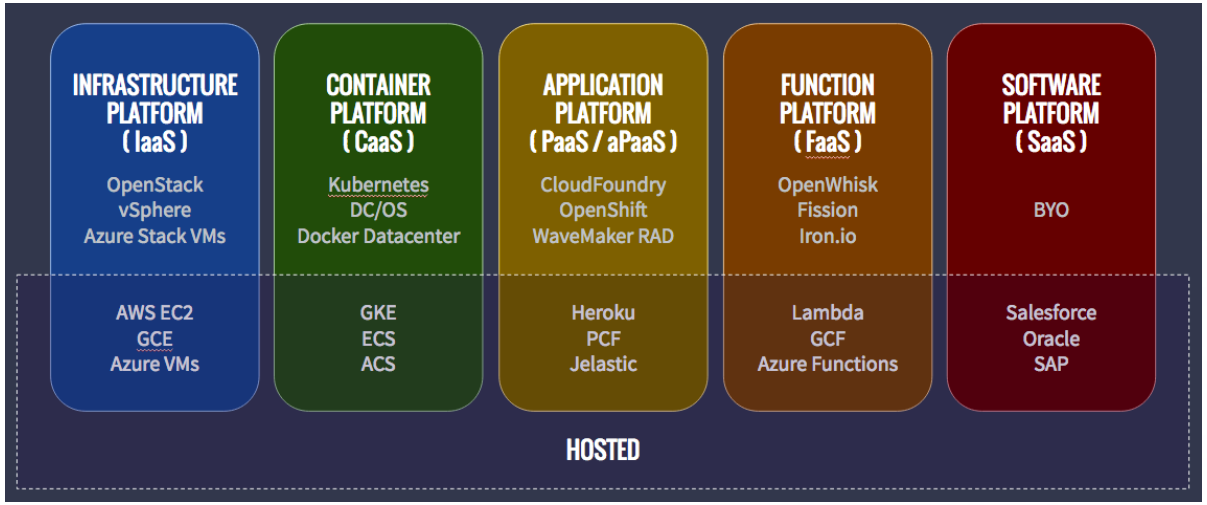
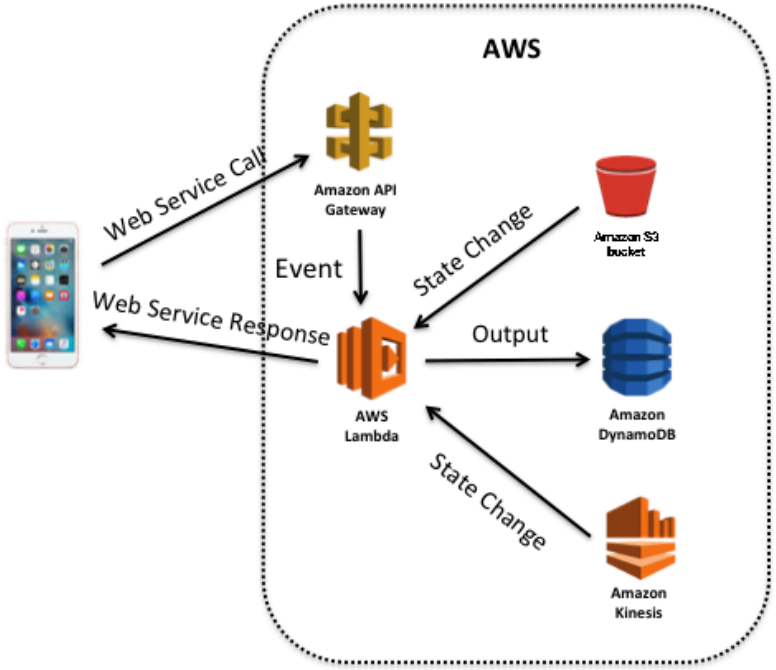
# Tackling Cold Start in Serverless Computing with Container Runtime Reusing

