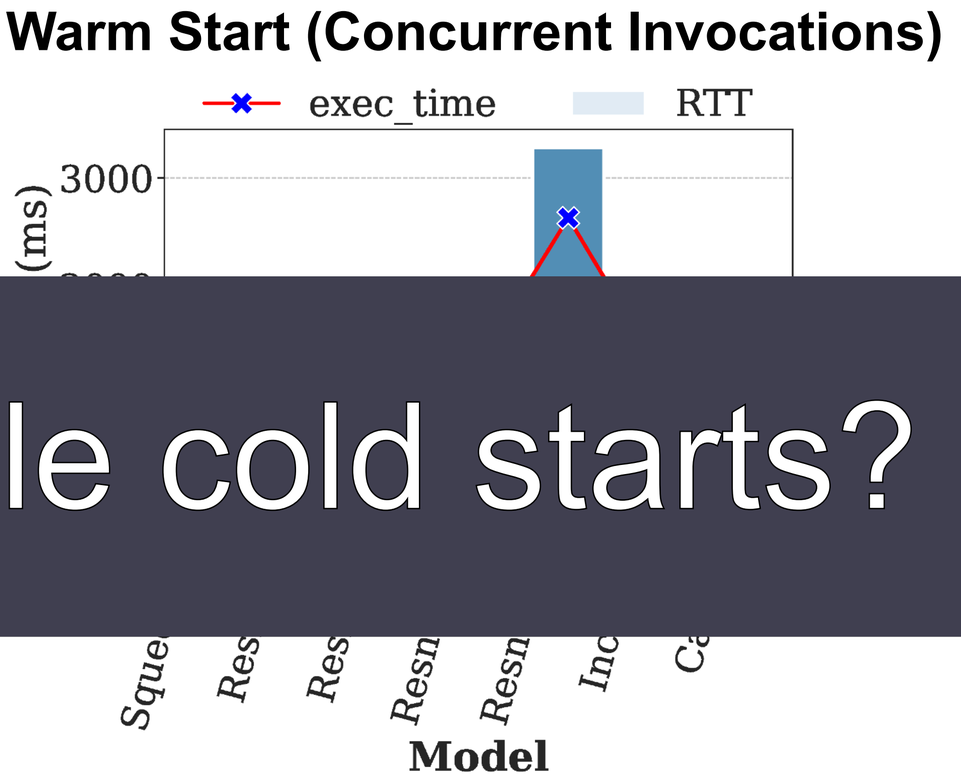
Warm Start (Concurrent Invocations)



Cold starts contribute ~2000 to 7500 ms on top of execution time

Why Cold Starts are bad?





How providers handle cold starts?

Cold starts contribute ~2000 to 7500 ms on top of execution time

Current Serverless Platforms

Wang et al, Peeking behind the curtains of Serverless Platforms in ATC'18

Current Serverless Platforms

- Spawn new containers if existing containers are busy.
  - ➡ Leads to SLO violations due to cold-starts.
  - ➡ Many idle containers. Wasted power and energy.

Wang et al, Peeking behind the curtains of Serverless Platforms in ATC'18

Current Serverless Platforms

- Spawn new containers if existing containers are busy.
  - ➡ Leads to SLO violations due to cold-starts.
  - ➡Many idle containers. Wasted power and energy.

- Employing static queuing of requests on fixed pool of containers
  - ➡Leads to SLO violations due to queuing.

Wang et al, Peeking behind the curtains of Serverless Platforms in ATC'18

Current Serverless Platforms

- Spawn new containers if existing containers are busy.
  - ➡ Leads to SLO violations due to cold-starts.
  - ➡ Many idle containers. Wasted power and energy.

- Employing static queuing of requests on fixed pool of containers
  - ➡ Leads to SLO violations due to queuing.

- Not aware of application execution times and response latency requirements.
  - ➡ Colossal container overprovisioning.

Wang et al, Peeking behind the curtains of Serverless Platforms in ATC'18

Current Serverless Platforms

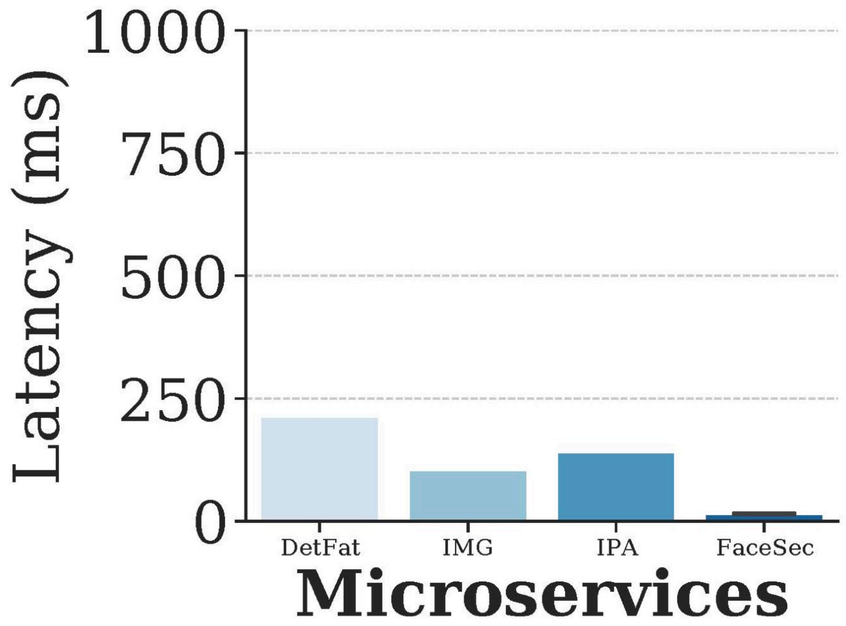
- Spawn new containers if existing containers are busy.
  - ➡ Leads to SLO violations due to cold-starts.
  - ➡ Many idle containers. Wasted power and energy.

How can we do better?

- Not aware of application execution times and response latency requirements.
  - ➡ Colossal container overprovisioning.

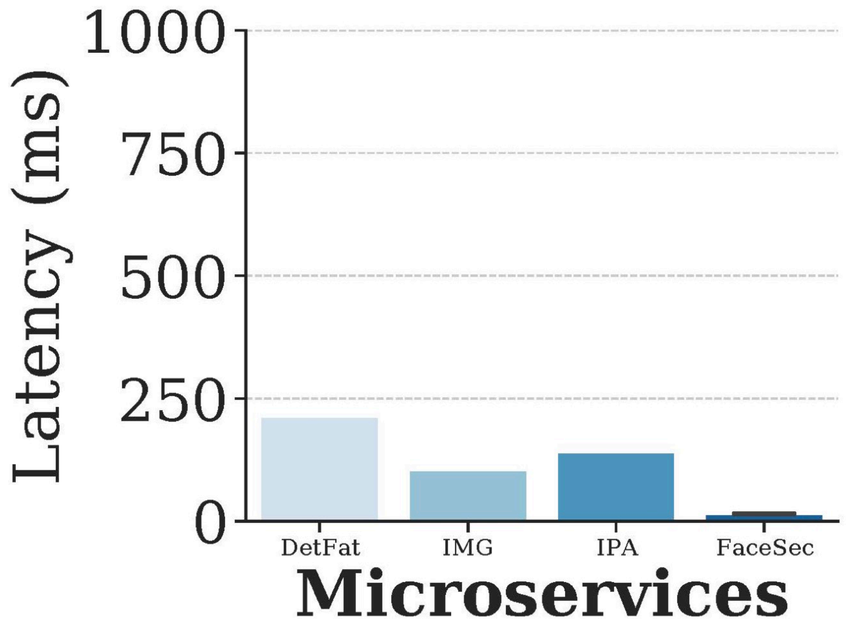
Wang et al, Peeking behind the curtains of Serverless Platforms in ATC'18

Application Characterization



Djinn and Tonic- DNN Inference Benchmark Suite-ISCA'15

Application Characterization

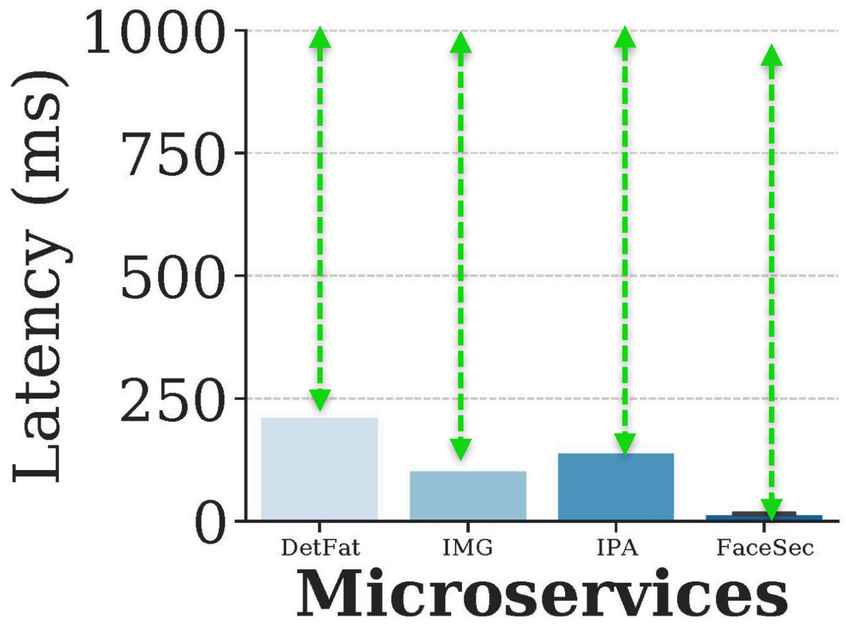


SLO's usually within 1.5s

Djinn and Tonic- DNN Inference Benchmark Suite-ISCA'15

Swayam: Distributed Autoscaling to Meet SLAs of Machine Learning Inference Services, Middleware'17

Application Characterization



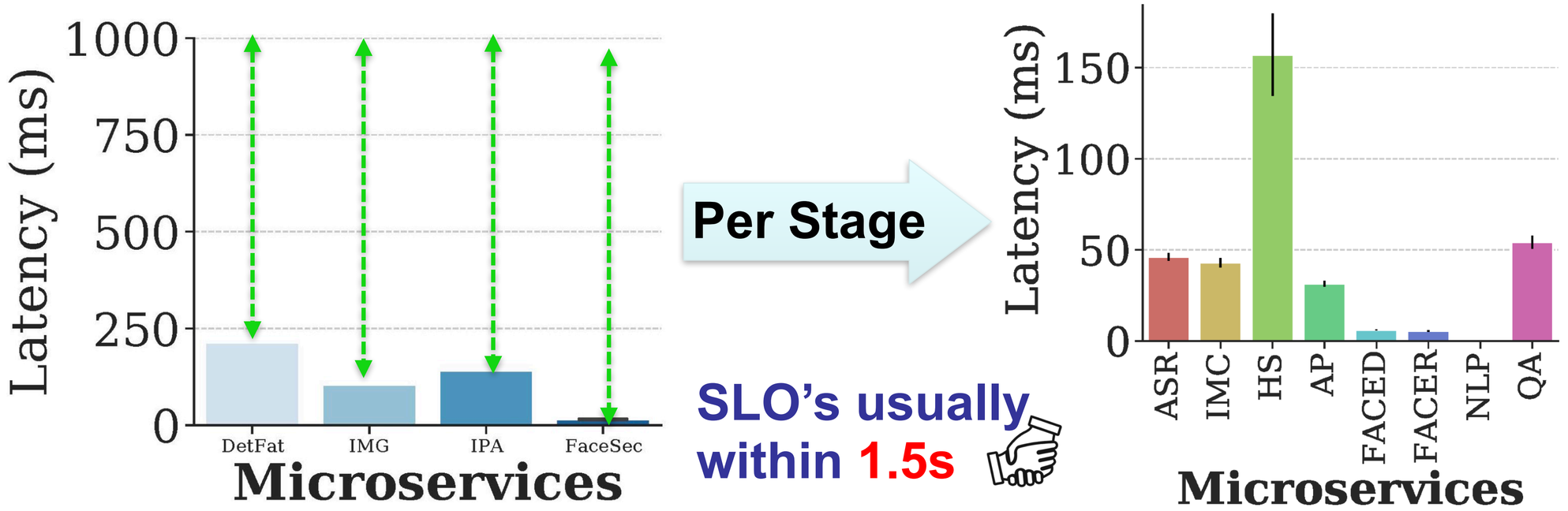
SLO's usually within 1.5s

Djinn and Tonic- DNN Inference Benchmark Suite-ISCA'15

Swayam: Distributed Autoscaling to Meet SLAs of Machine Learning Inference Services, Middleware'17

- Multi-staged applications have ample slack.

