Tackling Cold Start of Serverless Applications by Efficient and Adaptive Container Runtime Reusing

Kun Suo[*], Junggab Son[*], Dazhao Cheng[†], Wei Chen[‡], Sabur Baidya[§]

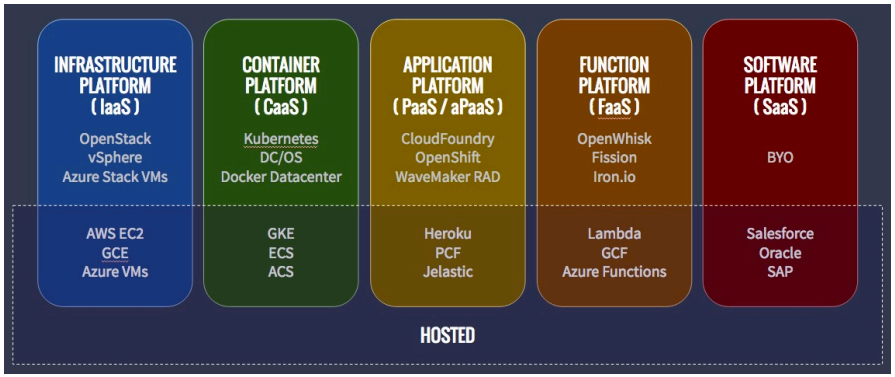
[*]Kennesaw State University, [†]University of North Carolina at Charlotte

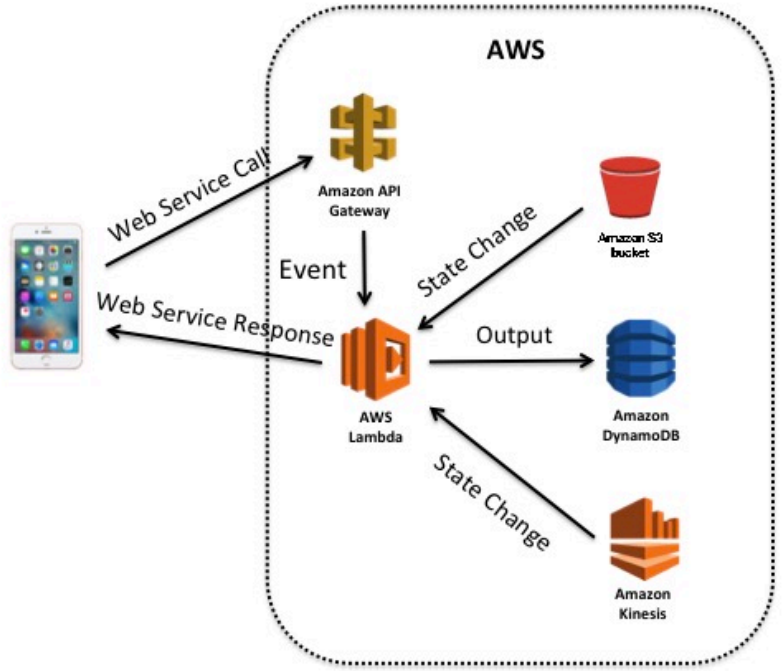
[‡]Automotive Vehicle Group, Nvidia, [§]University of Louisville

The Rise of Serverless Applications in the cloud

- Serverless computing is becoming increasingly important to on-demand, elastic, and cost-effective cloud services

- Key benefits of serverless applications
  - ✓ High performance
  - ✓ High scalability
  - ✓ Built-in availability
  - ✓ Fault tolerance

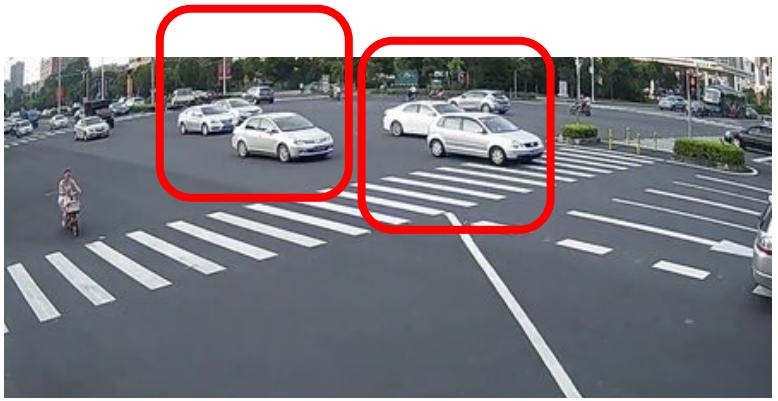




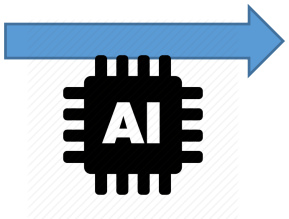
Serverless at the Edge

idle

Event
Trigger



Run ML/
DL code



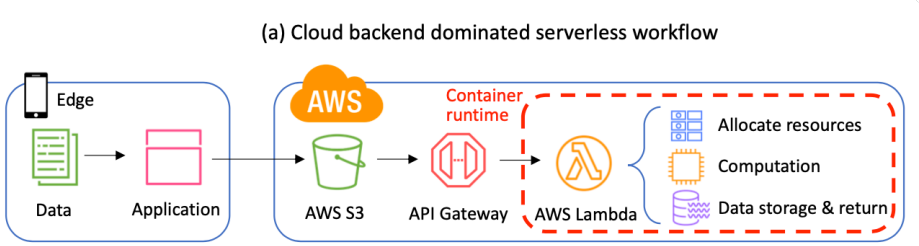
Size

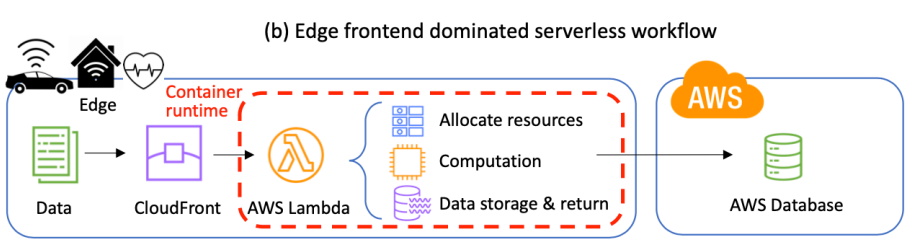
Speed

Plate

...

Serverless Pro vs Con





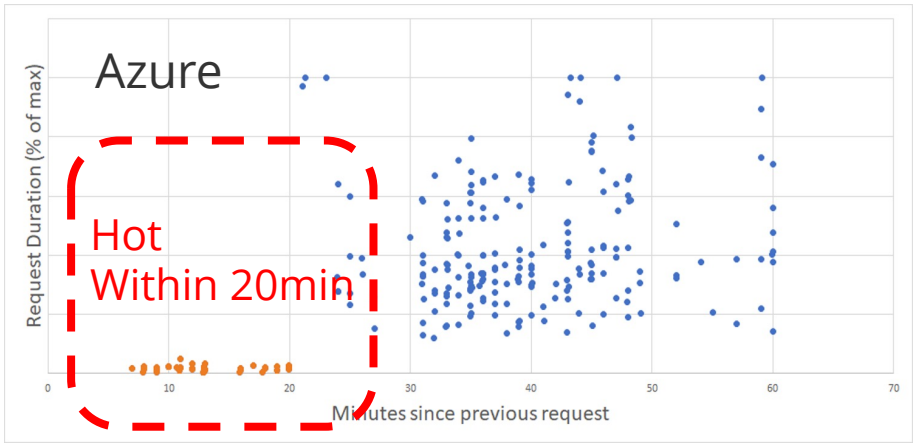
Benefits:

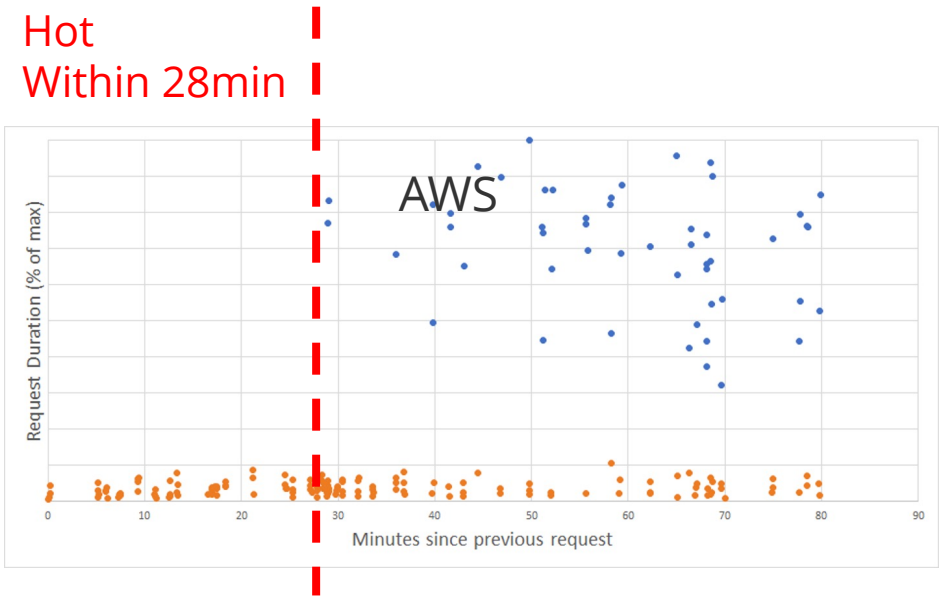
- Drop cost
- Fast Dev/deployment
- More security
- Microservice support
- Auto scale
- …

