Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider

Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke,

Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini

USENIX ATC'20

2022. 05. 31

Presented by Yejin Han

yj0225@dankook.ac.kr

What is Serverless?

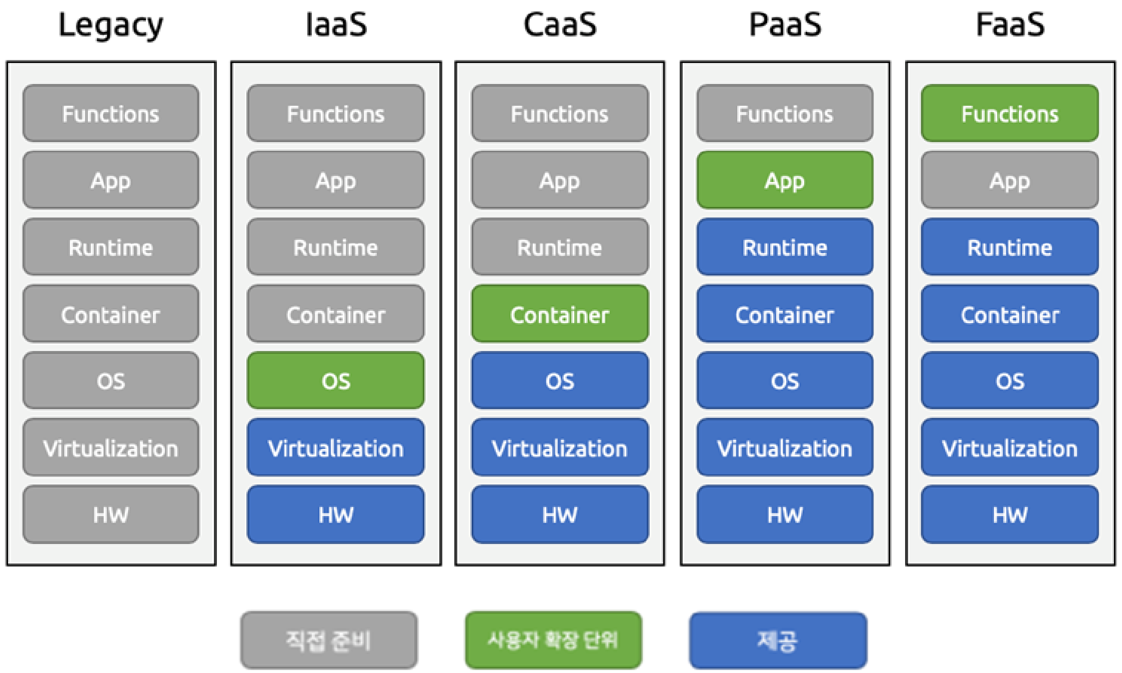
- Very attractive abtraction to users:
  - Pay for only usage
  - No worry about servers
    - Provisioning, reserving, configuring, patching, managing

- Most popular offering: Function-as-a-Service (FaaS)
  - Intuitive, event-based interface for developing cloud-based applications
  - Users upload the code, provider enables a handle for the code to run

serverless

FaaS

- FaaS (Function as a Service): users do not provision or configure resources
  - Function: basic unit of deployment
  - Application: consists of multiple functions, unit of resource allocation



(source: https://futurecreator.github.io/2019/03/14/serverless-architecture/)

Put yourself in provider's shoes..

- Challenges are:
  - You do worry about servers
    - Provisioning, scaling, allocating, securing, isolating
  - Optimize resource usage
  - Provide the illusion of always-available resources at the lowest cost
    - Fast function invocations without cold starts

Cloud Providers

Cold Starts

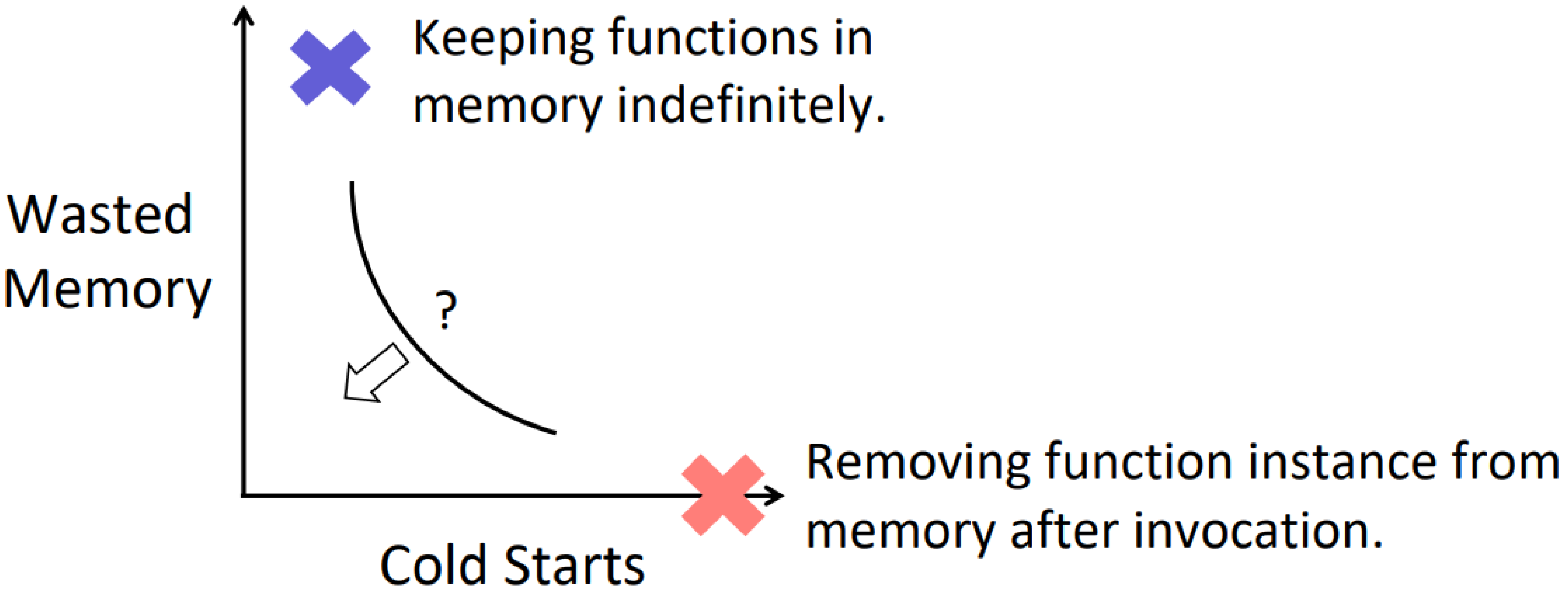
- Providers want to provide high function performance at the lowest cost
  - Function execution requires the needed code in memory
    - Warm start (from memory) vs **cold start** (from storage)
  - Keeping all resources in memory is prohibitively expensive
  - Functions have varying resource needs and invocation frequencies

