上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

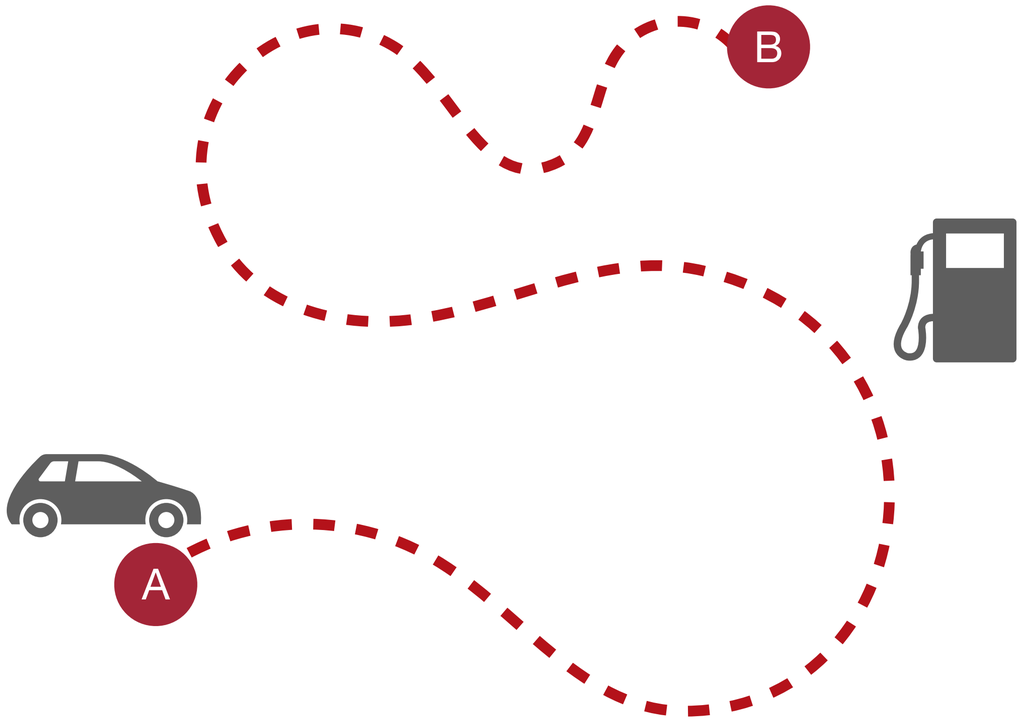
Pre-Warming Is Not Enough: Accelerating Serverless Inference With Opportunistic Pre-Loading

Yifan Sui[1], Hanfei Yu[2], Yitao Hu[3], Jianxun Li[1], Hao Wang[2]

Shanghai Jiao Tong University[1], Stevens Institute of Technology[2], Tianjin University[3]