Kun Suo*, Yong Shi*, Xiaohua Xu*
Dazhao Cheng+, Wei Chen^

*Kennesaw State University,
+University of North Carolina at Charlotte, ^Nvidia

# Serverless & Its Workflow

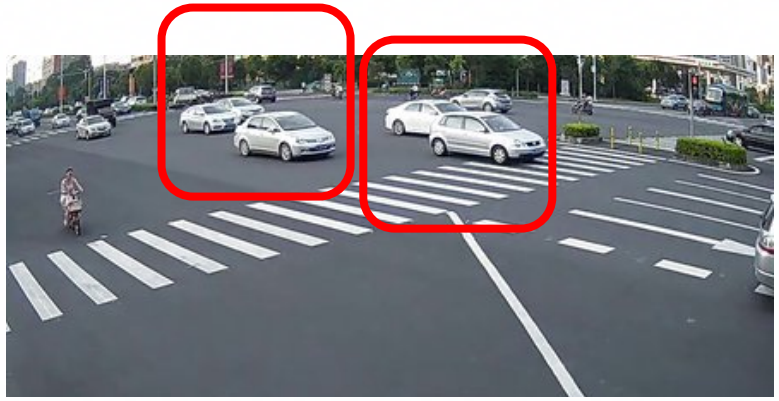


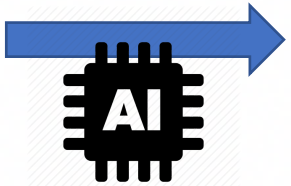
# Serverless at the Edge

idle

Event
Trigger



Run ML/
DL code




- Size
- Plate
- Speed
- ...

# Serverless Pro vs Con

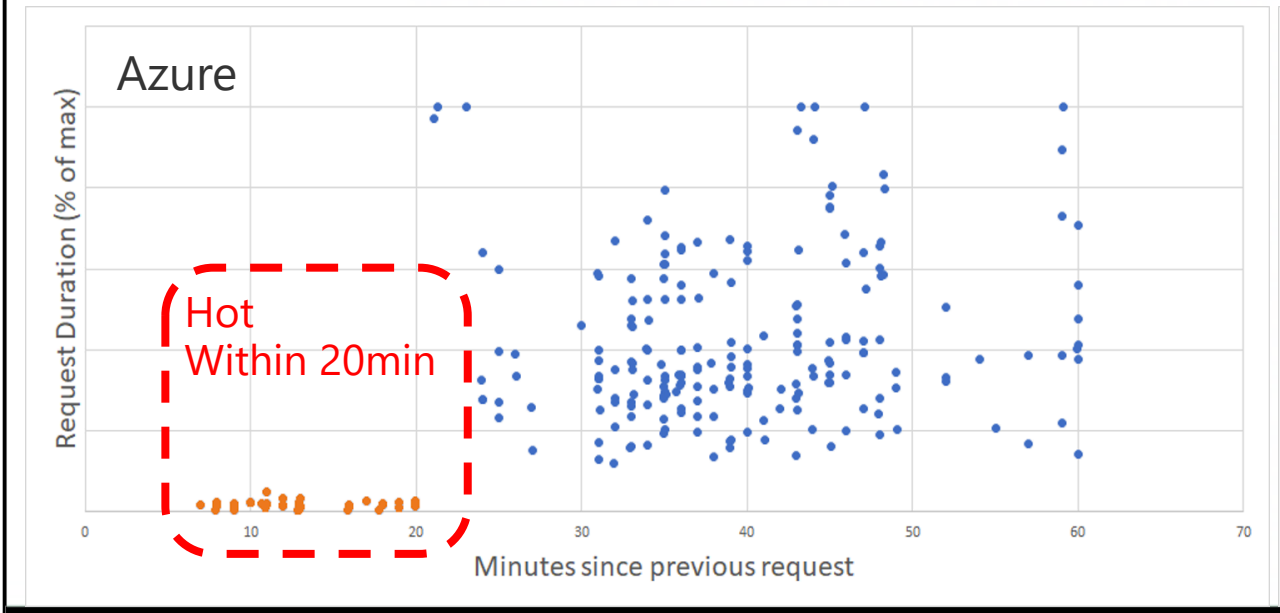
Benefits:

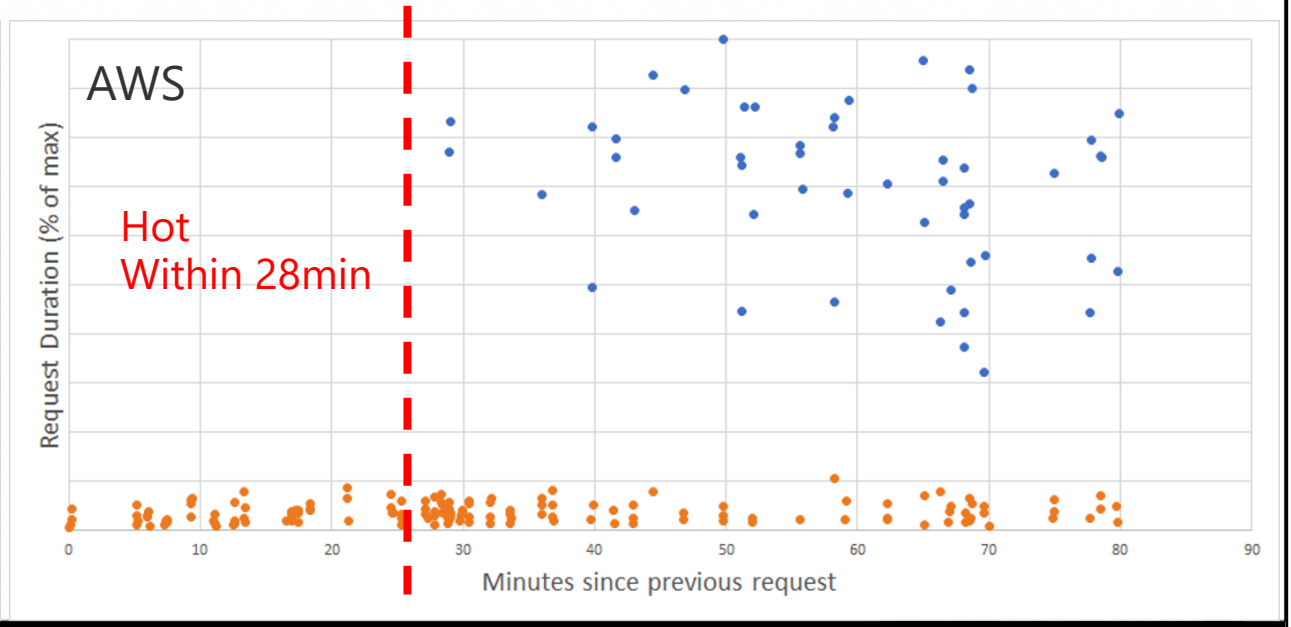
- Drop cost
- Fast Dev/deployment
- More security
- Microservice support
- Auto scale
- …

Drawbacks:

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

# Cold Start Impact

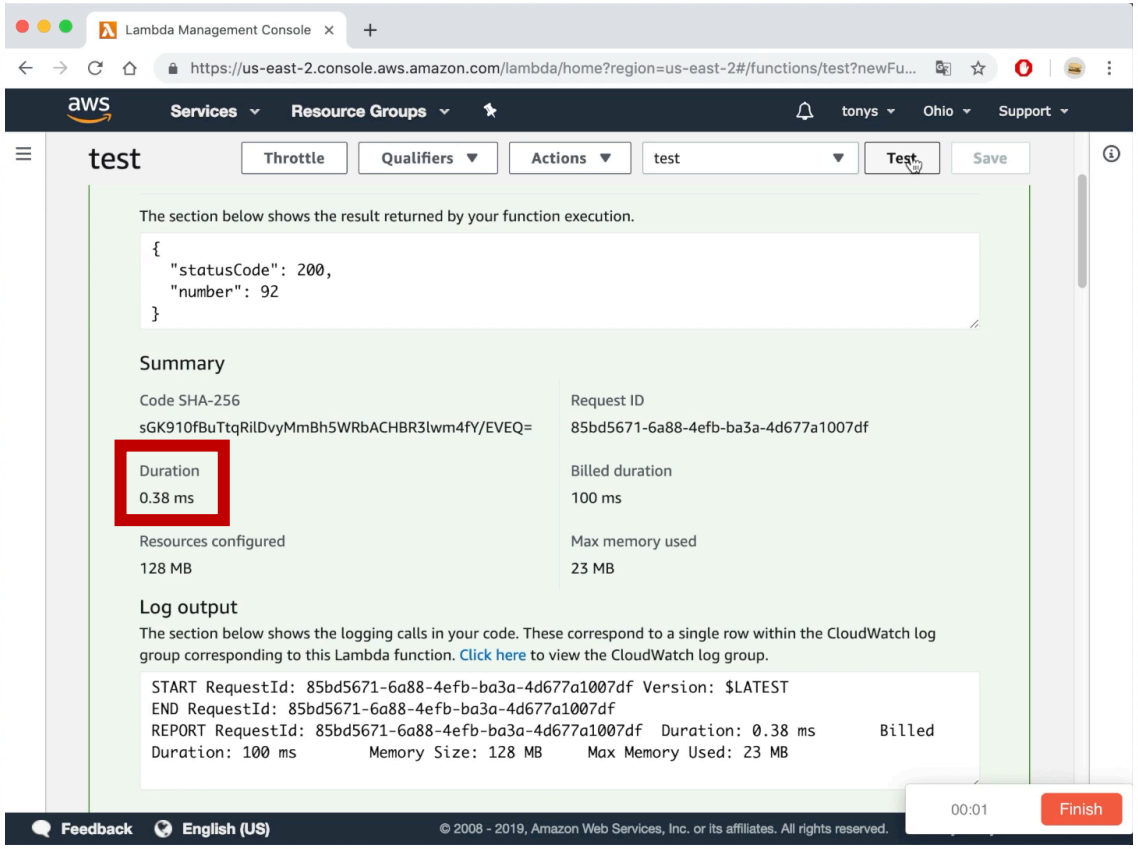



# AWS Lambda Cold Start Demo

Python:
http request return
random number

Measure the
request latency



# Containerized Application Characteristics

> 50%

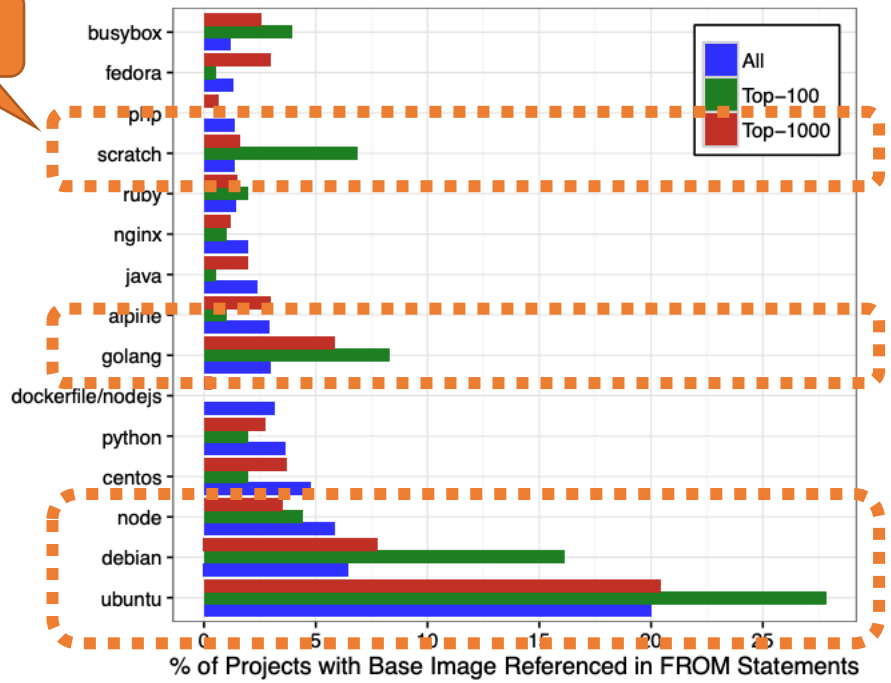
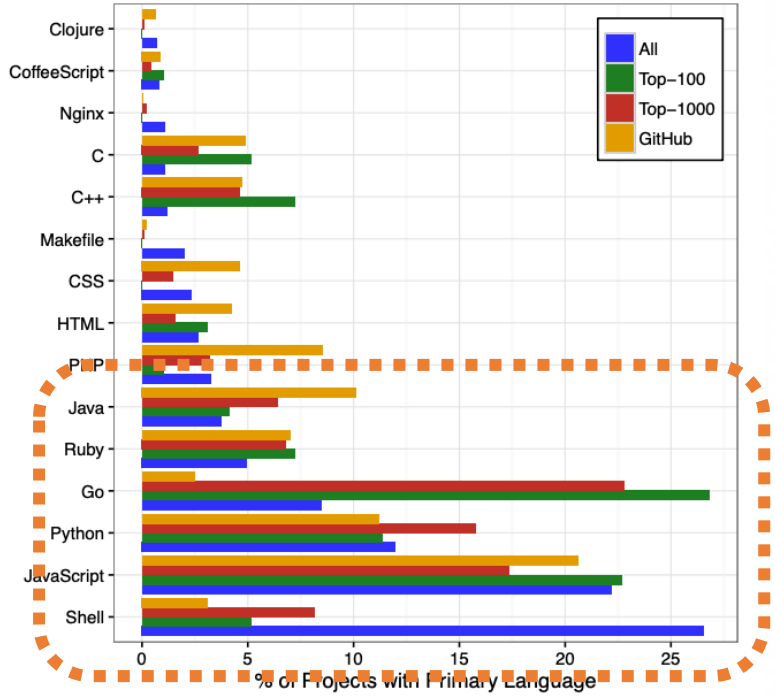