Drawbacks:

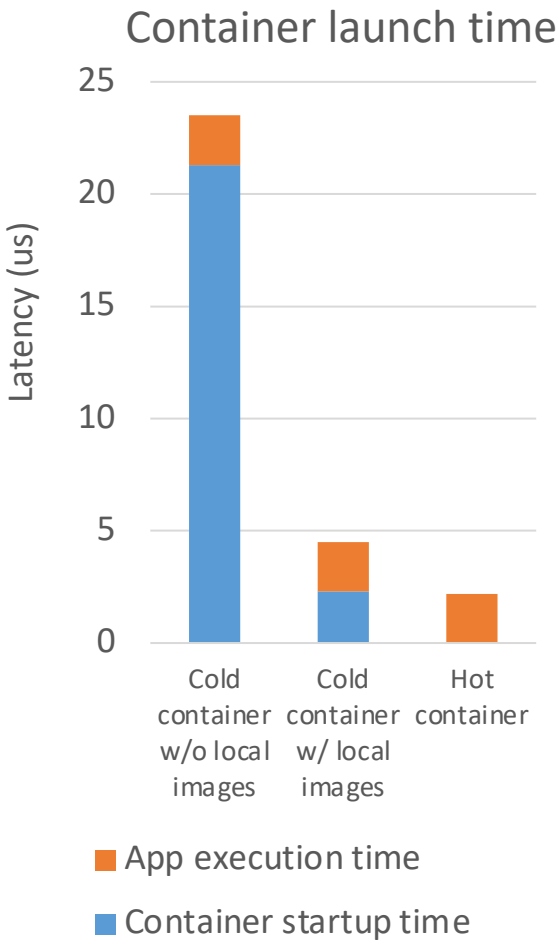
- Not good for long time apps
- Deep dependency on platform
- Cold start
- Lack of debugging or monitoring
- …

