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

Microsoft Azure Microsoft Research

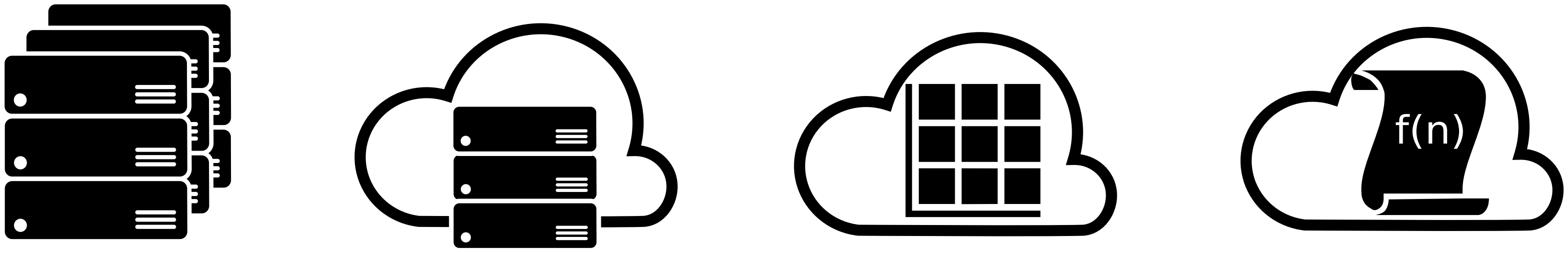
July 15, 2020

What is Serverless?

- Very attractive abstraction:
  - Pay for Use
  - Infinite elasticity from 0 (and back)
  - No worry about servers
    - Provisioning, Reserving, Configuring, patching, managing
- Most popular offering: Function-as-a-Service (FaaS)
  - Bounded-time functions with no persistent state among invocations
  - Upload code, get an endpoint, and go

For the rest of this talk, **Serverless = Serverless FaaS**

What is Serverless?



	Bare Metal	VMs (IaaS)	Containers	Functions (FaaS)
Unit of Scale	Server	VM	Application/Pod	Function
Provisioning	Ops	DevOps	DevOps	Cloud Provider
Init Time	Days	~1 min	Few seconds	Few seconds
Scaling	Buy new hardware	Allocate new VMs	1 to many, auto	0 to many, auto
Typical Lifetime	Years	Hours	Minutes	O(100ms)
Payment	Per allocation	Per allocation	Per allocation	Per use
State	Anywhere	Anywhere	Anywhere	Elsewhere

Serverless

"…more than 20 percent of global enterprises will have deployed serverless computing technologies by 2020."

Gartner, Dec 2018

7 Reasons

Why Serverless is the Future

claudiobernasconi.ch

A CLOUD GURU

Serverless — the future of software architecture?

The future is transitioning from 3-tiered architectures to thick-client apps connected to cloud-based microservice functions

Sam Kroonenburg Follow

Oct 21, 2015 · 7 min read

psc_ TRANSITION TECHNOLOGIES

13 September 2019 ★★★★★ 5 (4)

Why serverless is the future of software and apps

Survey Shows More than 75% Use or Plan to Use Serverless in Next 18 Months

Source: The New Stack Serverless Survey 2018. Q. Is your organization using a serverless architecture? n=608.

Serverless

Hosted Platform

- ALGORITHMIA
- Alibaba Cloud Function Compute
- AWS Lambda
- Azure Functions
- BINARIS
- CLOUDFLARE Workers
- Google Cloud Functions
- HUAWEI FunctionStage
- IBM Cloud Functions
- netlify Functions
- nimbella
- Nuweba
- Oracle Functions
- PubNub FUNCTIONS
- spotinst
- Standard Library
- Tencent Cloud Serverless Cloud Function
- twilio Functions
- ZEIT

Installable Platform

- APACHE Camel
- APACHE OpenWhisk
- AppScale
- fission
- fn
- Knative
- Kubeless
- Kyma
- nuclio
- OPENFAAS
- PipelineAI
- riff
- Virtual Kubelet

Source: CNCF Cloud Native Interactive Landscape
https://landscape.cncf.io/format=serverless

Serverless

December 2019

Serverless Computing: One Step Forward, Two Steps Back

Joseph M. Hellerstein, Jose Faleiro, Joseph E. Gonzalez, Johann Schleier-Smith, Vikram Sreekanti, Alexey Tumanov and Chenggang Wu
UC Berkeley
{hellerstein,jmfaleiro,jegonzal,jssmith,vikrams

Peeking Behind the Curtains of Serverless Platforms

Liang Wang [1], Mengyuan Li [2], Yinqian Zhang [2], Thomas Ristenpart[3], Michael Swift[1]
[1]UW-Madison, [2]Ohio State University, [3]Cornell Tech

Cloud Programming Simplified:
A Berkeley View on Serverless Computing

Eric Jonas, Anurag Khandelwal, Karl Krauth, Johann Schleier-Smith, Qifan Pu, Neeraja Yadwadkar, Ion Stoica, Vikram Sreekanti, Vaishaal Shankar, Joseph E. Gonzalez, David A. Patterson, Chia-Che Tsai, Joao Carreira, Raluca Ada Popa
UC Berkeley
serverlessview@berkeley.edu

"... we predict that (...) serverless computing will grow to dominate the future of cloud computing."

So what are people doing with FaaS?

- Many simple things
  - ETL workloads
  - IoT data collection / processing
  - Stateless processing
    - Image / Video transcoding
    - Translation
    - Check processing
  - Serving APIs, Mobile/Web Backends

- Interesting Explorations
  - MapReduce (pywren)
  - Linear Algebra (numpywren)
  - ExCamera
  - gg "burst-parallel" functions apps
  - ML training
- Limitations
  - Communication
  - Latency
  - Locality (lack)
  - State management

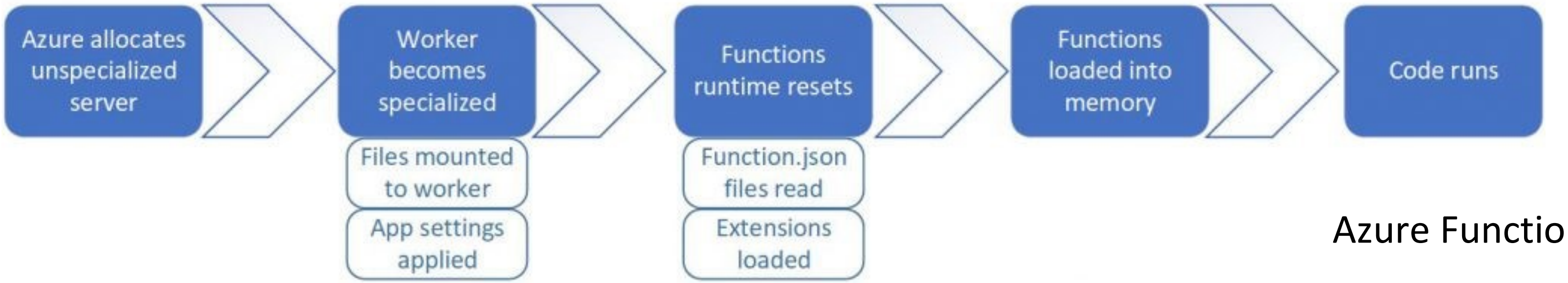
What is Serverless?

- Very attractive abstraction:
  - Pay for Use
  - Infinite elasticity from 0 (and back)
  - No worry about servers
    - Provisioning, Reserving, Configuring, patching, managing

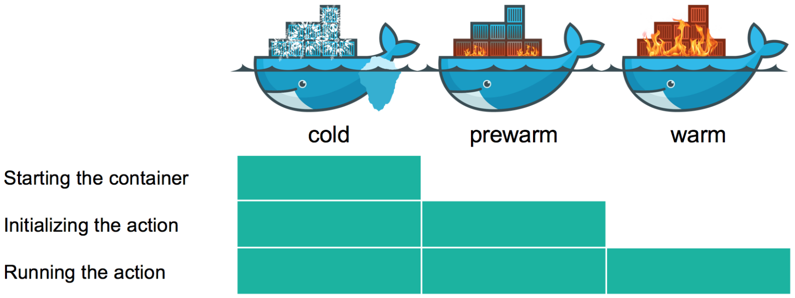
If you are a cloud provider…

- A big challenge
  - You do worry about servers!
    - Provisioning, scaling, allocating, securing, isolating
  - Illusion of infinite scalability
  - Optimize resource use
  - Fierce competition
- A bigger opportunity
  - Fine grained resource packing
  - Great space for innovating, and capturing new applications, new markets

Cold Starts



Azure Functions



OpenWhisk

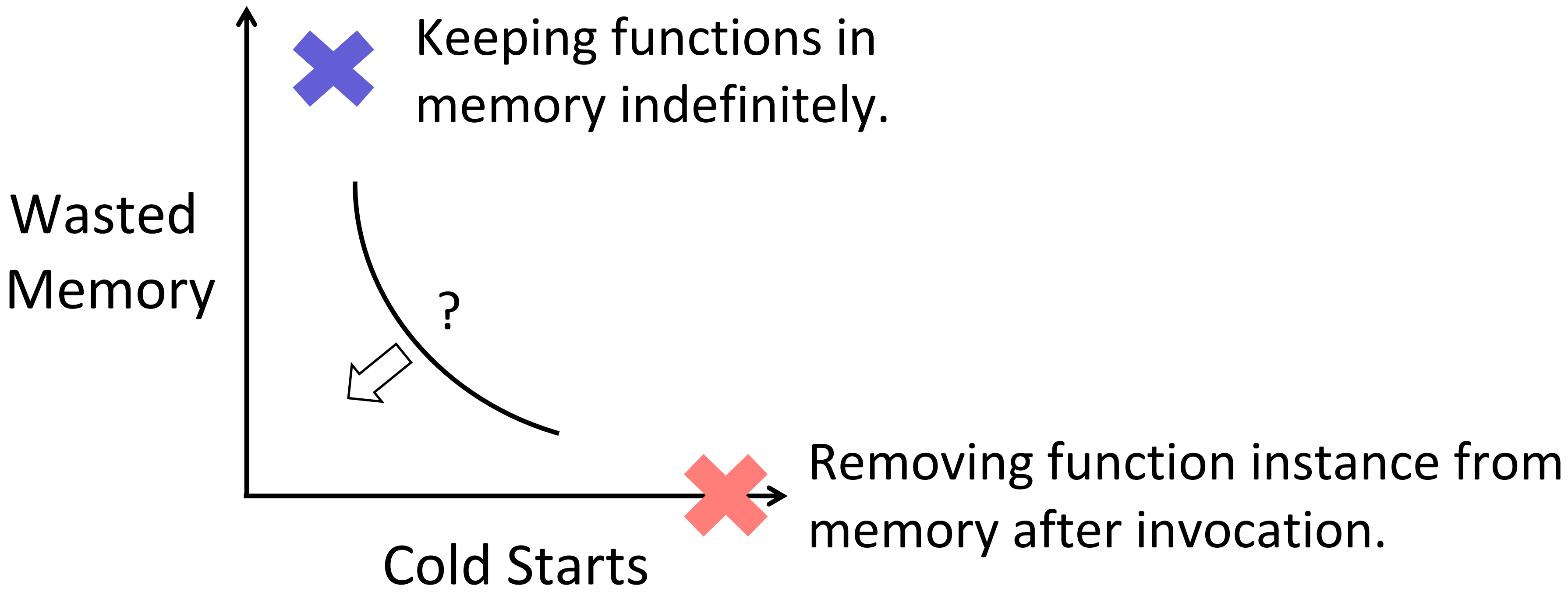
AWS Lambda

- Typically range between 0.2 to a few seconds[1,2]