Fig. 5. Percentage of usage of 15 most commonly used base images




> 70%

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

https://peerj.com/preprints/2905.pdf

# Our Idea: HotC

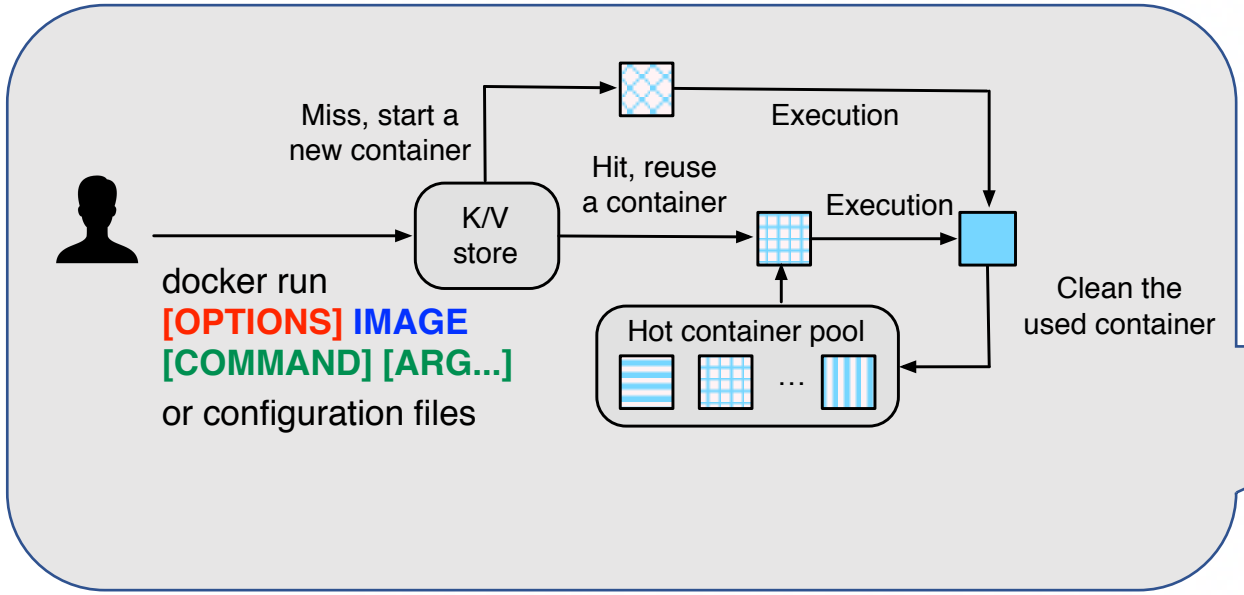
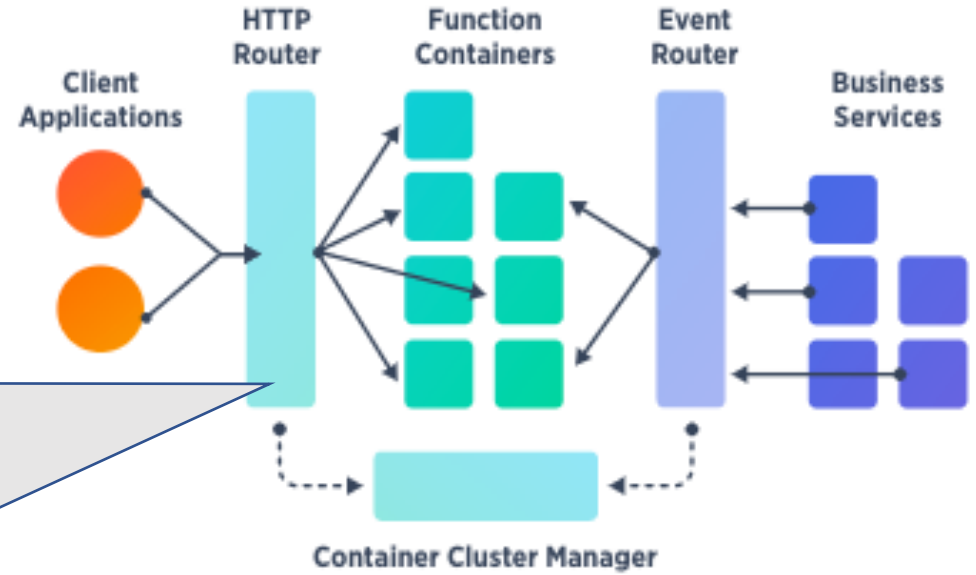