Application Characterization

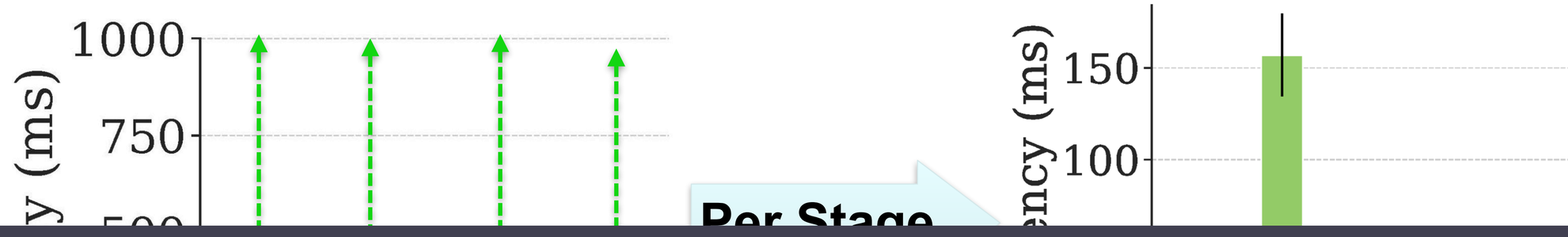


Djinn and Tonic- DNN Inference Benchmark Suite-ISCA'15

Swayam: Distributed Autoscaling to Meet SLAs of Machine Learning Inference Services, Middleware'17

- Multi-staged applications have ample slack.
- Execution times of each function is predictable.

Application Characterization



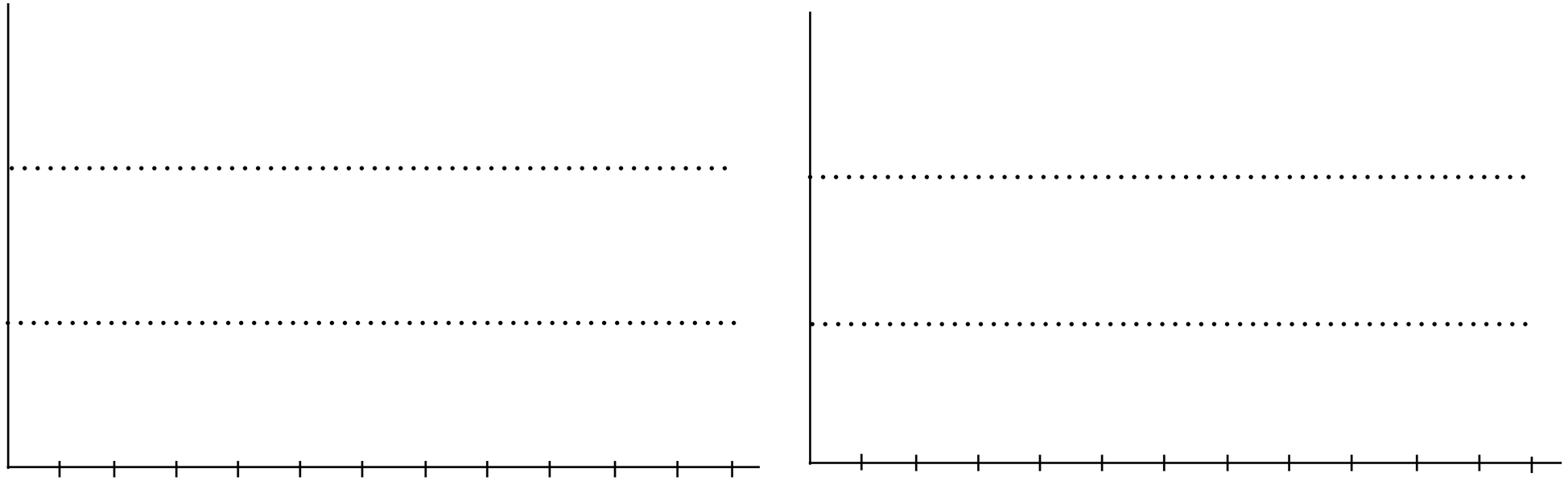
How to exploit the slack and execution time predictability?

Djinn and Tonic- DNN Inference Benchmark Suite-ISCA'15

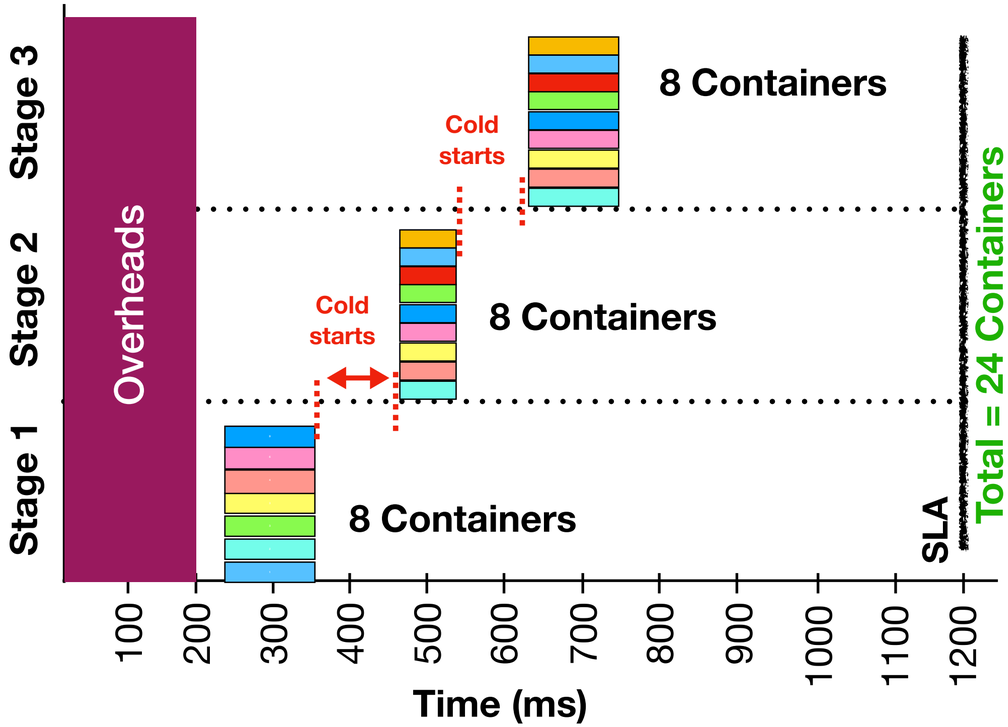
Swayam: Distributed Autoscaling to Meet SLAs of Machine Learning Inference Services, Middleware'17

- Multi-staged applications have ample slack.
- Execution times of each function is predictable.