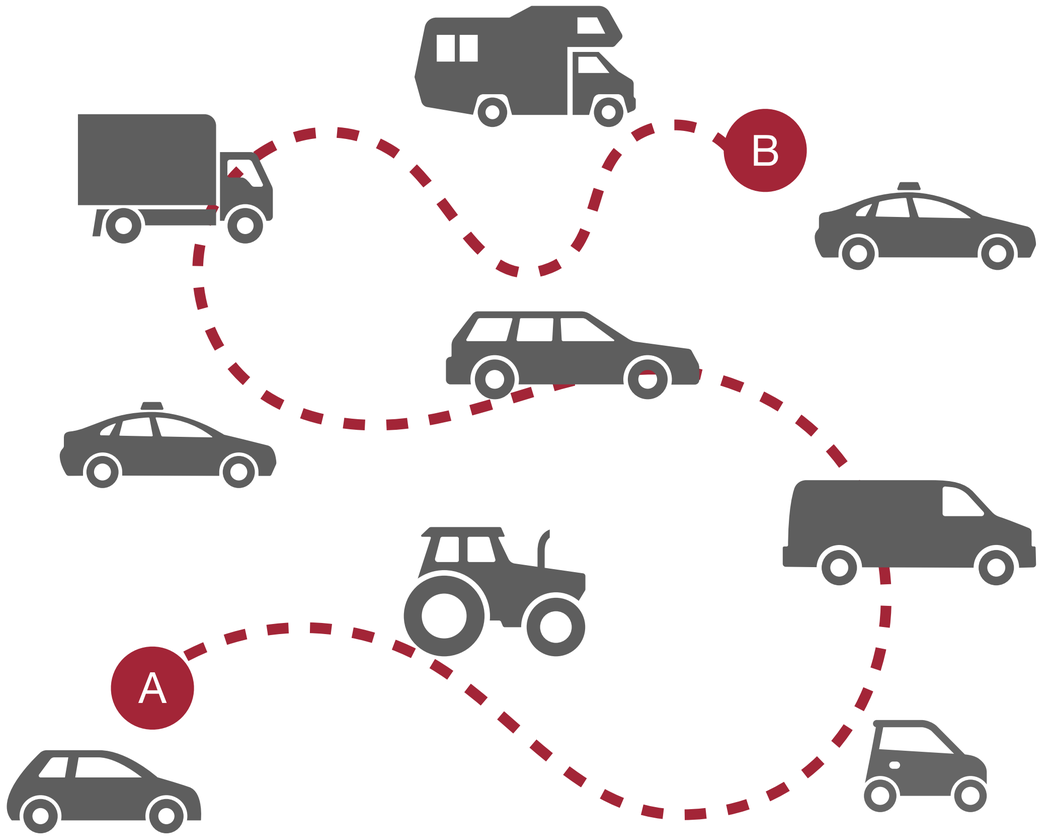
IntelliSys Lab

Cloud



Car rental

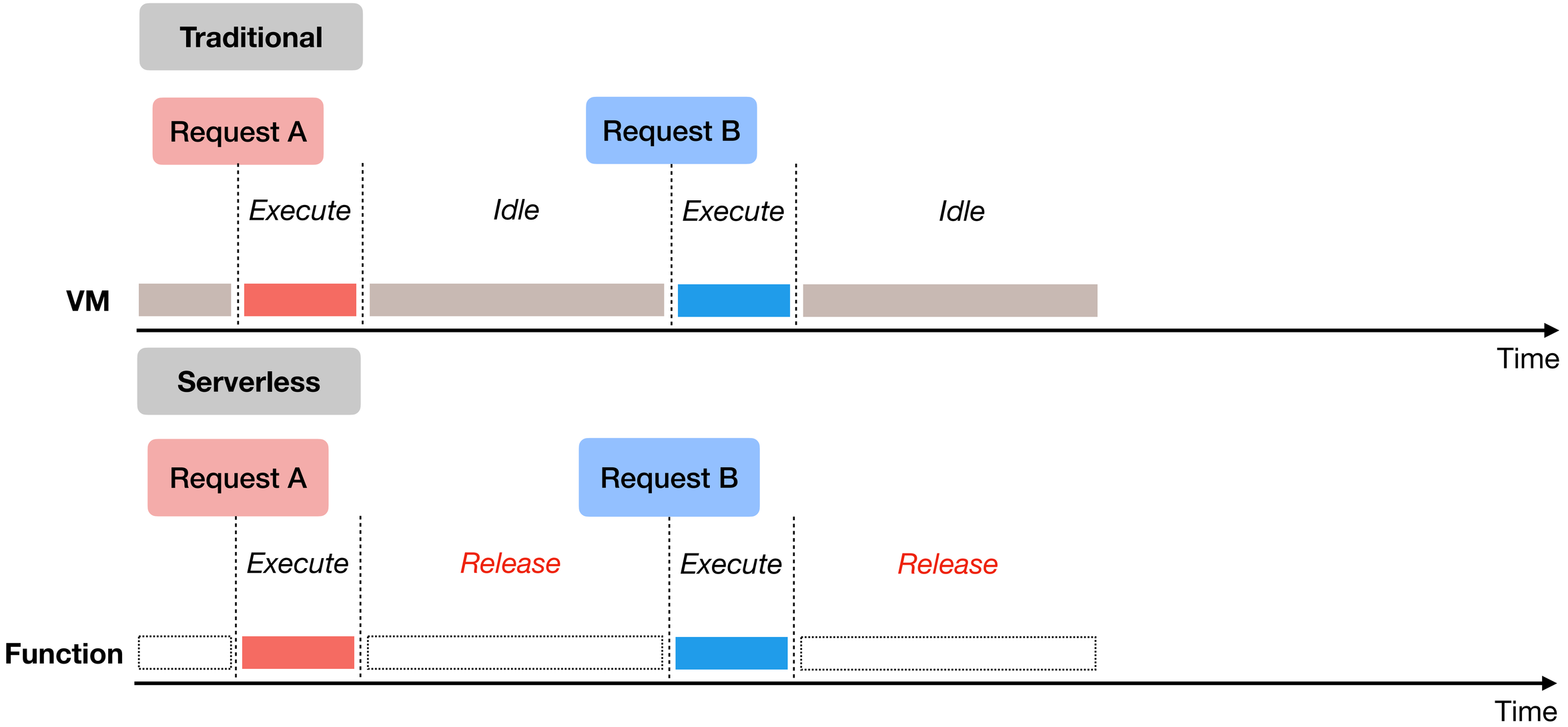
Serverless



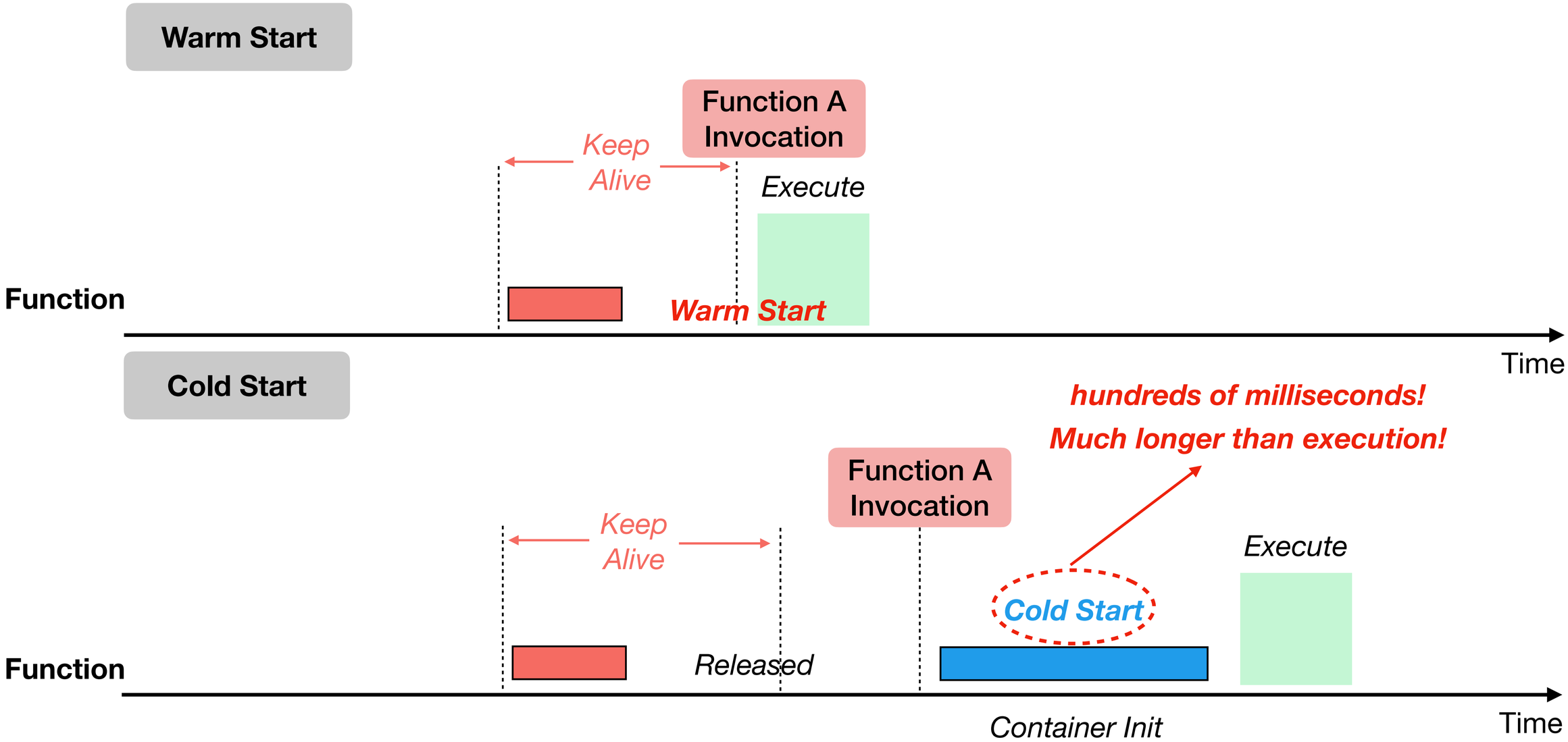
Uber

From *A Serverless Vision for Cloud Computing*
by Prof. Ana Klimovic

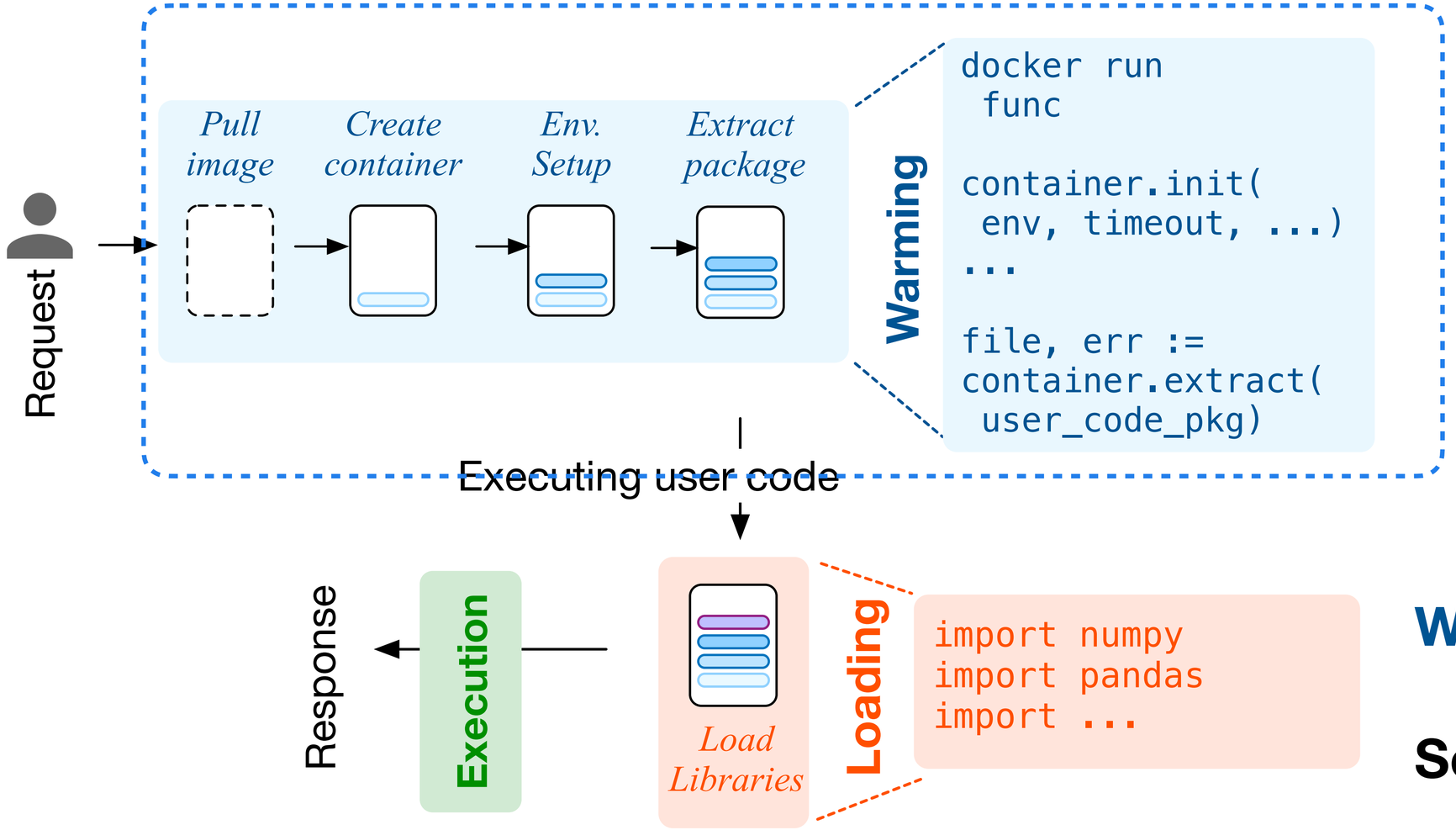
Traditional Cloud vs. Serverless



Cold-starts in Serverless



Serverless Cold Start

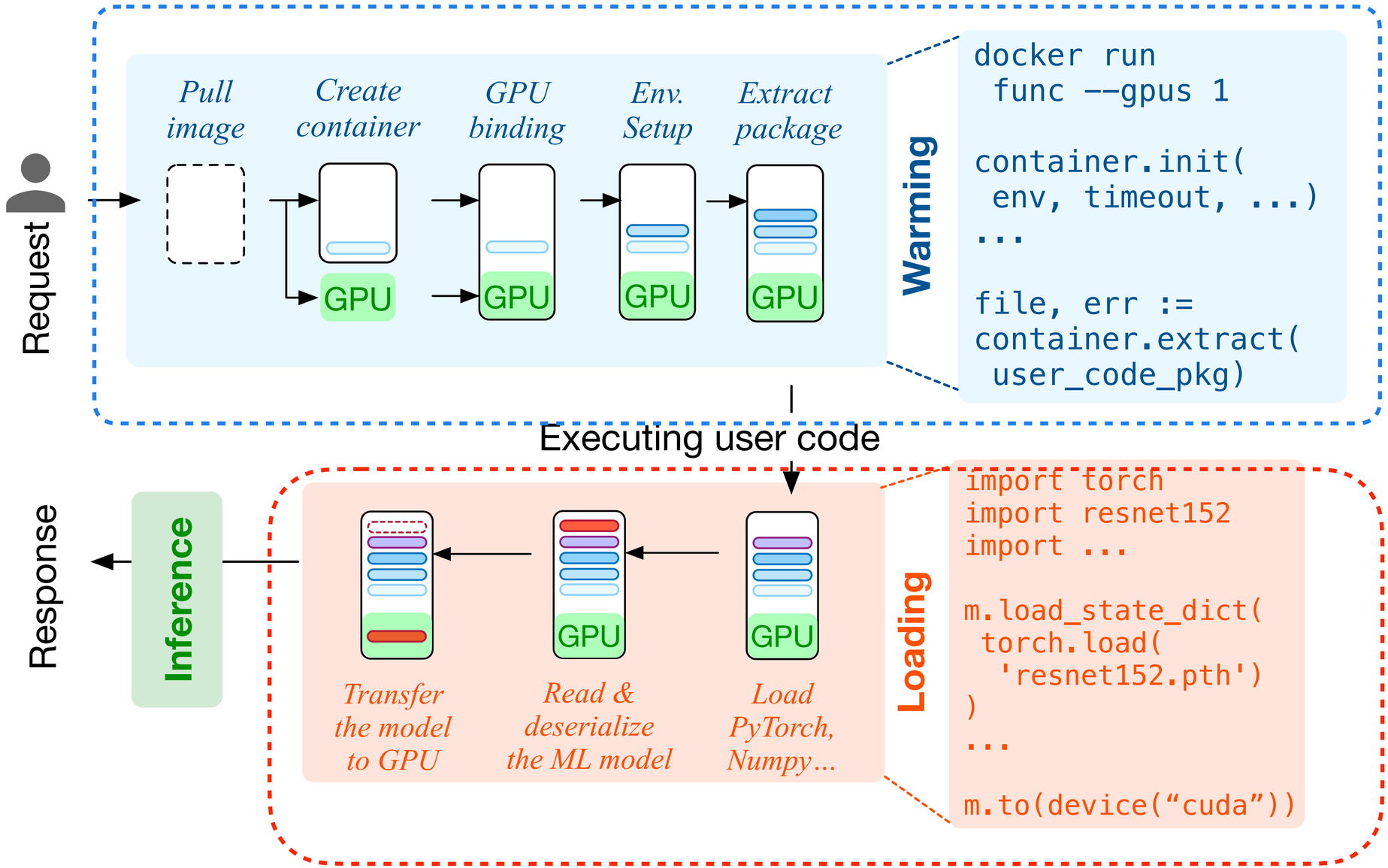


Pre-Warming:
Warm containers before request arrives

Warming >> Loading ?