☞ ***a comprehensive characterization of the FaaS workload is needed!!***

(Source: https://medium.com/ssense-tech/the-trade-offs-with-serverless-functions-71ea860d446d)

Problem: Cold Starts and Resource Wastage

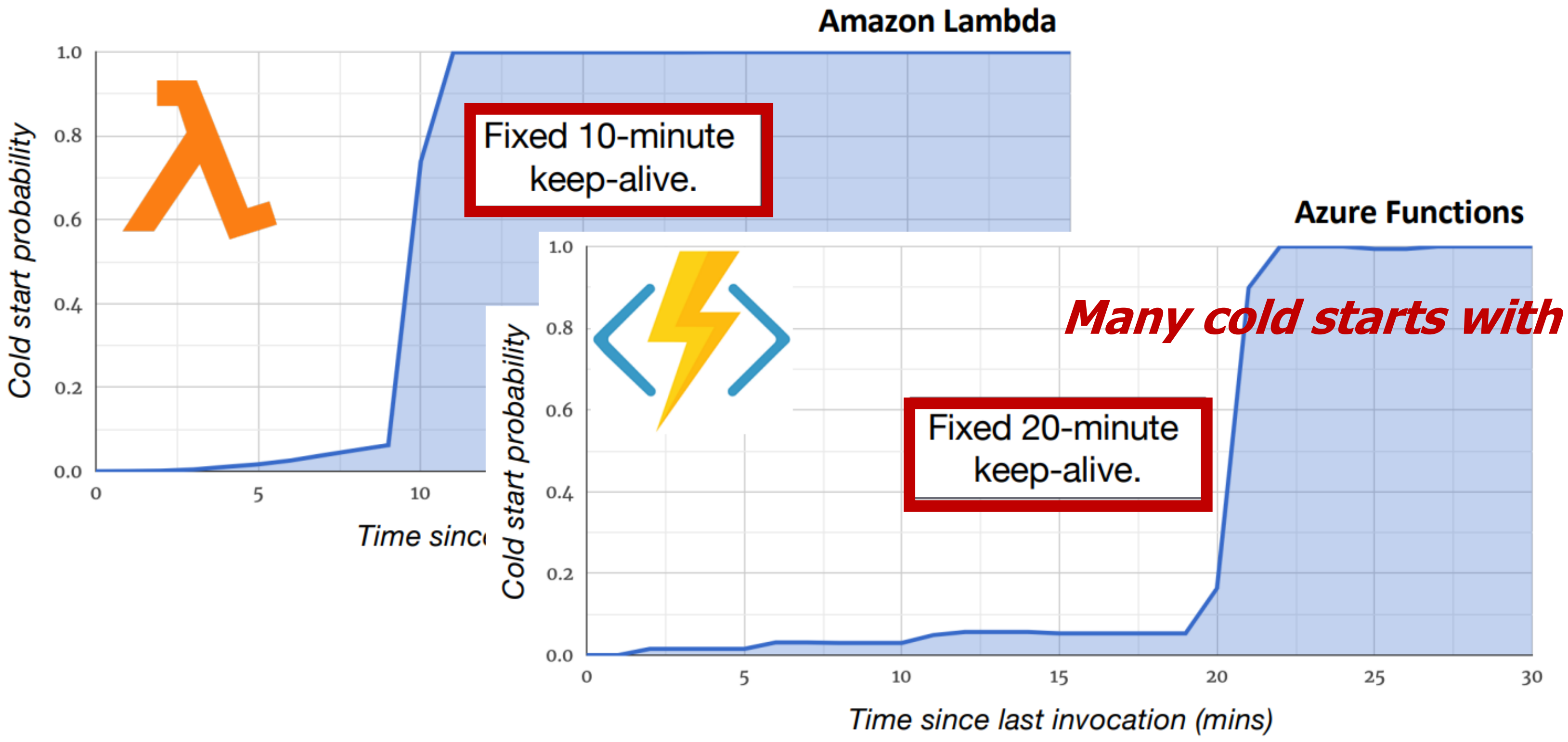
- Trade-off between reducing cold starts and the resources they need



(Source: https://www.usenix.org/system/files/atc20-paper593-slides-shahrad.pdf)

Problem: Fixed keep-alive policy

- Application instances are kept loaded in memory for a fixed time
  - It does not consider the wide variety of application behaviors



Many cold starts with wasting resources

(Source: https://www.usenix.org/system/files/atc20-paper593-slides-shahrad.pdf)

FaaS Workload

2 weeks of all invocations to Azure Functions in July 2019
First characterization of the workload of a large serverless provider

- Functions and applications
  - 54% applications only have one function and 95% applications have at most 10 functions