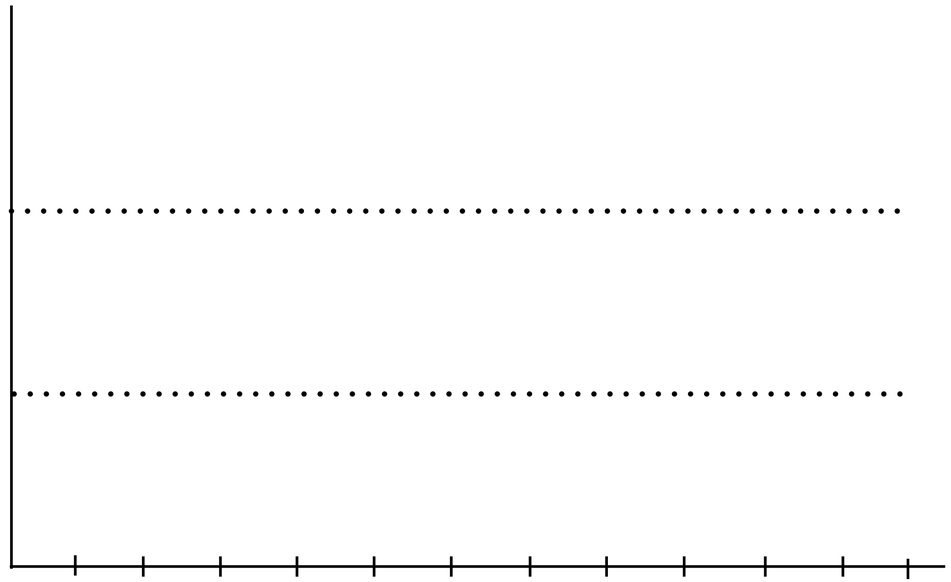
Slack aware queuing



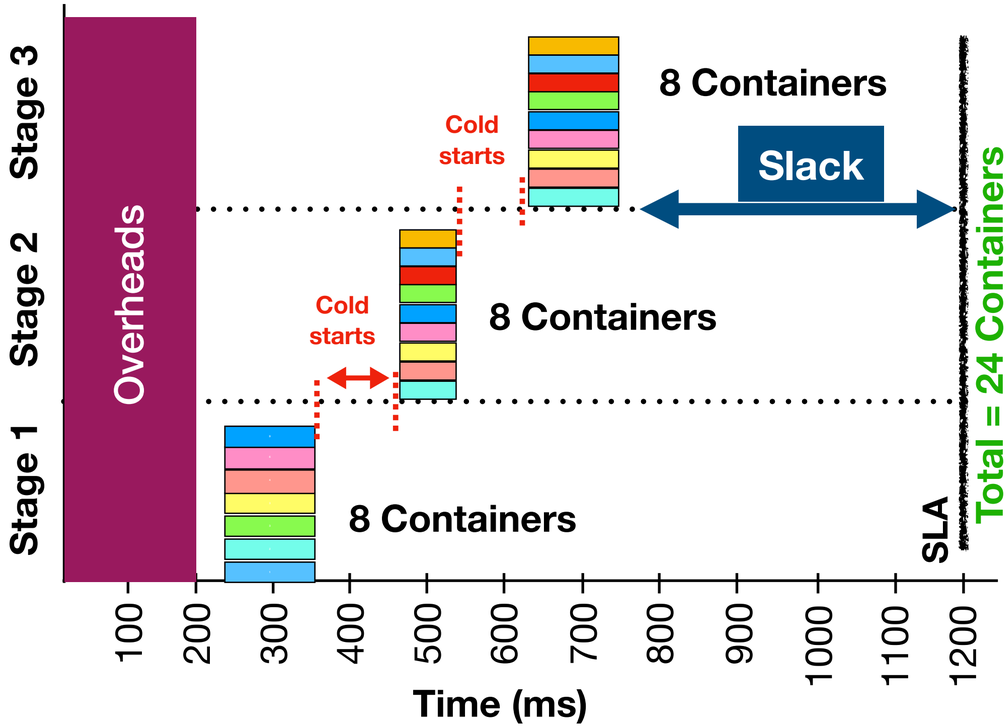
Slack aware queuing



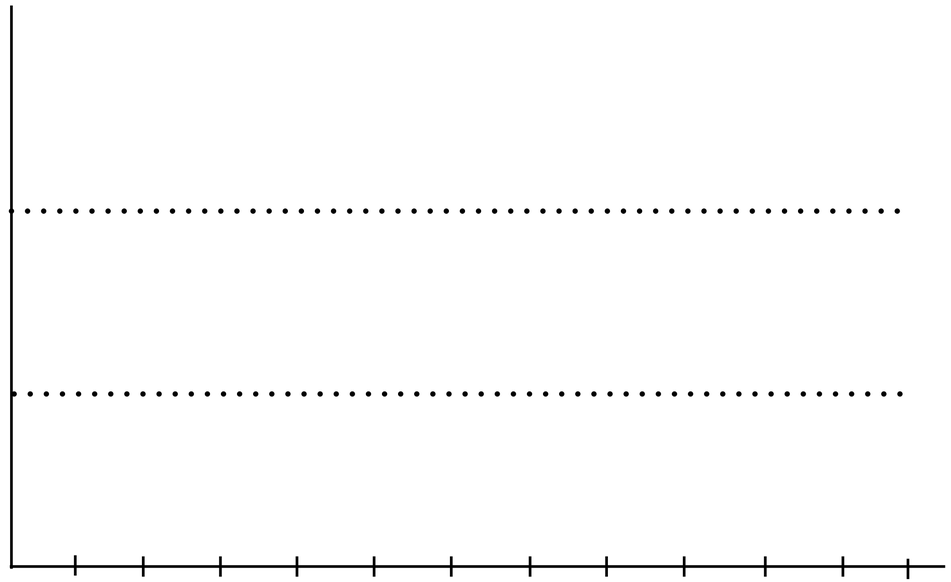
(a) Baseline RM



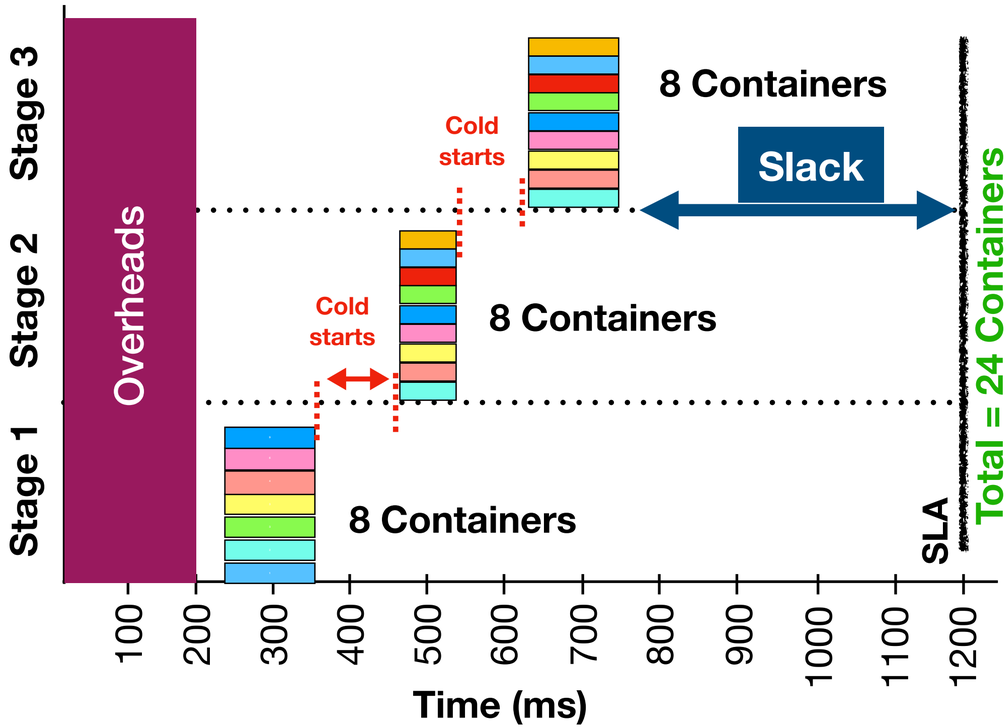
Slack aware queuing



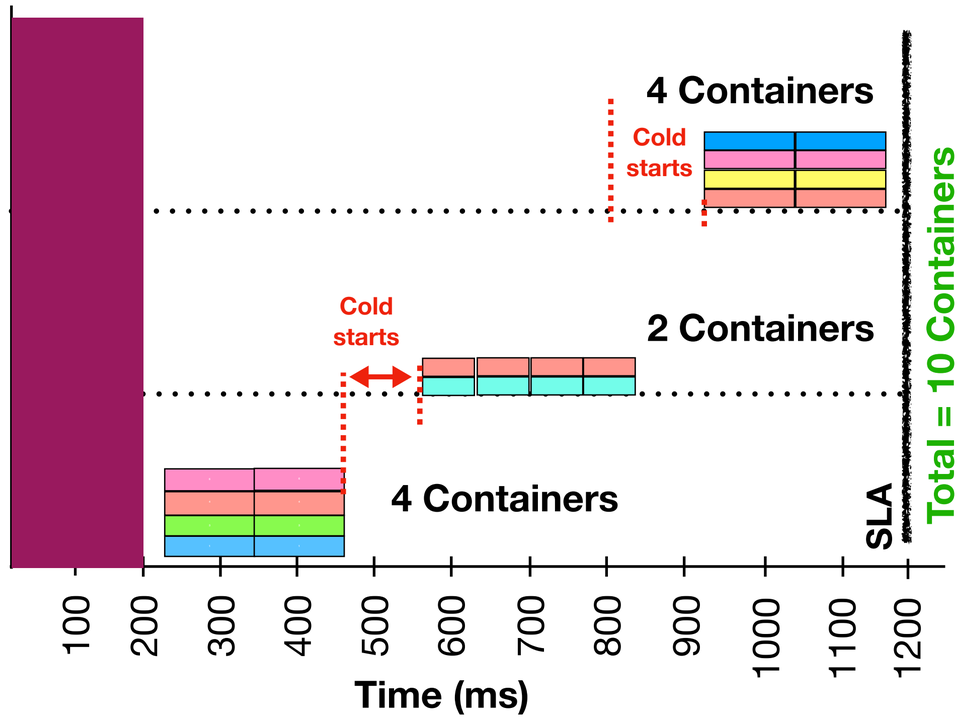
(a) Baseline RM



Slack aware queuing

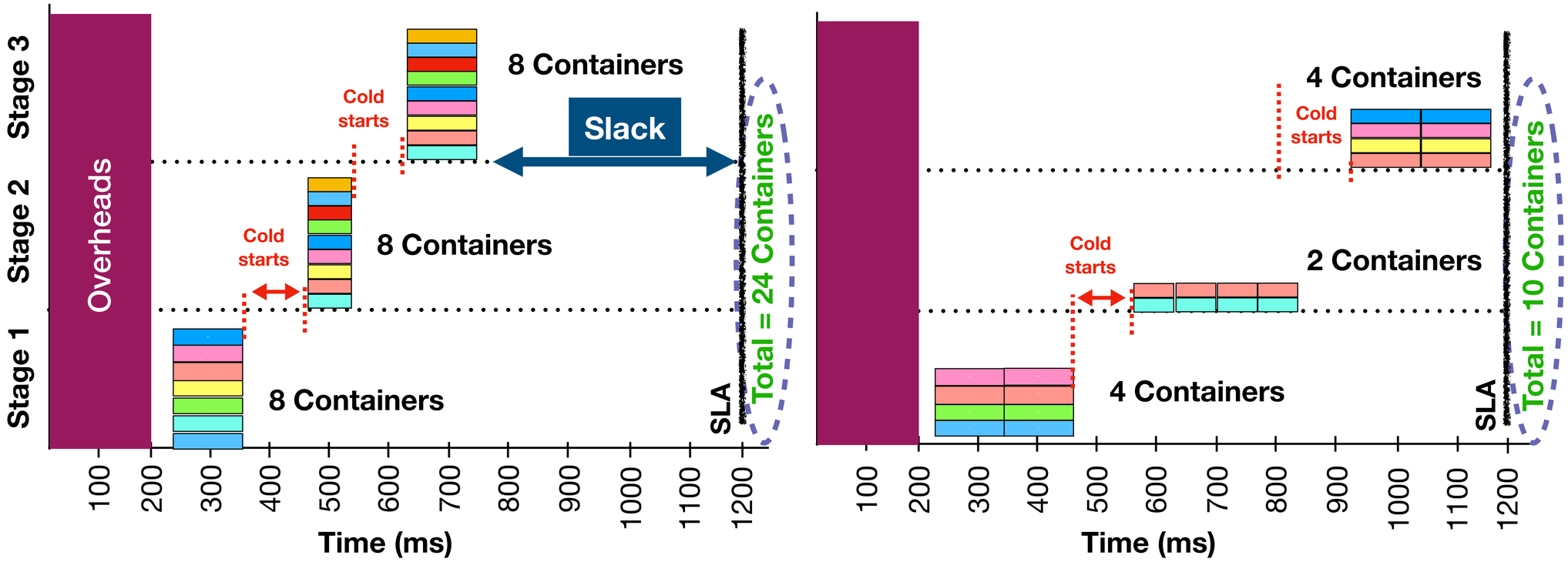


(a) Baseline RM



(b) Request-Batching RM

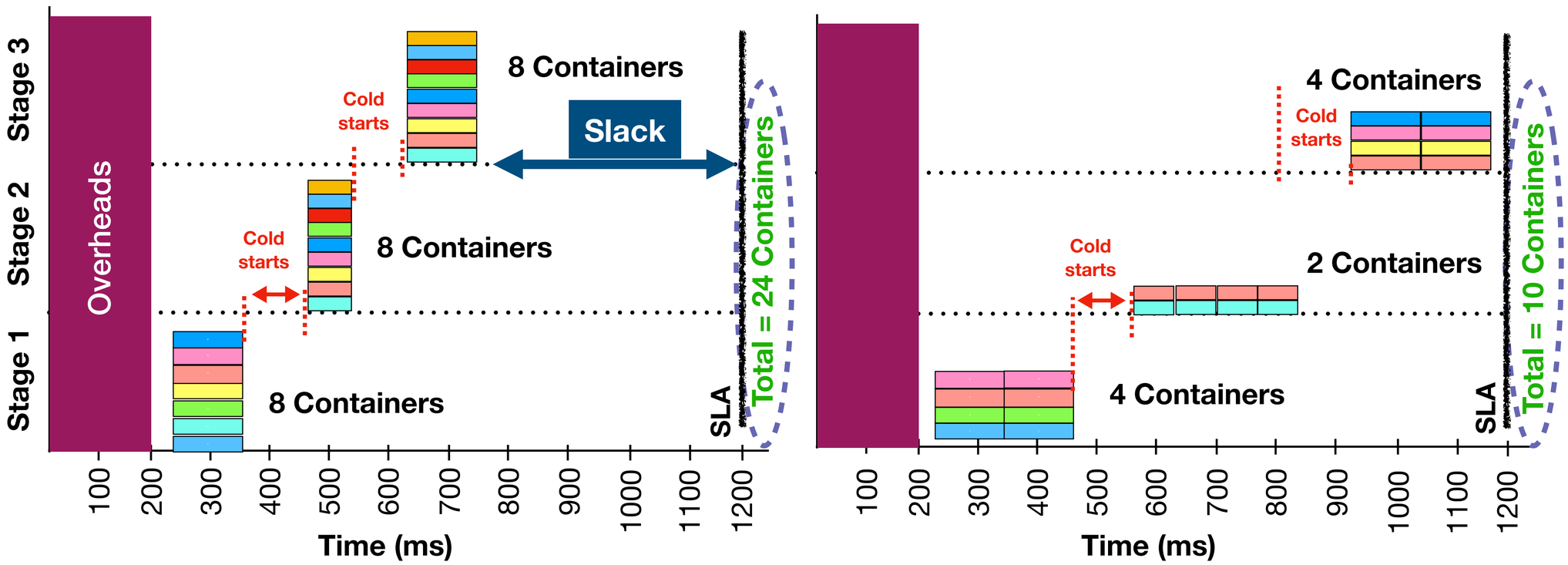
Slack aware queuing



(a) Baseline RM

(b) Request-Batching RM

Slack aware queuing

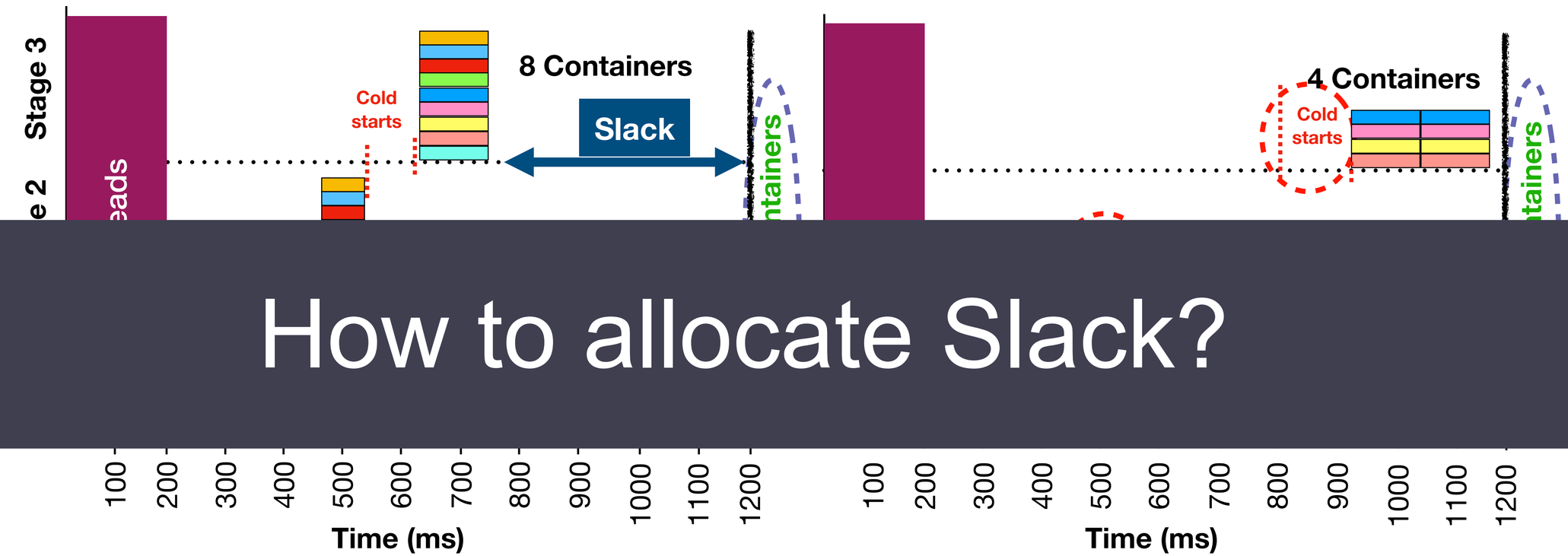


(a) Baseline RM

(b) Request-Batching RM

Exploiting Slack to Queue requests can save up to **14 containers**.