Solution: Pre-Warming

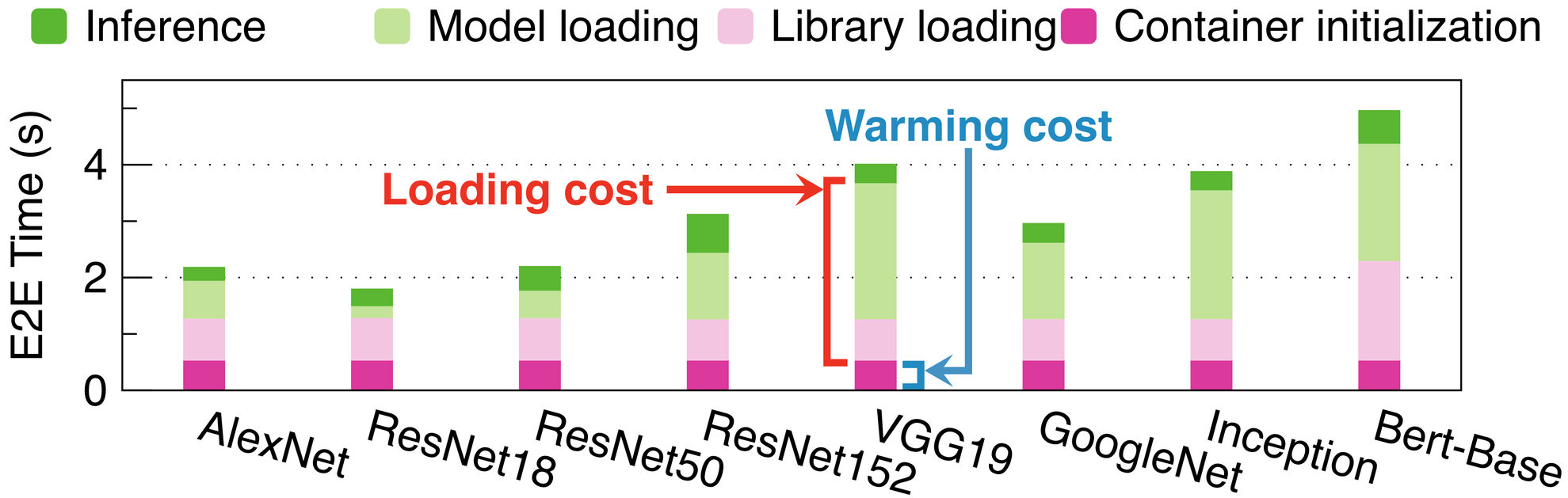
ML Inference's Cold Start



Pre-Warming's Scope:
Complete warming before request arrives

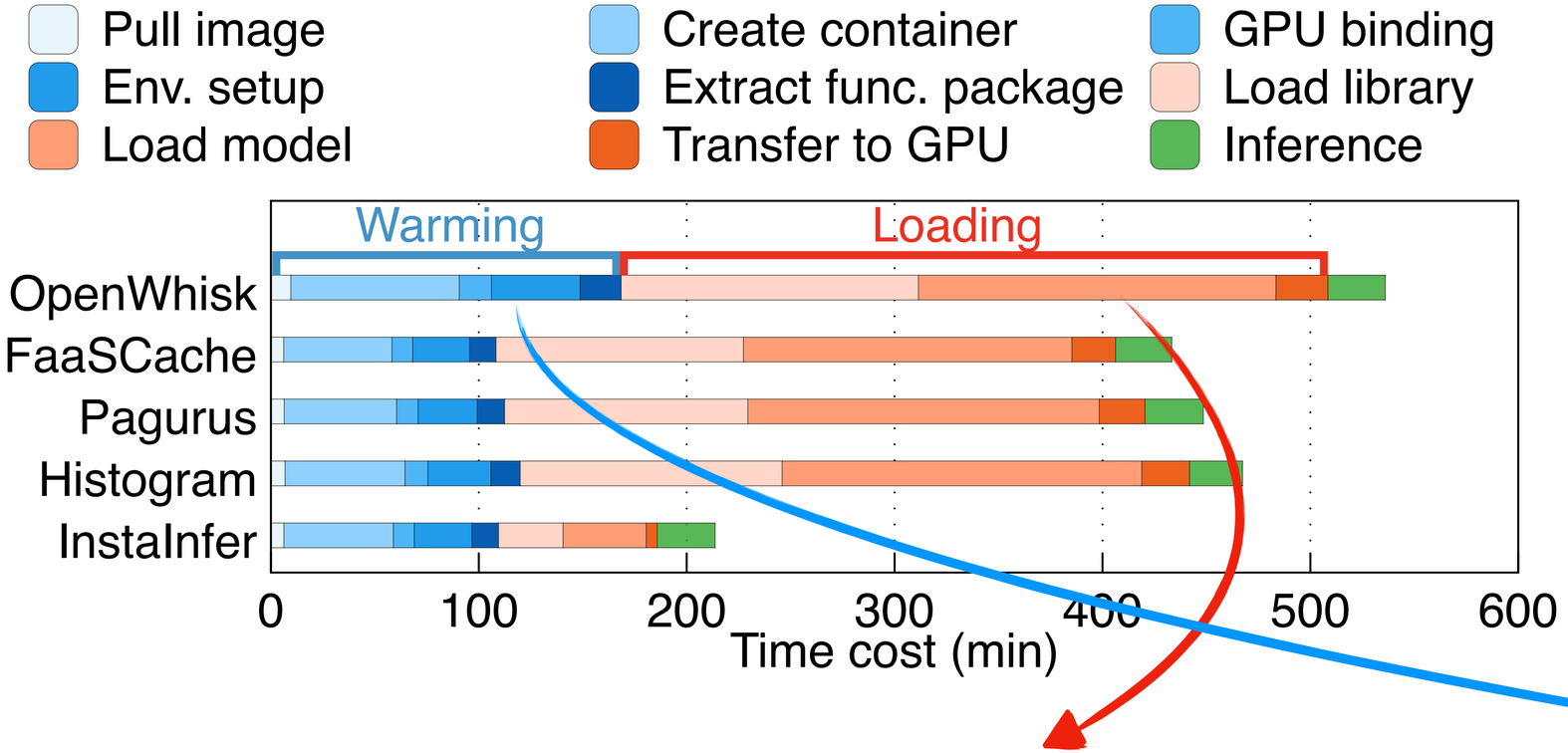
Loading is out of pre-warming's scope.

Existing Works:



In a cold start, loading takes far longer than warming !

Limitations of Existing Works



We run 4 pre-warming methods using the above inference functions in Azure Trace:

- FaaSCache [1]
- Pagurus [2]
- Histogram [3]
- OpenWhisk [4]

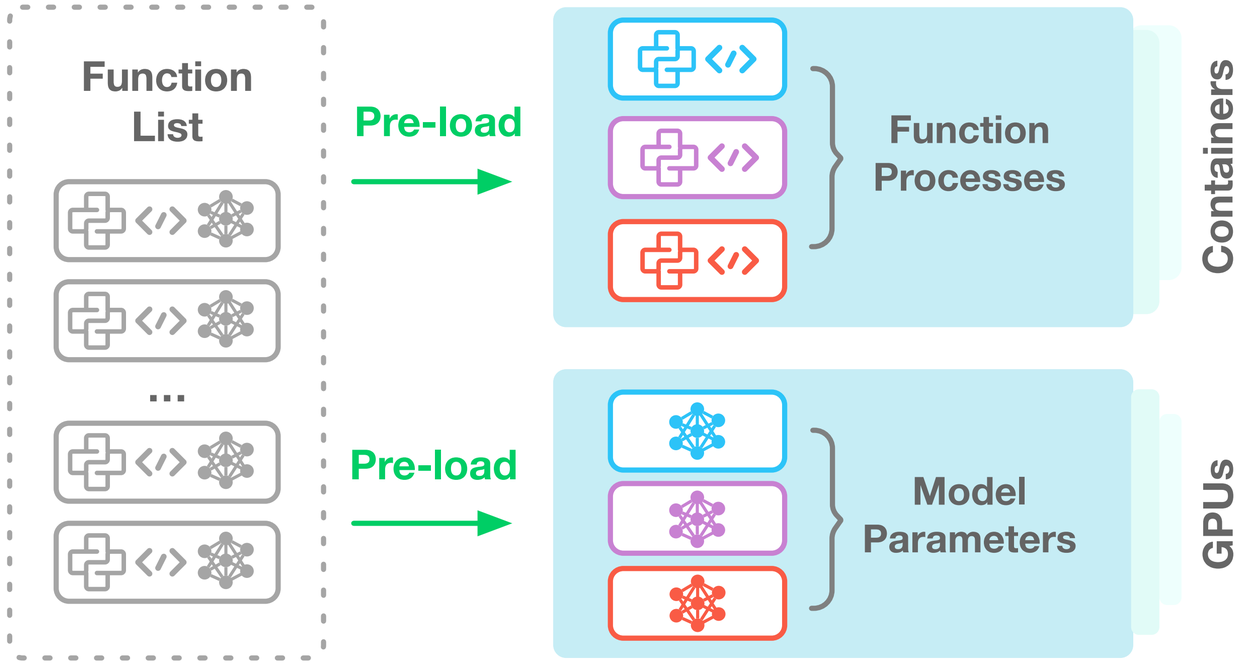
Loading takes far longer than warming !