[1] https://levelup.gitconnected.com/1946d32a0244

[2] https://mikhail.io/serverless/coldstarts/big3/

Cold Starts and Resource Wastage



Stepping Back: Characterizing the Workload

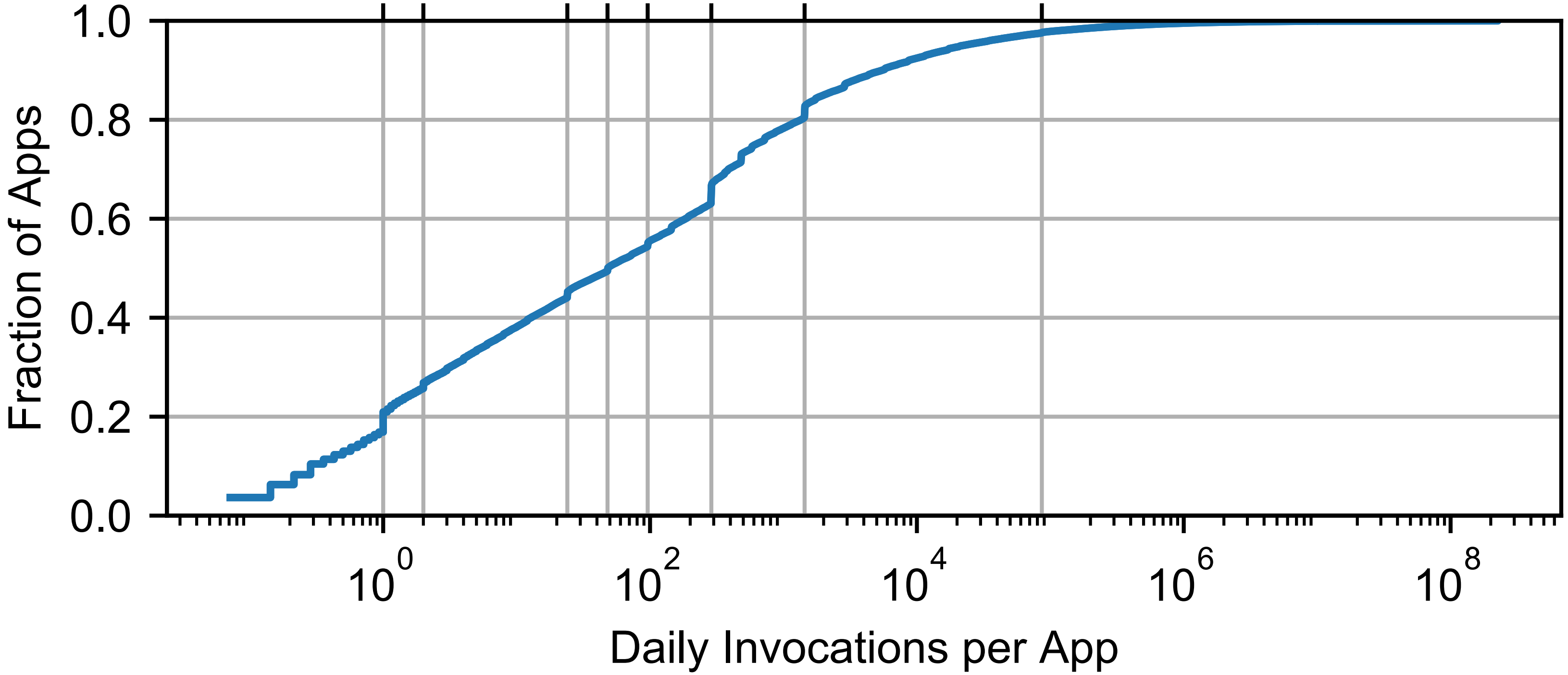
- How are functions accessed
- What resources do they use
- How long do functions take

2 weeks of all invocations to Azure Functions in July 2019

First characterization of the workload of a large serverless provider

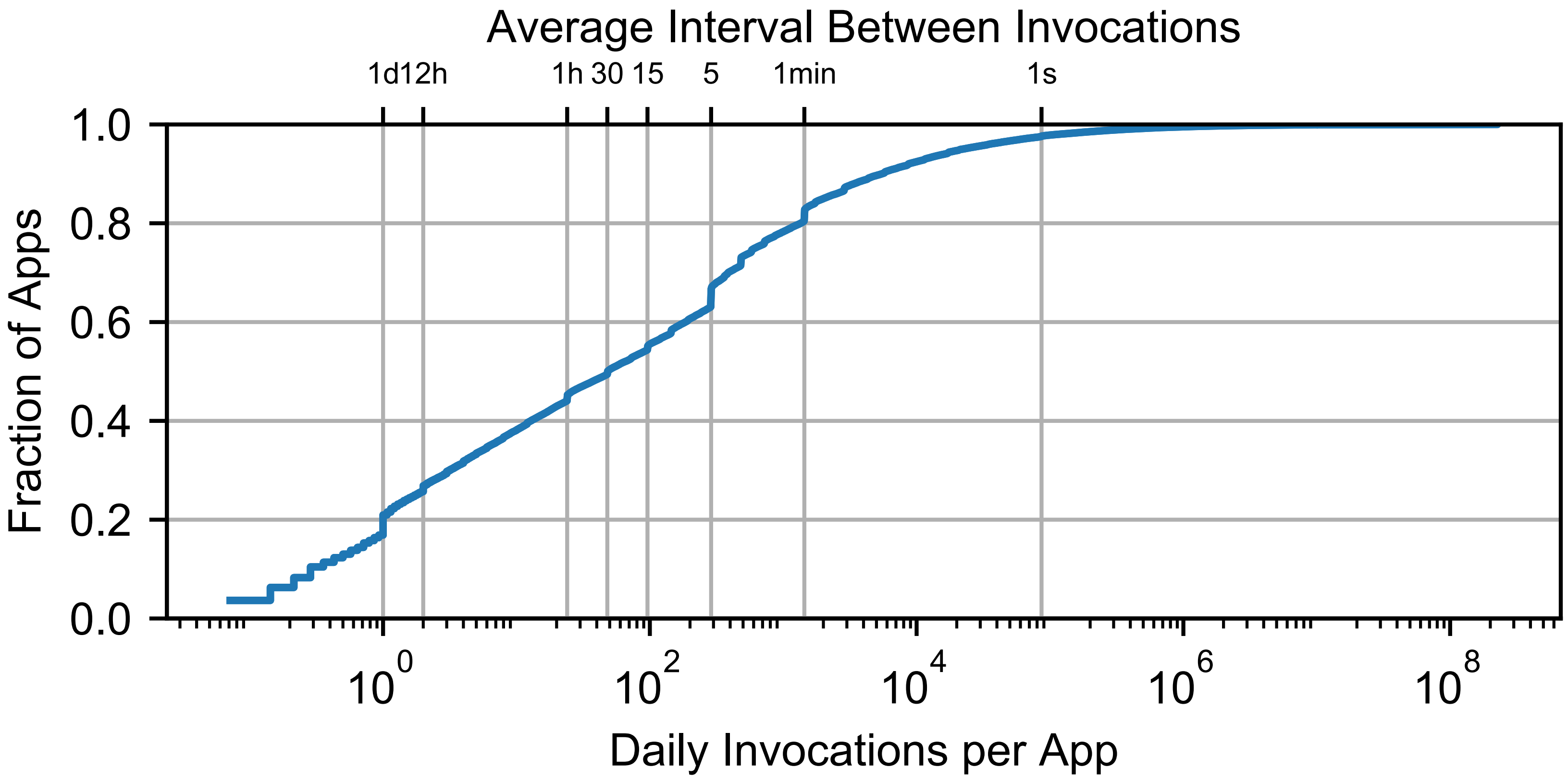
Subset of the traces available for research:
https://github.com/Azure/AzurePublicDataset

Invocations per Application*



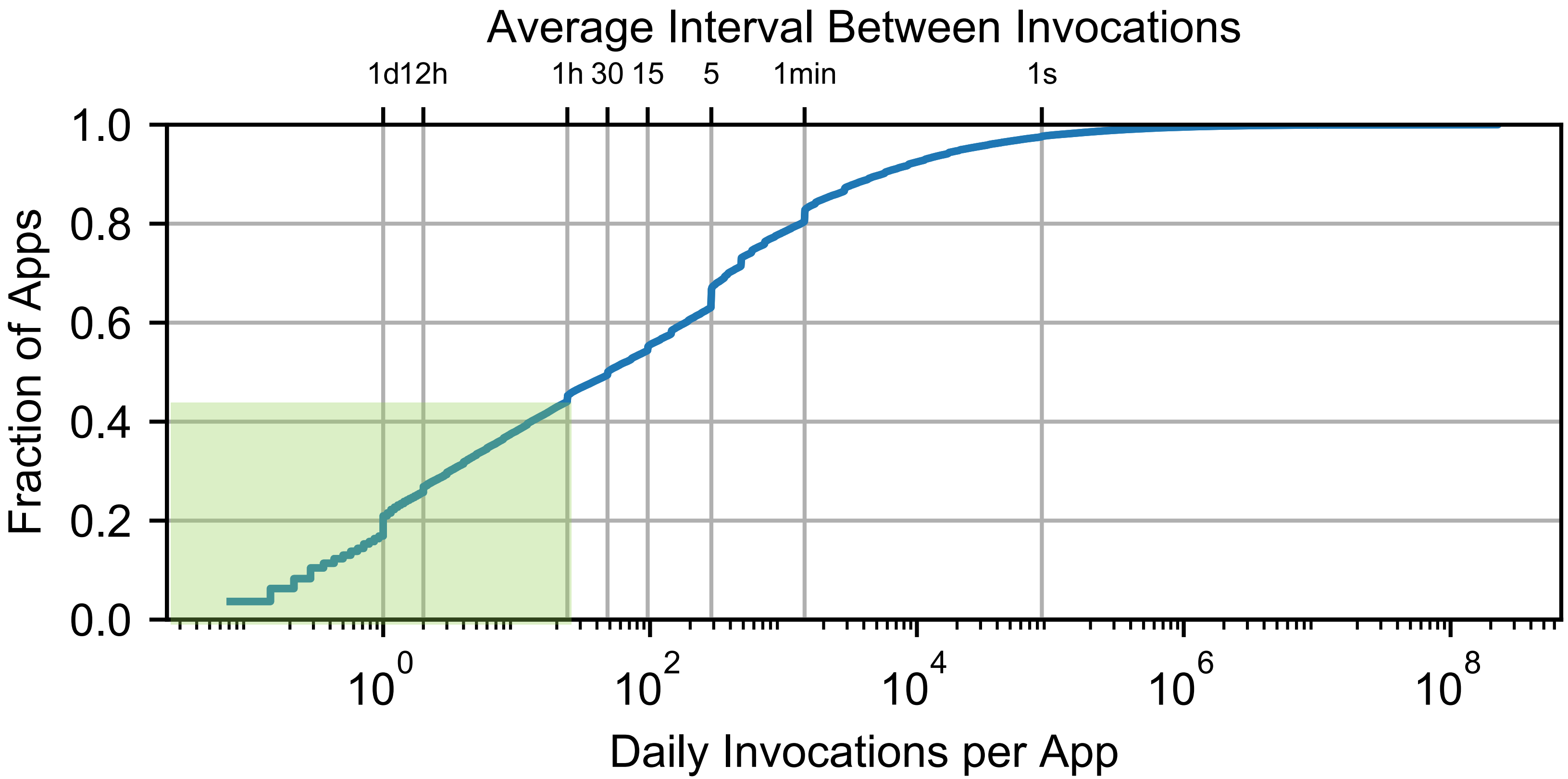
This graph is from a representative subset of the workload. See paper for details.