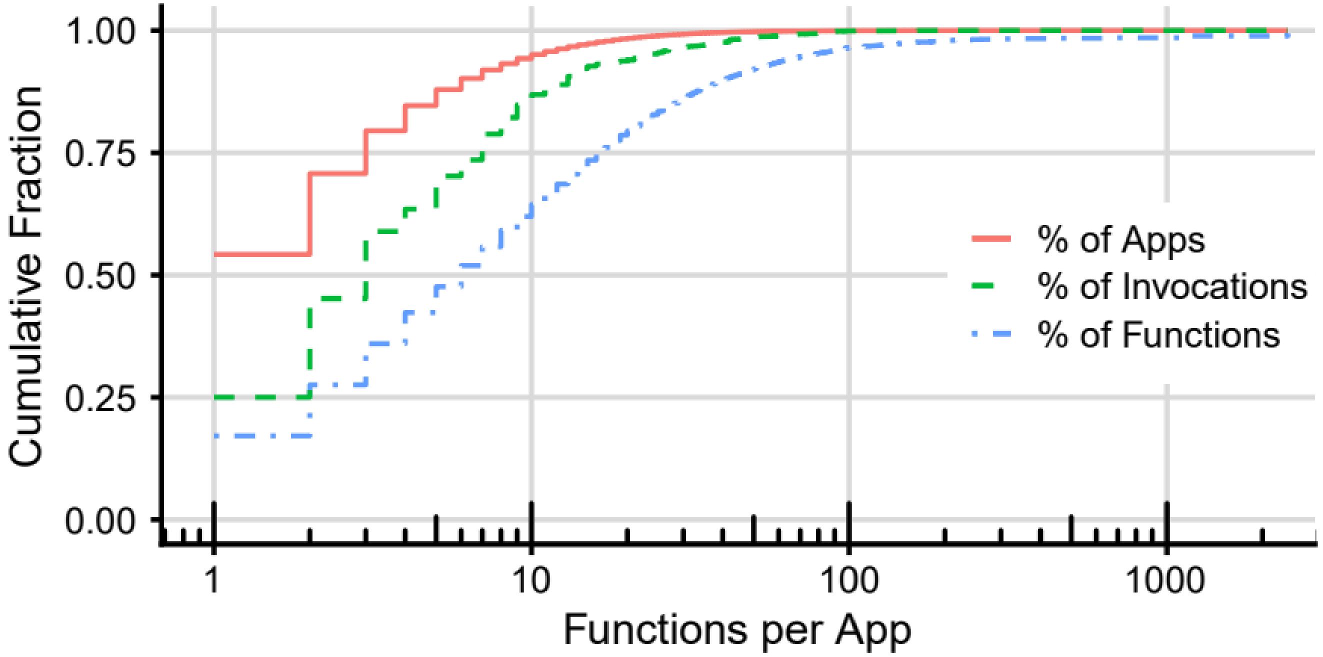


Figure 1: Distribution of the number of functions per app.

The number of functions in an applications is not useful in resource management

FaaS Workload

2 weeks of all invocations to Azure Functions in July 2019
First characterization of the workload of a large serverless provider

- Triggers and applications

Trigger	%Functions	%Invocations
HTTP	55.0	35.9
Queue	15.2	33.5
Event	2.2	24.7
Orchestration	6.9	2.3
Timer	15.6	2.0
Storage	2.8	0.7
Others	2.2	1.0

Figure 2: Functions and invocations per trigger type.

Trigger Type	% Apps
HTTP (H)	64.07
Timer (T)	29.15
Queue (Q)	23.70
Storage (S)	6.83
Event (E)	5.79
Orchestration (O)	3.09
Others (o)	6.28

(a) Apps with ≥ 1 of each trigger.

Trigger Types	Fraction of Apps (%)	Cum. Frac. (%)
H	43.27	43.27
T	13.36	56.63
Q	9.47	66.10
HT	4.59	70.69
HQ	4.22	74.92
E	3.01	77.92
S	2.80	80.73
TQ	2.57	83.30
HTQ	2.48	85.78
Ho	1.69	87.48
HS	1.05	88.53
HO	1.03	89.56

(b) Popular trigger combinations.

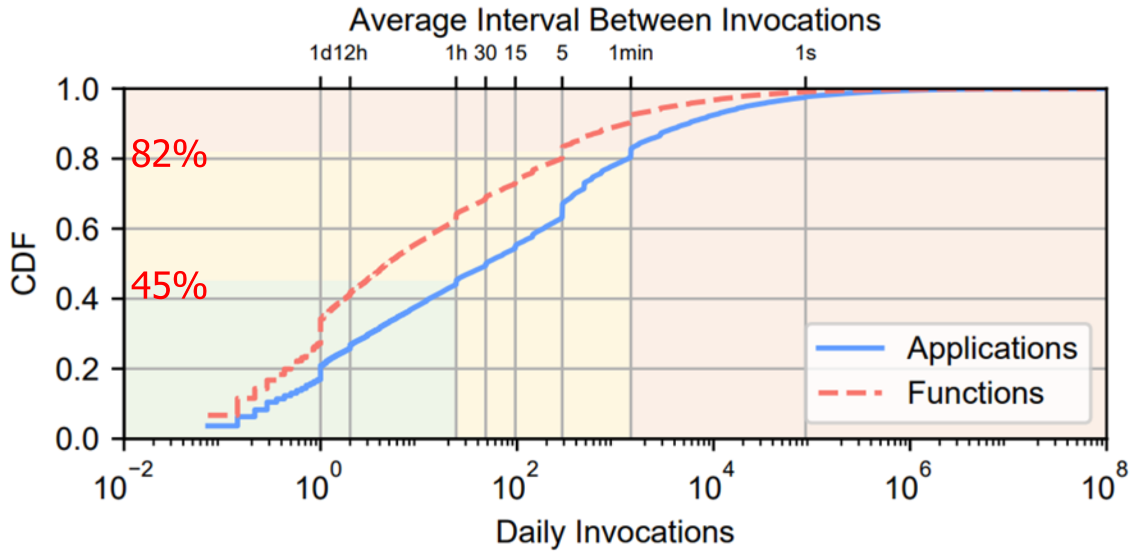
Figure 3: Trigger types in applications.

For predicting invocations: timers are very predictable, other applications with no timers aren't

FaaS Workload

2 weeks of all invocations to Azure Functions in July 2019
First characterization of the workload of a large serverless provider

- Invocations per application
  - Daily invocations varies by over 8 orders of magnitude
  - Vast majority of functions are invoked very infrequently