[1] **FaaSCache:** Fuerst, Alexander, et al. "SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing." ASPLOS'21
[2] **Pagurus:** Li, Zijun, et al. "Help Rather Than Recycle: Alleviating Cold Startup in Serverless Computing…" ATC'22
[3] **Histogram:** Shahrad, Mohammad, et al. "Serverless in the Wild: Characterizing and Optimizing the Serverless…" ATC'20
[4] **OpenWhisk:** Apache OpenWhisk. https://openwhisk.apache.org.

Our Contribution:

Opprtunistic Pre-loading:

- Pre-load inference functions in idle containers and transfer the model to GPU instances.
- Keep pre-loading until all idle instances are full.



InstaInfer

Design Goals

Minimizing Loading Latency

Pre-load multiple functions within a container

Zero Wastage

Only utilize existing container/GPU

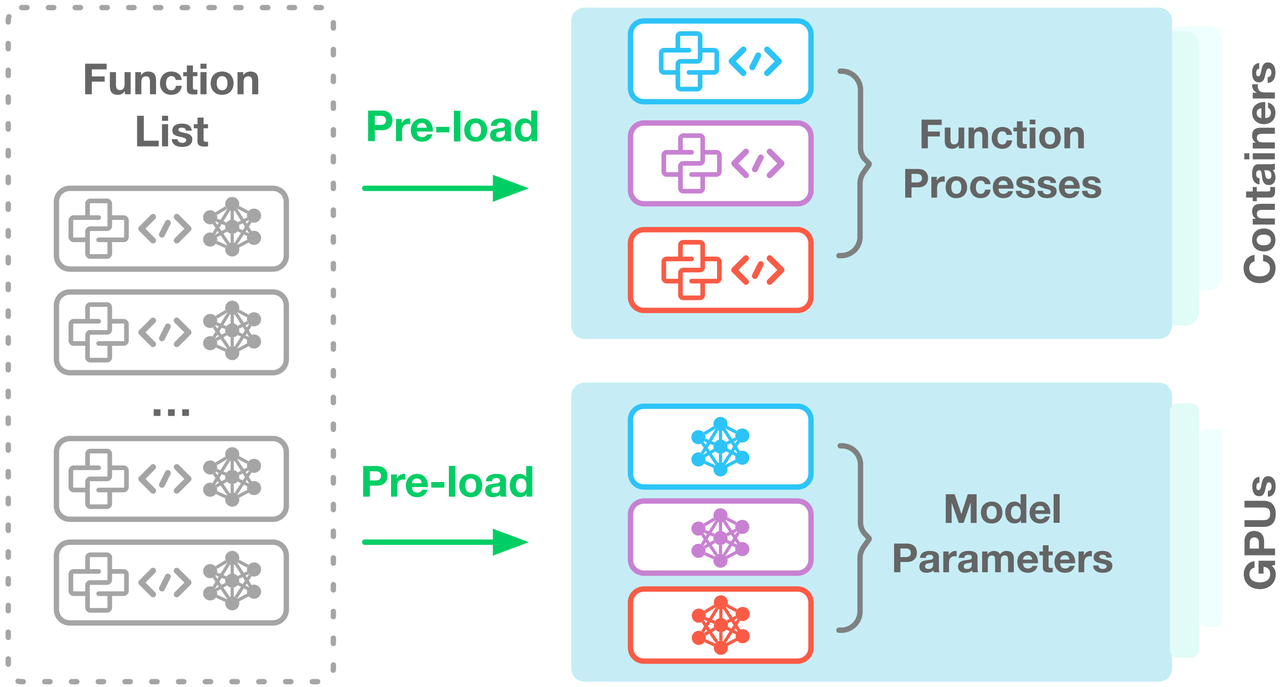
Compatible with pre-warming

Never affect container creation & removal

Opportunistic Pre-loading

So many inference functions, only limited containers.

- **Instance level:** Pre-load multipe functions within a container/GPU until full.
- **Platform level:** Pre-loading until all idle instances are full.



Opportunistic:

- Only use idle instances.
- Never create new one.

Zero Wastage

- Never affect the lifetime of pre-warmed instances.

Compatible with Pre-warming

Challenges

When to pre-load / off-load a function?

Based on its request's arrival rate

Estimated using the prediction model of pre-warming methods.

How to minimize loading latency within the limited instances?

Priority-based Bin-Packing

$$E(f) = P_{arrive} \times T_{loading}$$

Challenges

When to pre-load / off-load a function?

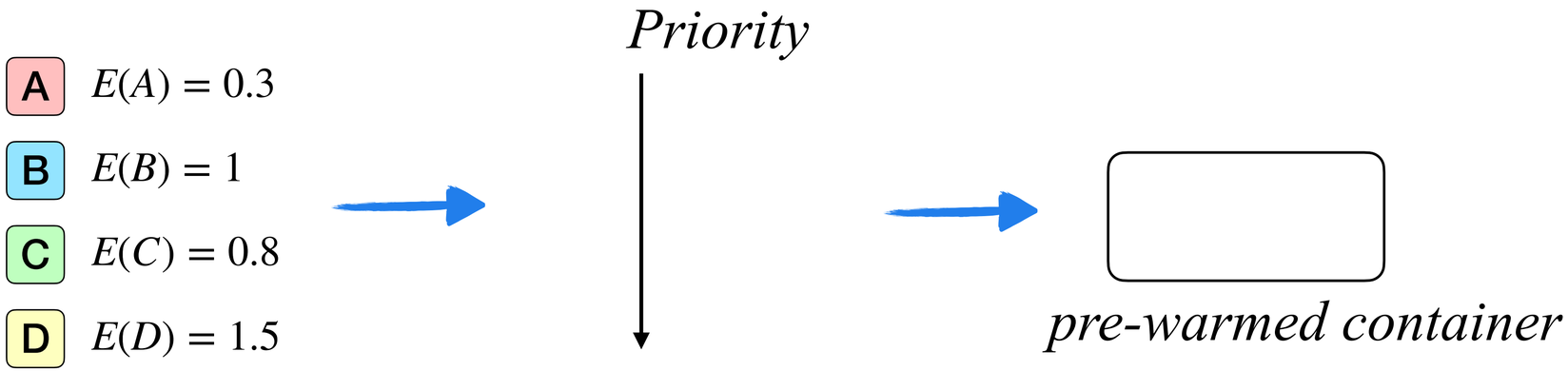
Based on its request's arrival probability

Estimated using the prediction model of pre-warming methods.

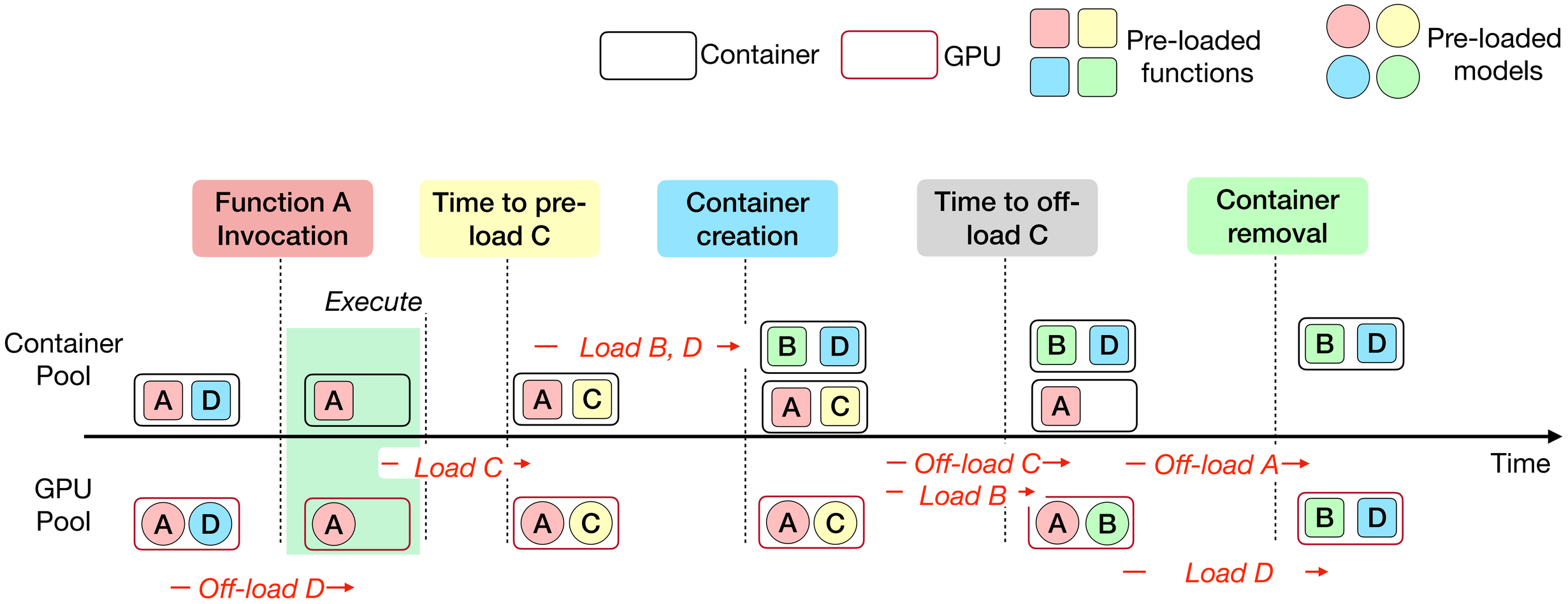
How to maximize acceleration within the limited instances?