Invocations per Application



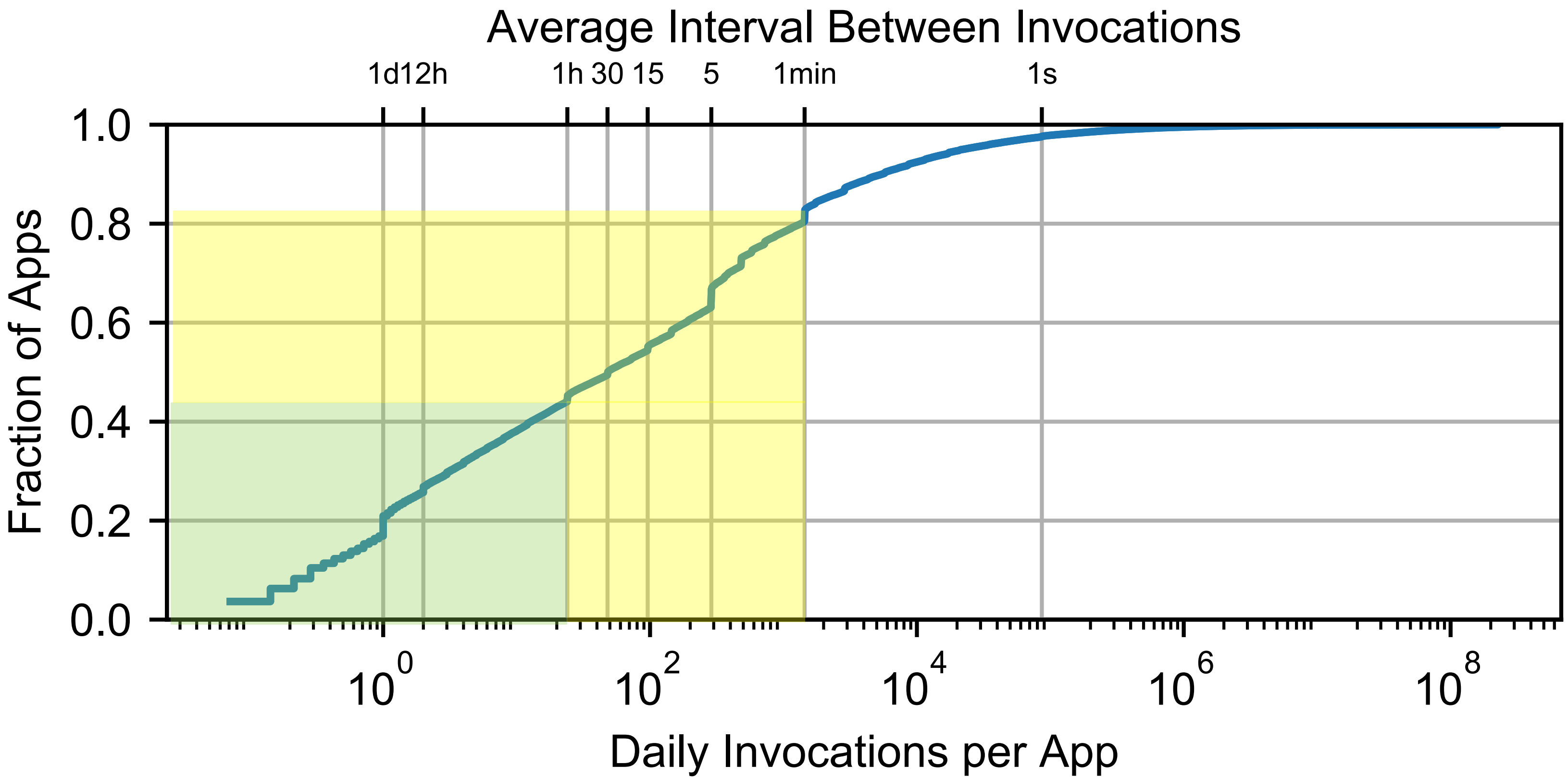
This graph is from a representative subset of the workload. See paper for details.

Invocations per Application



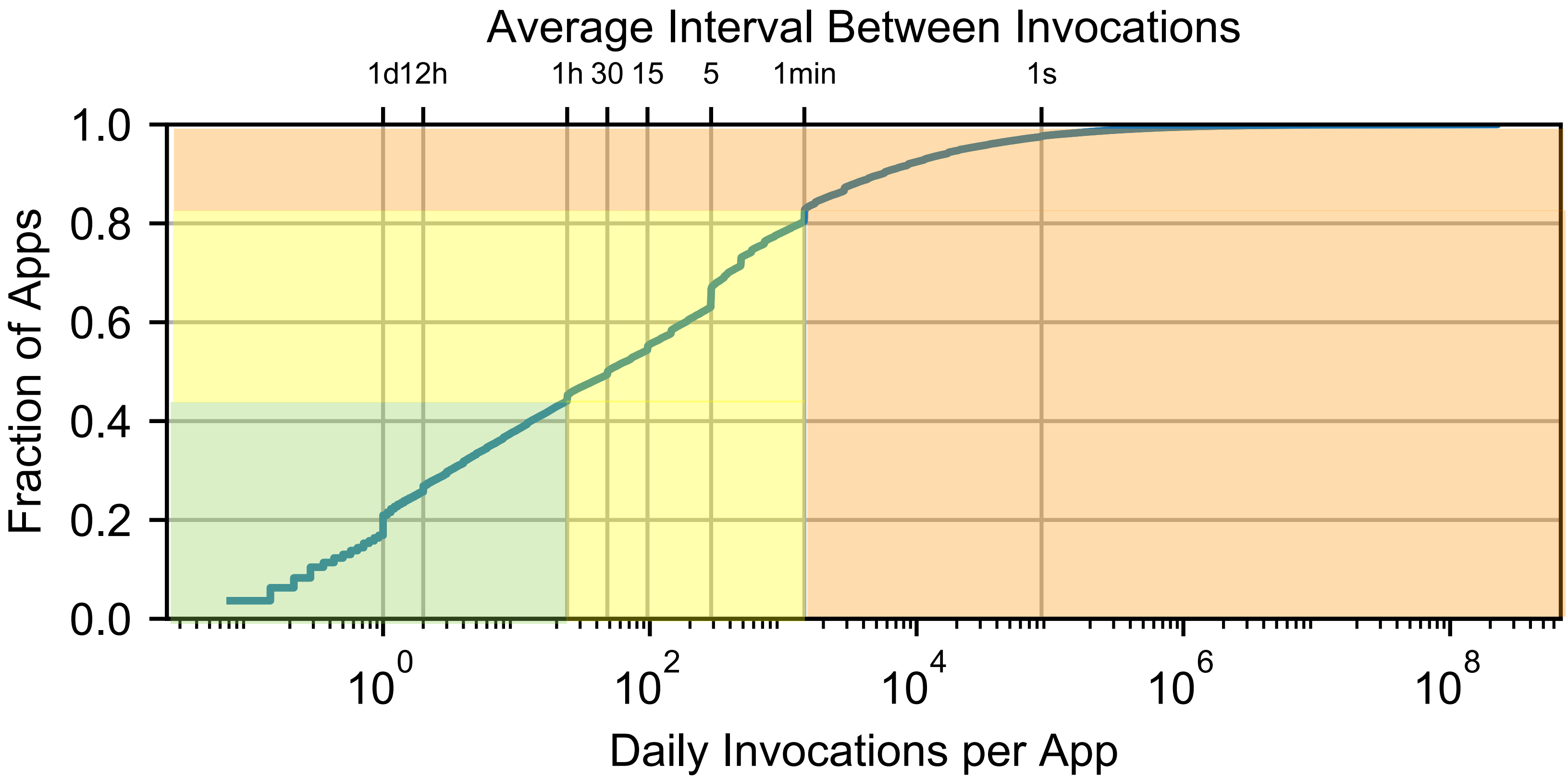
This graph is from a representative subset of the workload. See paper for details.

Invocations per Application



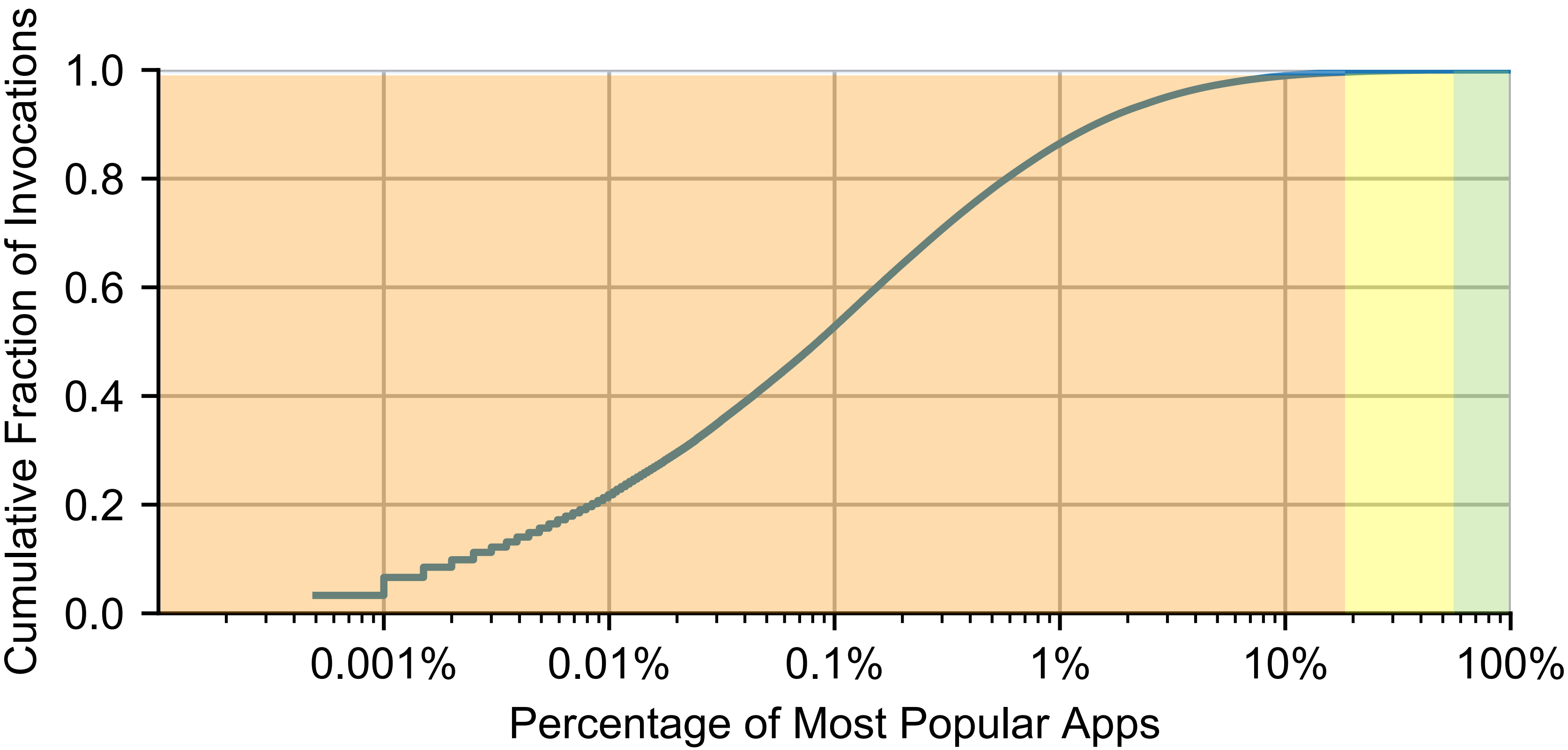
This graph is from a representative subset of the workload. See paper for details.