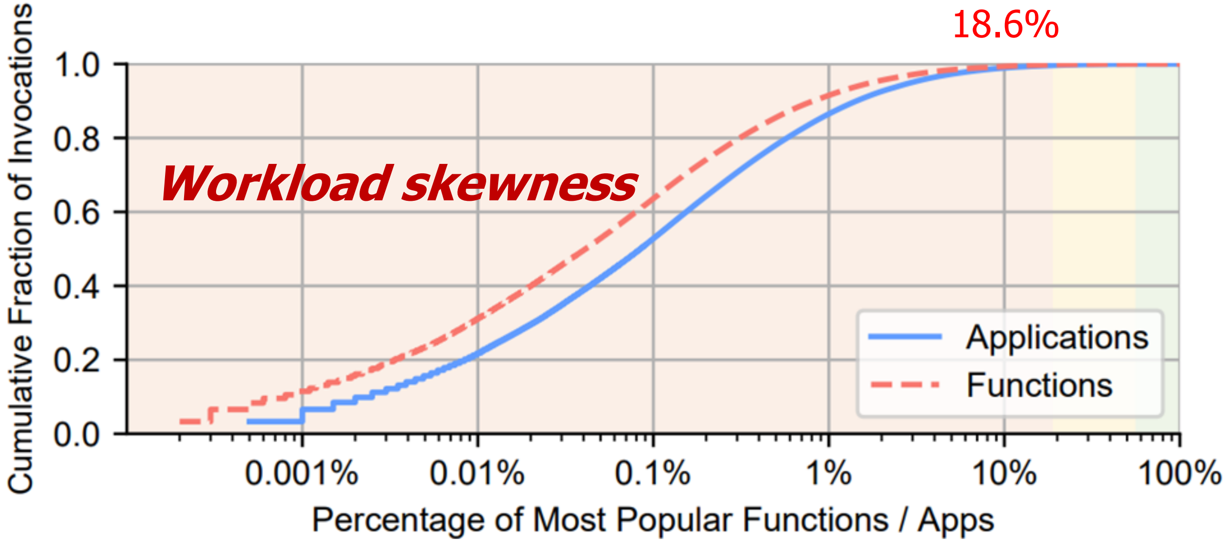


Figure 5: Invocations per application and per function for a representative sample of the dataset.

From IAT (Inter-arrival time) of invocation, we can pre-warming the application

FaaS Workload

2 weeks of all invocations to Azure Functions in July 2019
First characterization of the workload of a large serverless provider

- Function execution times
  - Same sale as the cold start times
  - Execution times are short

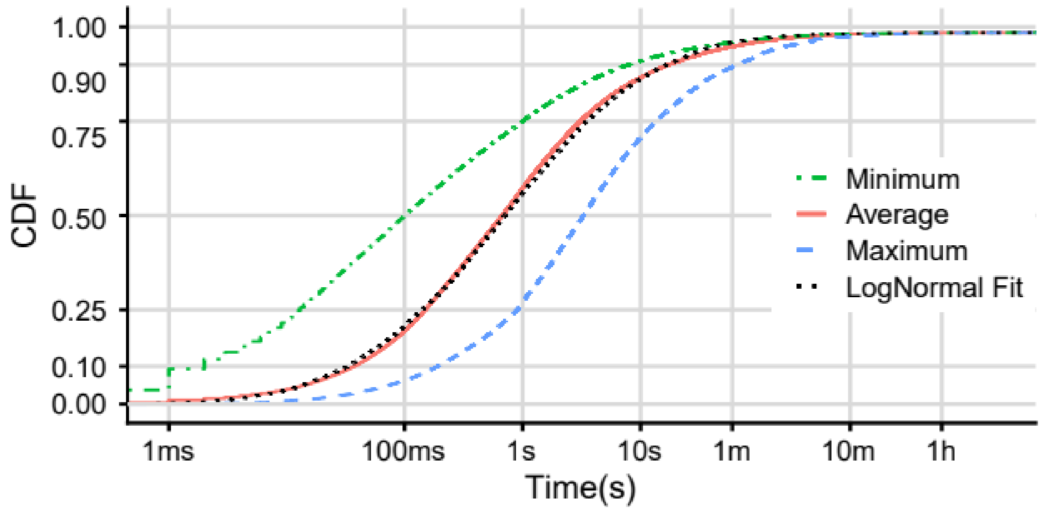


Figure 7: Distribution of function execution times. Min, avg, and max are separate CDFs, and use independent sorting.

Optimizing cold starts is important

- Memory Usage
  - Applications tend to remain memory resident for longer

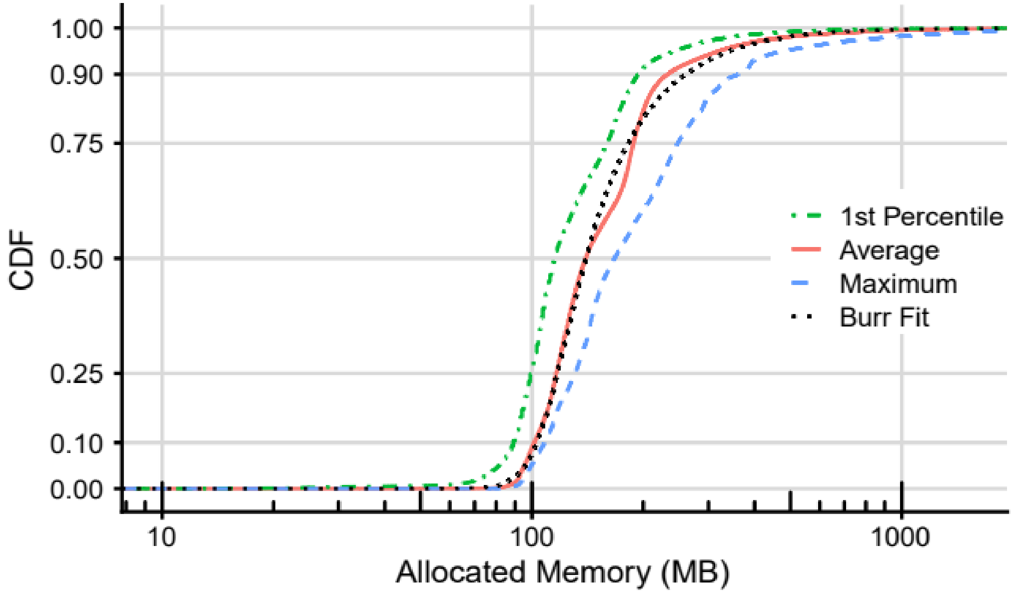
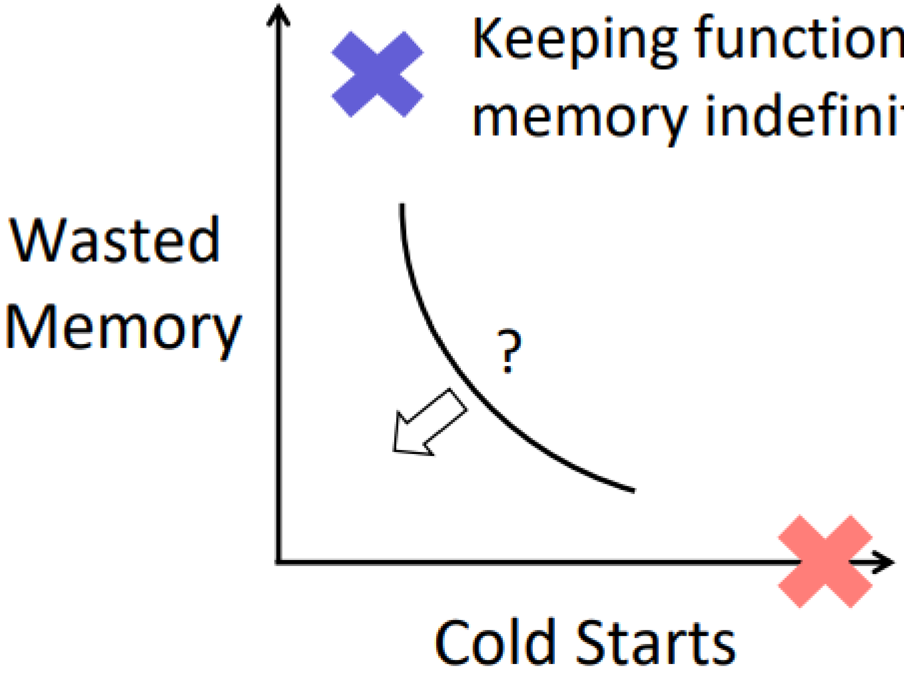