Priority-based Bin-Packing

$$E(f) = P_{arrive} \times T_{loading}$$



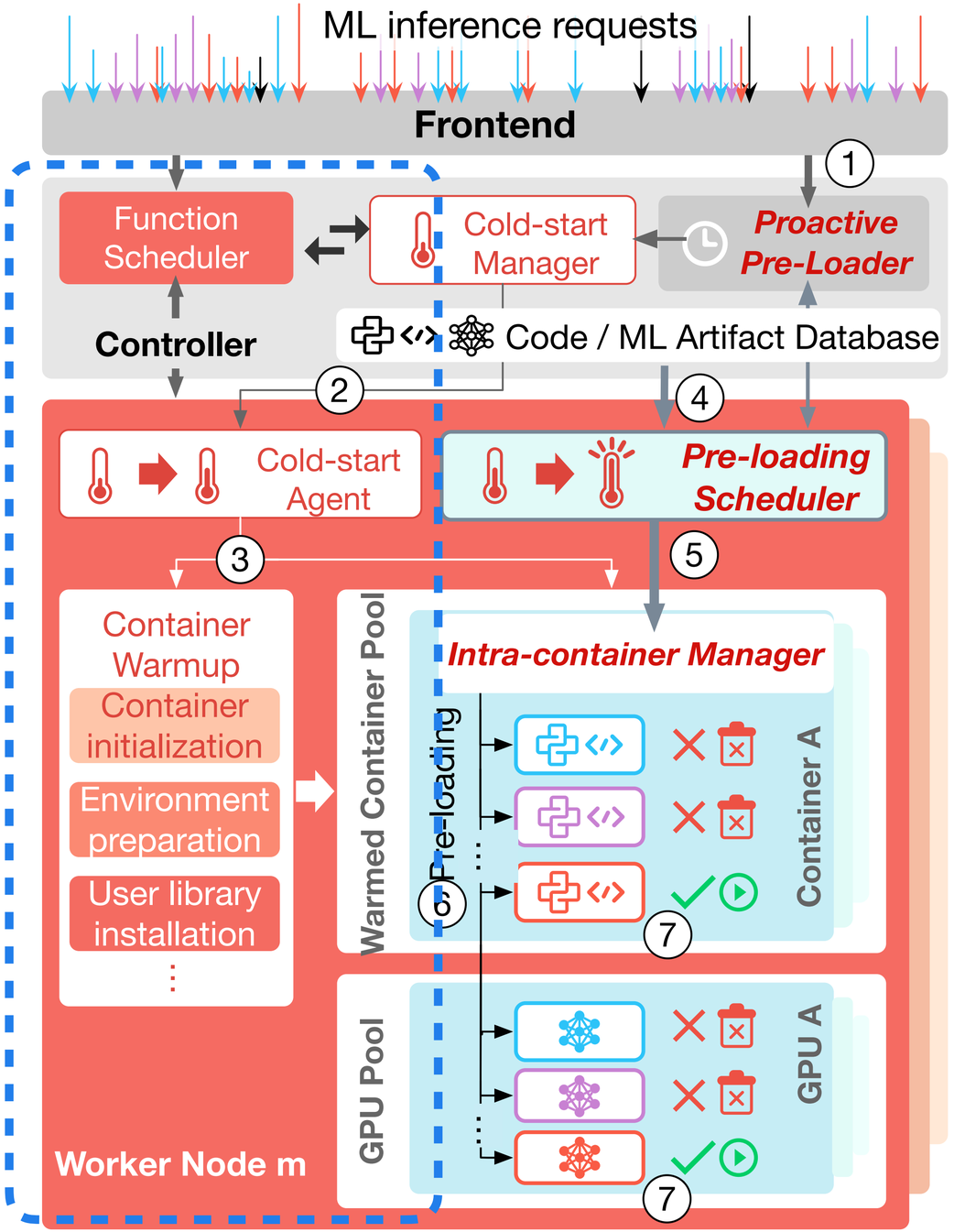
InstaInfer Workflow



Pre-Loading Workflow

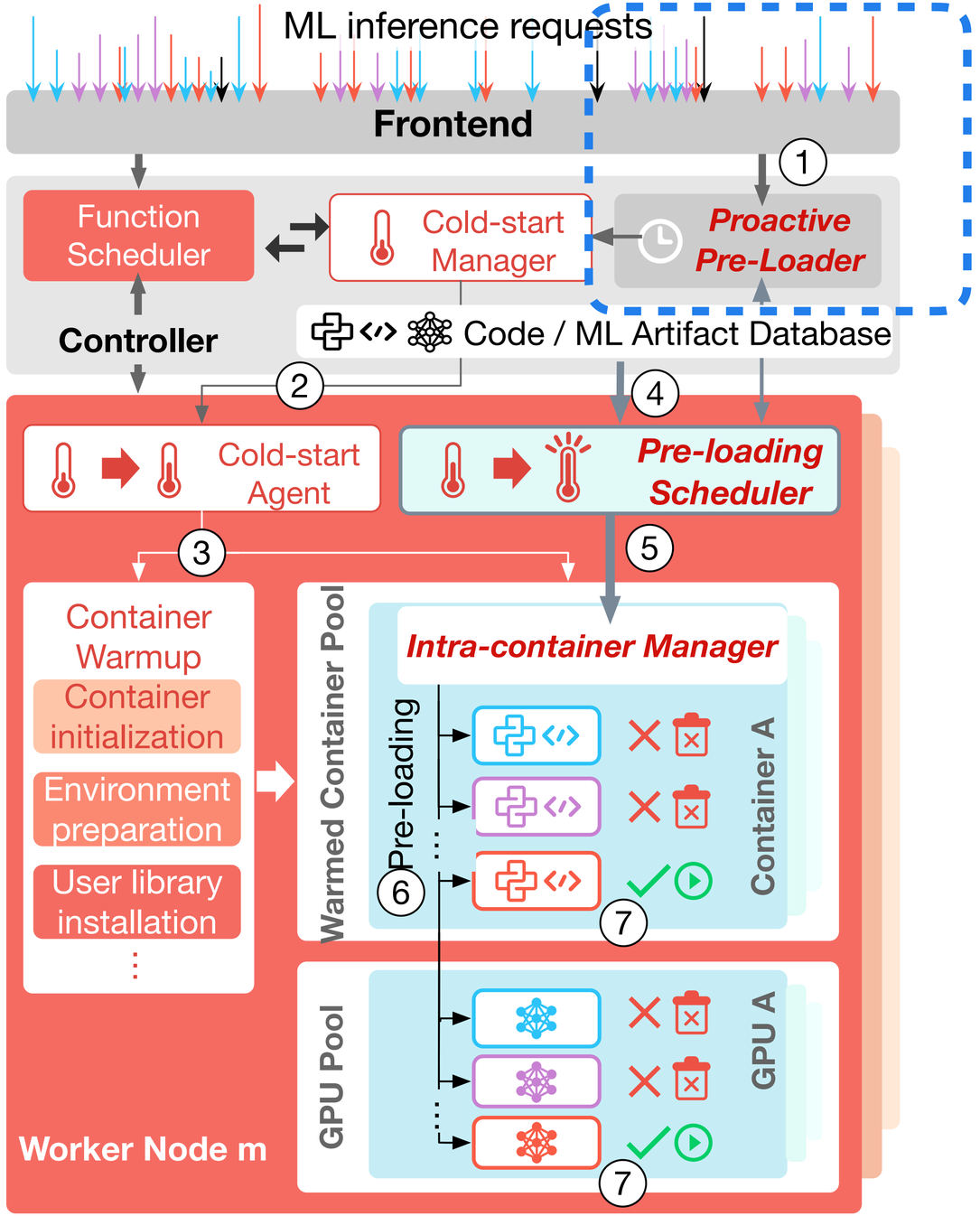
The pre-warming mechanism creates idle containers.

InstaInfer's Architecture



Pre-Loading Workflow

Records past invocations to decide when to load each function.