Invocations per Application



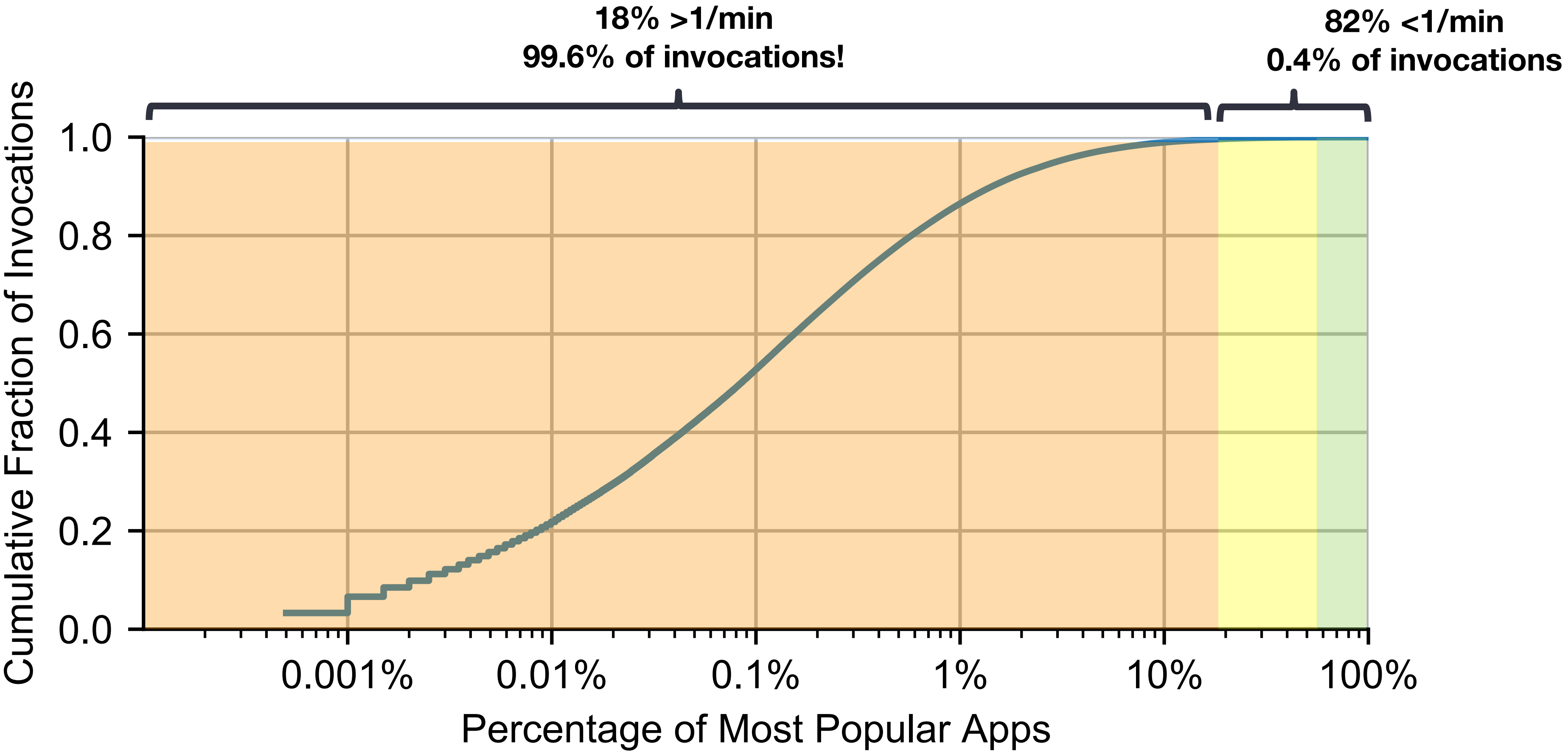
This graph is from a representative subset of the workload. See paper for details.

Invocations per Application



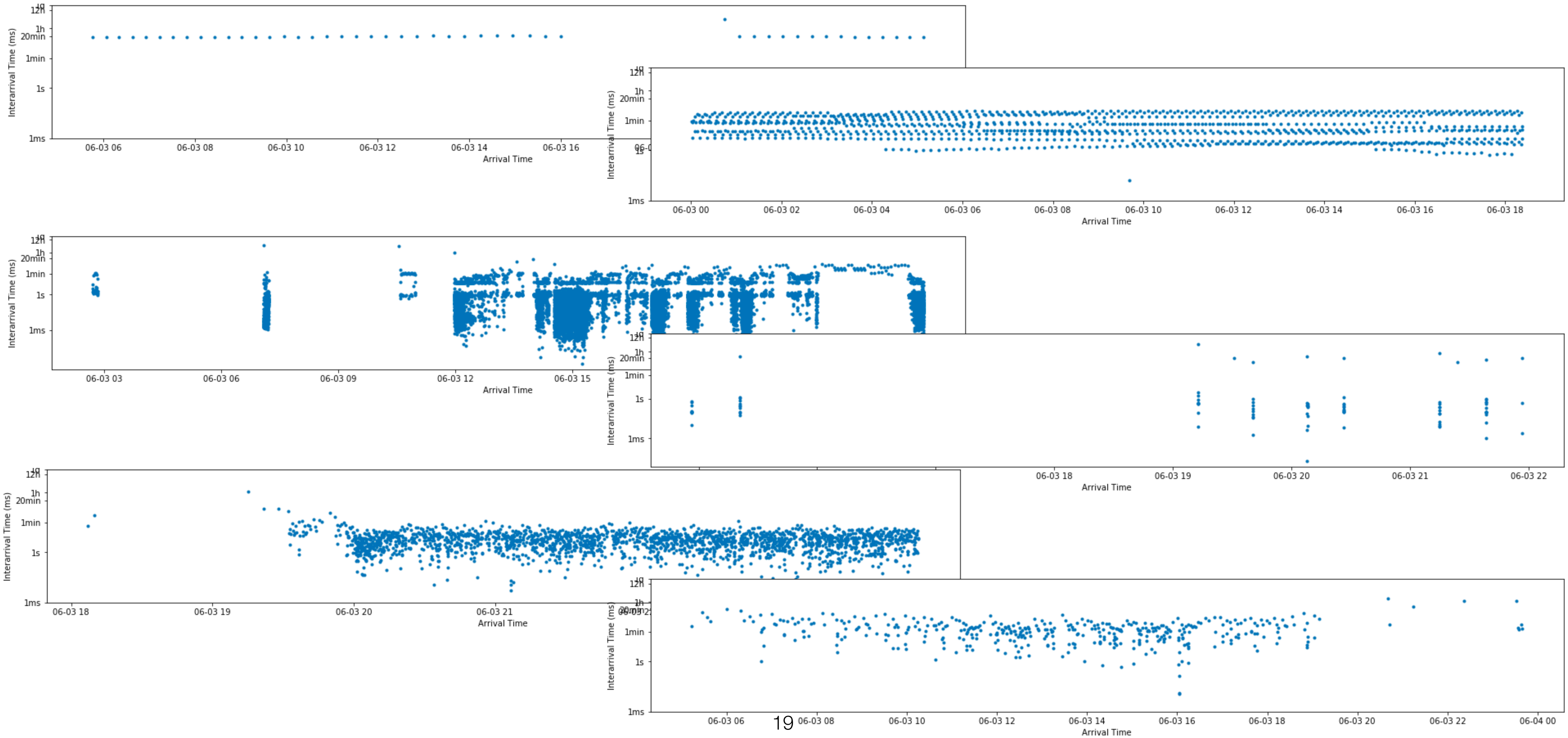
This graph is from a representative subset of the workload. See paper for details.

Invocations per Application



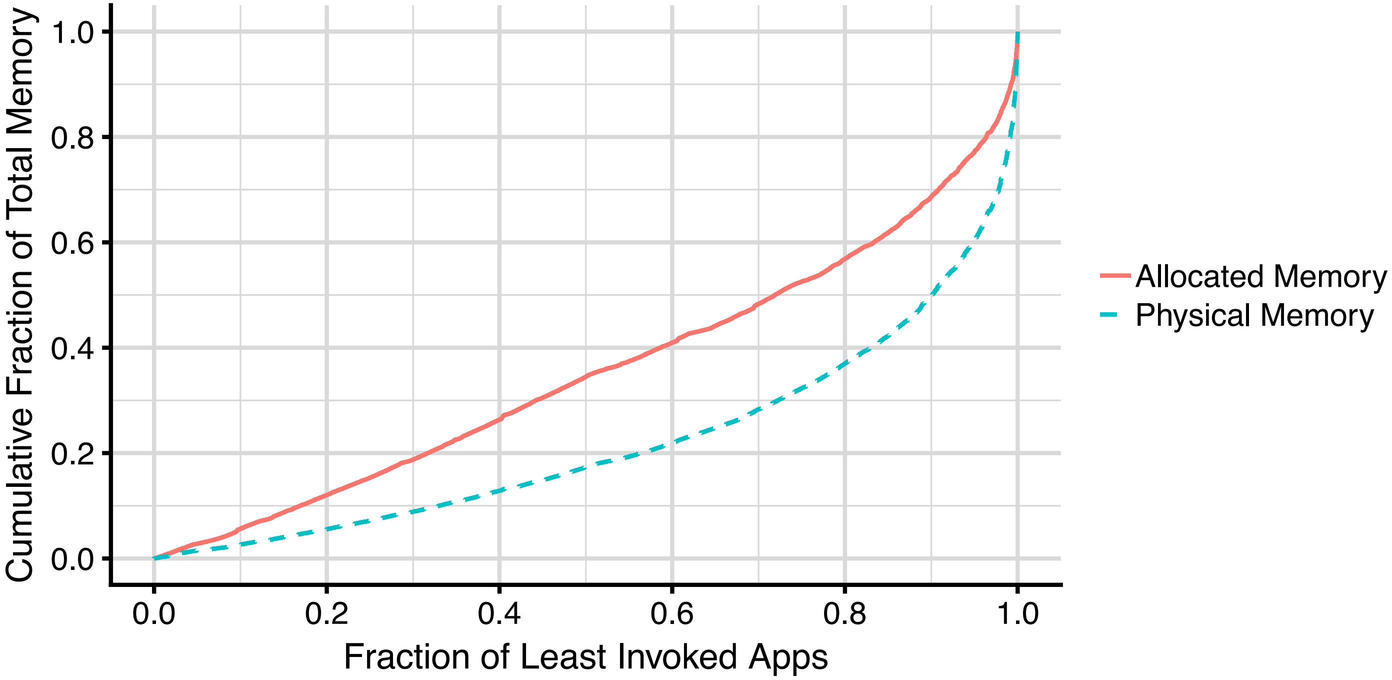
This graph is from a representative subset of the workload. See paper for details.