FIGURE 3: SERVERLESS ARCHITECTURE OVERVIEW



# Overhead of Keeping Live Containers

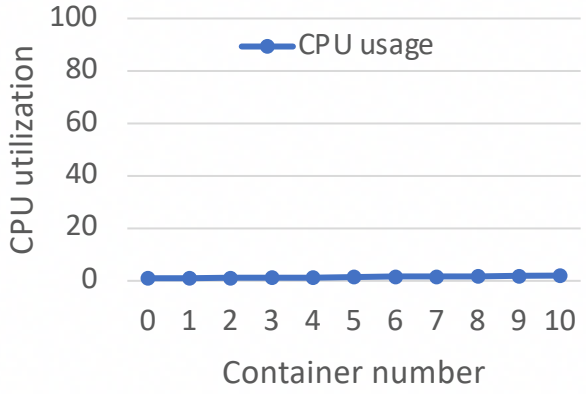
The CPU usage is almost 0% when no application executes.

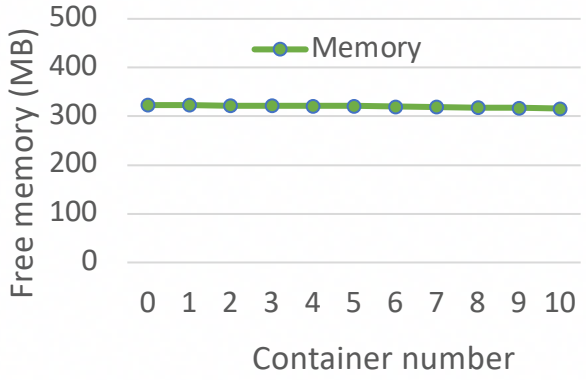
The memory usage increased by 0.7MB for each individual live container.

- Alpine Linux container only takes hundreds of KBs

Most of the resource consumption comes from the applications instead of the container itself

Overhead cannot be limited by precisely controlling the number of live containers



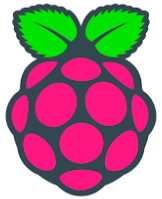
# Evaluation

DELL PowerEdge T430 server:

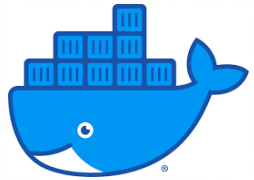
- Dual ten-core Intel Xeon E5-2640 2.6GHz processors, 64GB memory, Gigabit Network and a 2TB 7200RPM hard drive
- Ubuntu 16.04 and Linux kernel version 4.4.20

Raspberry Pi 3:

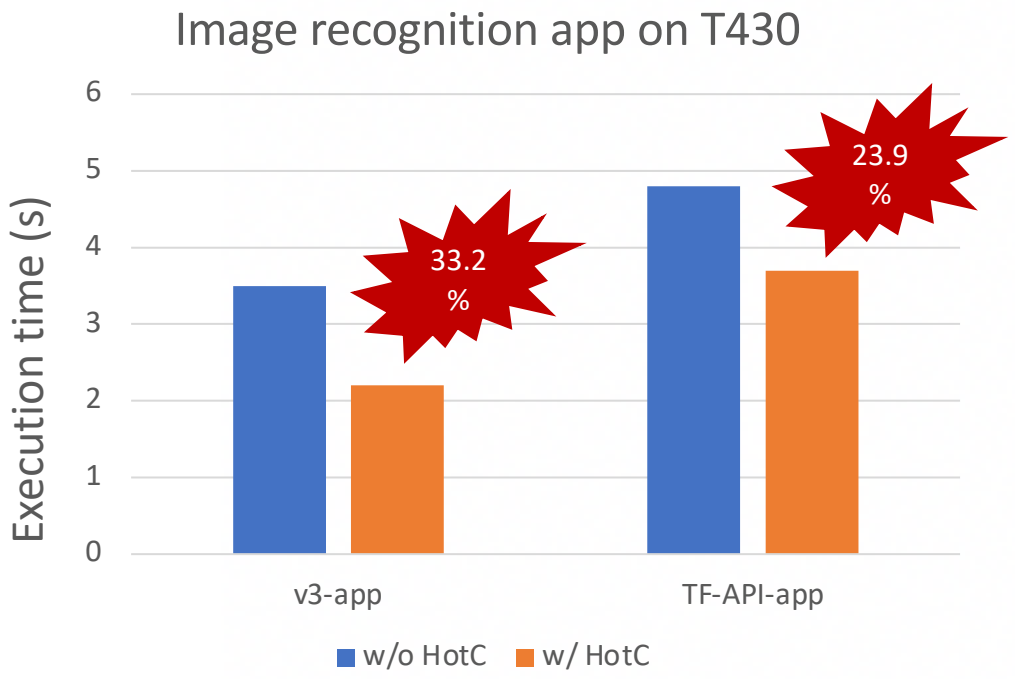
- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU, 1GB memory and 32GB storage
- Linux Raspberrypi 4.14