Slack aware queuing



(a) Baseline RM

(b) Request-Batching RM

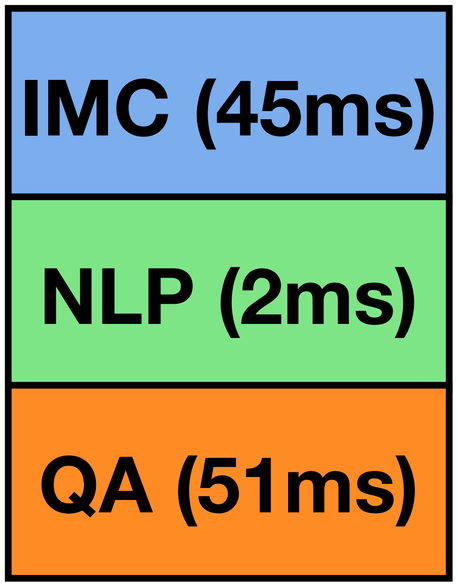
How to allocate Slack?

Exploiting Slack to Queue requests can save up to **14 containers**.

Slack Allocation

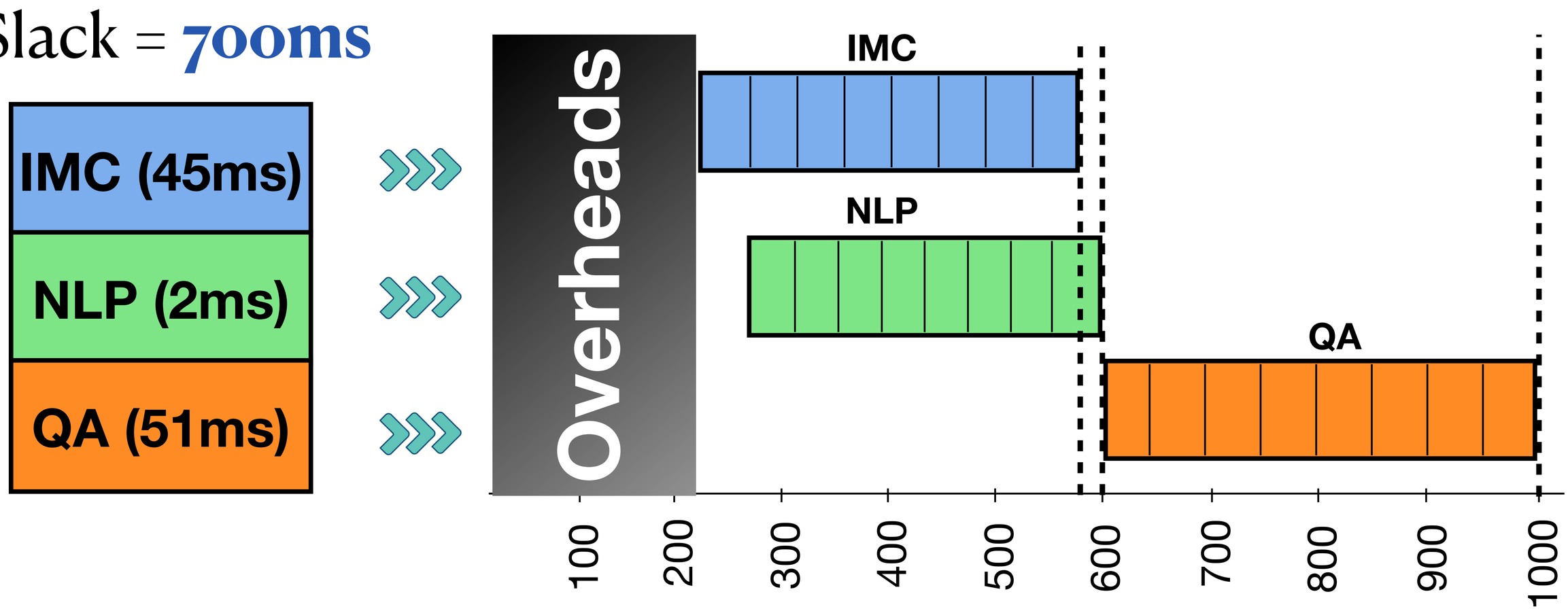
Slack Allocation

Slack = **700ms**



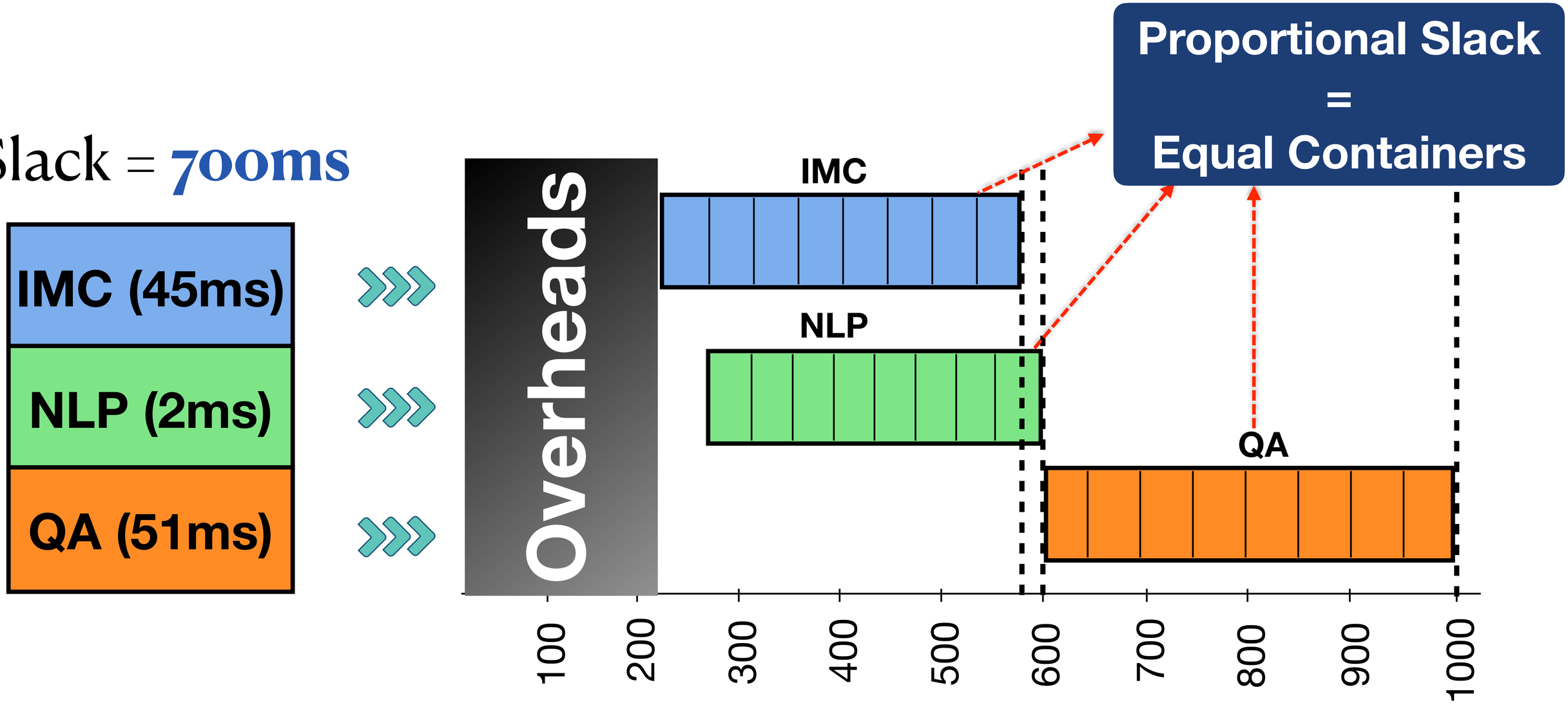
Slack Allocation

Slack = 700ms

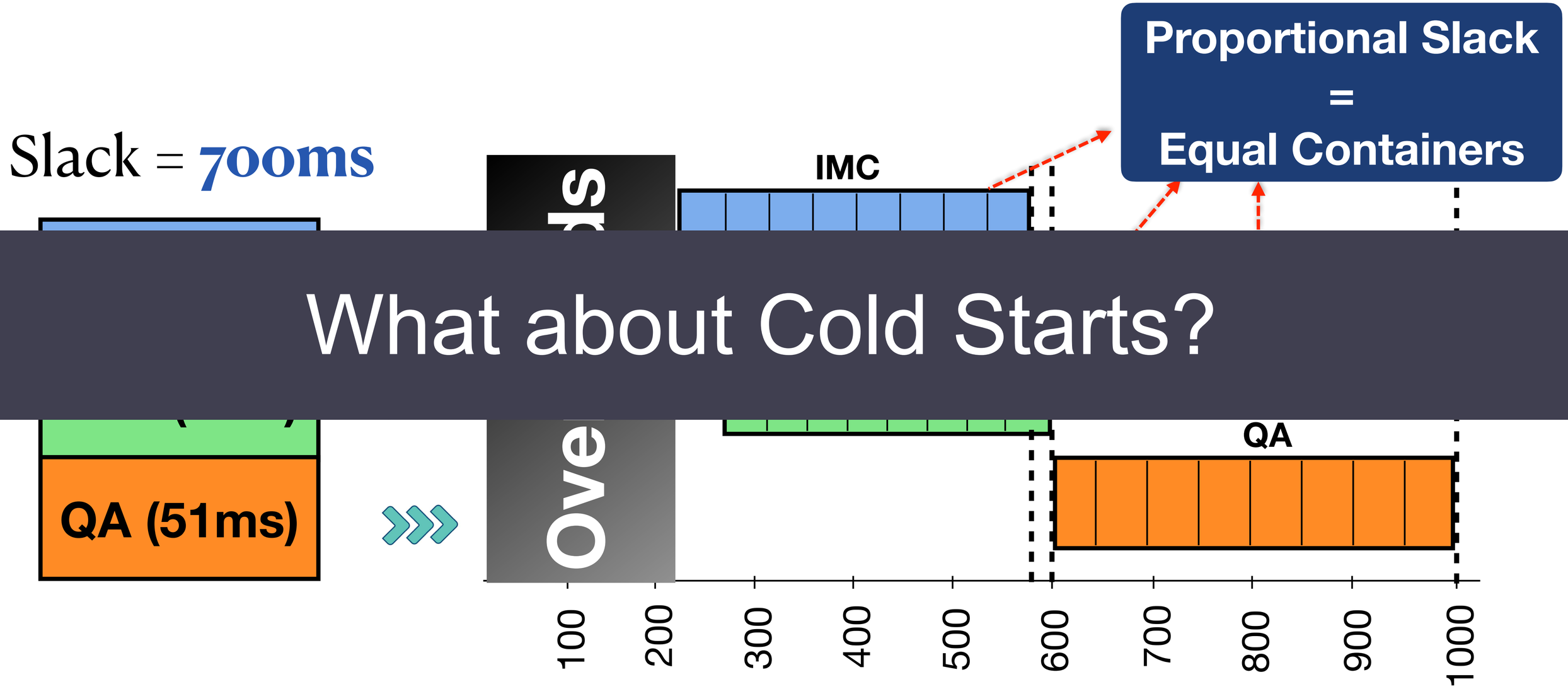


Slack Allocation

Slack = 700ms

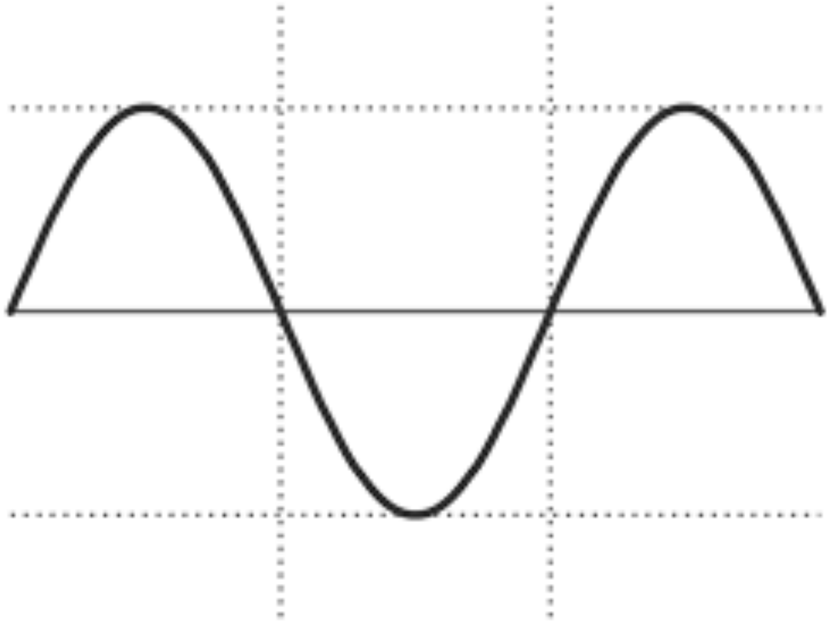


Slack Allocation

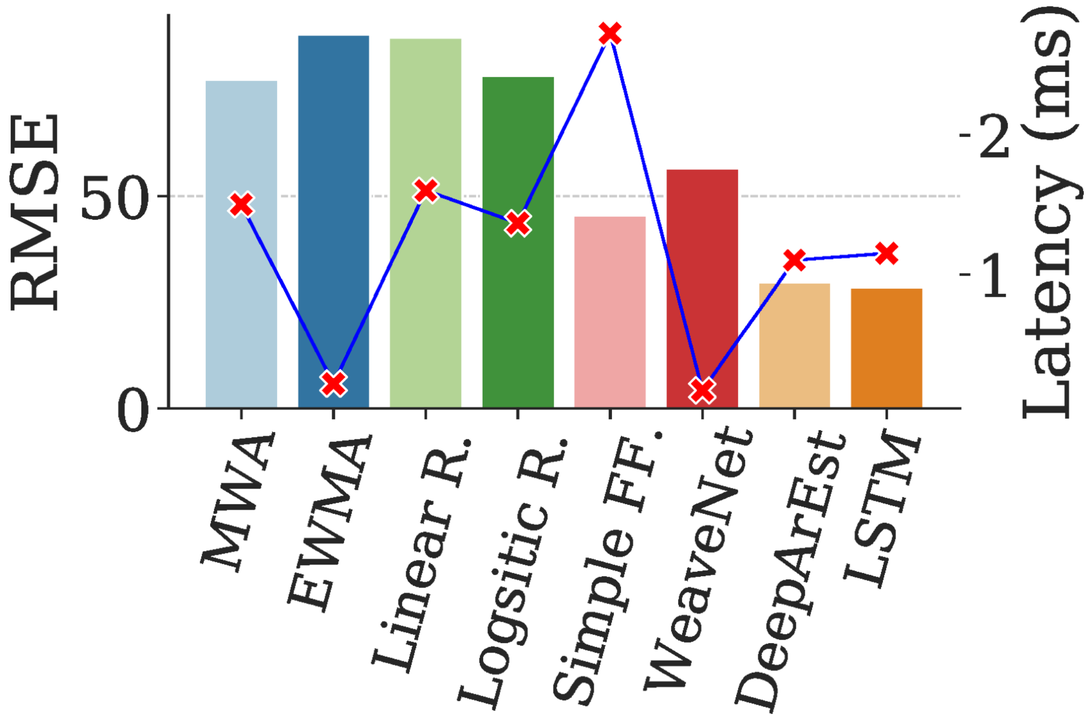


What about Cold Starts?