Apps are highly heterogeneous



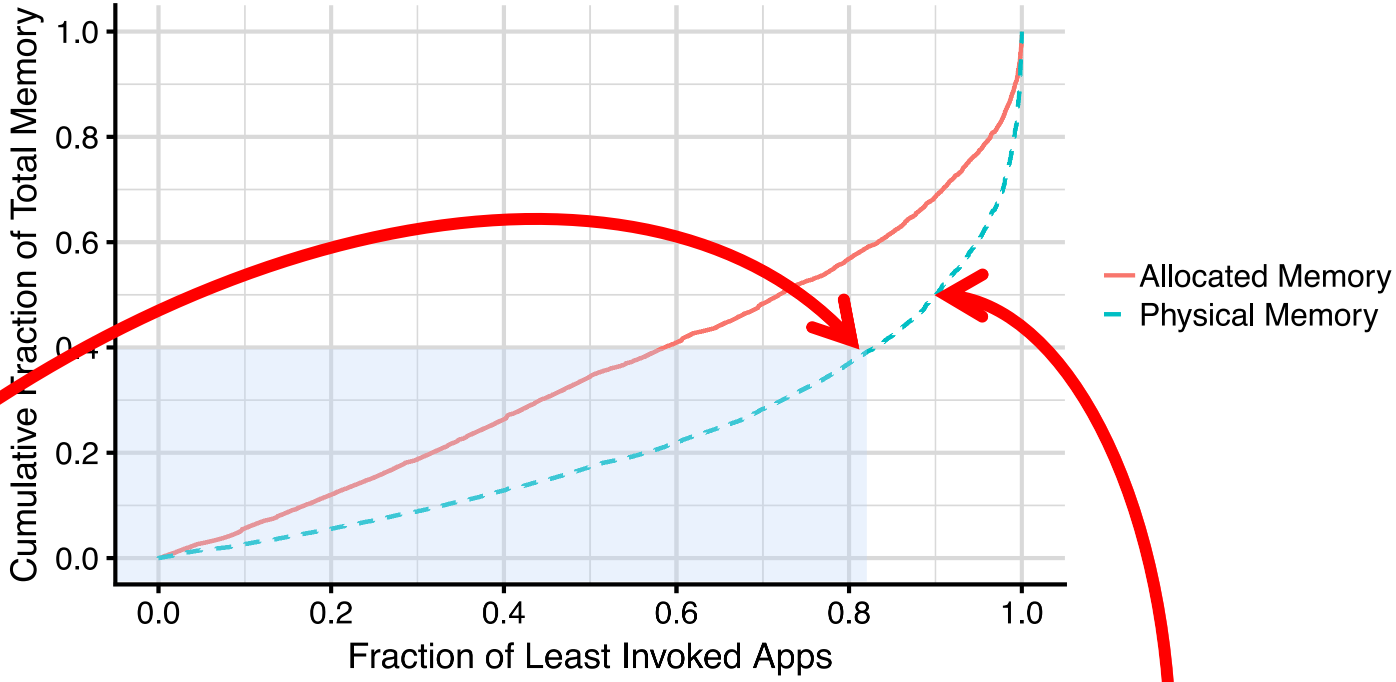
What about memory?

If we wanted to keep all apps warm…



What about memory?

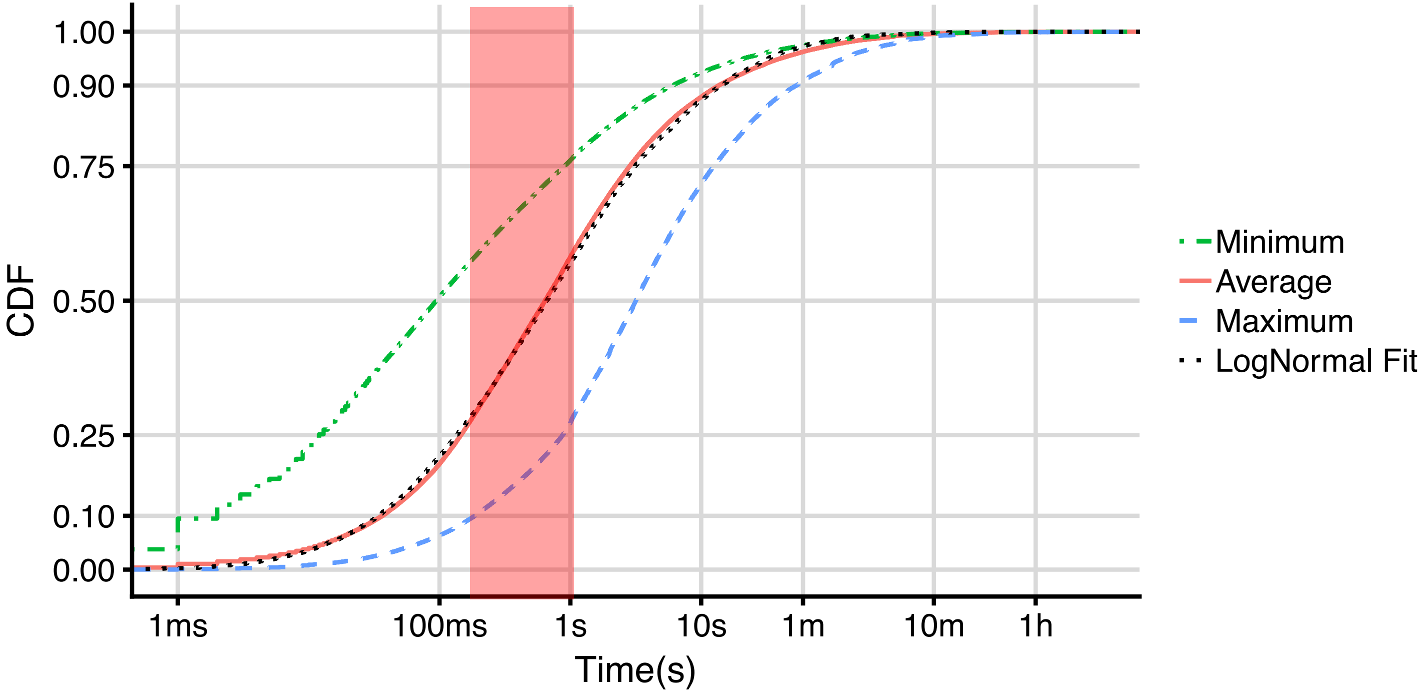
If we wanted to keep all apps warm…



82% of apps -> 0.4% of invocations -> 40% of all physical memory, 60% of virtual memory

90% of apps -> 1.05% of invocations -> 50% of all physical memory

Function Execution Duration



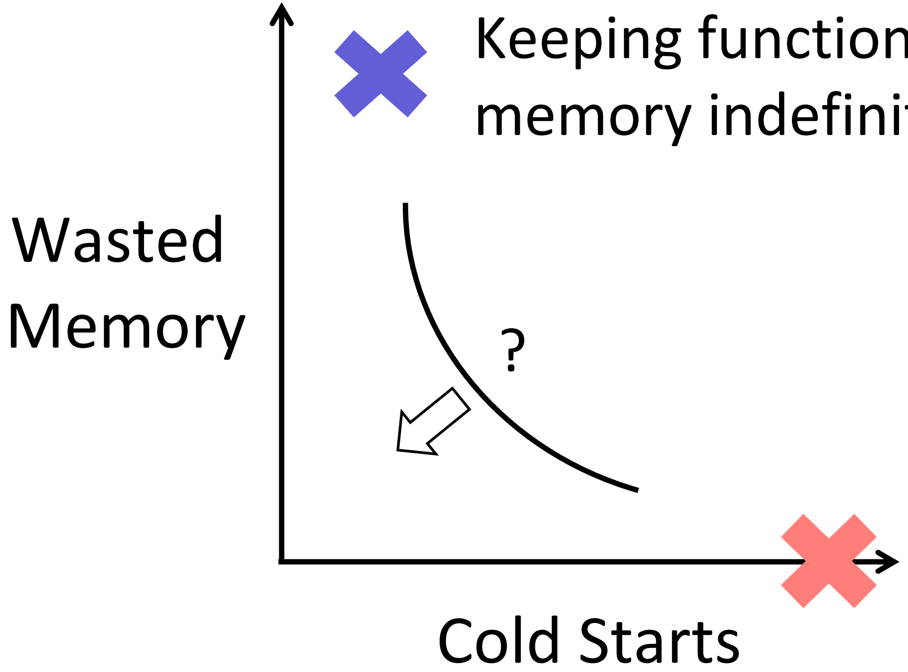
- Executions are short
  - 50% of apps on average run for <= 0.67s
  - 75% of apps on run for <= 10s max
- Times at the same scale as cold start times[1,2]

[1]https://levelup.gitconnected.com/1946d32a0244 [2]https://mikhail.io/serverless/coldstarts/big3/

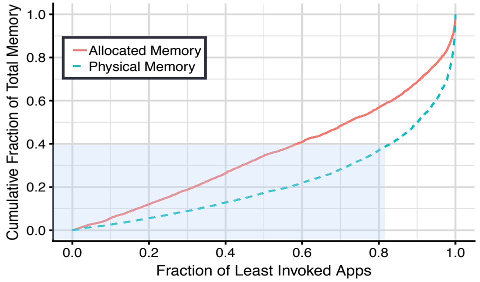
Key Takeaways

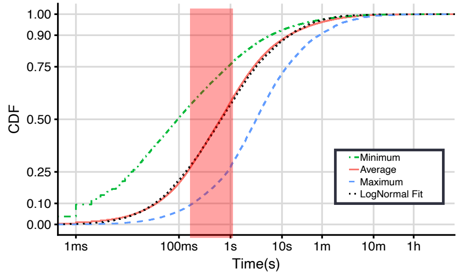
- Highly concentrated accesses
  - 82% of the apps are accessed <1/min on average
  - Correspond to 0.4% of all accesses
  - But in aggregate would take 40% of the service memory if kept warm
- Arrival processes are highly variable
- Execution times are short
  - Same OOM as cold start times

Cold Starts and Resource Wastage



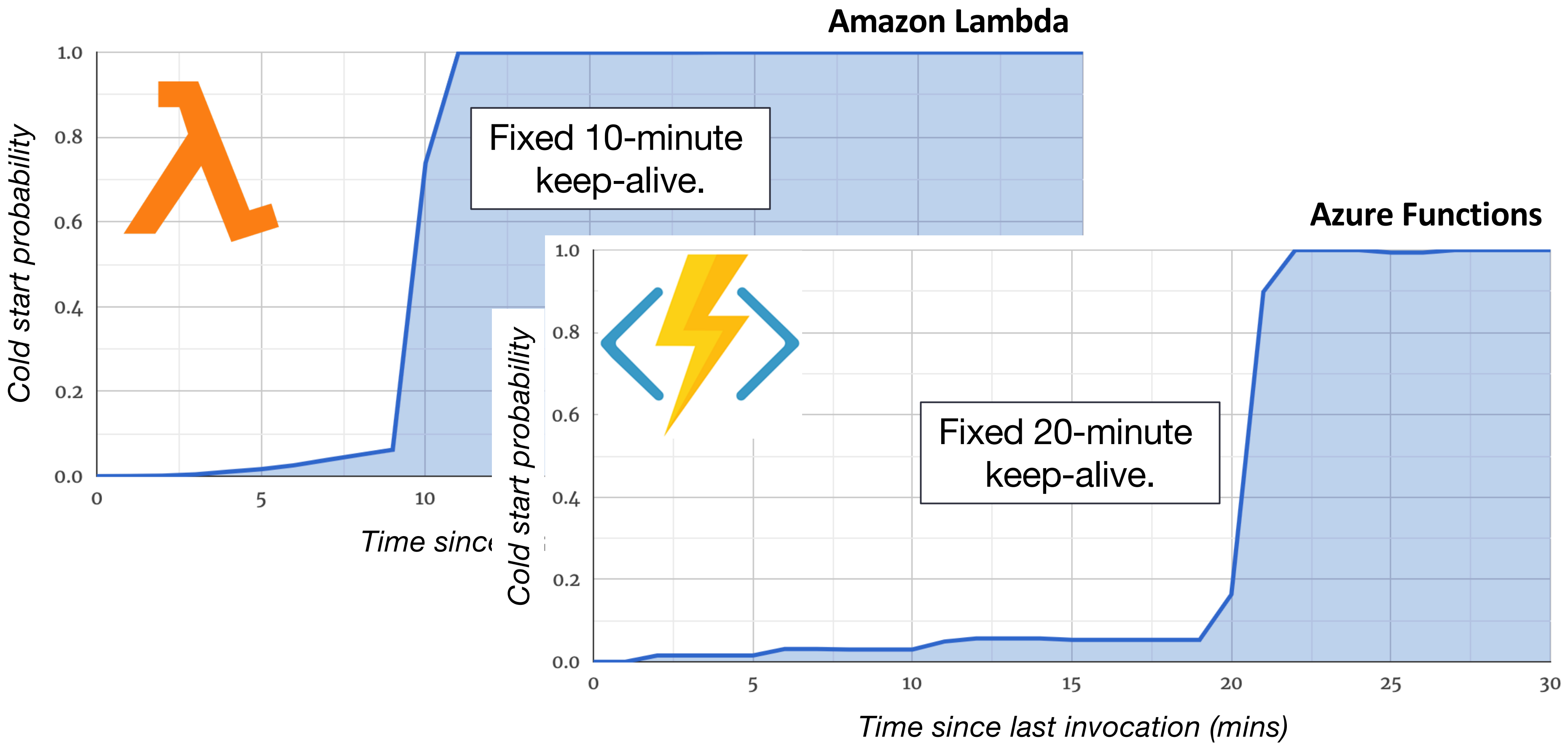
Keeping functions in memory indefinitely.