Figure 8: Distribution of allocated memory per application.

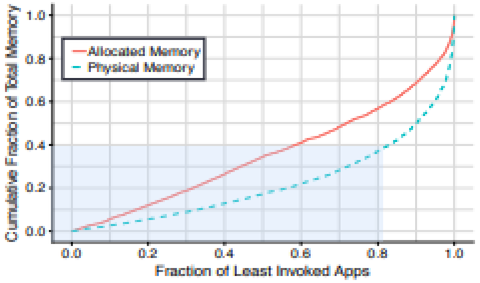
Memory is important in keep-alive decision for FaaS

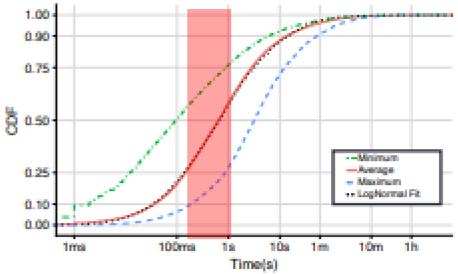
Cold Starts and Resource Wastage

- How to manage cold starts in FaaS?



Keeping functions in memory indefinitely.





Removing function instance from memory after invocation.

Hybrid Histogram Policy

- The policy of hybrid histogram
  - Pre-warming window / keep-alive window

* ARIMA: Autoregressive Integrated Moving Average

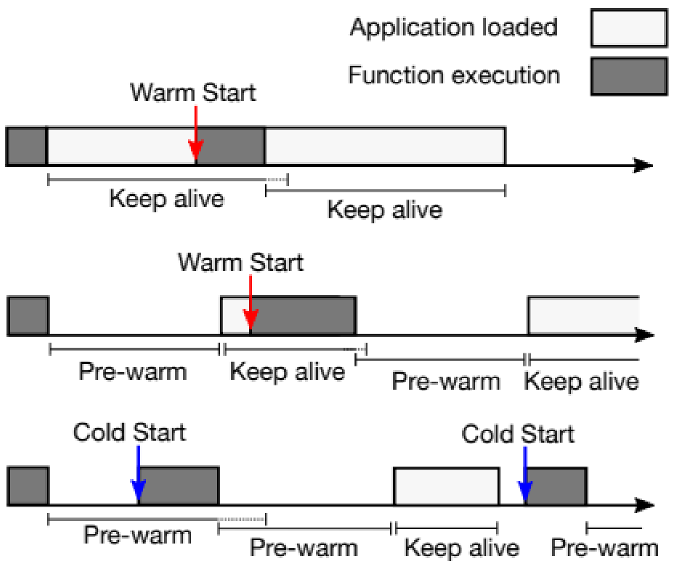


Figure 9: Timelines showing a warm start with keep alives and no pre-warming (top); a warm start following a pre-warm (middle); and two cold starts, before a pre-warm, and after a keep alive (bottom).

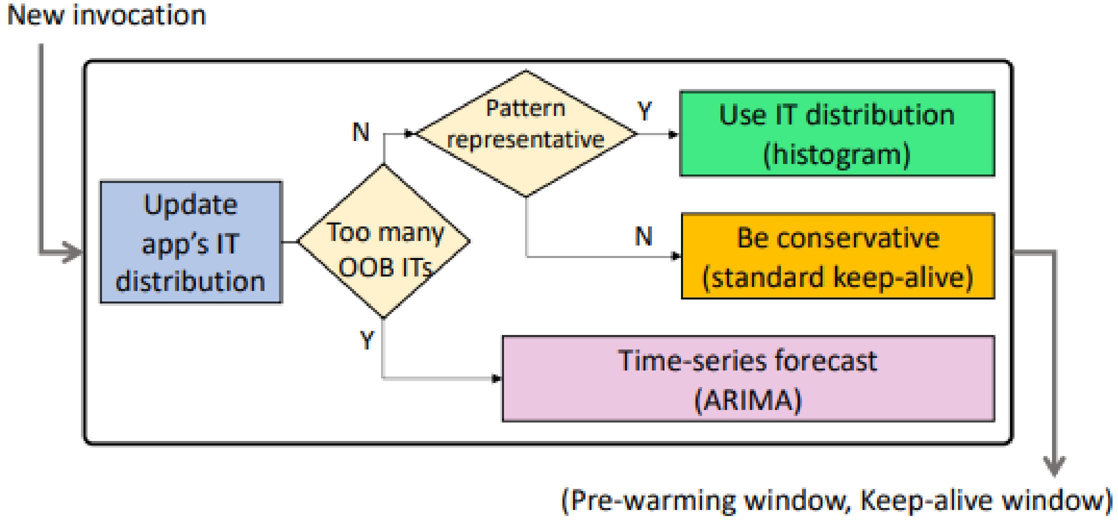


Figure 10: Overview of the hybrid histogram policy.