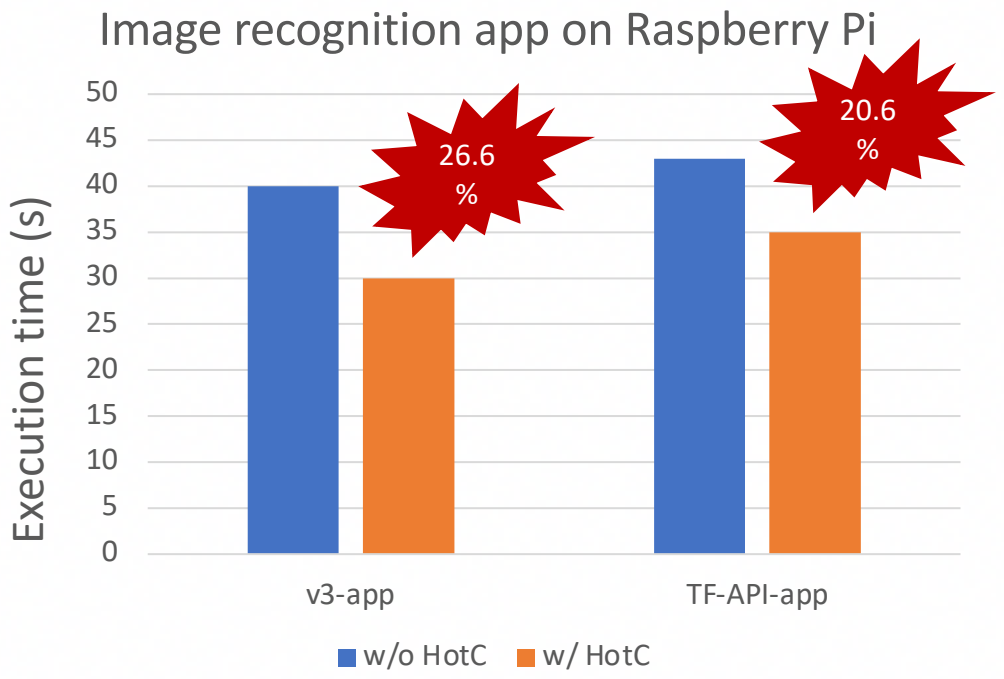


Docker 1.17, OpenFaaS 0.8.5



# Preliminary Result



# Conclusion

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.

Our evaluation results show HotC introduces negligible overhead and can efficiently improve the performance of various applications in both cloud servers and edge devices.