Removing function instance from memory after invocation.

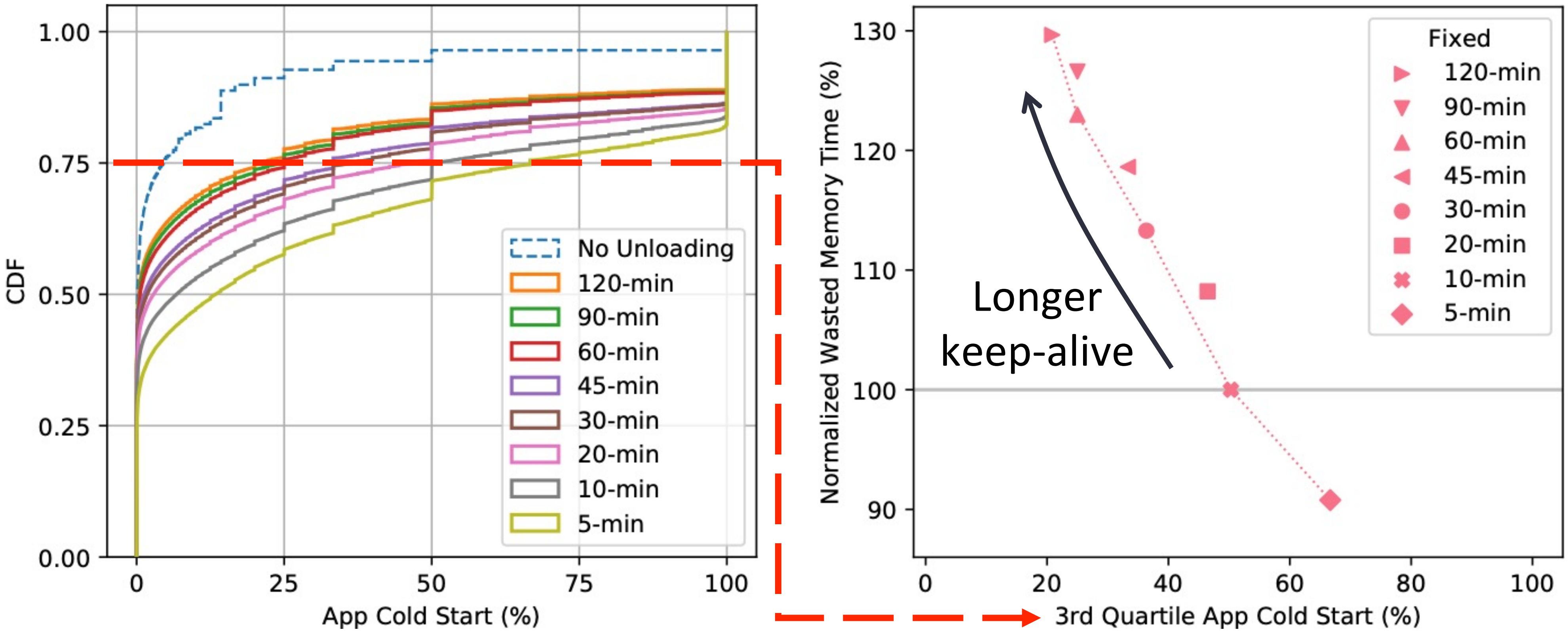
What do serverless providers do?



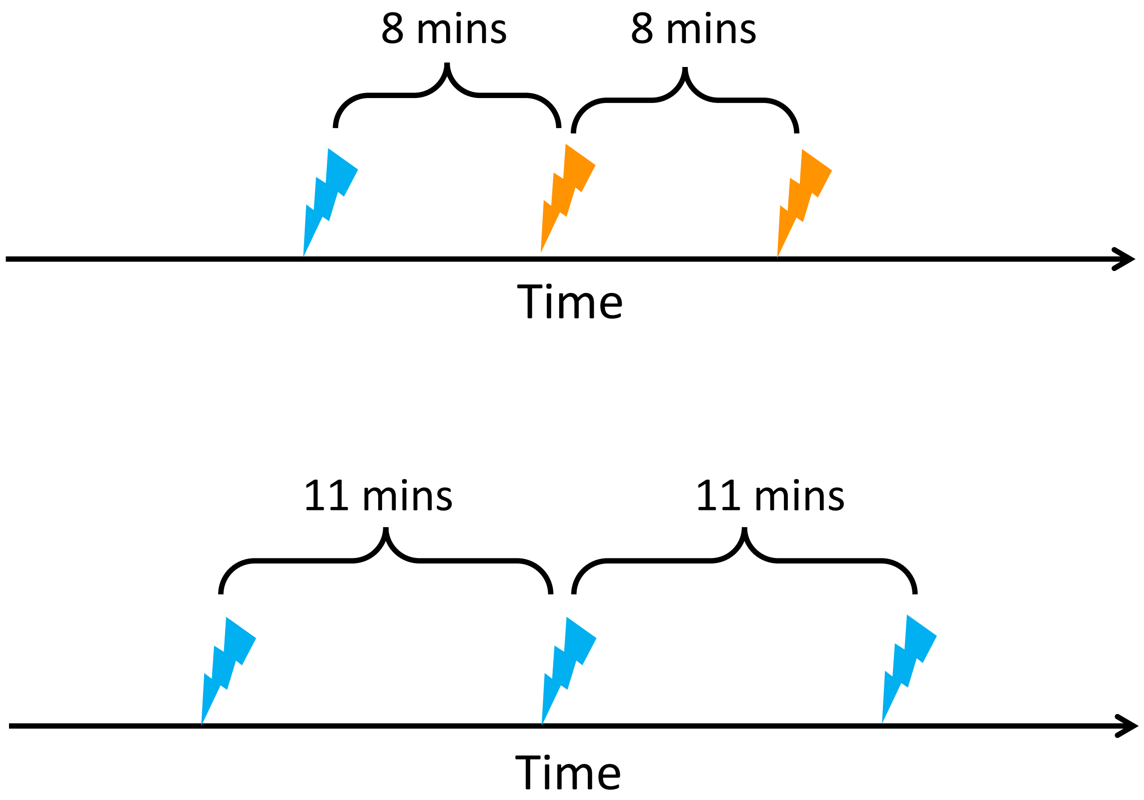
Mikhail Shilkov, Cold Starts in Serverless Functions, https://mikhail.io/serverless/coldstarts/

Fixed Keep-Alive Policy

Results from simulation of the entire workload for a week.