Hybrid Histogram Policy

- Three main components of the policy
  - A range-limited histogram
  - Standard keep-alive approach
  - Time-series forecast component

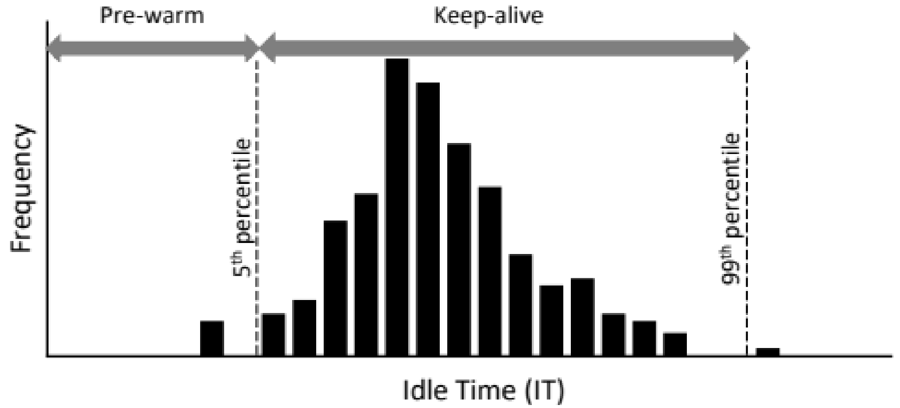


Figure 11: Example application idle time (IT) distribution used to select pre-warming times and keep-alive windows.

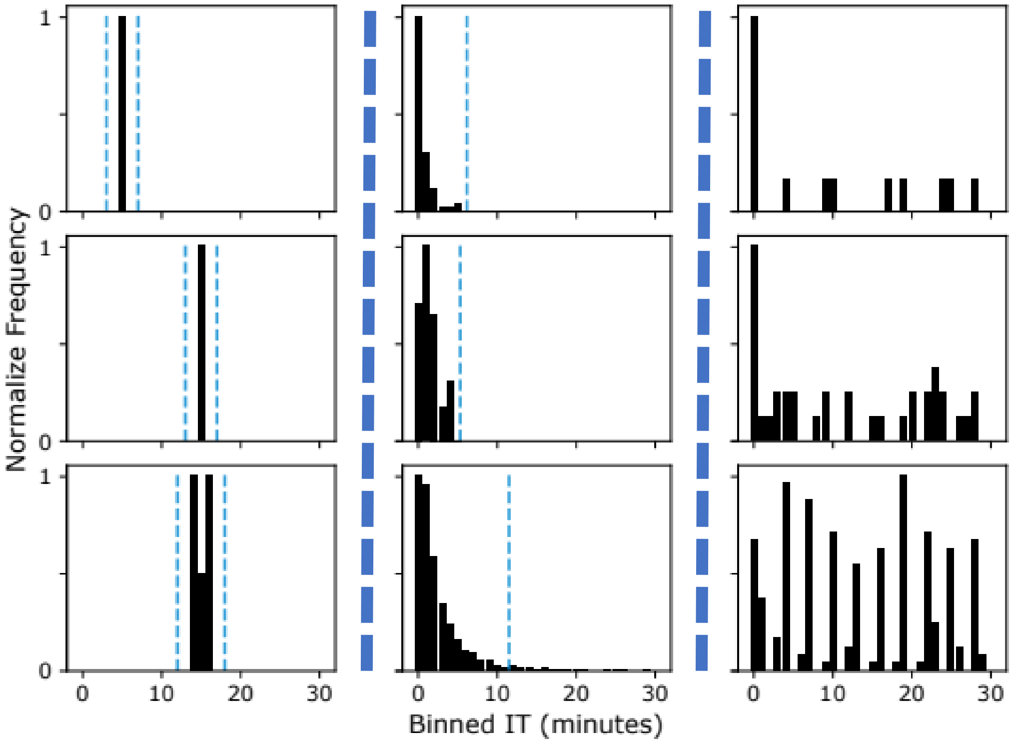


Figure 12: Nine normalized IT distributions from real FaaS workloads over a week.

Implementation in Apache OpenWhisk

- Apache OpenWhisk
  - Open-sourced industry-grade (IBM Cloud Functions)
  - Functions run in docker containers
  - Uses 10-minute fixed keep-alive
  - Built a distributed setup with 19 VMs

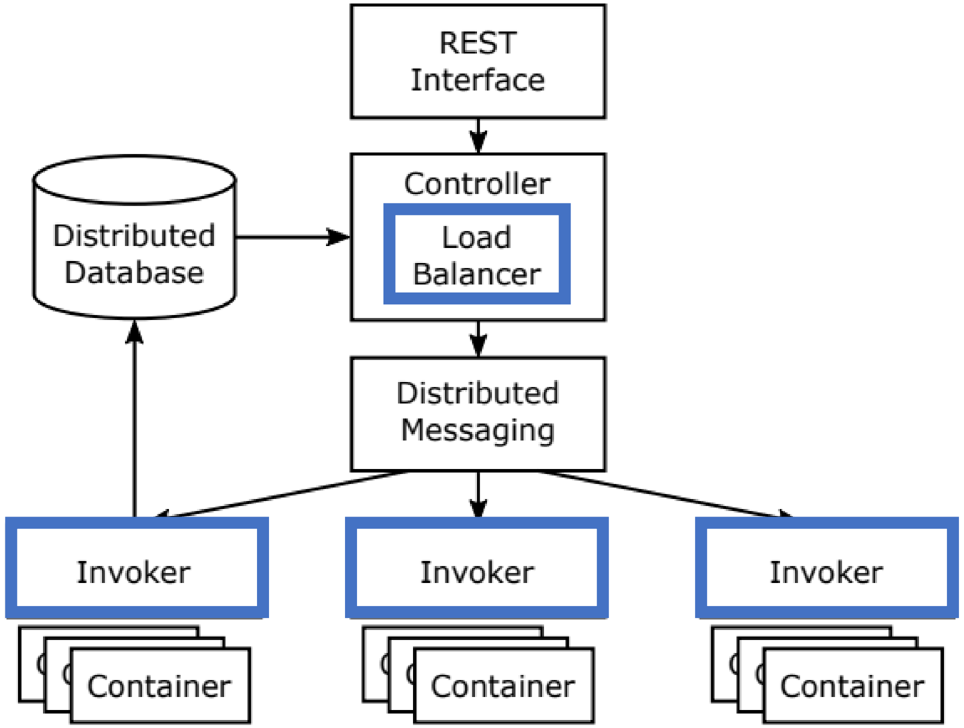


Figure 13: OpenWhisk architecture.