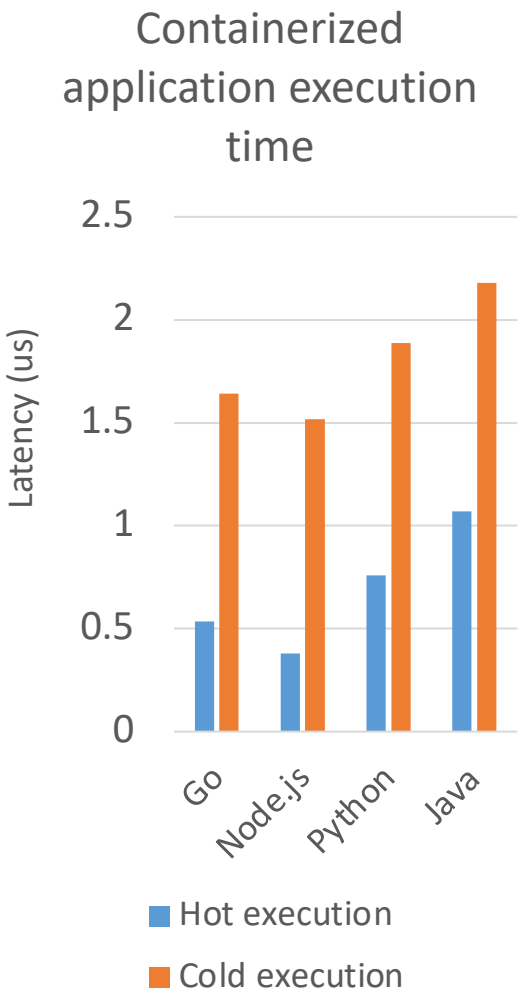
Cold Start Impact

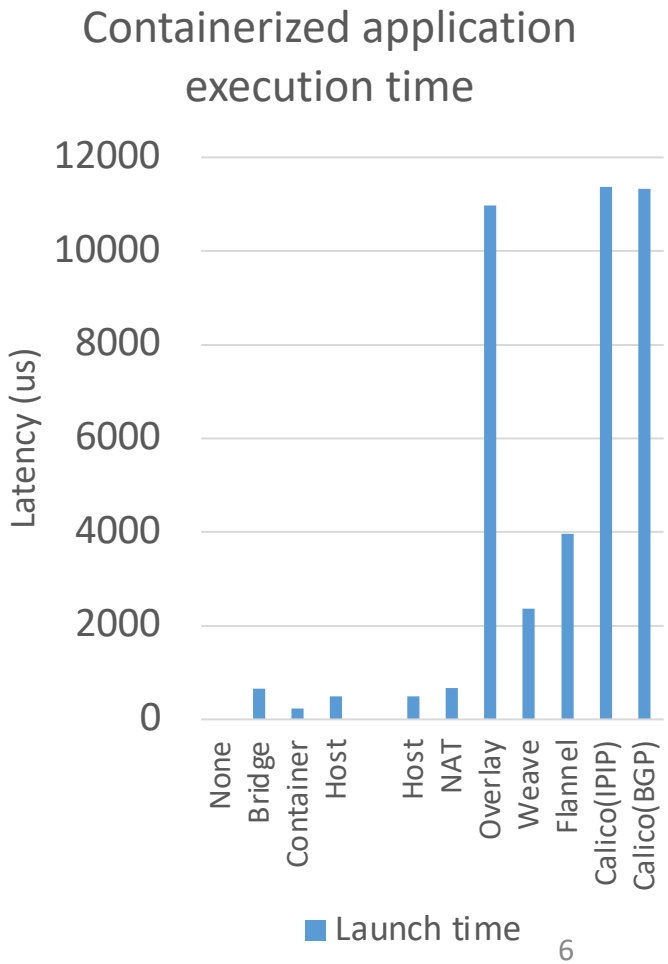




Motivation examples of cold start in serverless applications



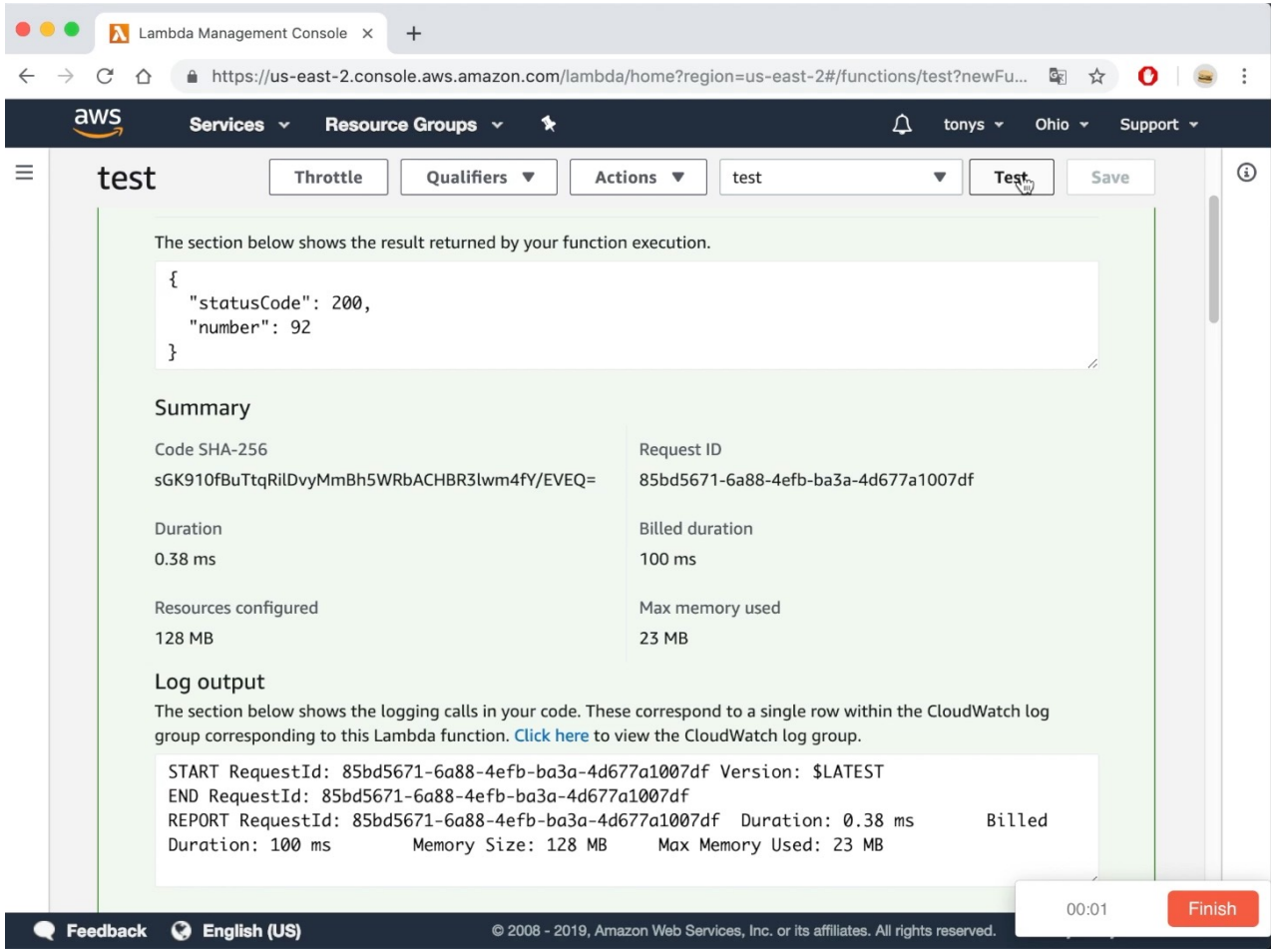




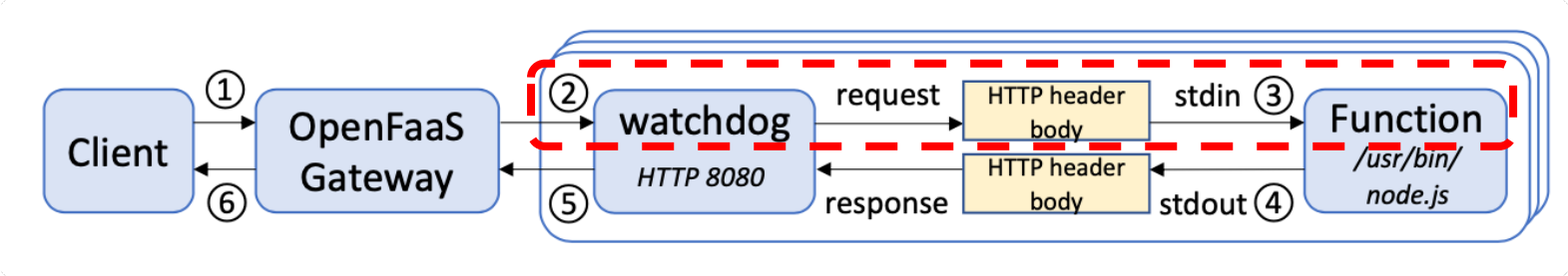
AWS Lambda Cold Start Demo

Python:
http request return random number

Measure the request latency



Analysis of cold start overhead



- OpenFaaS, an event-driven functions and microservices platform
- We added timestamps in source code and record six moments during the workflow path
- We found that function initiation time (2→3) dominates the total latency

Containerized Application Characteristics

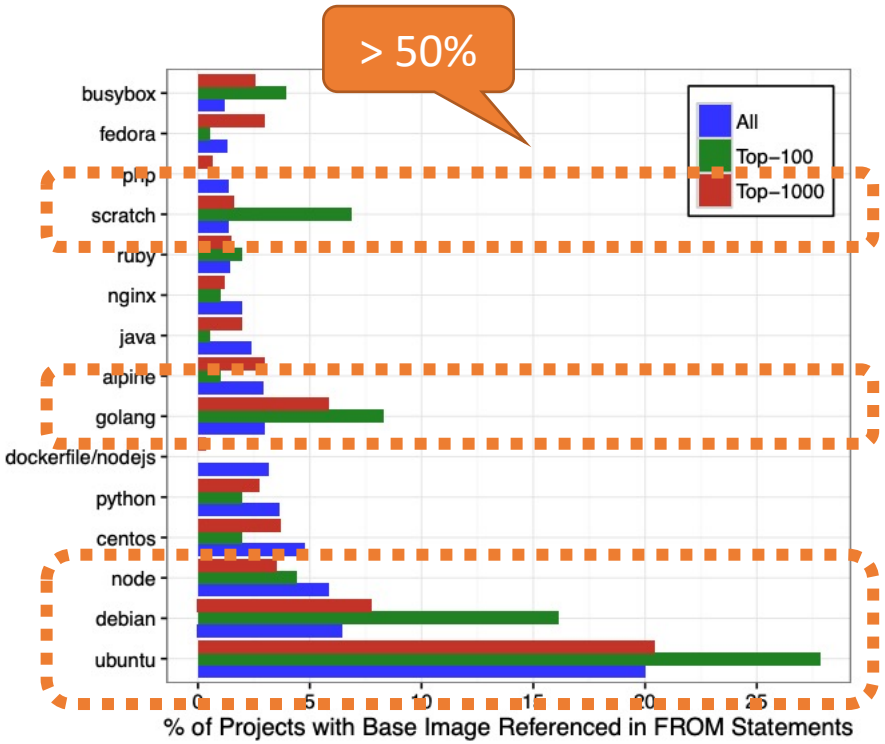


Fig. 5. Percentage of usage of 15 most commonly used base images

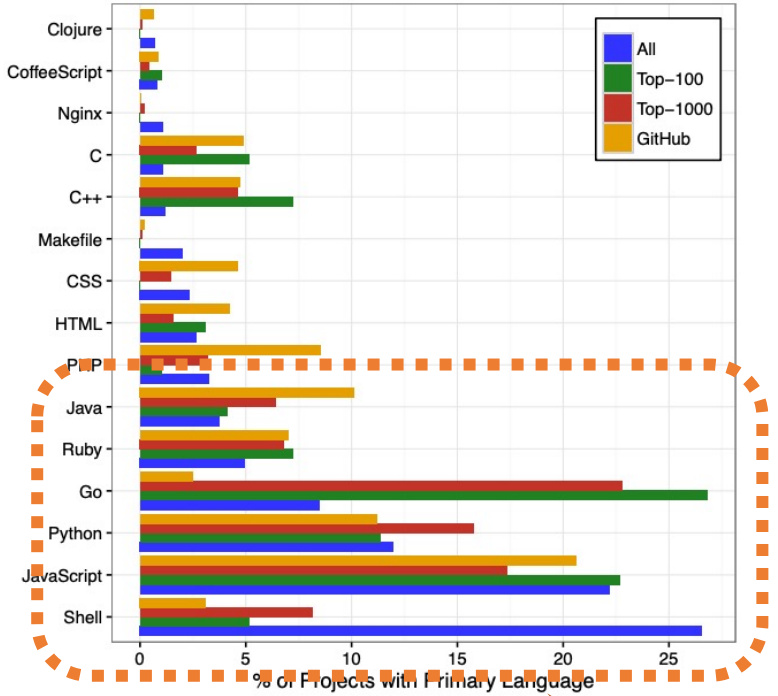


Fig. 4. Distribution of top 15 languages in our dataset

> 50%

> 70%

Related Work

- **Accelerating Container Startup**
  - ✓ Slacker [FAST-16], SAND [ATC-18], SOCK [ATC-18], Catalyzer [ASPLOS-20]
  - ✓ Industry: periodically booting the containers, reducing artifact program size, prefetching the hot data

✗ Not focus on container runtime

- **Resource Management and Allocation**
  - ✓ Elastic resource management: Huang [HPDC-19], X-containers [ASPLOS-19], Mohan [Hotcloud-19]
  - ✓ Resource prediction and dynamic allocation: Emars [Cloud-18]

✗ Only effective within a protected domain, not serverless

- **Optimizing Container Architecture**
  - ✓ Seuss [EuroSys-20], Iron [NSDI-18], Lloyd [IC2E-18], Nguyen [Serverless-19]