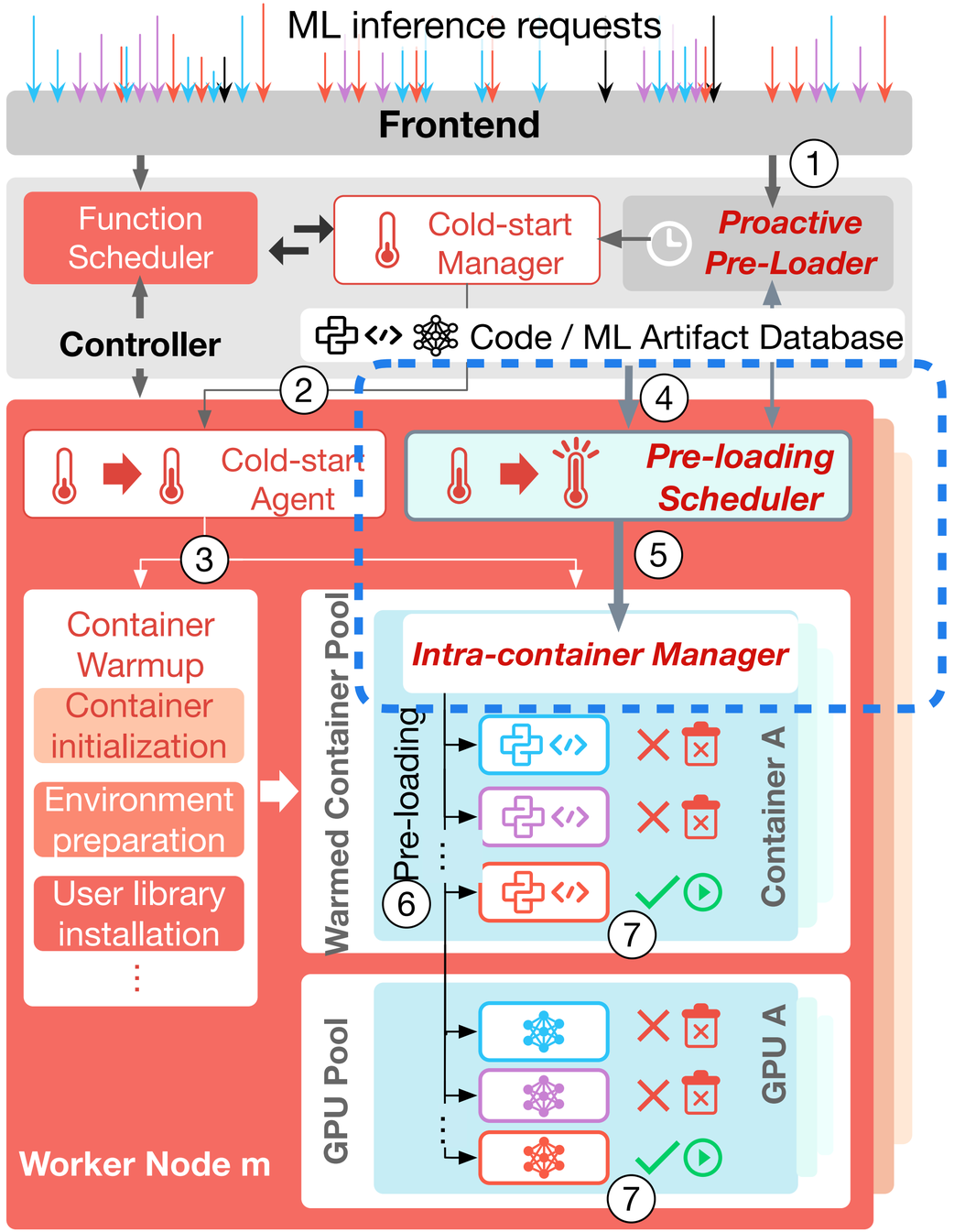


InstaInfer's Architecture

Pre-Loading Workflow

Operates bin-packing, pre-loading functions into instances.

InstaInfer's Architecture

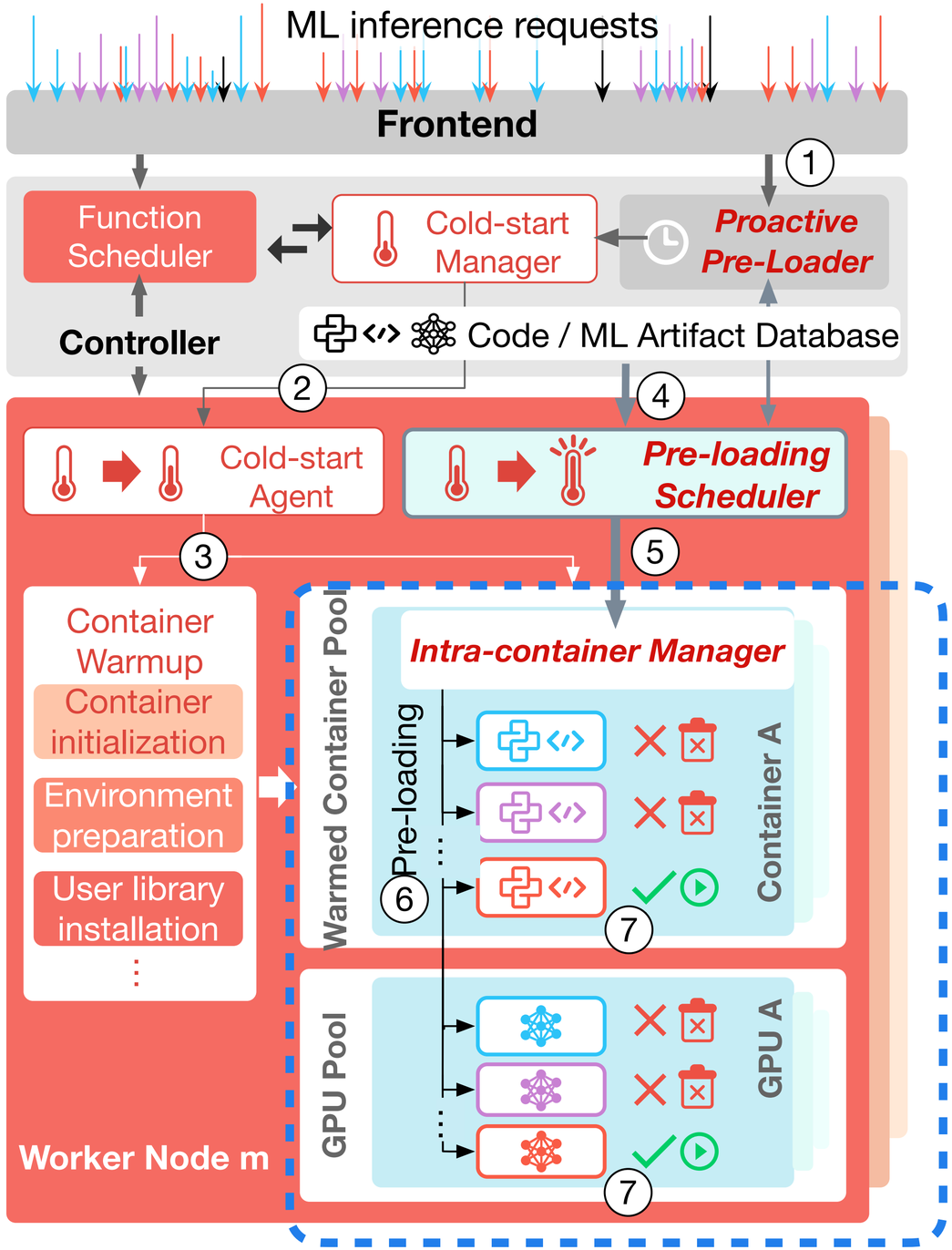


Pre-Loading Workflow

Operate loading and off-loading;

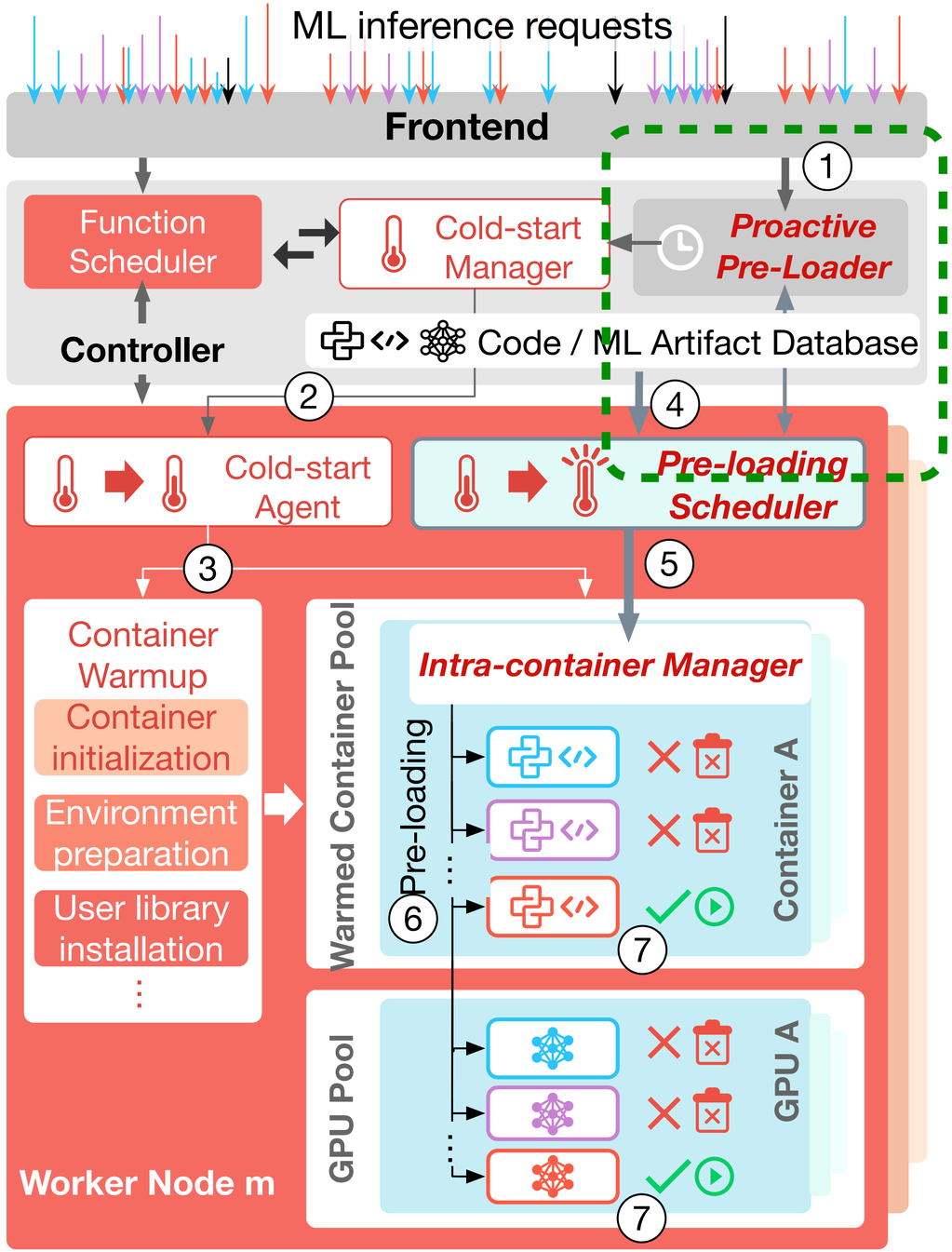
Guarantee security

InstaInfer's Architecture



Invocation Reaction Workflow

Choose a worker node with an instance that loads the function.