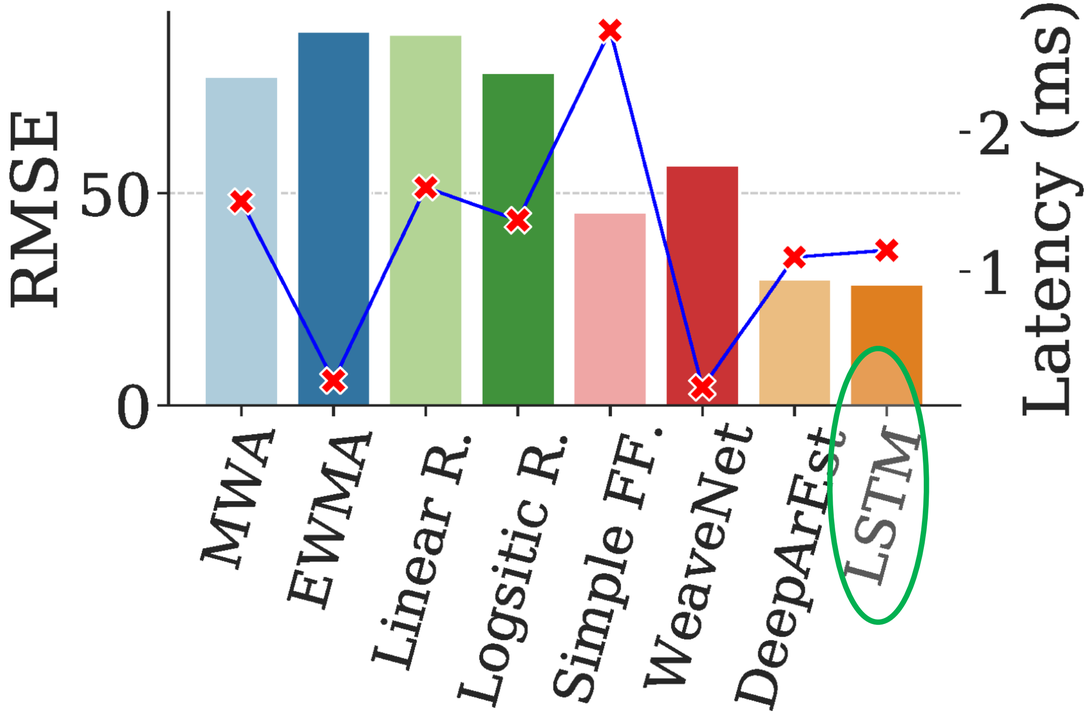
Reactive Scaling + Load Prediction

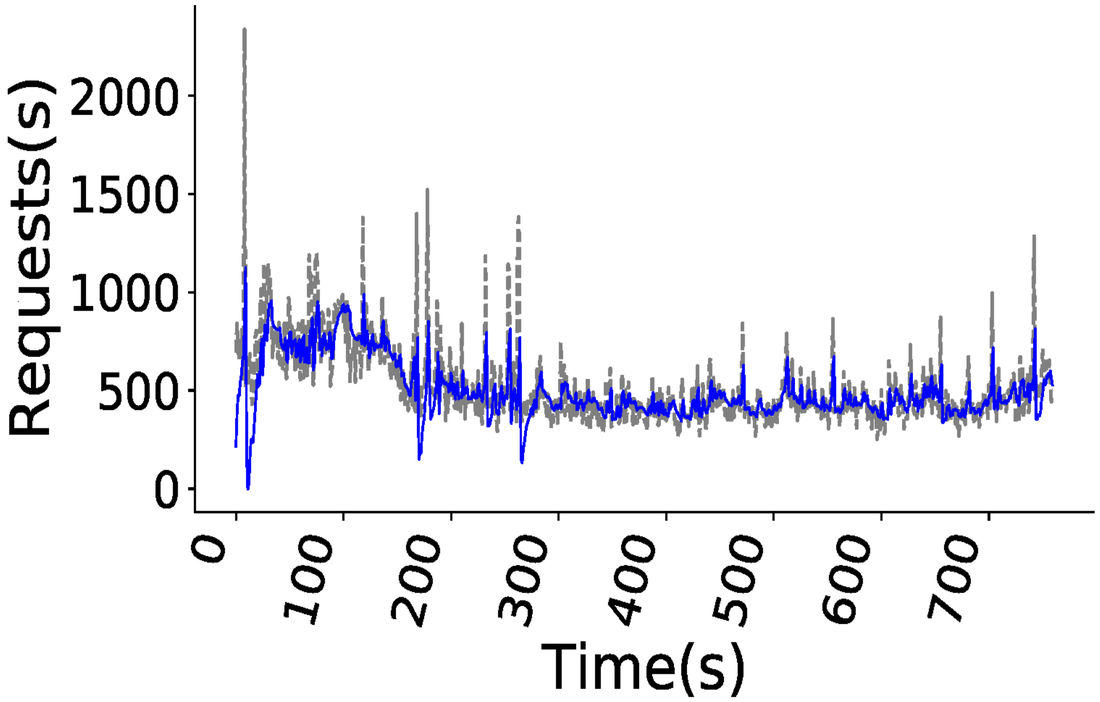


Prediction Model

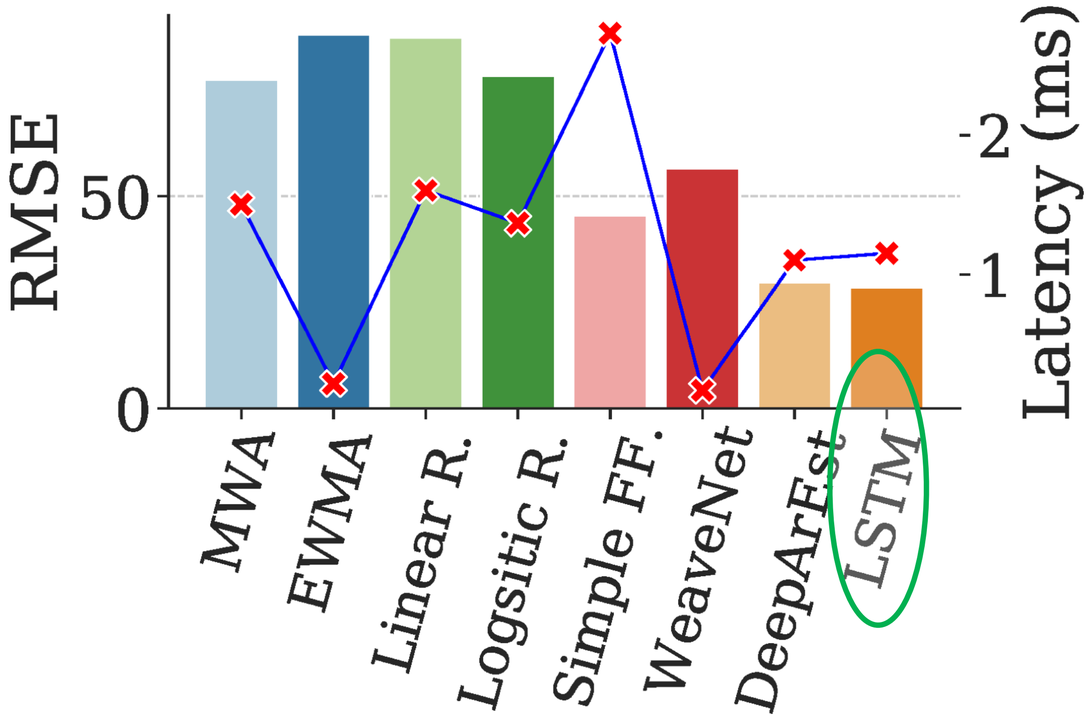


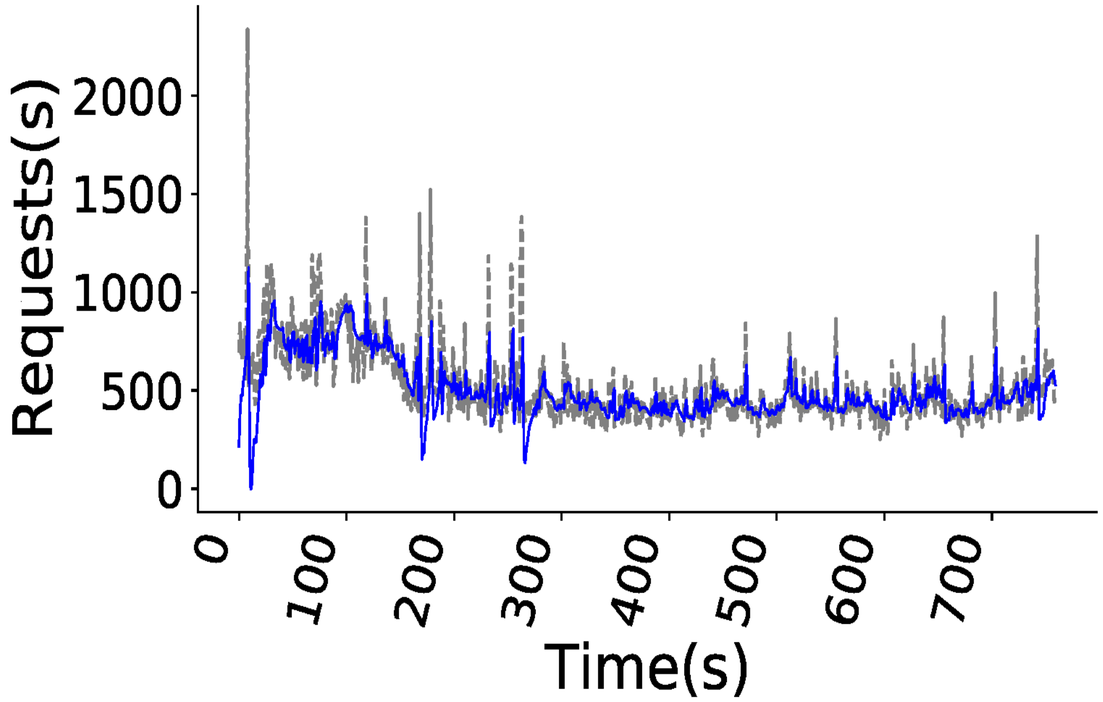
Prediction Model





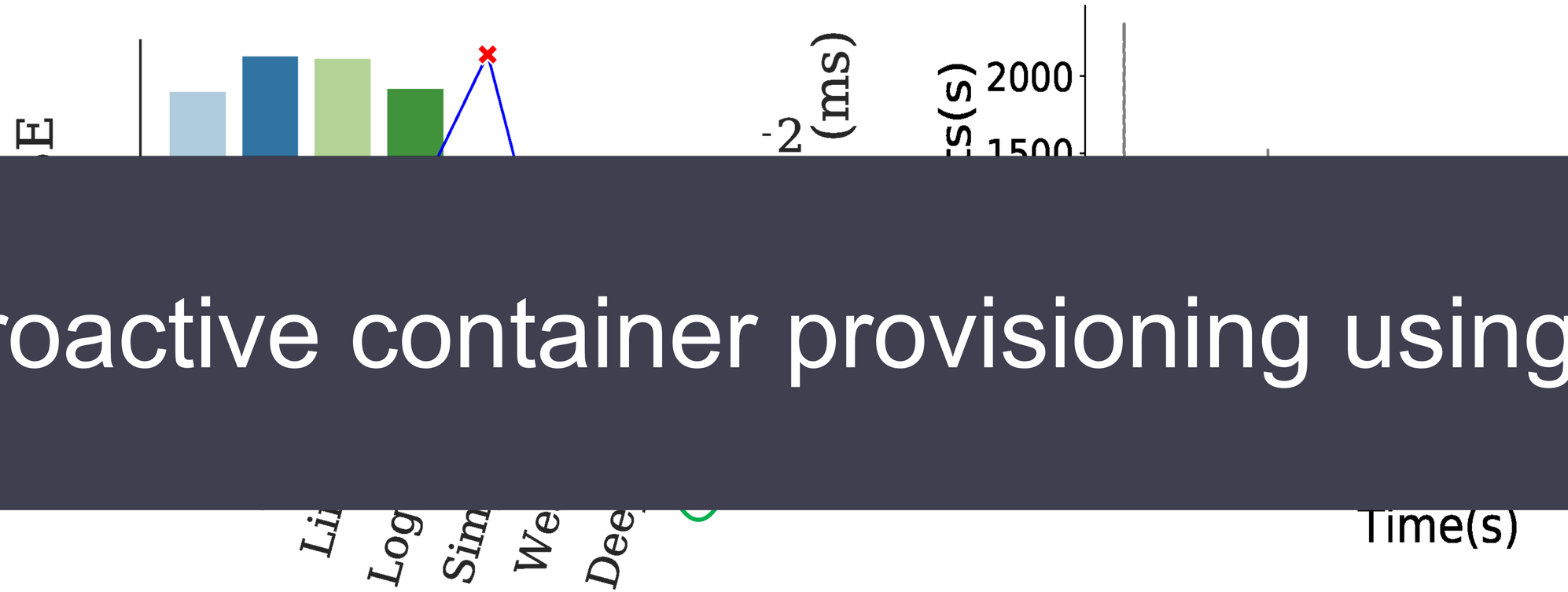
Prediction Model





LSTM is the best with least RMSE

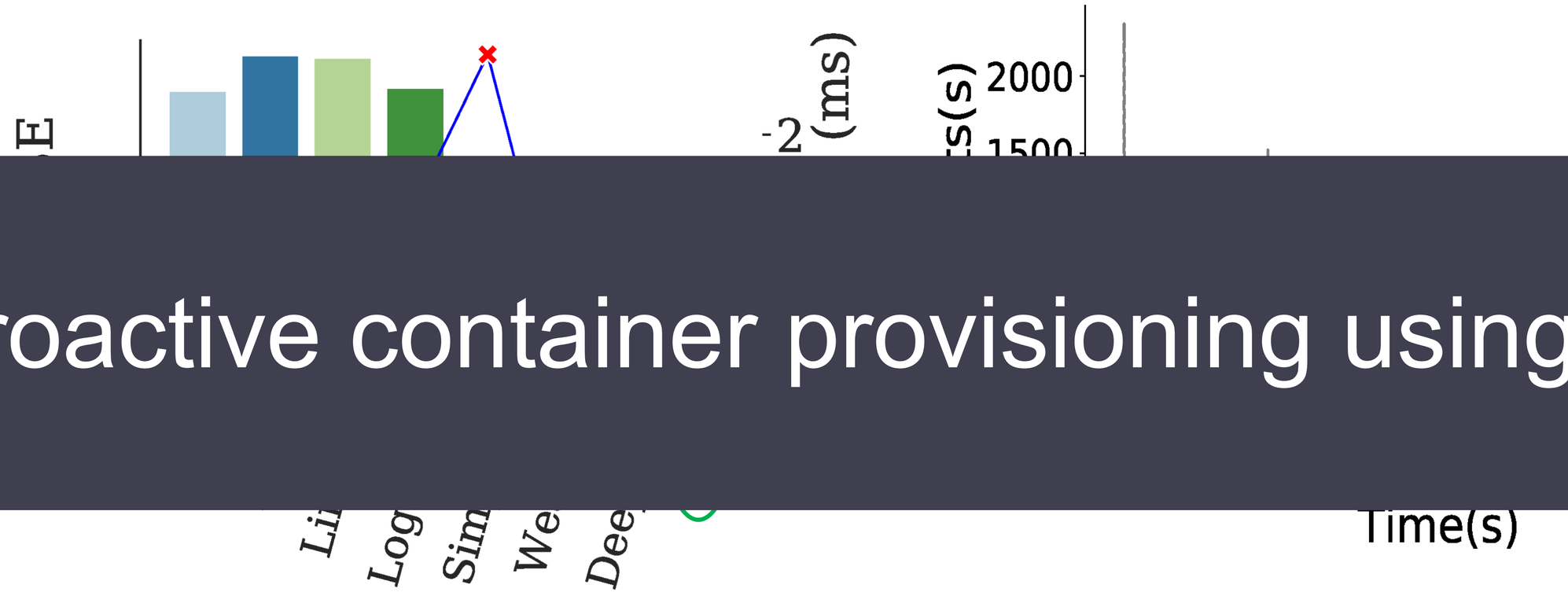