✗ Require changes To applications

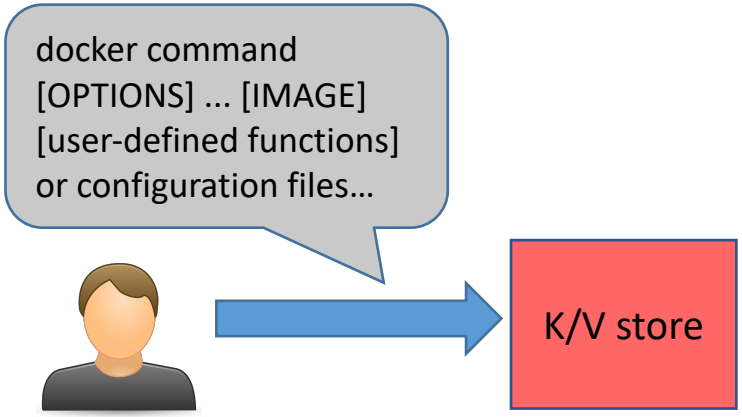
Industry Practices

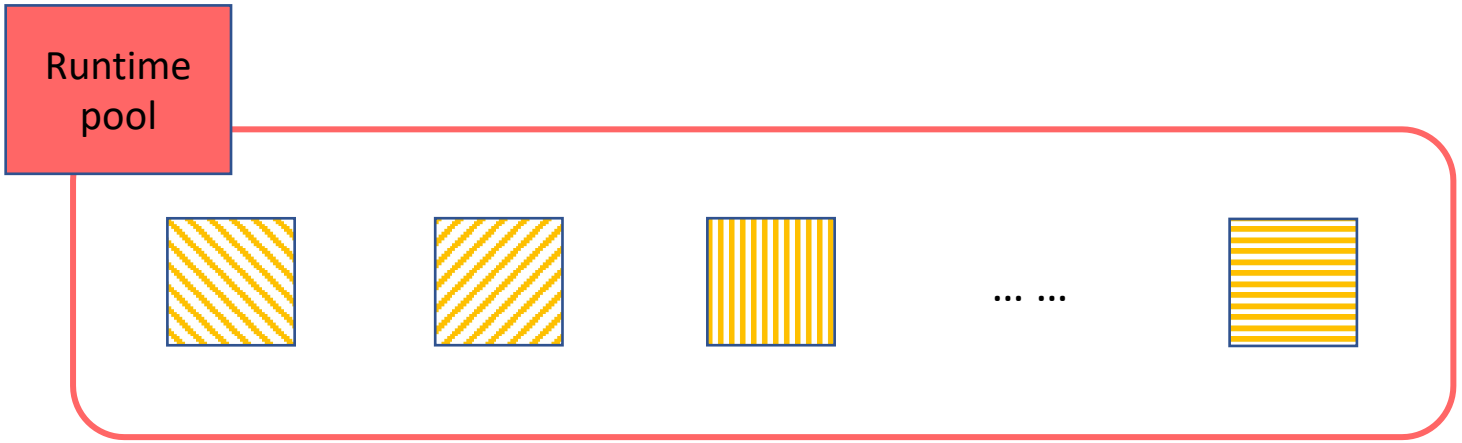
- **Alibaba:**
  - ✓ New image format
  - ✓ P2P network pair for data distribution
  - ✓ Efficient decompression algorithm
- **Tencent**
  - ✓ Deploys their functions based on lightweight virtualization
  - ✓ New scheduling policies for active functions
- **AWS**
  - ✓ Keep-alive policy that retains the resources in memory for 10x minutes after a function execution
  - ✓ AWS Hyperplane create interfaces in advance for each serverless application instead of creating Virtual Private Cloud (VPC) for each function during execution

Our Approach: HotC

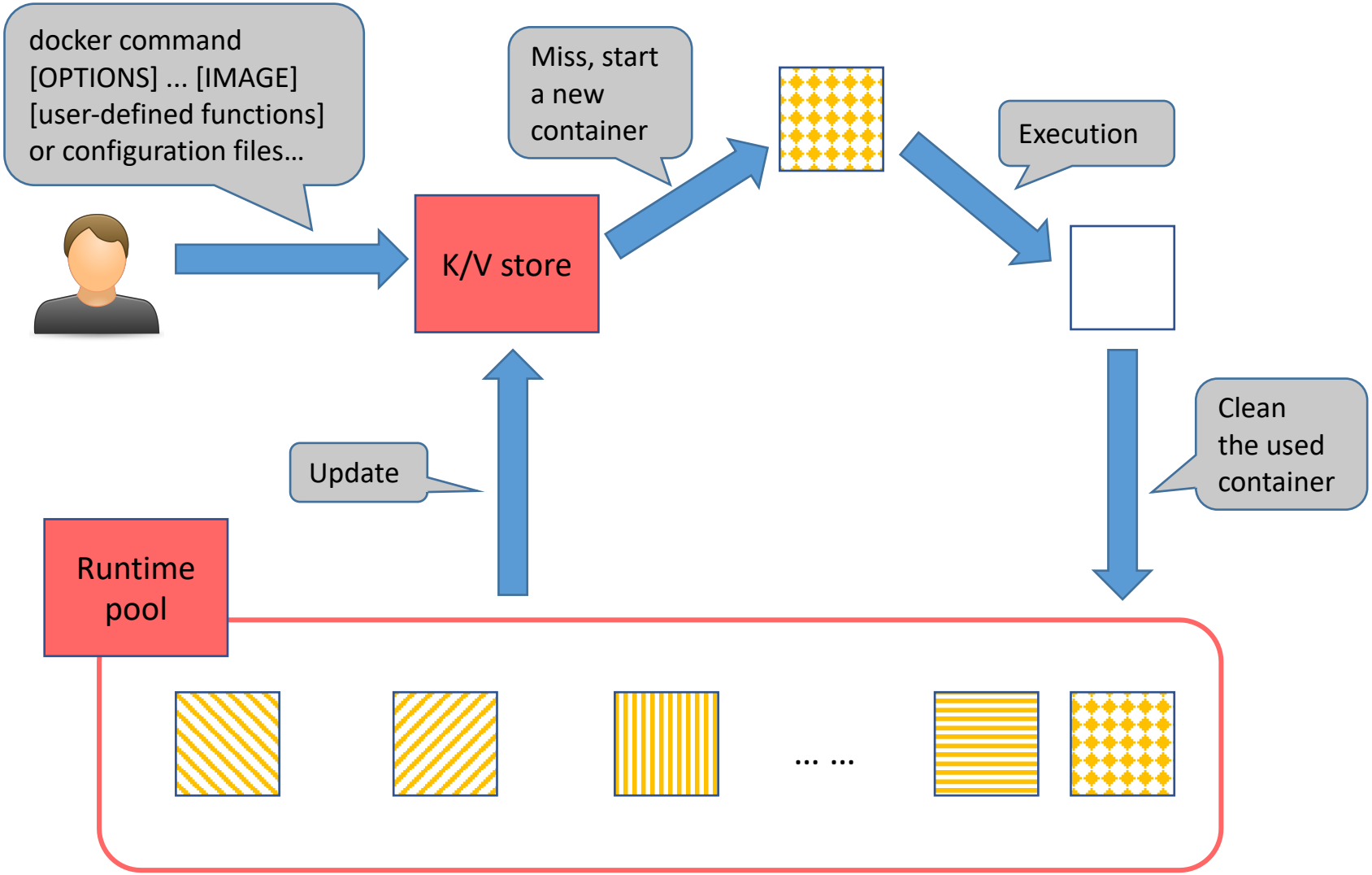
- **What is HotC?**
  - ✓ An approach to leveraging lightweight containers and an efficient reuse mechanism to prevent the unnecessary cold start, excessive resource allocation, and reclaim
- **Key designs of HotC**
  - ✓ A middleware between clients and backend servers
  - ✓ Maintain a live container runtime pool and adaptively update the pool over time
  - ✓ Reuse the same type of available container runtime to new requests