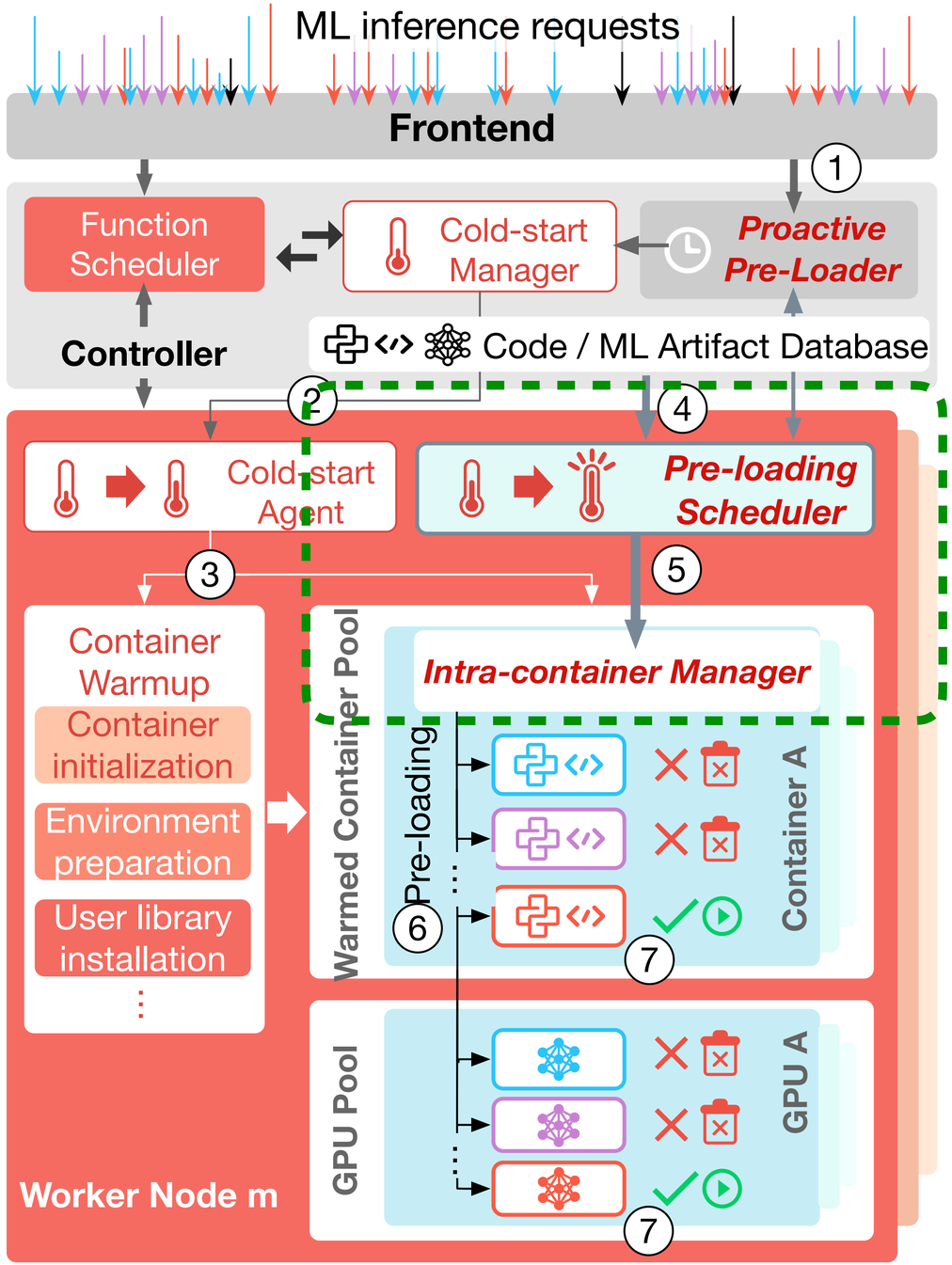


InstaInfer's Architecture

Invocation Reaction Workflow

Send the invocation to the container that loaded the function

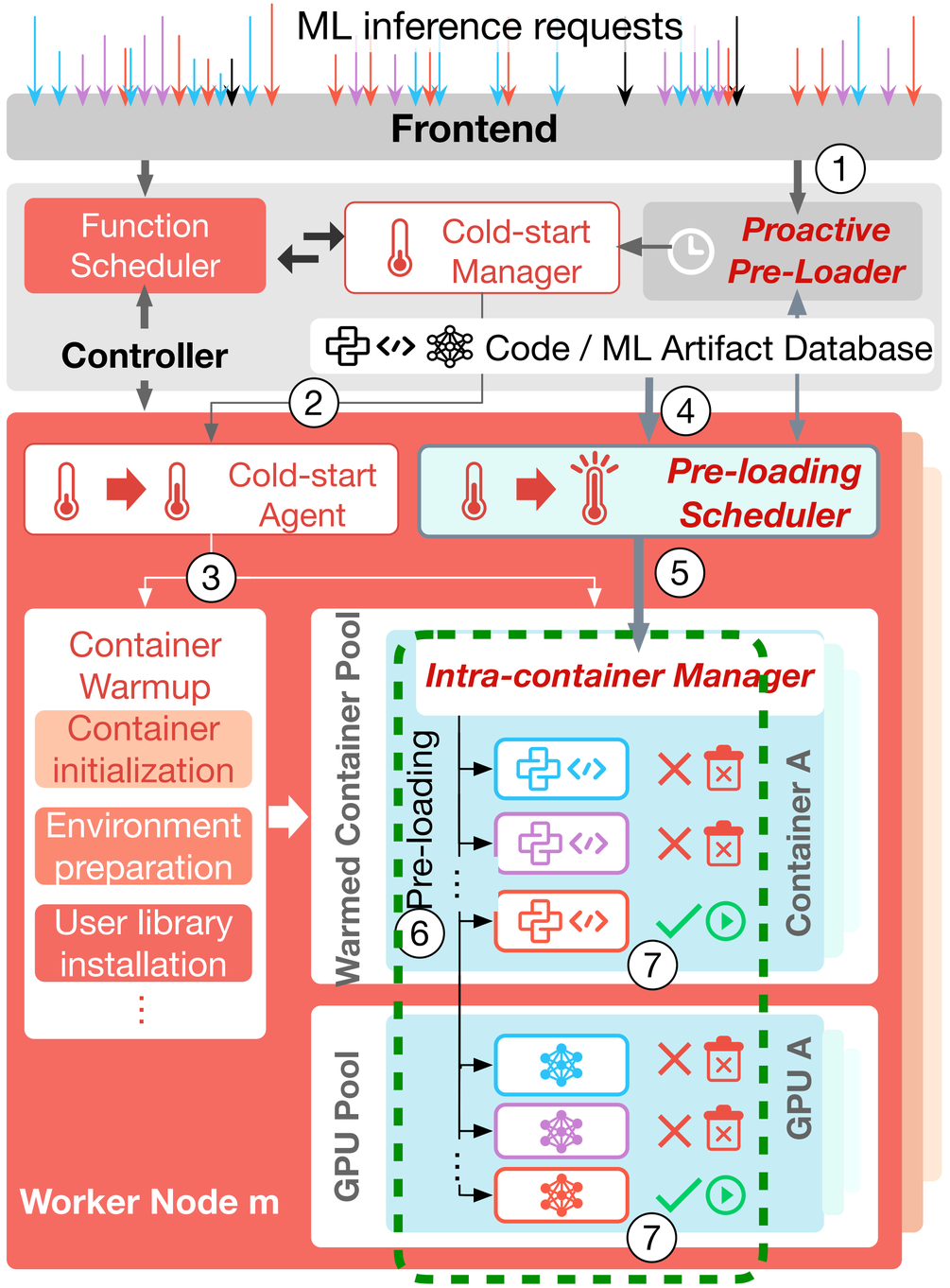
InstaInfer's Architecture



Invocation Reaction Workflow

The container terminates all other functions and starts inference.

InstaInfer's Architecture



Implementation

InstaInfer is prototyped on top of Docker and Apache OpenWhisk

Proactive Pre-Loader

OpenWhisk's Load Balancer

Pre-Loading Scheduler

OpenWhisk's Container Pool

Intra-Continaer Manager

Docker Container's Proxy

GPU Support

Nvidia MPS
&
Nvidia Container

Evaluation

Testbed

5 nodes

160 AMD EPYC CPU cores

640 GB Memory

4 Nvidia A10 GPUs

Metrics

Function response latency

Memory cost

Baselines

OpenWhisk default

Histogram

FaaSCache

Pagurus

REAP

Azure Premium

Pre-warming methods

Traces

Azure Functions traces

4-hour workloads

Histogram: Shahrad, Mohammad, et al. “Serverless in the Wild: Characterizing and Optimizing the Serverless…” ATC’20
FaaSCache: Fuerst, Alexander, et al. "SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing." ASPLOS’21
Pagurus: Li, Zijun, et al. "Help Rather Than Recycle: Alleviating Cold Startup in Serverless Computing…” ATC’22
REAP: Dmitrii Ustiugov, et al. “Benchmarking, analysis, and optimization of serverless function snapshot” ASPLOS’21
Azure Premium: Microsoft. Azure Functions warmup trigger. 2023

Evaluation

Testbed

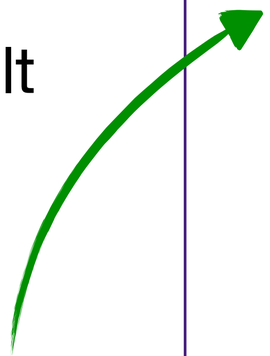
5 nodes
160 AMD EPYC CPU cores
640 GB Memory