Prediction Model



Proactive container provisioning using LSTM

LSTM is the best with least RMSE

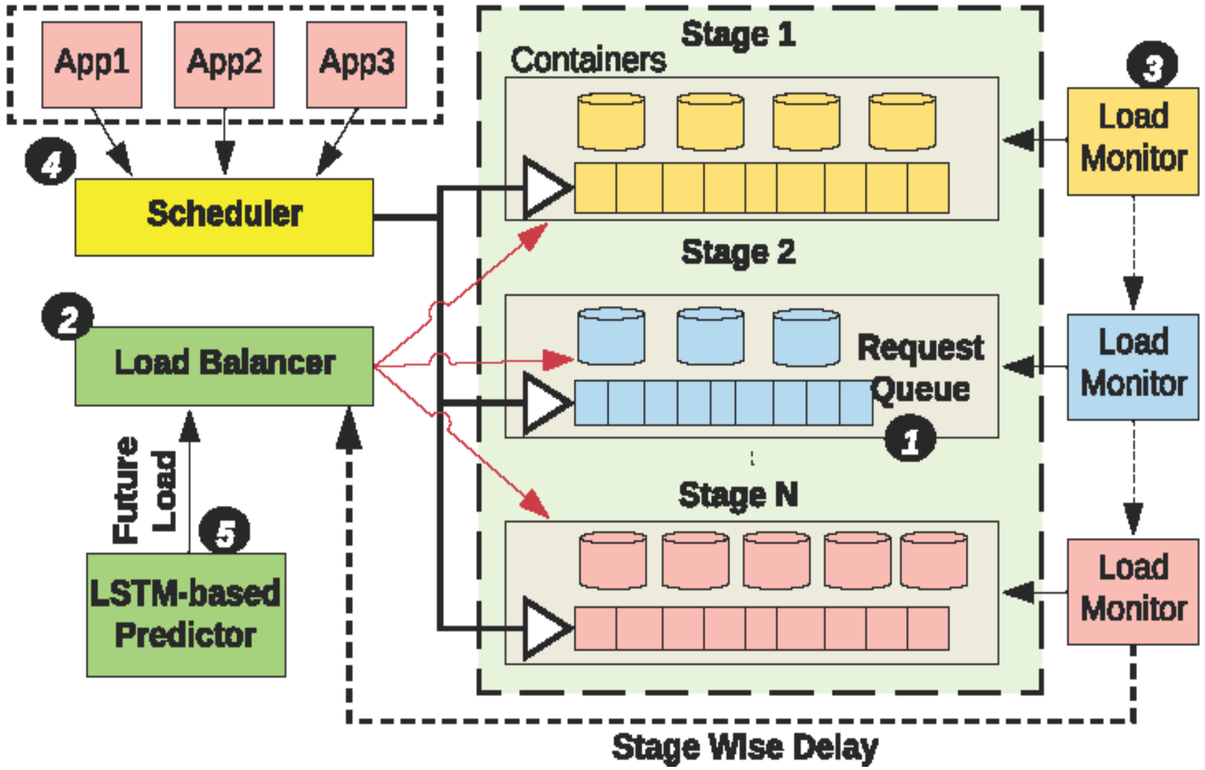
Prediction Model



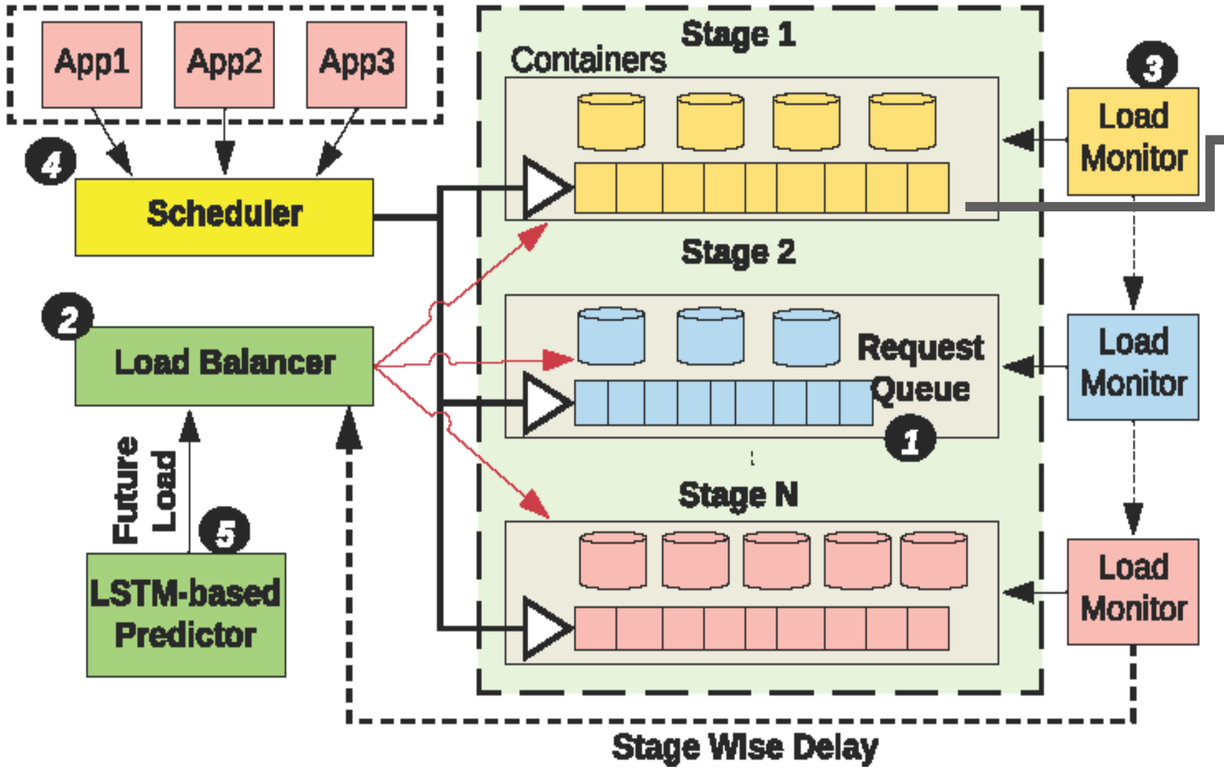
Proactive container provisioning using LSTM

LSTM is the best with least RMSE

Fifer: Stage-aware Proactive container provisioning and management of function chains