HotC Overview

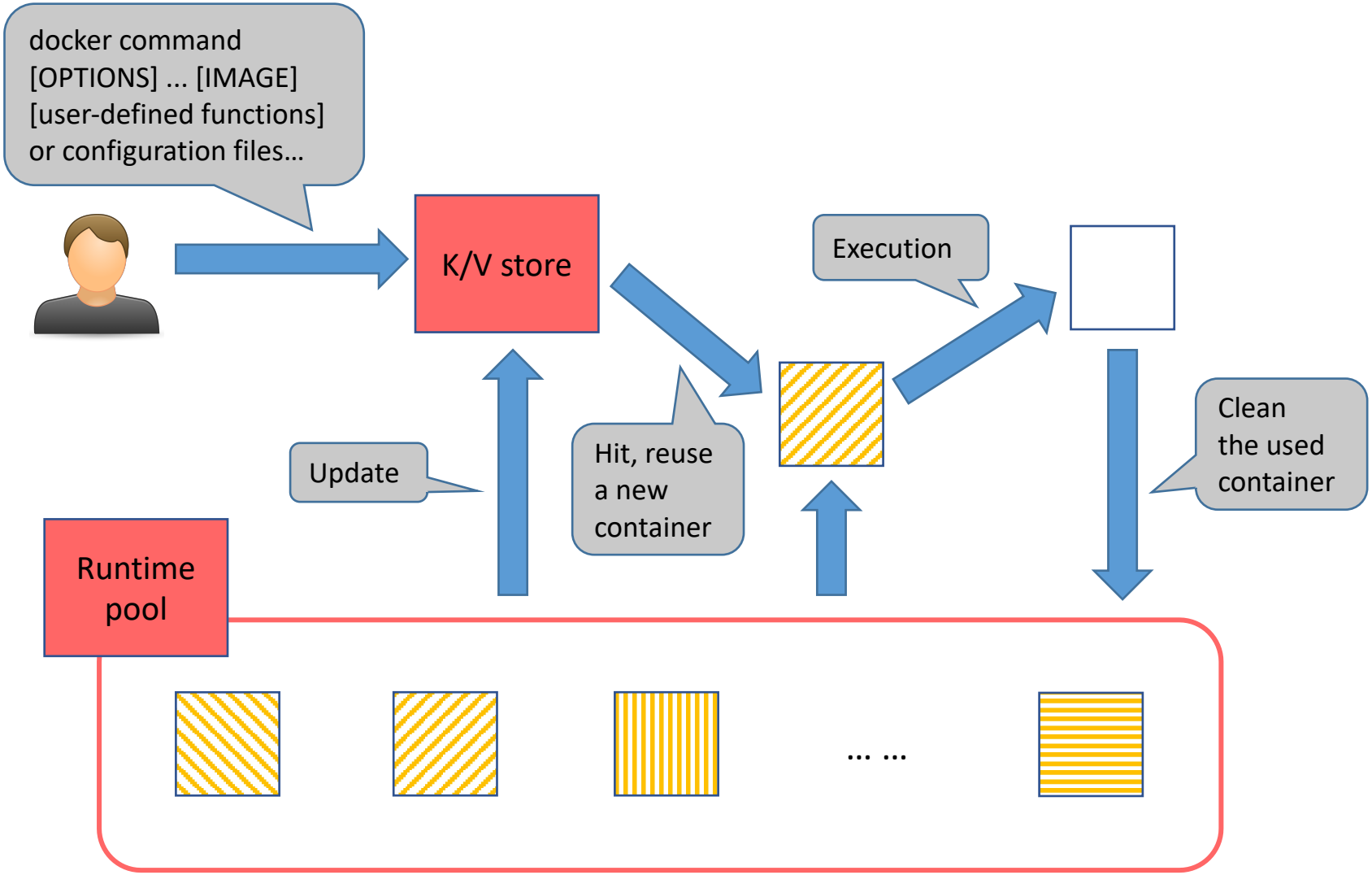




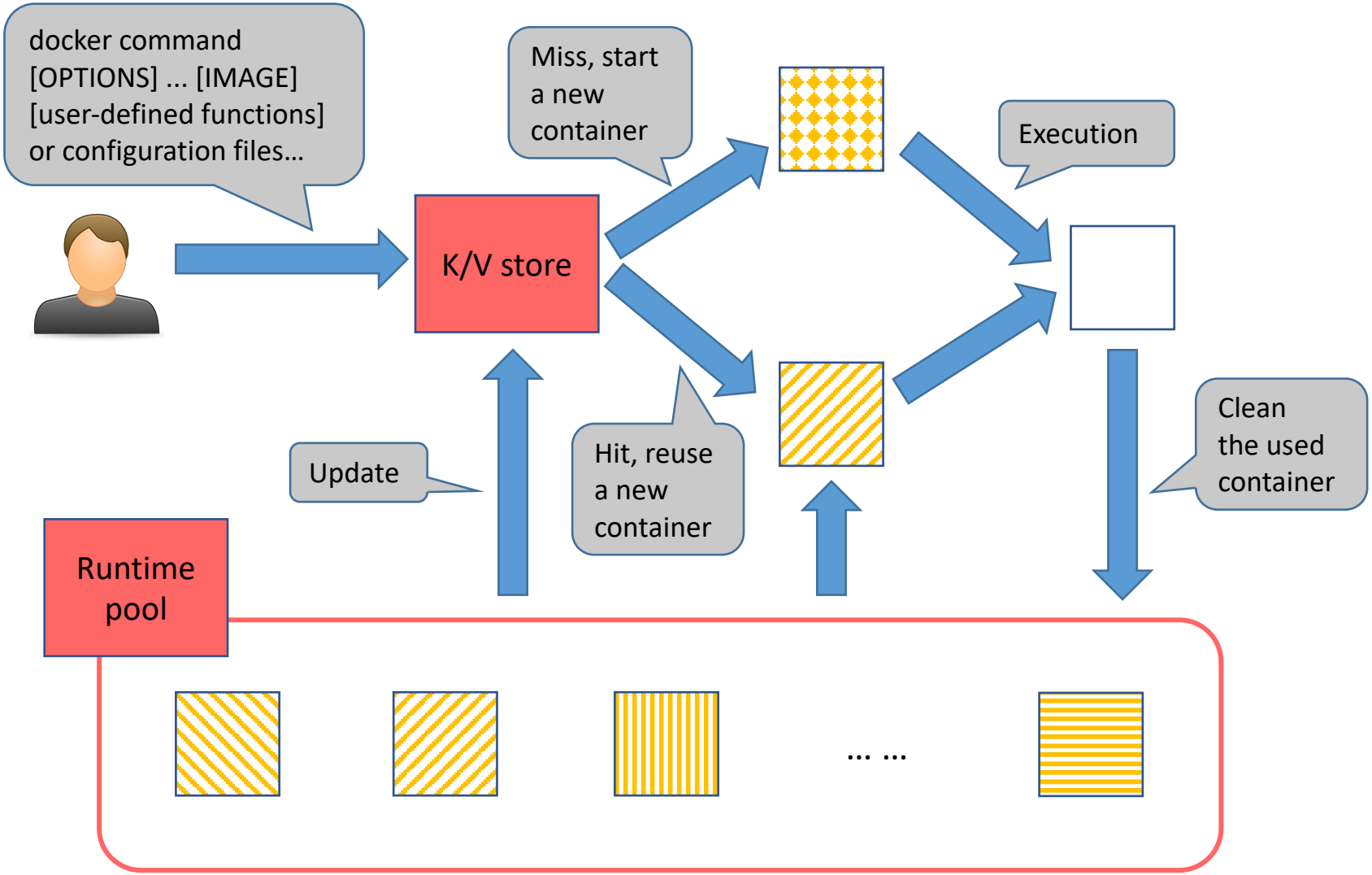
HotC Overview



HotC Overview



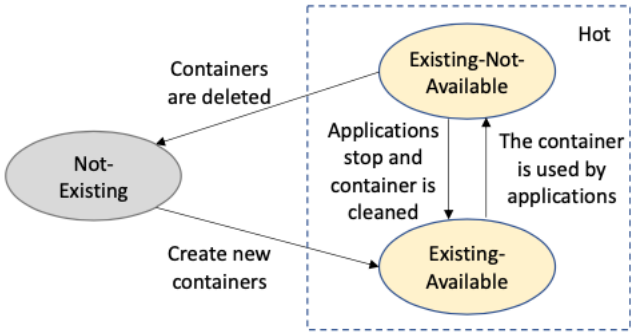
HotC Overview



Parameter Analysis

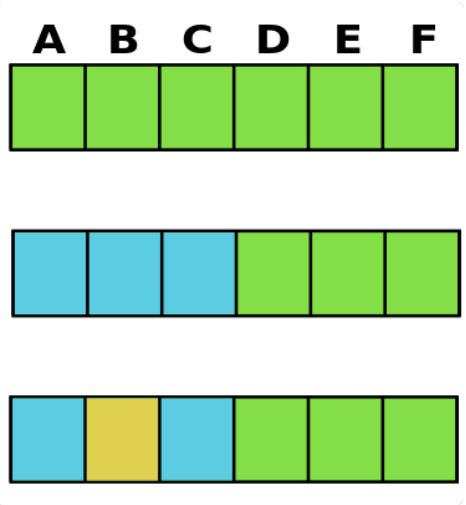
- **Analyze user command or configuration file**
  - ✓ Parameters: container images, network configuration, UTS settings, IPC settings, execution options, etc.
- **HotC treats containers with identical parameter configurations as the same**
  - ✓ HotC maintains a key value store to track the available containers
  - ✓ Three states for the container: *Not-Existing, Existing-Not- Available* and *Existing-Available*
  - ✓ Client reuses the first available container

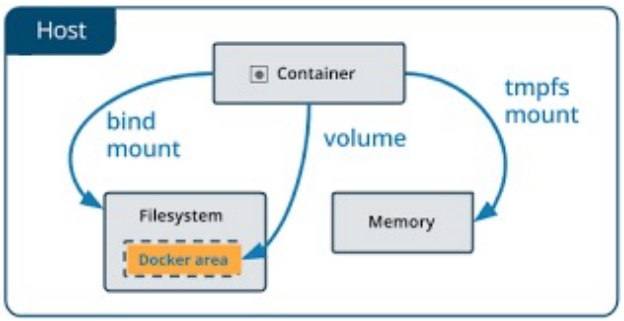
```
nginx:
    image: waldemarnt/cakephp-nginx
    ports:
        - 80:80
    links:
        - php
    volumes_from:
        - application
    volumes:
        - ./logs/nginx/:/var/log/nginx
```



Container Runtime Pool & Cleanup

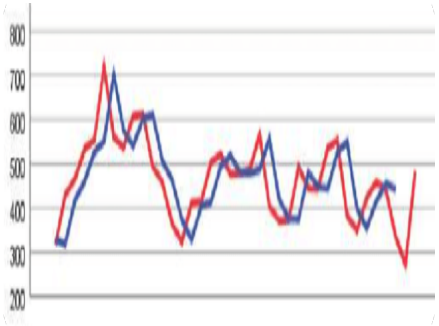
- HotC maintains a container pool inside the memory
- Maximum number of live containers to 500 and the memory usage threshold as 80% in the host
- Heuristically identify the memory pressure and terminated the oldest if necessary
- Cleanup the used containers using mounted volumes