# Q & A

Email: ksuo@kennesaw.edu

Website: https://kevinsuo.github.io/

# Alibaba cloud optimization for container cold start

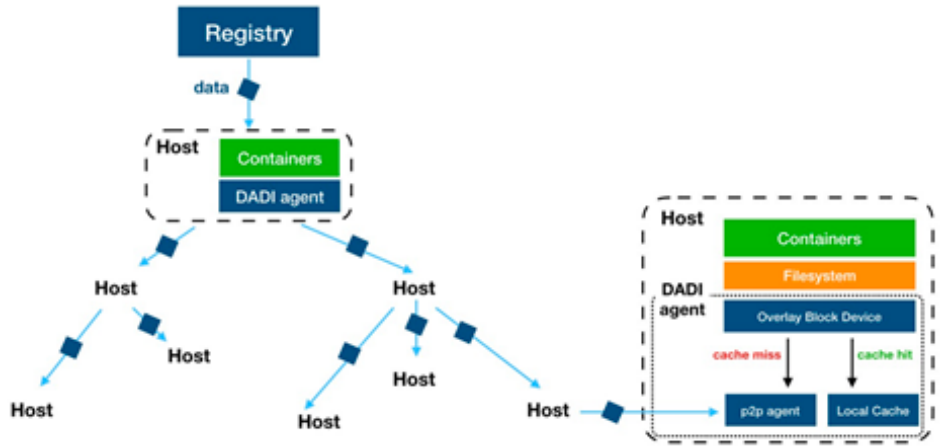
1, New image format

No need to download and unzip the full image

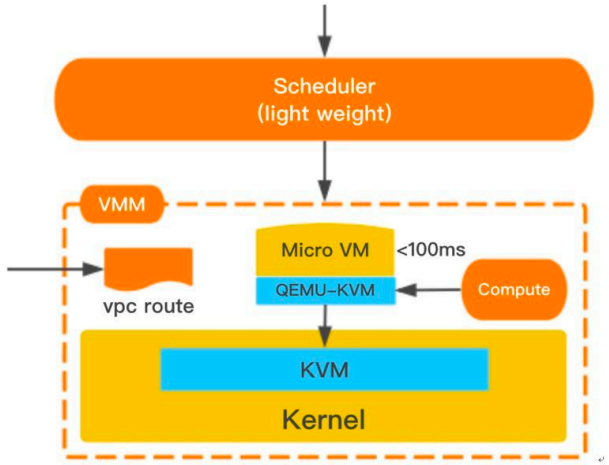
2, P2P network pair for data distribution

Reduce the overall load on a single point Registry

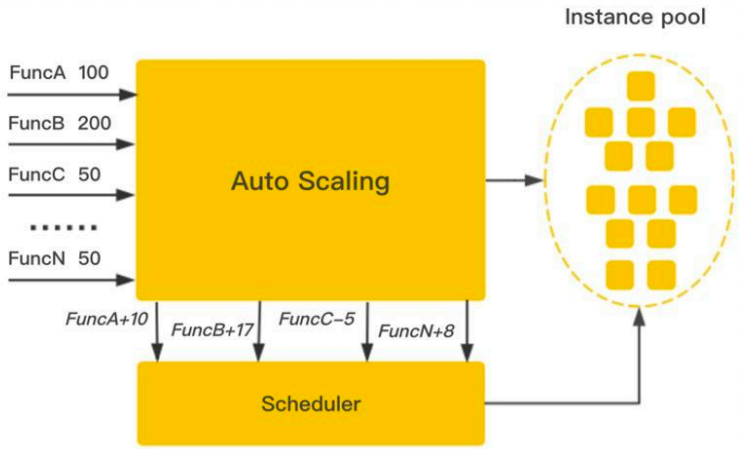
3, Efficient decompression algorithm



# Tencent optimization for container cold start




1, Deploys their functions based on lightweight virtualization named MicroVm




2, New scheduling policies for active functions

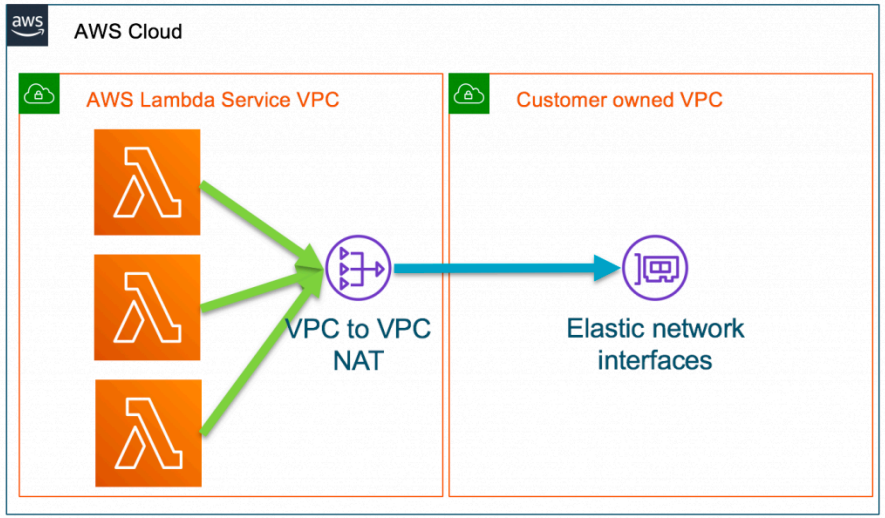
real-time auto scale system which can expand or contract in second-level based on the system metrics and monitoring data

# AWS optimization for container cold start

1, Keep-alive policy that retains the resources in memory for 10x minutes after a function execution

P.S. OpenWhisk also uses a 10-minute period

2, AWS Hyperplane create interfaces in advance for each serverless application instead of creating Virtual Private Cloud (VPC) for each function during execution



# Azure optimization for container cold start

1, Keep-alive value for each user's workload, according to its actual invocation frequency and pattern

2, Enables the provider in many cases to pre-warm a function execution just before its invocation happens