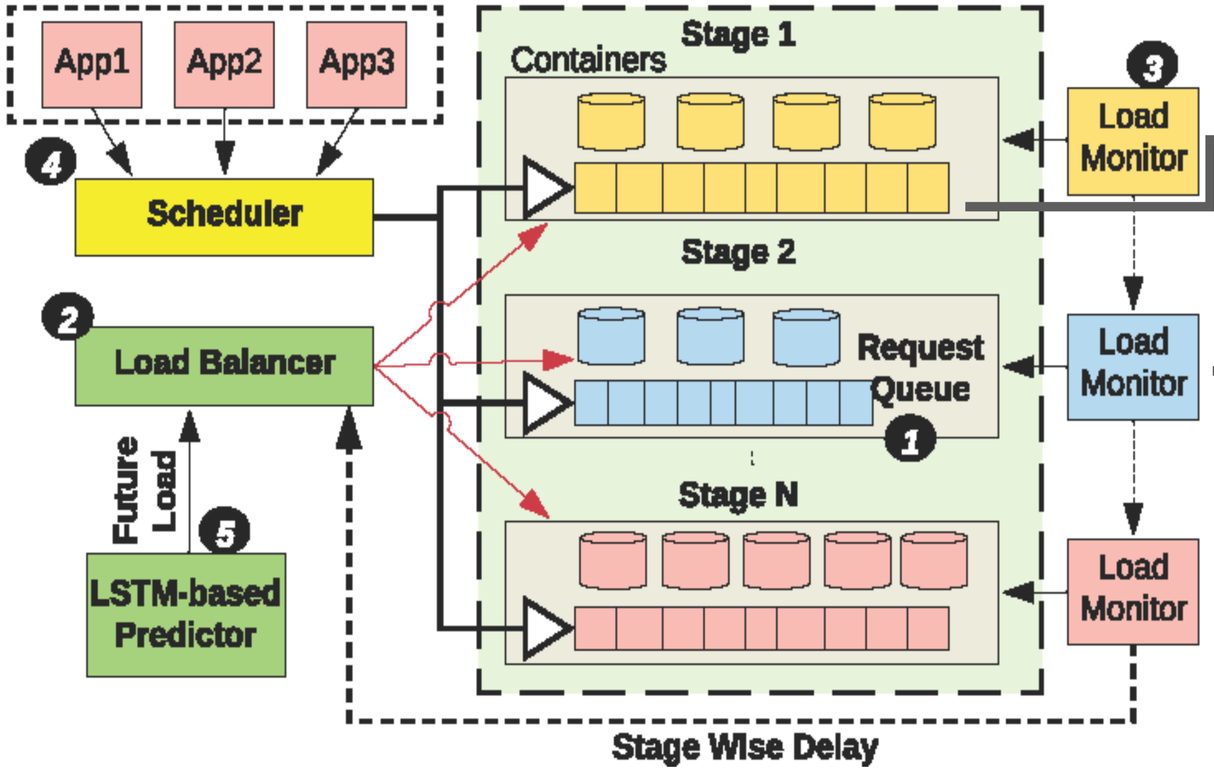


Fifer: Stage-aware Proactive container provisioning and management of function chains



Stage-aware proportionate queuing

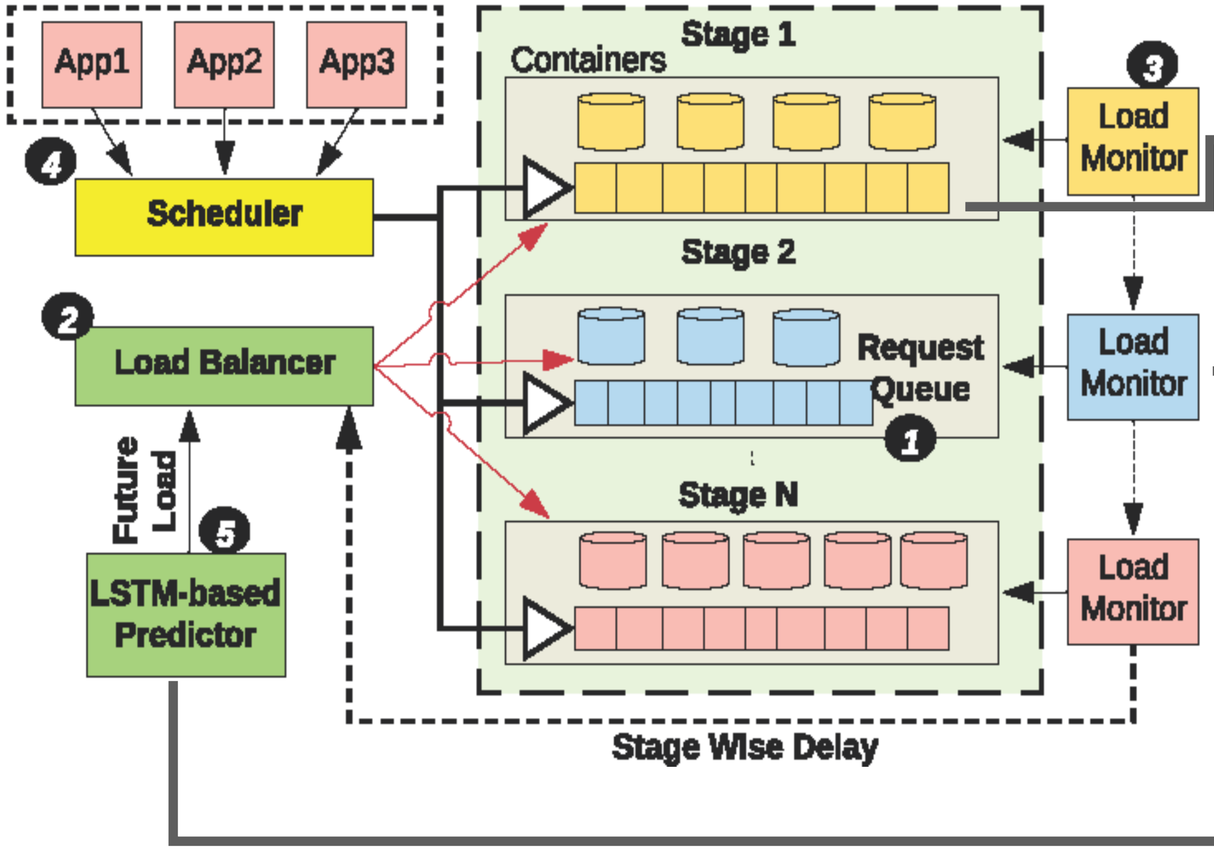
Fifer: Stage-aware Proactive container provisioning and management of function chains



Stage-aware proportionate queuing

Queuing delay-based Reactive container scaling

Fifer: Stage-aware Proactive container provisioning and management of function chains

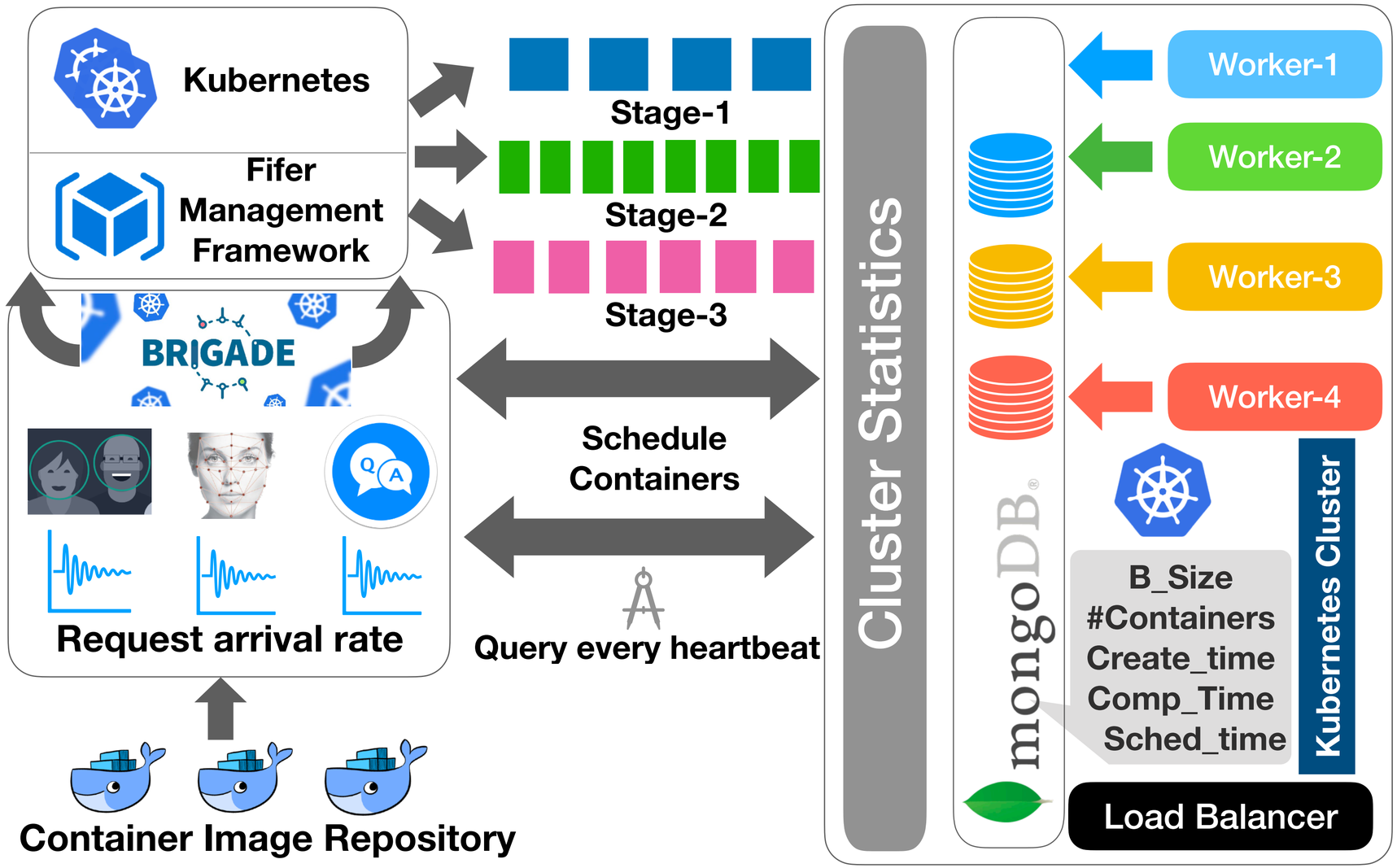


Stage-aware proportionate queuing

Queuing delay-based Reactive container scaling

Proactive container scaling

IMPLEMENTATION

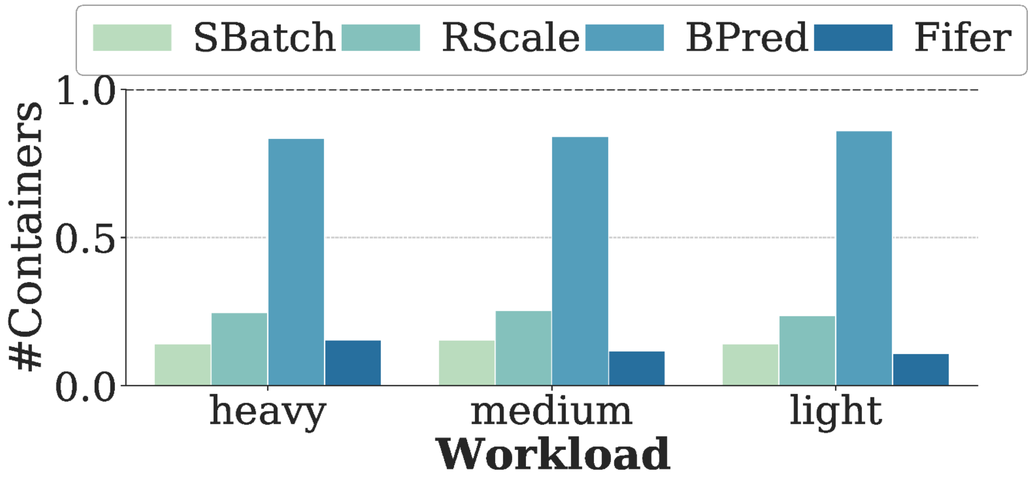


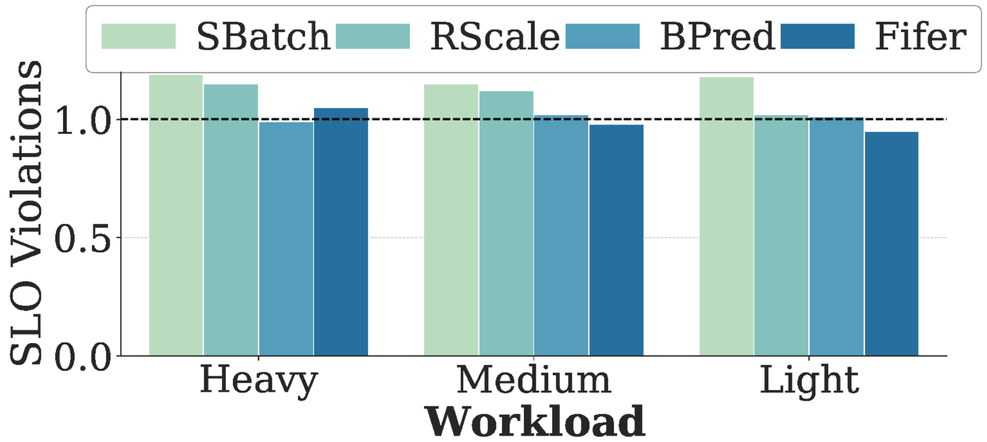
Hardware	Config
CPU	Xeon gold
Sockets	2
Core	16
Threads	2
Clock	2.8Ghz
DRAM	192GB