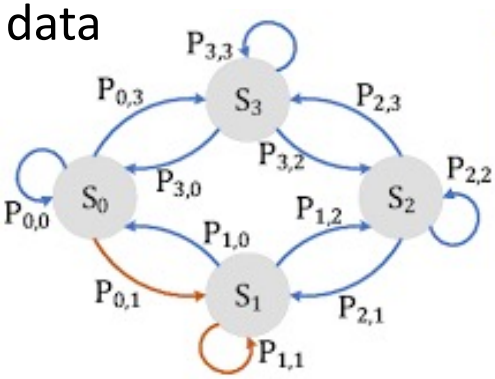


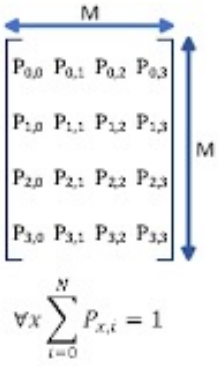
Adaptive Live Container Management

- Container prediction: whether one container will be reused
  - ✓ The number of specific type of container runtimes $e_{k_i,t}$ with configuration k_i at time t = $\alpha * history[k_i][t] + (1 - \alpha) * e_{k_i,t-1}$, where $history[k_i][t]$ are time series data of how many specific type of container runtime in the pool and α is the exponential smoothing coefficient



- Markov Prediction
  - ✓ Overcome short-term forecasting with fewer data and volatility
  - ✓ Predict the results through the transition probability between states and can better compensate for above limitations

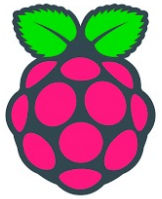


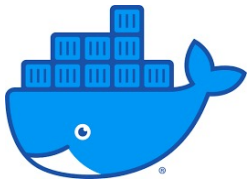


Evaluation Settings

- DELL PowerEdge T430 server:
  - ✓ Dual ten-core Intel Xeon E5-2640 2.6GHz processors, 64GB memory, Gigabit Network and a 2TB 7200RPM hard drive
  - ✓ Ubuntu 16.04 and Linux kernel version 4.4.20

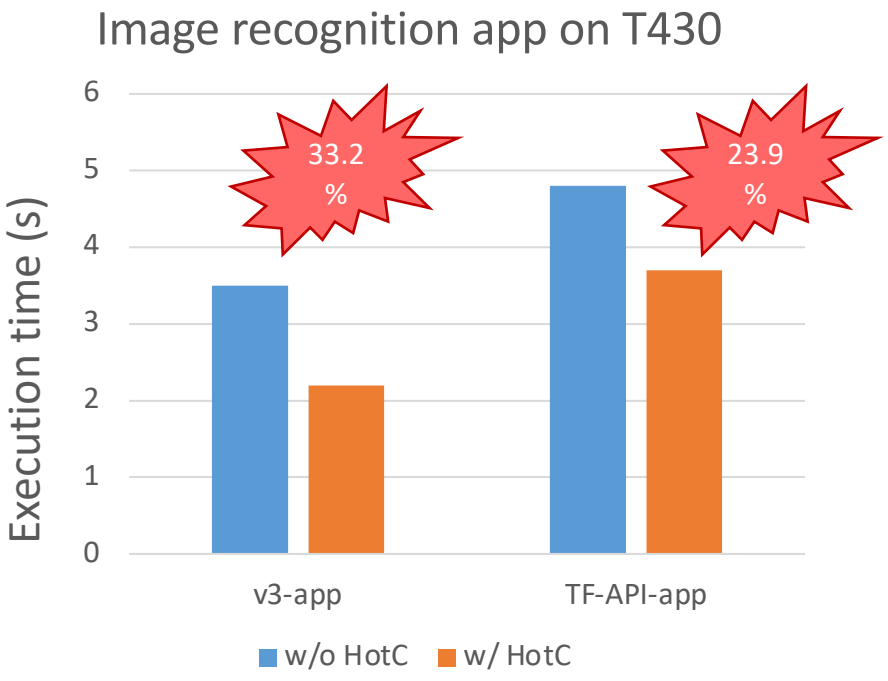
- Raspberry Pi 3:
  - ✓ Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB memory and 32GB storage
  - ✓ Linux Raspberrypi 4.14