Evaluation

- Experiment Setup
  - Simulation
    - Workload: all function invocations across Azure between July 15th and Jylu 20th
    - Generates an array of invocation times and infer whether each invocation would be a cold start
  - Real experiments
    - Workload: scaled-down version of the trace
    - 19VM: One VM with 8 cores, 8GB of memory hosts / 18 VM with 2 cores, 4 GB of memory

☞ ***Simulation and experimental results show the same trends in both cold start and memory consumption behaviors***

Evaluation: simulation

- Fixed keep-alive policy

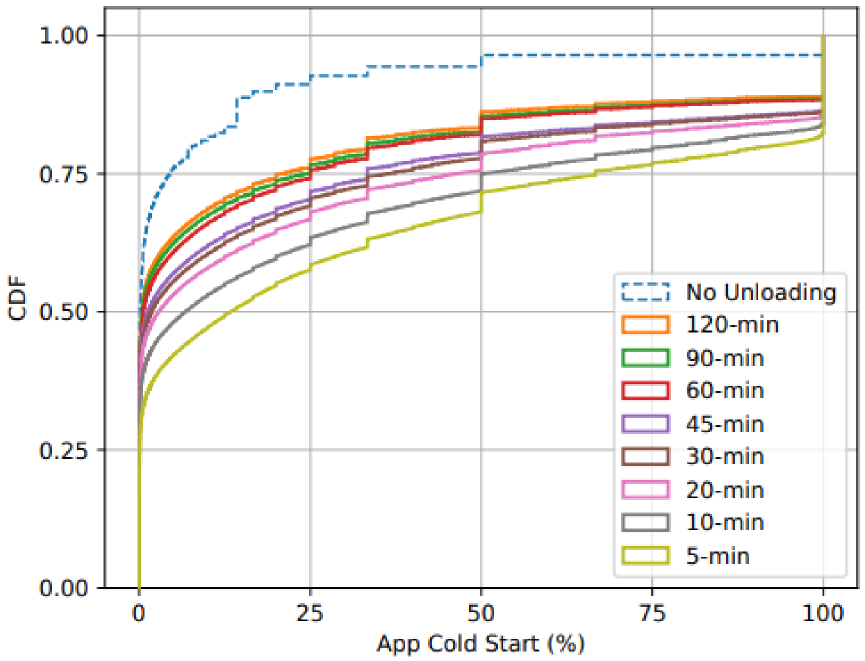


Figure 14: Cold start behavior of the fixed keep-alive policy, as a function of the keep-alive length.

Longer keep-alive reduces cold starts significantly

- Hybrid policy using a histogram

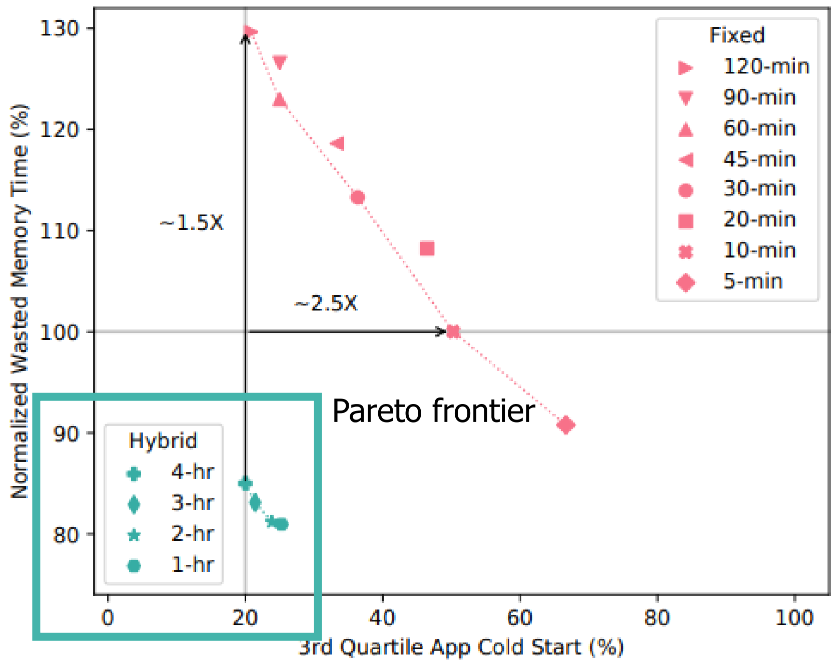


Figure 15: Trade-off between cold starts and wasted memory time for the fixed keep-alive policy and our hybrid policy.

Hybrid policies reduce cold starts significantly with lower memory waste

Evaluation: simulation

- Impact of the histogram cutoff percentiles

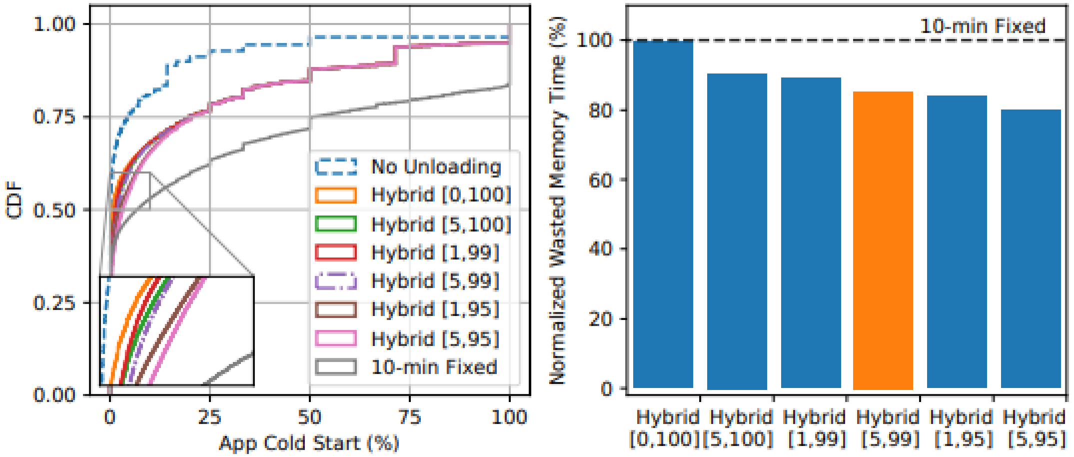


Figure 16: Wasted memory time can be significantly reduced by excluding outliers from the IT distribution.

wasted memory time goes down by 15%, compared to with no cutoff [0,100]