Metrics

Function response latency
Memory cost

Baselines

OpenWhisk default
Histogram
FaaSCache
Pagurus
REAP
Azure Premium



Traces

Azure Functions traces
4-hour workloads

Histogram: Shahrad, Mohammad, et al. “Serverless in the Wild: Characterizing and Optimizing the Serverless…” ATC’20
FaaSCache: Fuerst, Alexander, et al. "SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing." ASPLOS’21
Pagurus: Li, Zijun, et al. "Help Rather Than Recycle: Alleviating Cold Startup in Serverless Computing…” ATC’22
REAP: Dmitrii Ustiugov, et al. “Benchmarking, analysis, and optimization of serverless function snapshot” ASPLOS’21
Azure Premium: Microsoft. Azure Functions warmup trigger. 2023

Evaluation

Testbed

5 nodes

160 AMD EPYC CPU cores

640 GB Memory

20 A10 GPUs

Metrics

Function response latency

Memory cost

Baselines

OpenWhisk default

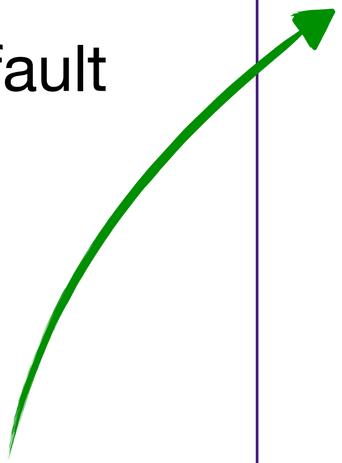
Histogram

FaaSCache

Pagurus

REAP

Azure Premium



Pre-loading method

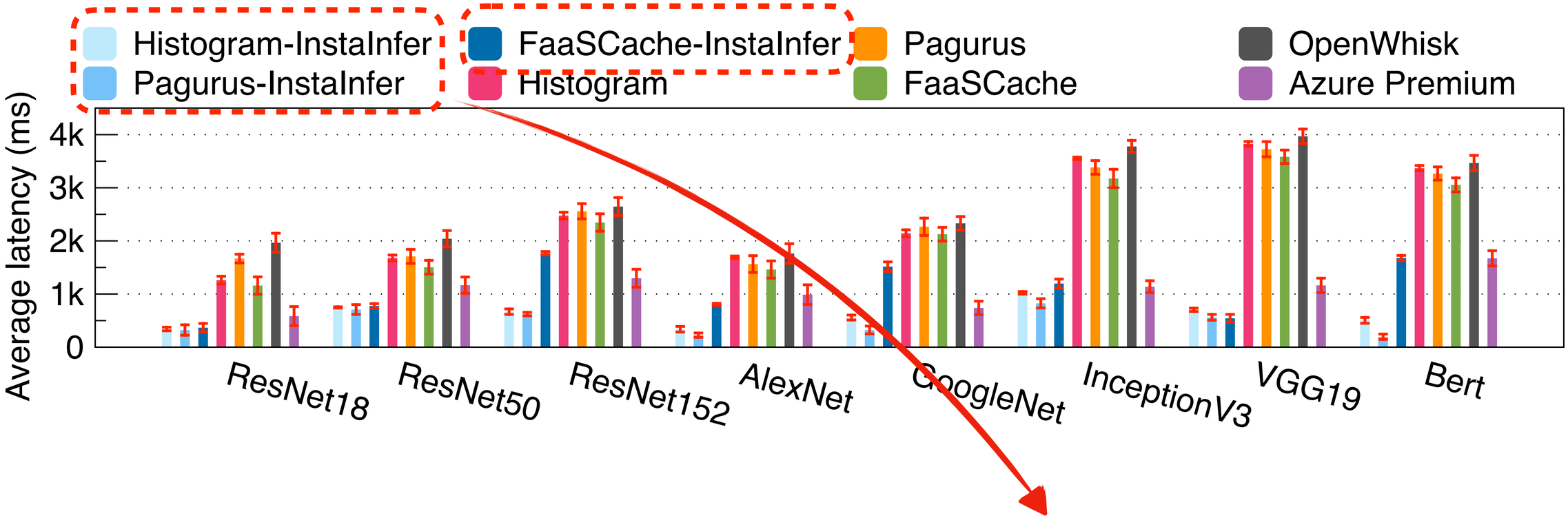
Traces

Azure Functions traces

4-hour workloads

Histogram: Shahrad, Mohammad, et al. “Serverless in the Wild: Characterizing and Optimizing the Serverless…” ATC’20
FaaSCache: Fuerst, Alexander, et al. "SeBS: A Serverless Benchmark Suite for Function-as-a-Service Computing." ASPLOS’21
Pagurus: Li, Zijun, et al. "Help Rather Than Recycle: Alleviating Cold Startup in Serverless Computing…” ATC’22
REAP: Dmitrii Ustiugov, et al. “Benchmarking, analysis, and optimization of serverless function snapshot” ASPLOS’21
Azure Premium: Microsoft. Azure Functions warmup trigger. 2023

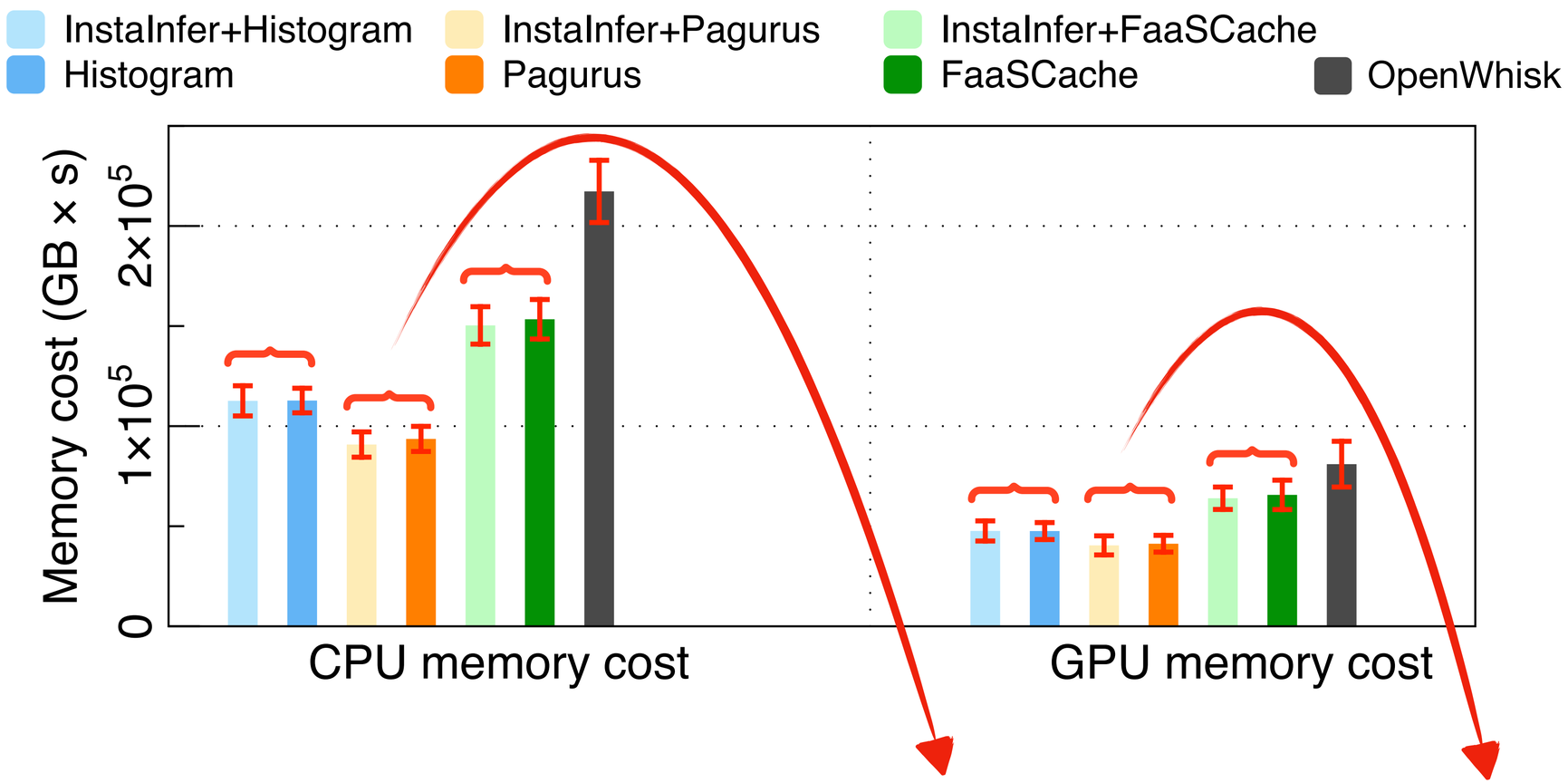
End-to-end Latency



InstaInfer is **compatible** with each pre-warming baseline

InstaInfer achieves better **invocation** latency than other baselines

Memory Cost



InstaInfer does not introduce additional **memory cost** and **GPU cost** after combining with pre-warming baselines

Opprtunistic pre-loading

Minimize loading latency

No additional cost

Transparent to providers

InstaInfer

93%

Function loading latency reduction

<1%

Memory waste

4

Compatible pre-warming methods

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

InstaInfer Code Repo:

https://github.com/IntelliSys-Lab/InstaInfer

Corresponding Author:

Jianxun Li <lijx@sjtu.edu.cn>

IntelliSys Lab

Q & A

Security & Privacy

1. User layer: Only functions belonging to the same user can be pre-loaded on the same machine.

2. Process layer: Deletes all other functions's data when a function is invoked

3. OS layer: Each pre-loaded functions' code, librairies, and data are in an unique non-root Linux user.

Proviacy: We treat each function's package is a blackbox.
(Developers are asked to modify two lines of code)

Modification Guide

Developers are asked to modify two lines of code:

Original:

```
model.load_state_dict(torch.load(model_path))
# inference…
```

InstaInfer:

```
model.load_state_dict(InstaInfer_load_model(model_path))
sys.stdin.readline()  # wait for request
# inference...
```