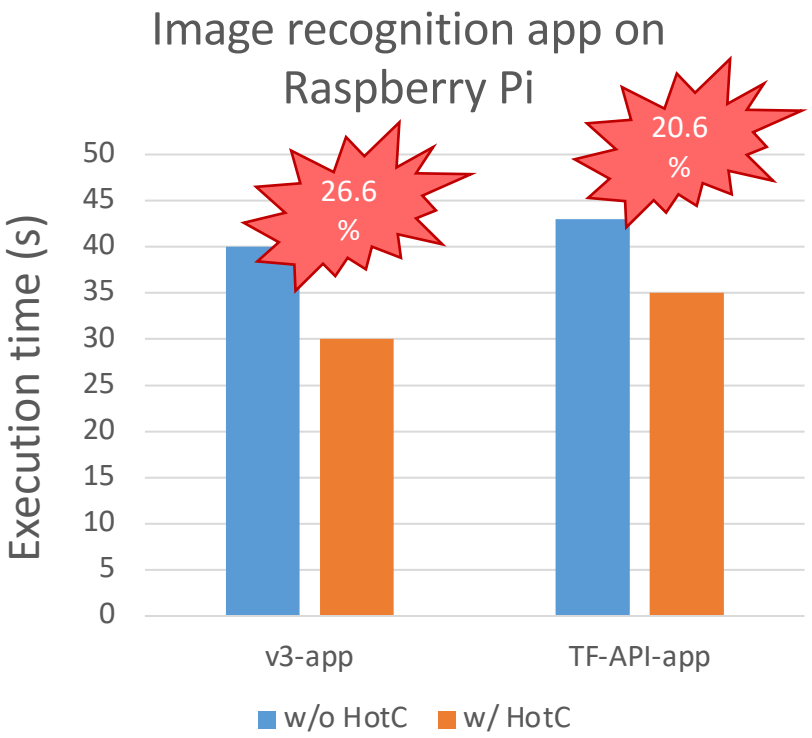




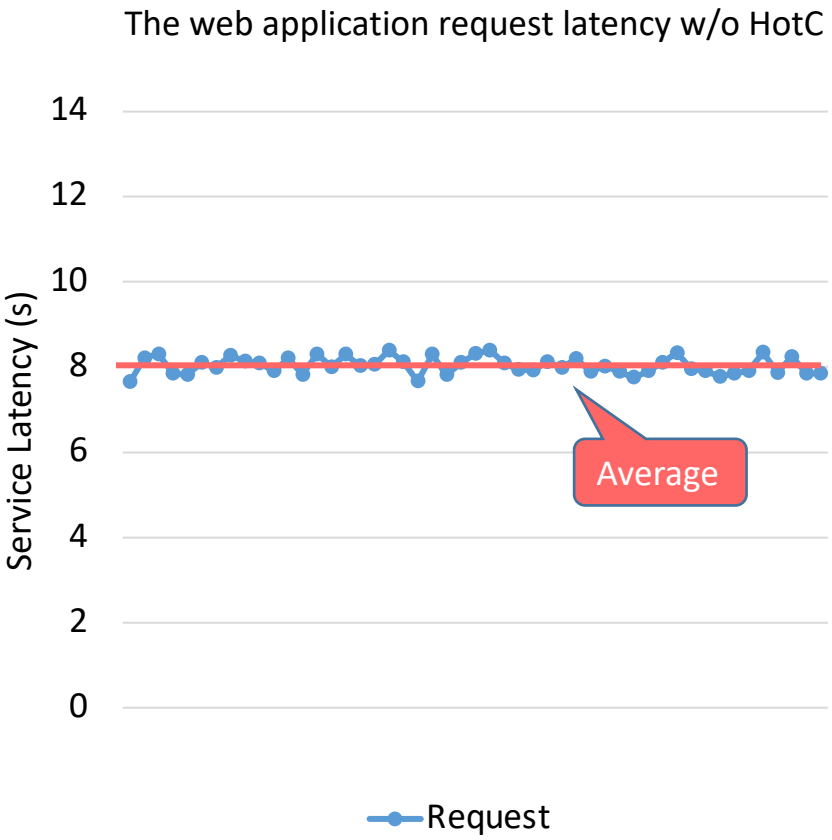
- Docker 1.17, OpenFaaS 0.8.5

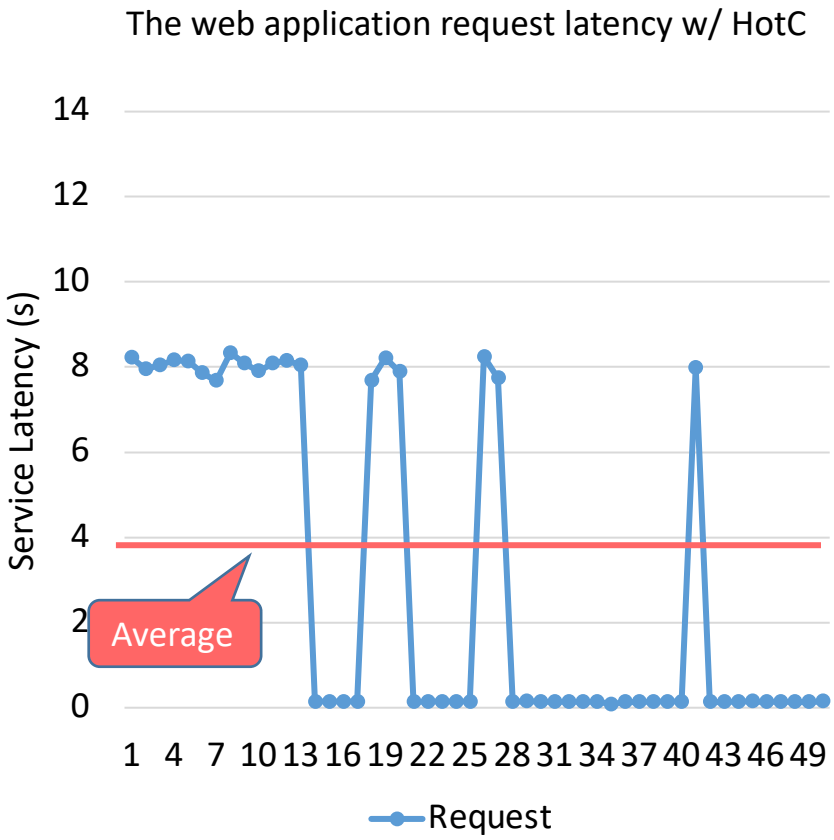
The image recognition application execution time