- Impact of unloading and pre-warming

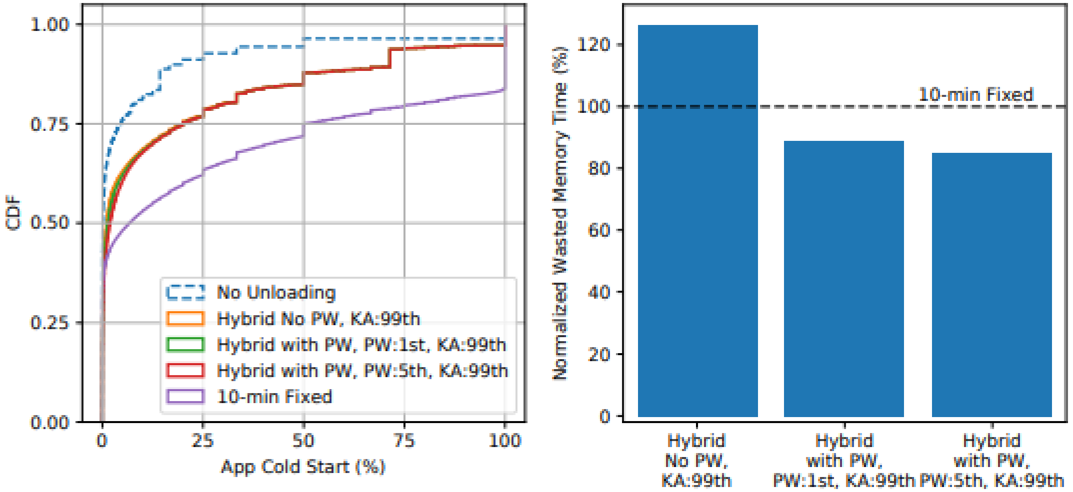
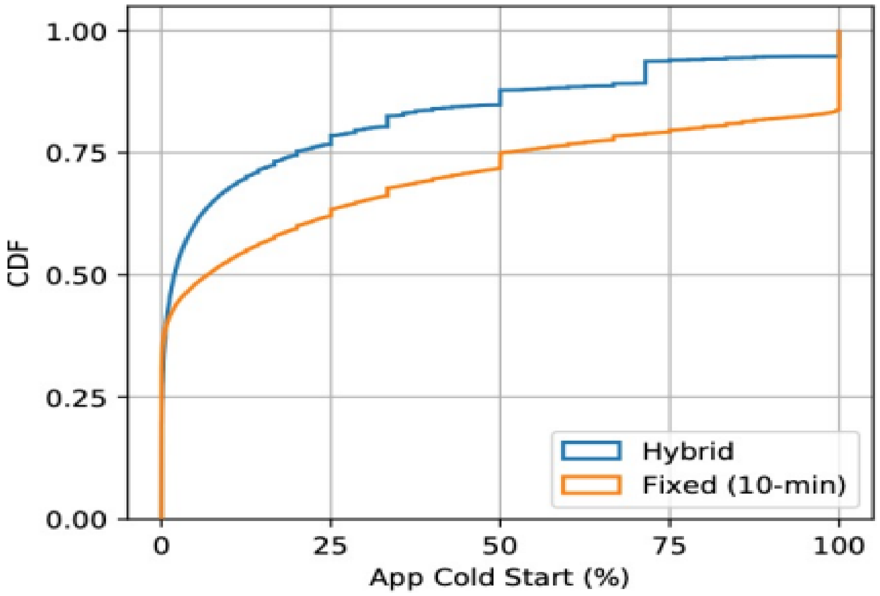


Figure 17: Pre-warming reduces the wasted memory time significantly. The cost is slight increase in cold starts.

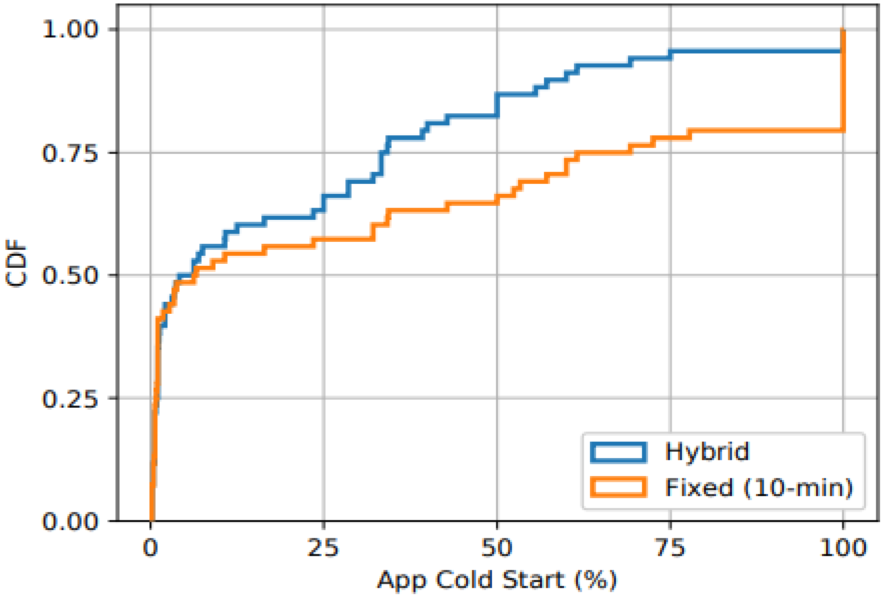
PW (Pre-warming) reduce the wasted memory time significantly

Evaluation: real experiment

- Simulation results



- Experimental results



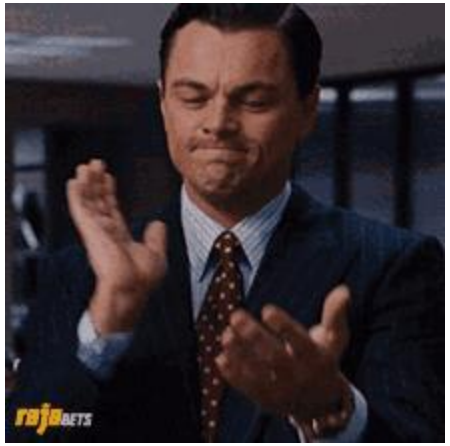
- ***Average exec time reduction: 32.5%***
- ***99th-percentile exec time reduction: 82.4%***

- ***Container memory reduction: 15.6%***
- ***Latency overhead: < 1ms (835.7us)***

Conclusion

Microsoft Azure Microsoft Research

- Characterize the users' real FaaS workloads of Azure Functions from provider's perspective
- A dynamic policy leveraging small histogram to reduce cold starts at a low resource cost
- Azure Functions sanitized traces available:
  https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsDataset2019.md



Serverless in the Wild: Characterizing and Optimizing the Serverless Workload at a Large Cloud Provider

Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke,

Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini

USENIX ATC'20

Thank You !

2022. 05. 31

Presented by Yejin Han

yj0225@dankook.ac.kr