Fixed Keep-Alive Won't Fit All

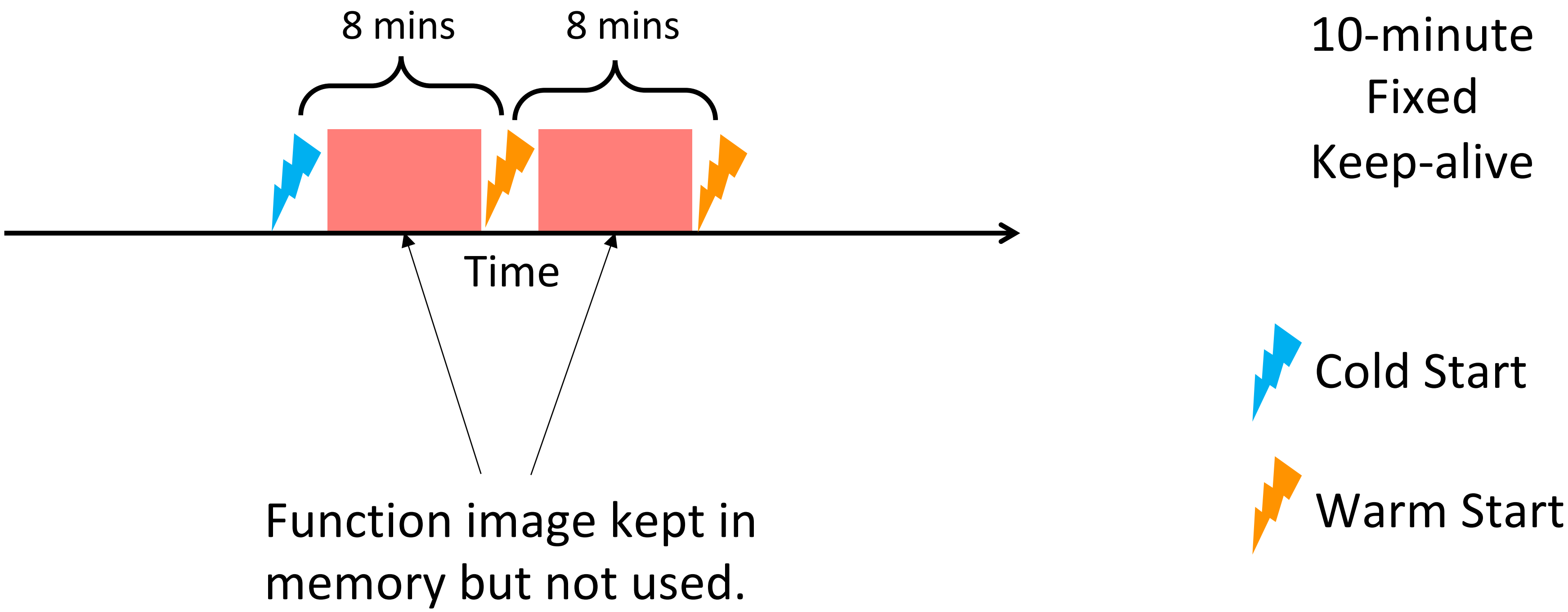


10-minute
Fixed
Keep-alive

Cold Start

Warm Start

Fixed Keep-Alive Is Wasteful



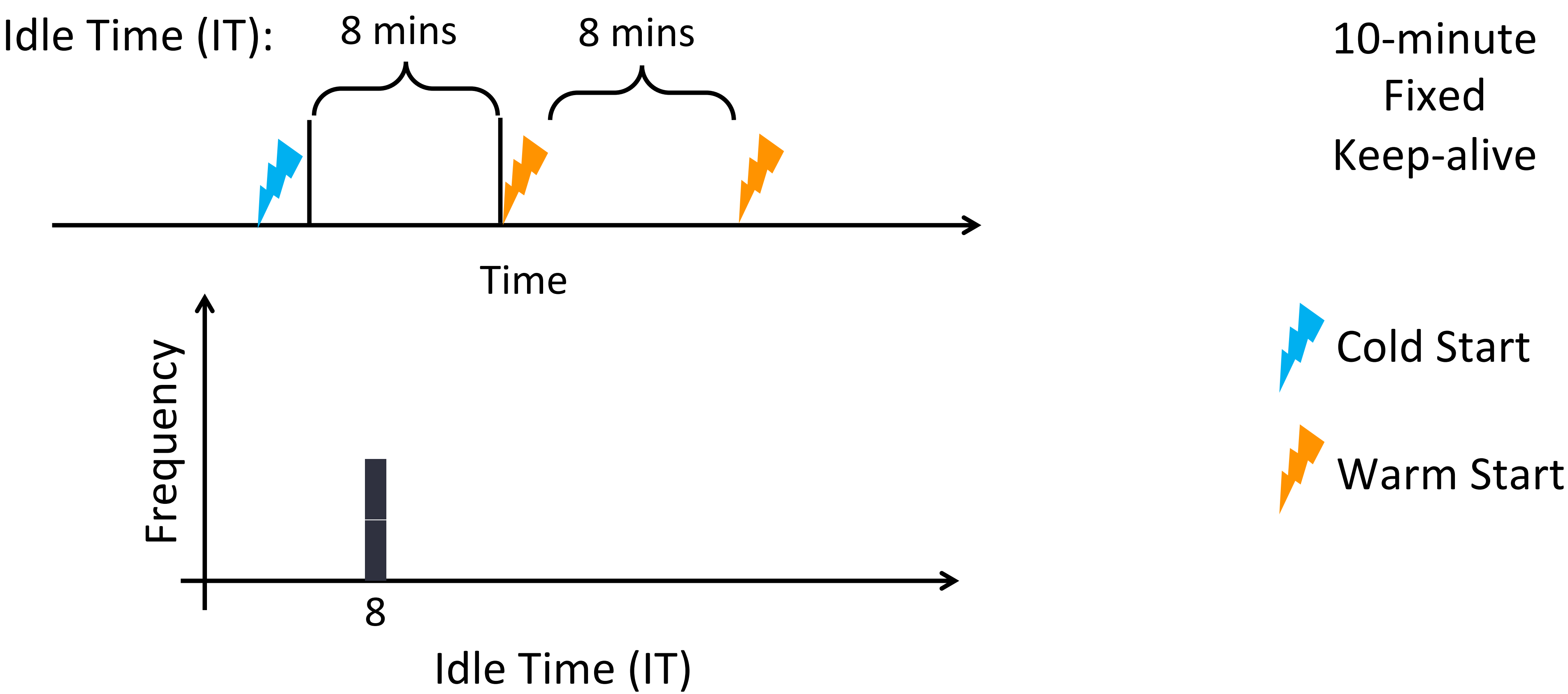
Hybrid Histogram Policy

Adapt to each application

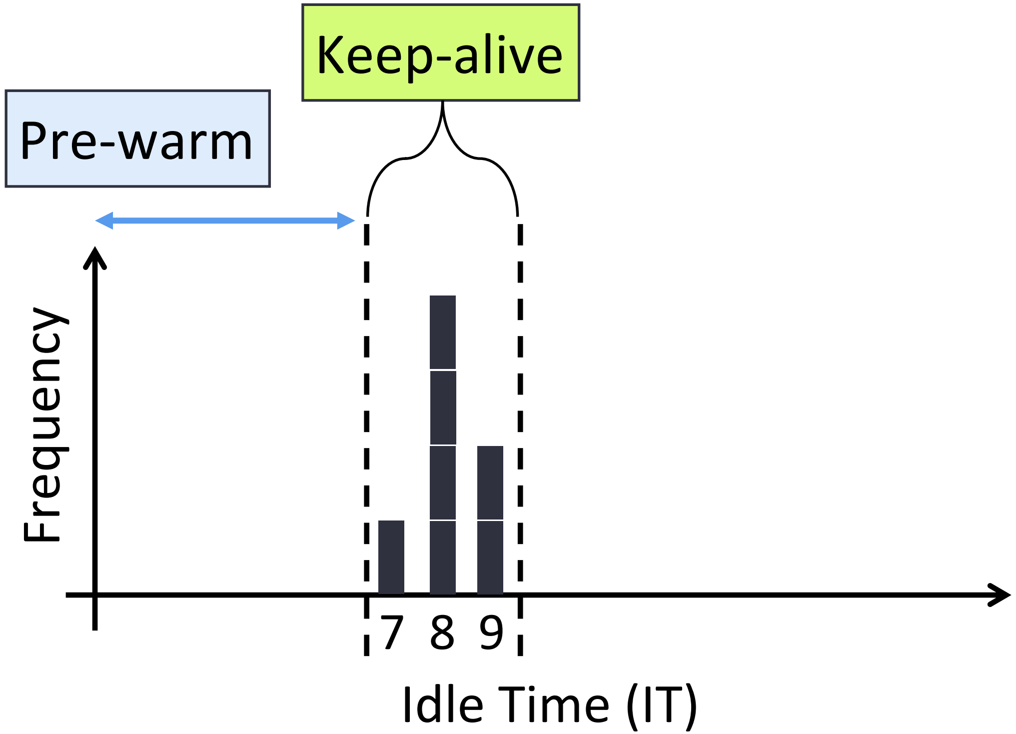
Pre-warm in addition to keep-alive

Lightweight implementation

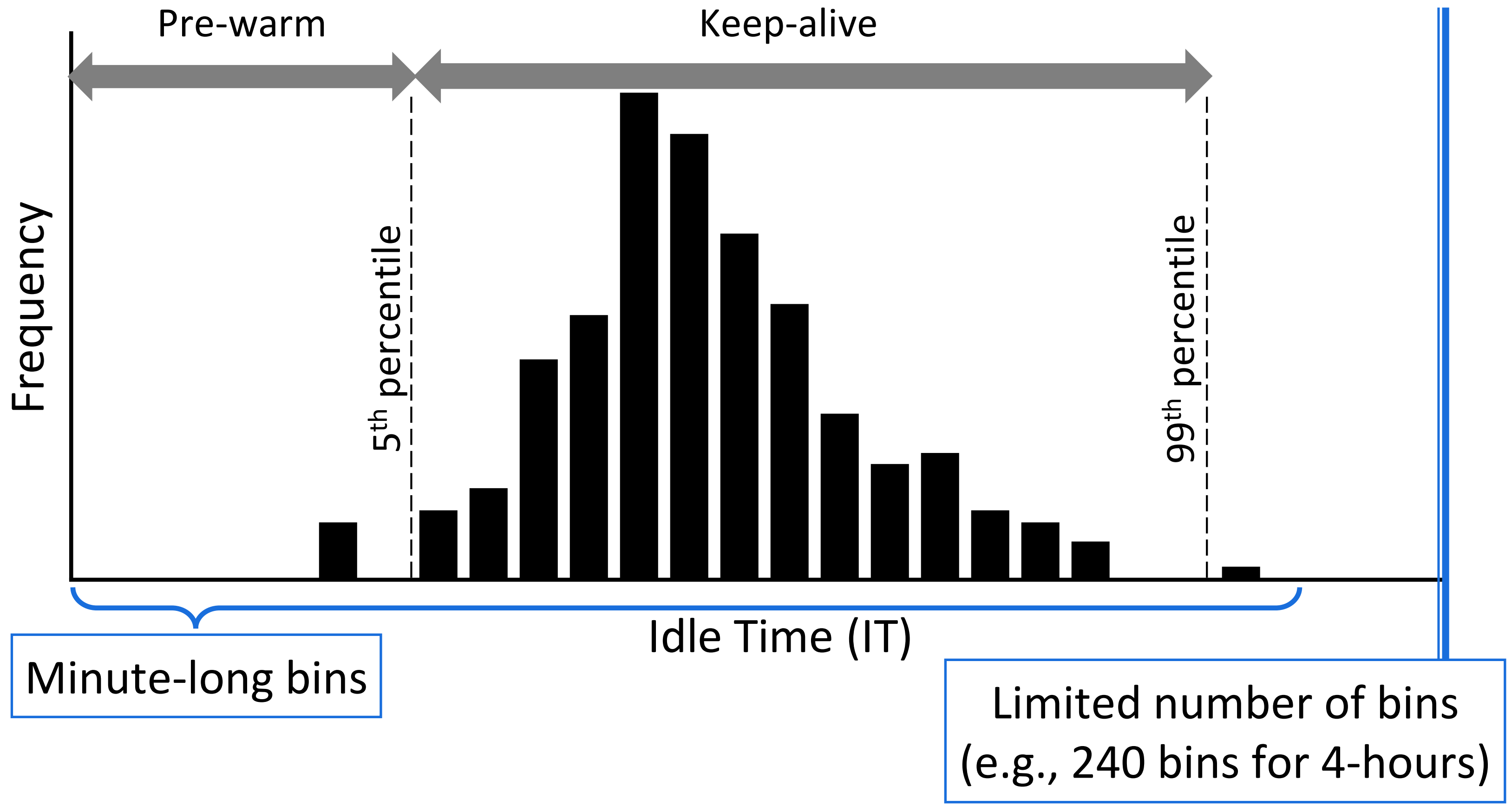
A Histogram Policy To Learn Idle Times



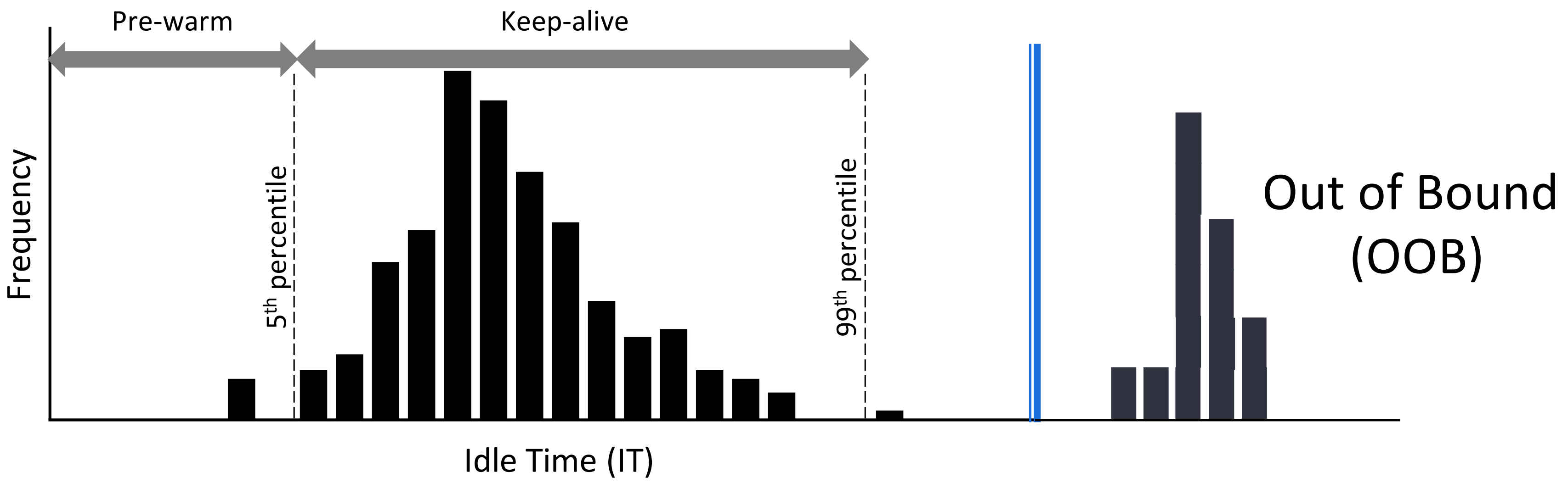
A Histogram Policy To Learn Idle Times



A Histogram Policy To Learn Idle Times



The Hybrid Histogram Policy

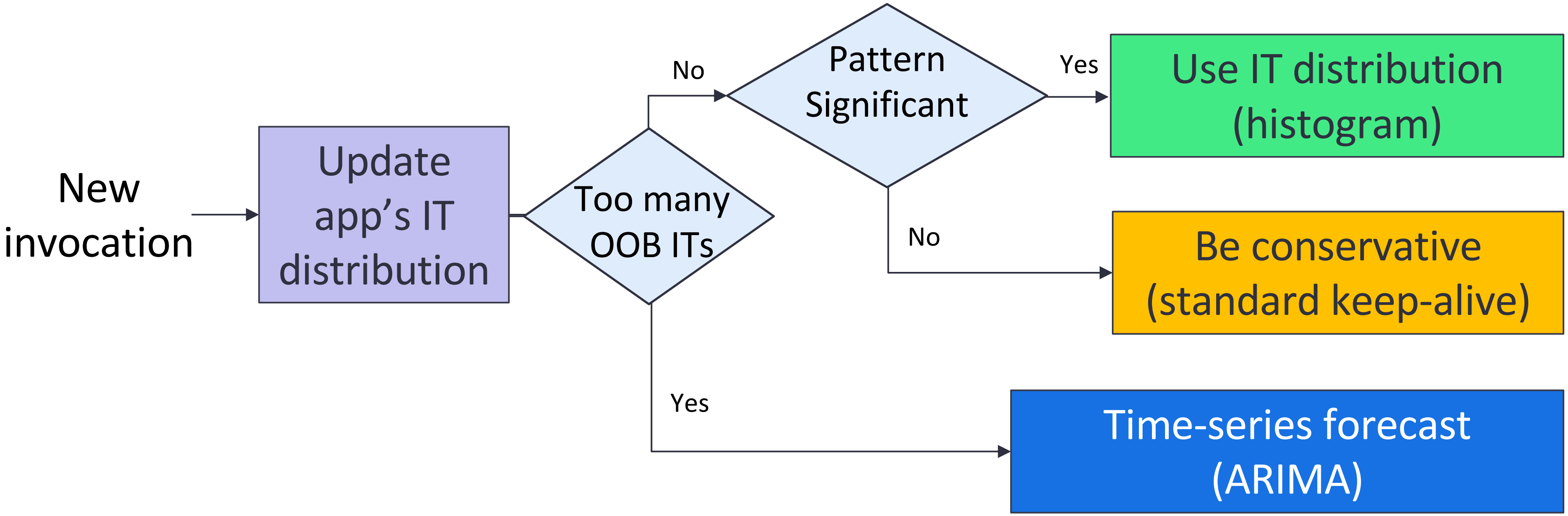


We can afford to run complex predictors given the low arrival rate.