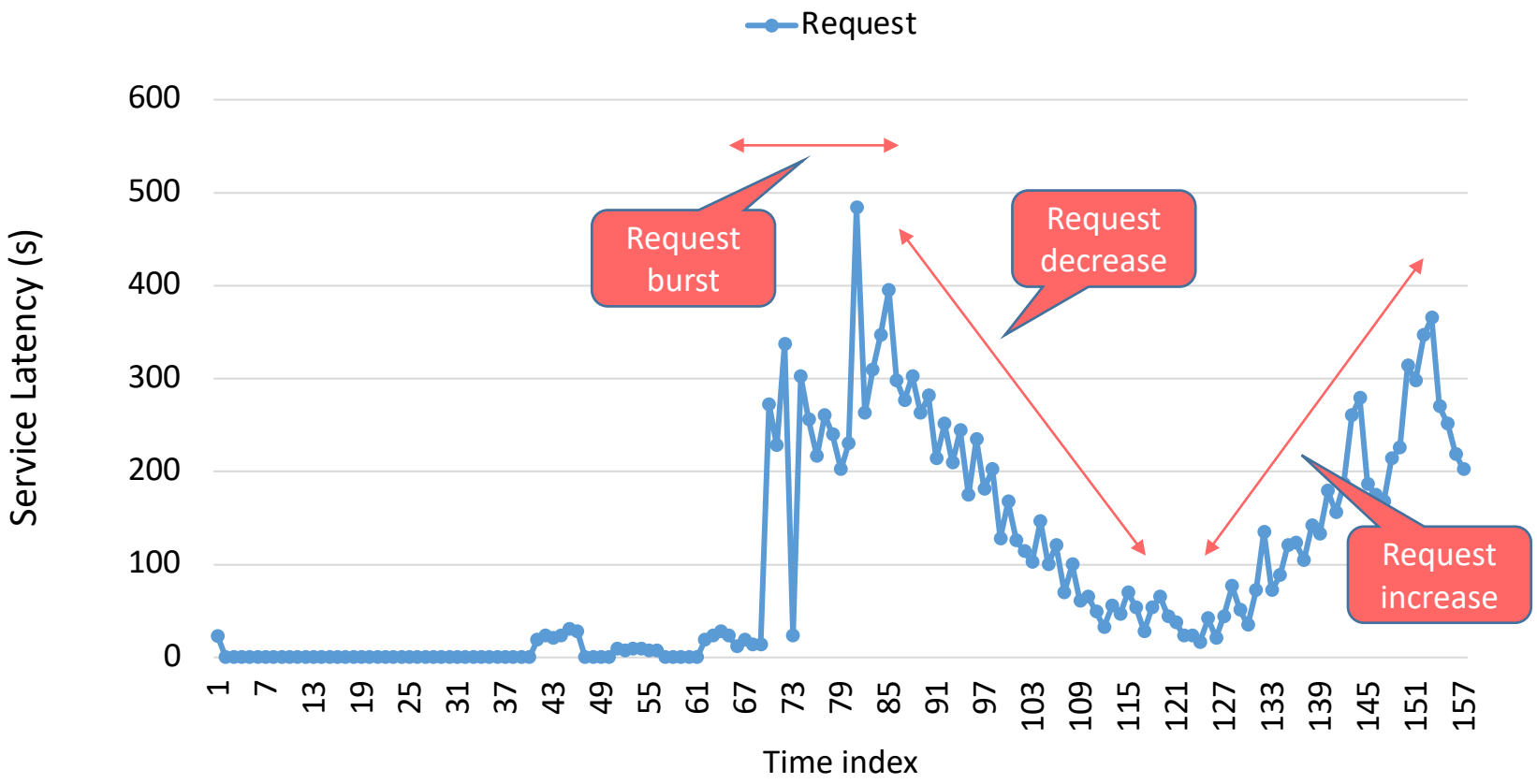


Web Application Latency

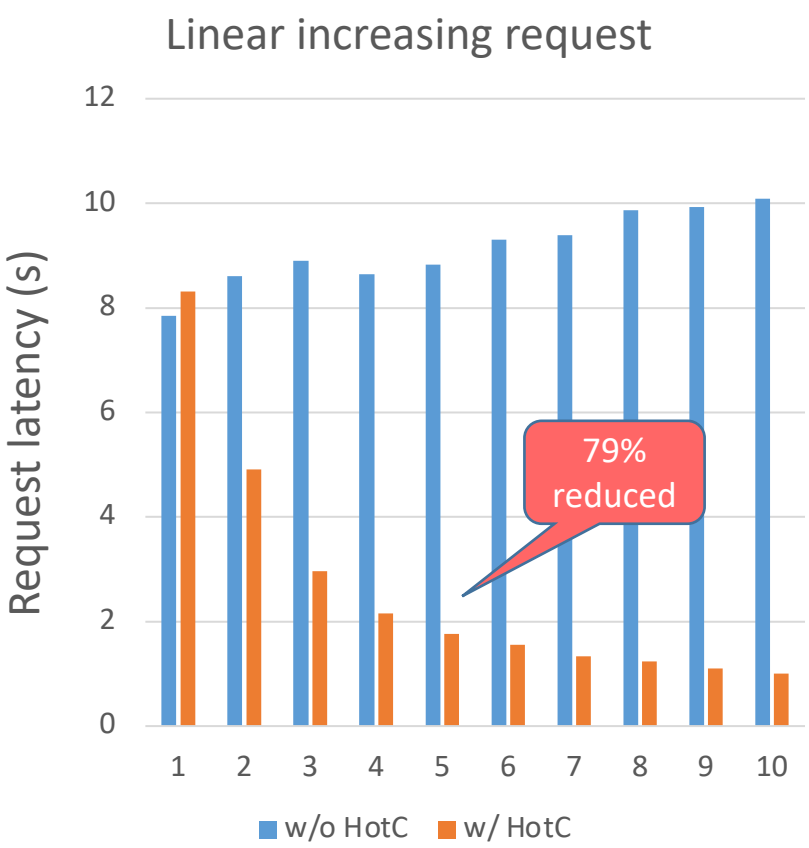


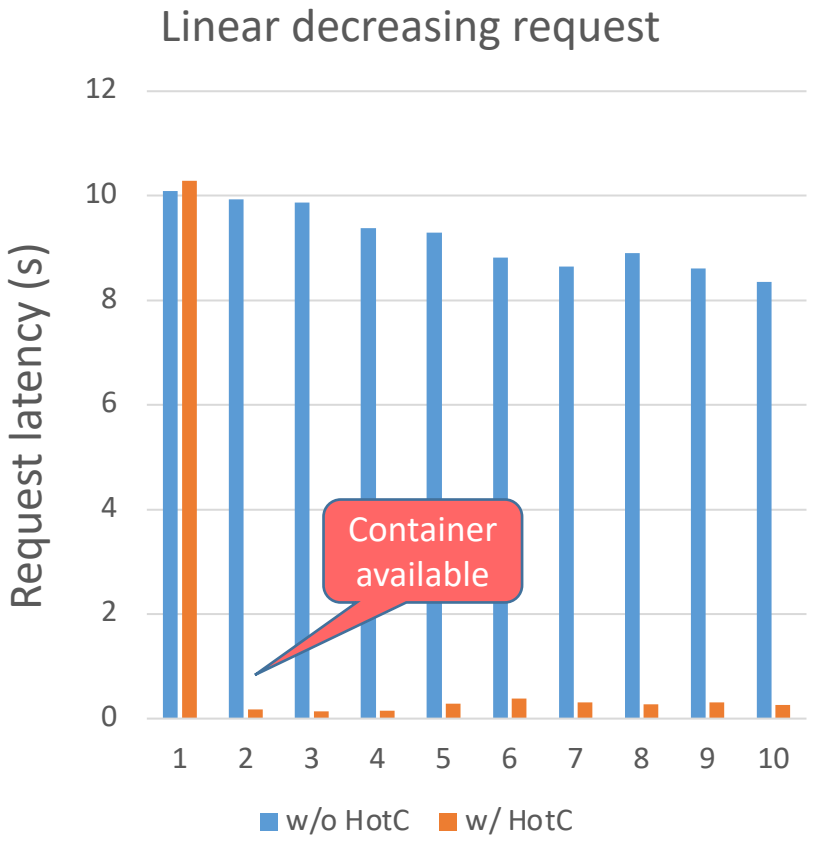


Youtube request statistics of Umass

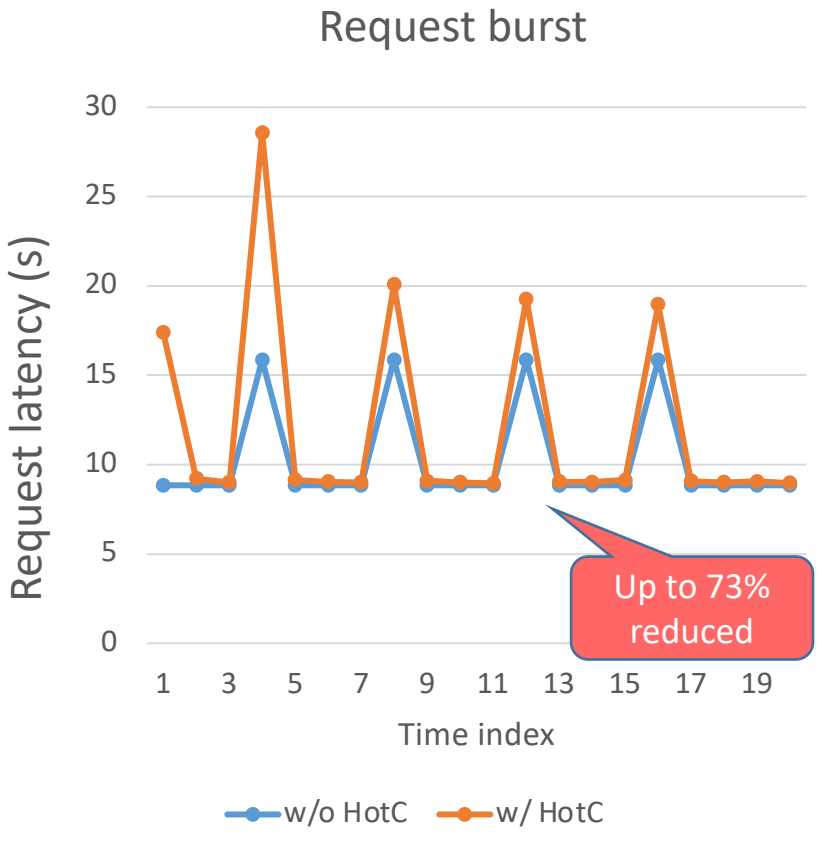


Linear network throughput

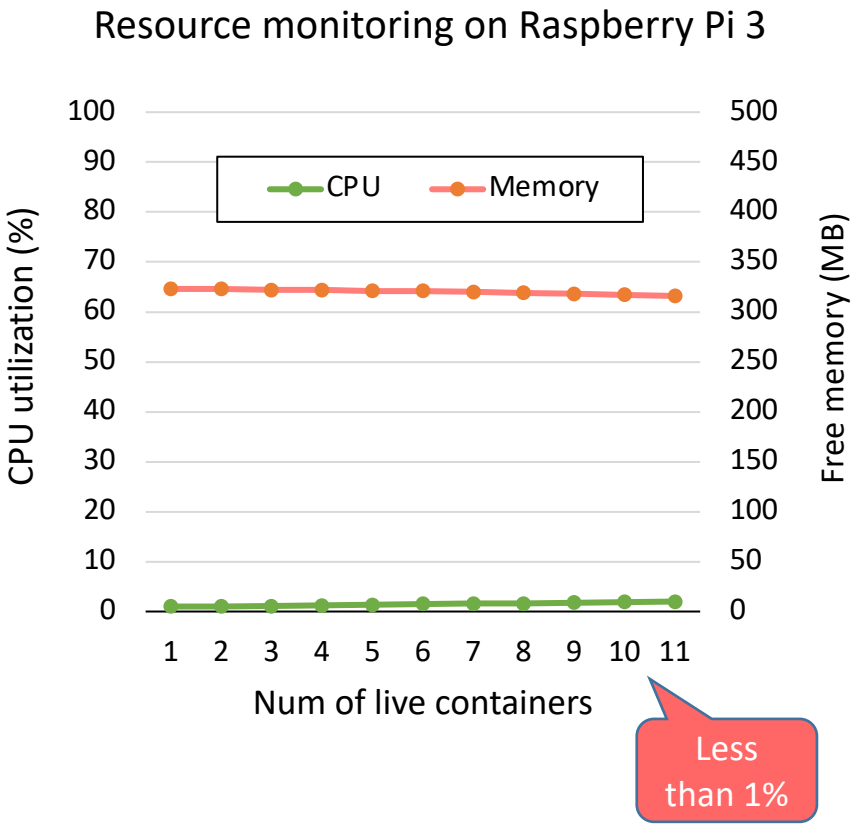


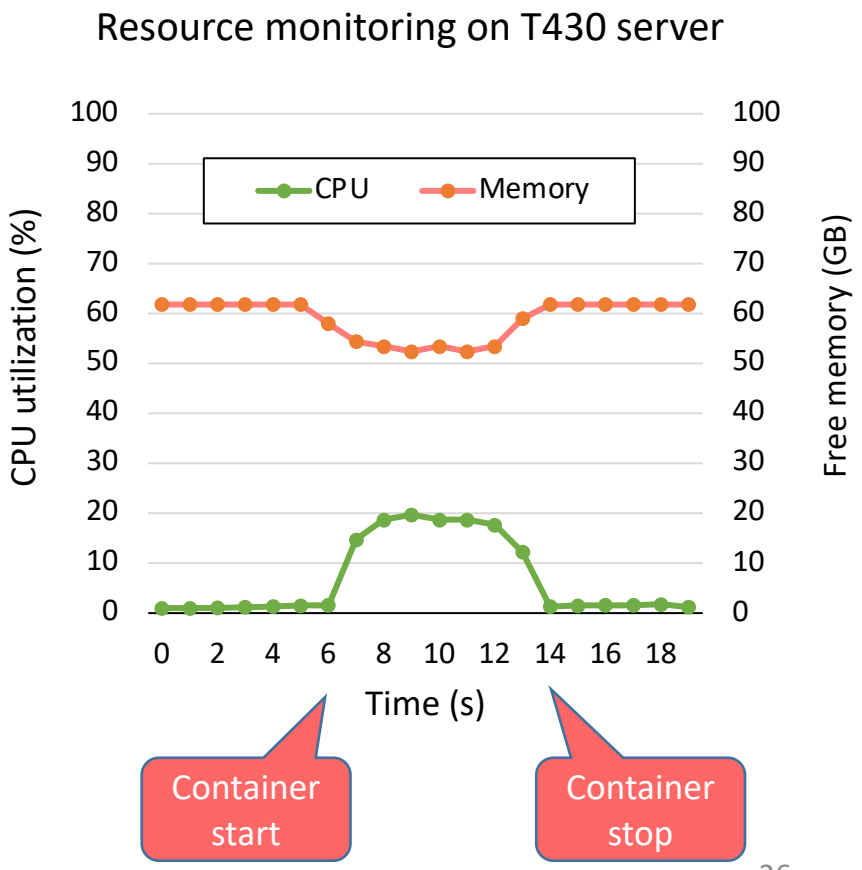


Exponential and burst request



The resource consumption of live containers





Conclusions

- **Background & Challenges**
  - ✓ Serverless computing is widely used but facing challenges such as long latency due to the container cold start.
- **HotC**
  - ✓ A container-based runtime management framework which leverages the lightweight containers to mitigate the cold start and improve network performance.
- **Results**
  - ✓ Our evaluation results show HotC introduces negligible overhead and can efficiently improve the performance of various applications in both cloud servers and edge devices.

Thank you !

Questions?