SLO violations and Containers

SLO violations and Containers

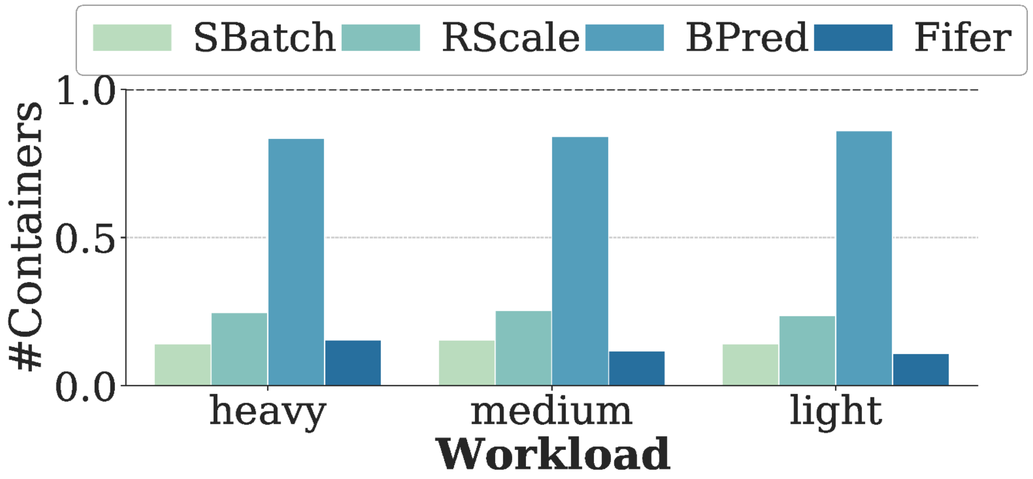
Better

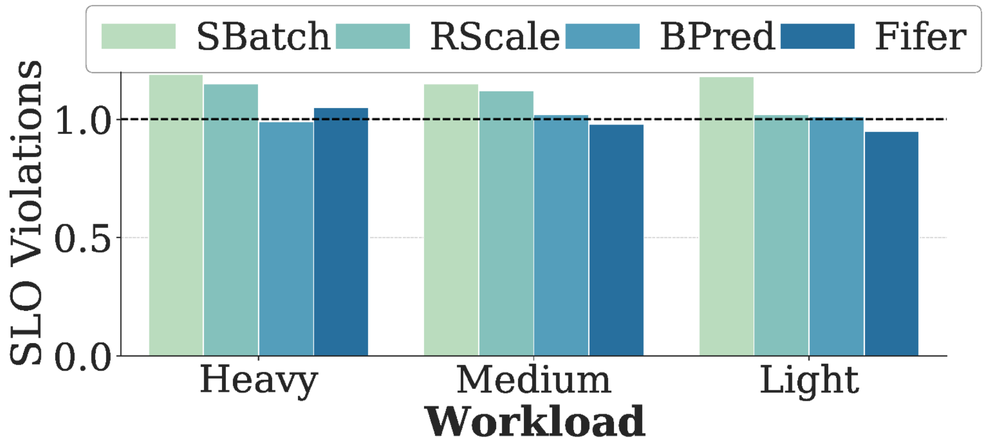




SLO violations and Containers

Better

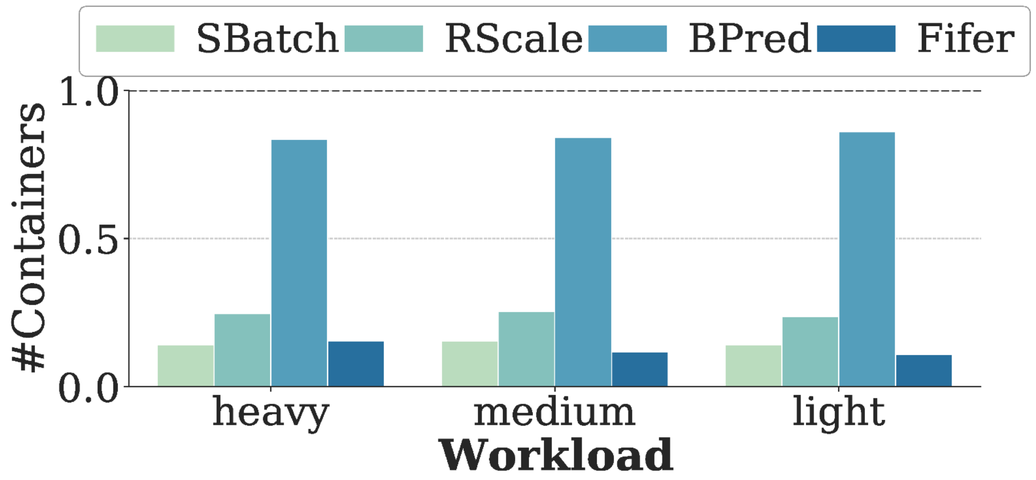




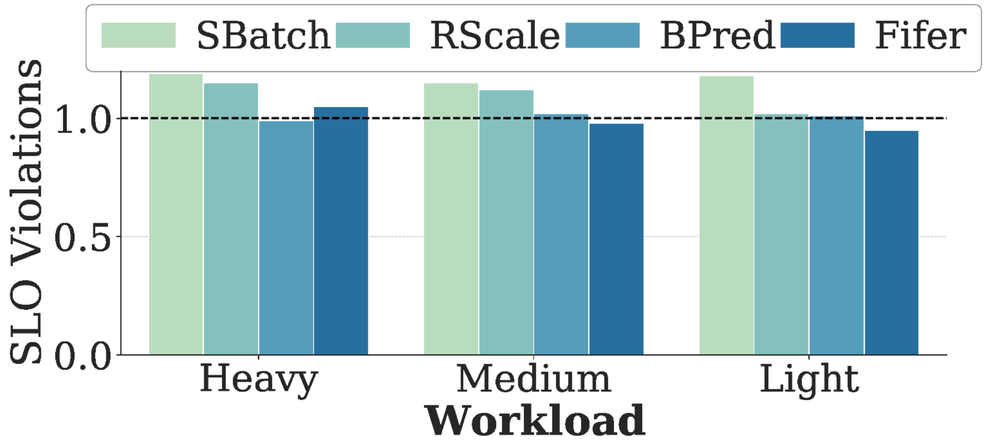
- #Containers normalized to baseline
- Fifer is spawns **20%** less containers

SLO violations and Containers

Better



- #Containers normalized to baseline
- Fifer is spawns **20%** less containers

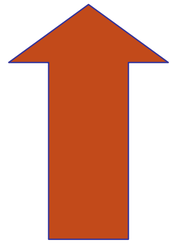


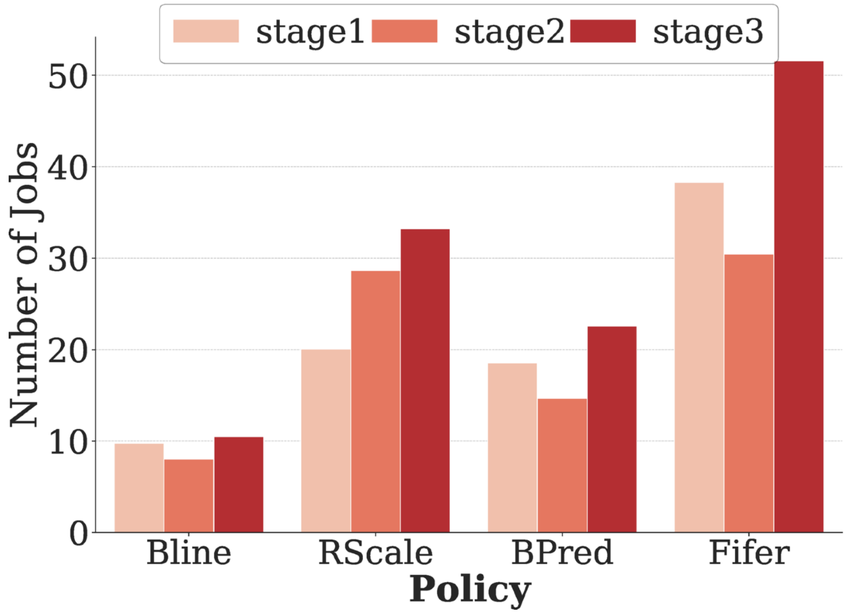
- SLO violations normalized to baseline
- Fifer is similar to baseline with fewer containers

Utilization and Energy

Utilization and Energy

Better

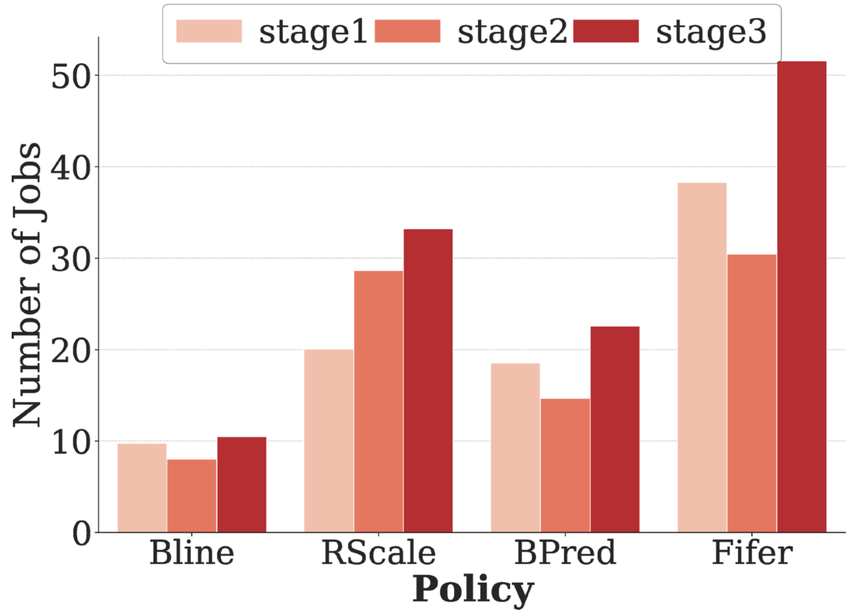




- Average #Requests executed per container (RPC).
- Fifer improves container utilization by **34%**

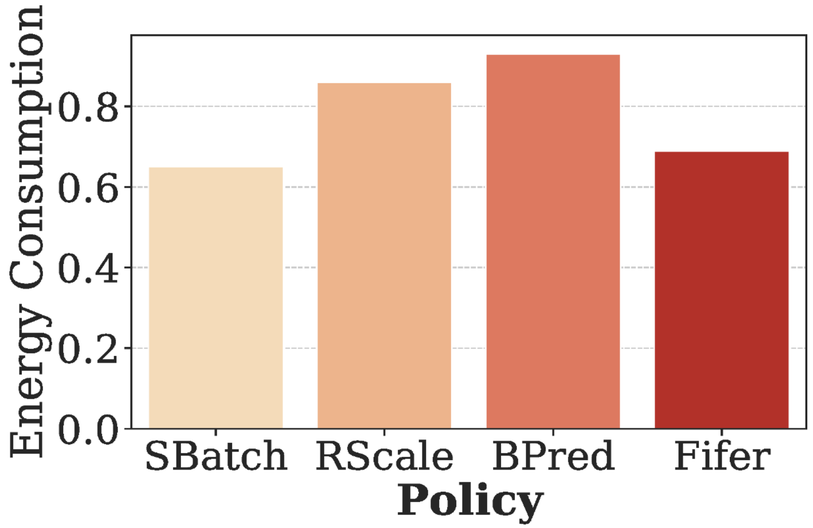
Utilization and Energy

Better



- Average #Requests executed per container (RPC).
- Fifer improves container utilization by **34%**

Better



- Energy consumption normalized to Bline.
- Fifer is **31%** more energy efficient

- Details of the workload used.
- Evaluated schemes and policies.
- Details about LSTM training.