A histogram might be too wasteful.

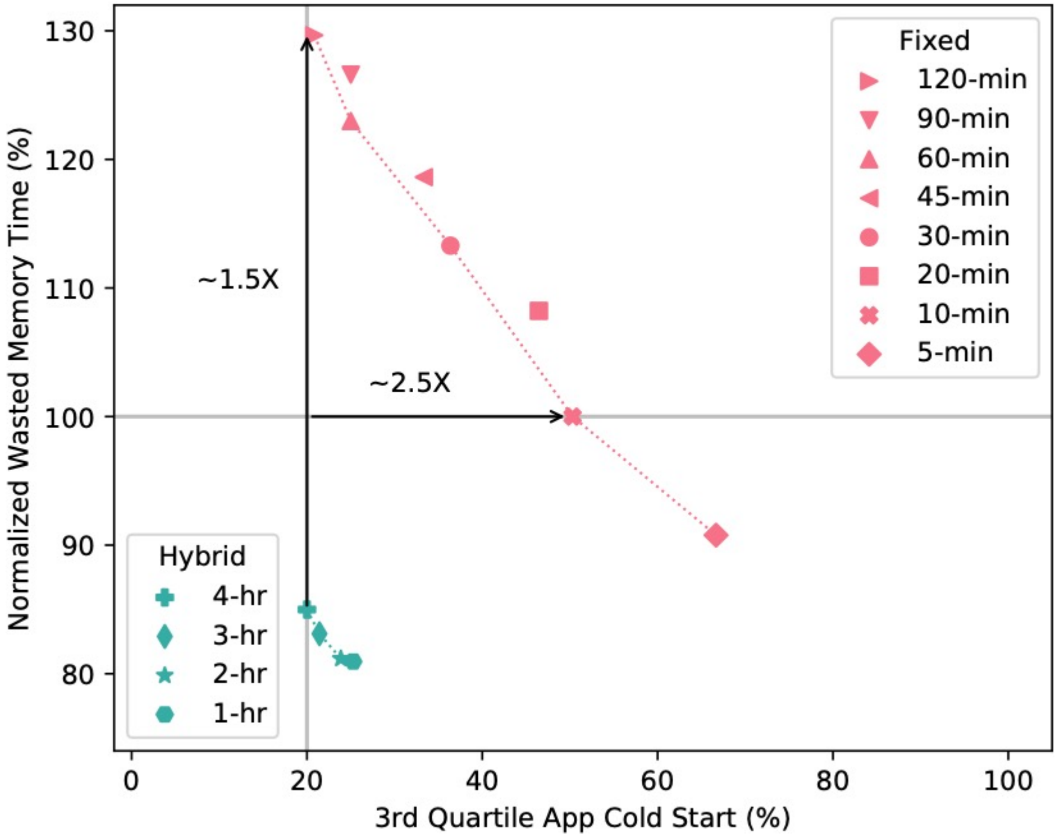
Time Series Forecast

The Hybrid Histogram Policy



ARIMA: Autoregressive Integrated Moving Average

More Optimal Pareto Frontier

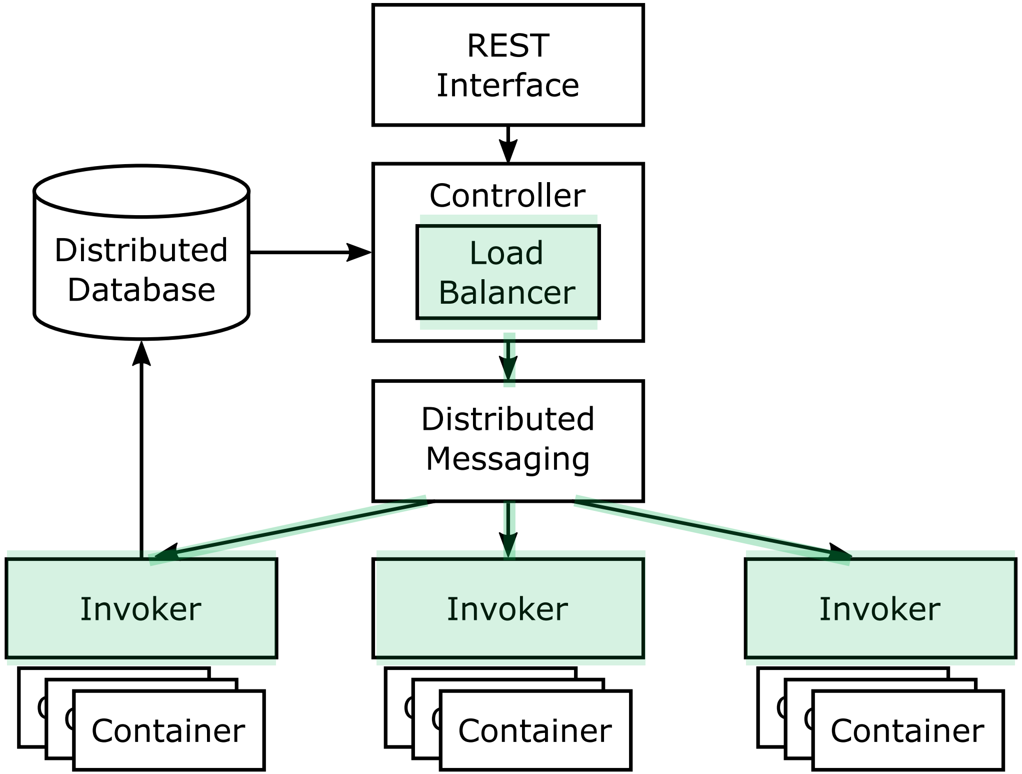


Implemented in OpenWhisk

APACHE
OpenWhisk™

- Open-sourced industry-grade (IBM Cloud Functions)
- Functions run in docker containers
- Uses 10-minute fixed keep-alive

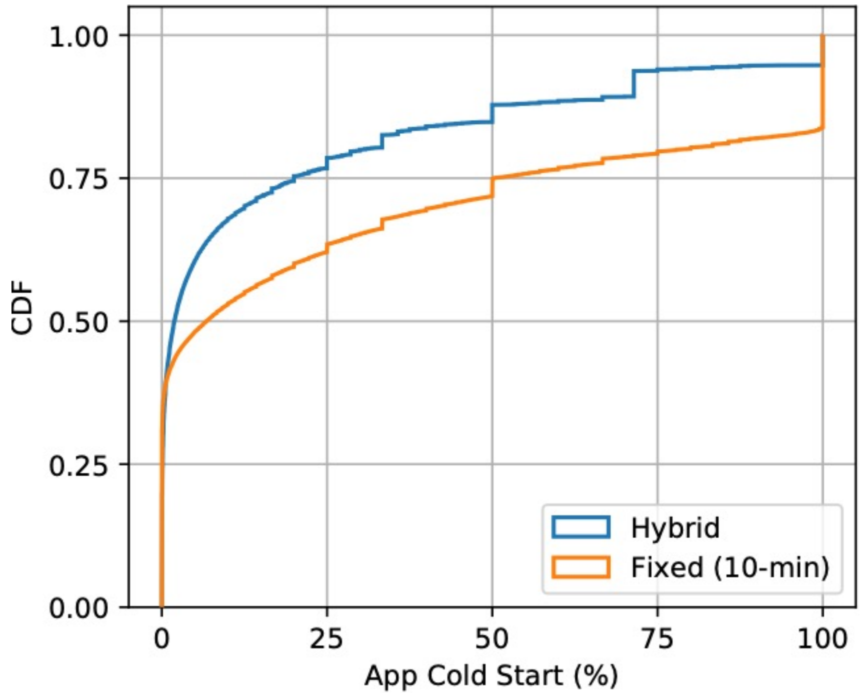
- Built a distributed setup with 19 VMs

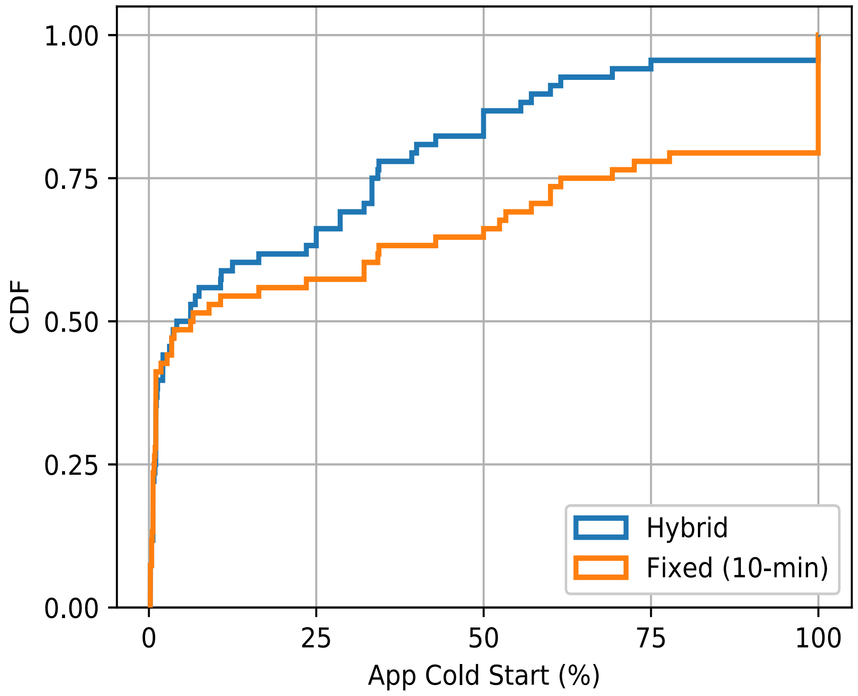


Simulation

4-Hour Hybrid Histogram

Experimental





Average exec time reduction: 32.5%

Container memory reduction: 15.6%

99th–percentile exec time reduction: 82.4%

Latency overhead: < 1ms (835.7µs)

Closing the loop

- First serverless characterization from a provider's point of view

- A dynamic policy to manage serverless workloads more efficiently
  (First elements now running in production.)

- Azure Functions traces available to download:

https://github.com/Azure/AzurePublicDataset/blob/master/AzureFunctionsDataset2019.md