Defuse: A Dependency-Guided Function Scheduler to Mitigate Cold Starts on FaaS Platforms

Jiacheng Shen[1], Tianyi Yang[1], Yuxin Su[1], Yangfan Zhou[2,3] and Michael Lyu[1]

[1] Department of Computer Science and Engineering, The Chinese University of Hong Kong

[2] School of Computer Science, Fudan University

[3] Shanghai Key Laboratory of Intelligent Information Processing

ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

Contents

- Background
- Approach
- Evaluation
- Conclusion

Background – FaaS

FaaS = Function as a Service

Serverless functions -> monolithic applications

Less management burden

Emerging Paradigm:

- AWS Lambda, Google Cloud Functions, Azure Functions, …

Background – FaaS

The process of invoking a serverless function (Critical Path):

- Create and initialize a container.
- Load user code to the container.
- Execute user code.
- Reply with the results.

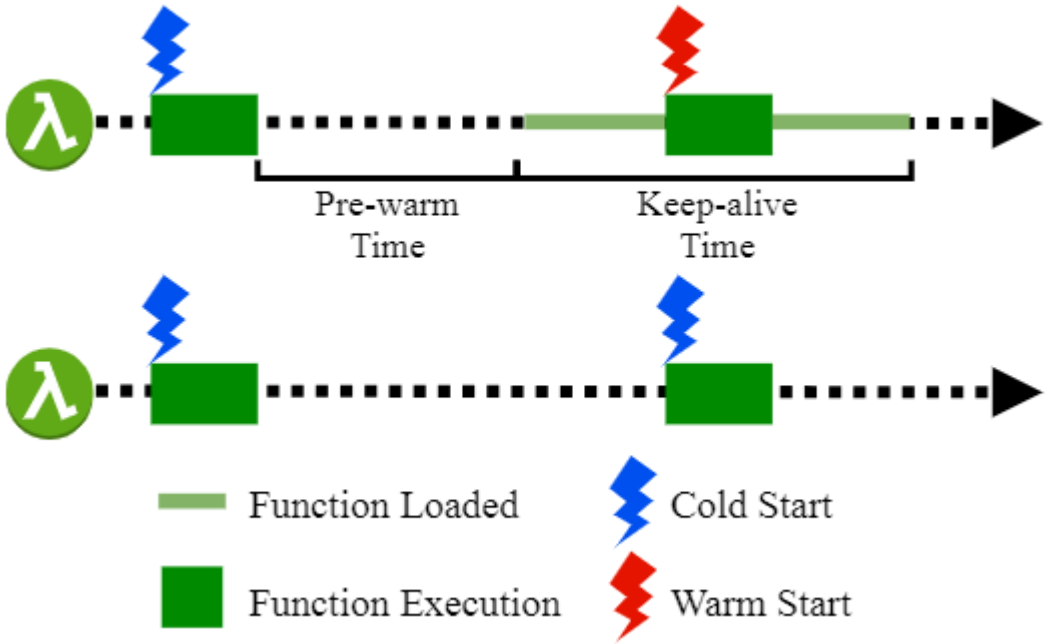
Speedup: pre-create some containers in memory.

Cold Start: Function is invoked without any instance of it loaded.

Cold starts are inevitable!

Background – Scheduling Problem

Scheduling Method



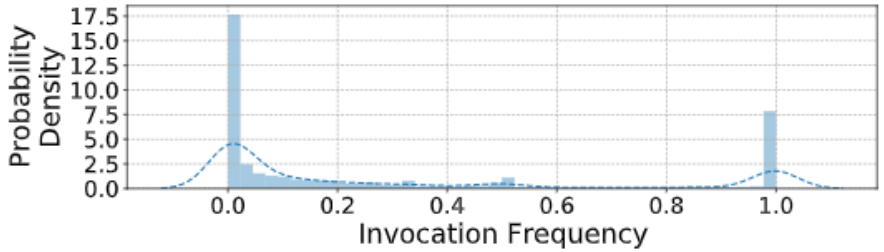
Determine two parameters:

1. Pre-warm time
2. Keep-alive time

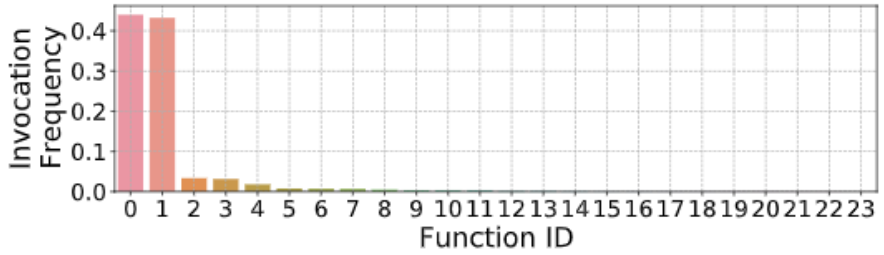
Background – Current Method

Coarse-grained Scheduling

- Schedules functions at the granularity of applications.
- Not all functions in an application are required for an invocation.



(a) Histogram of Function Invocation Frequency



(b) Invocation Frequencies of Functions in an Application

64.7% of functions are invoked less than 0.25.

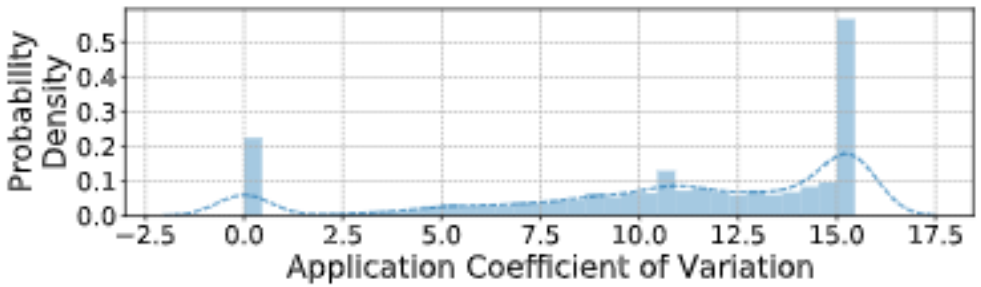
Problems:

1. Memory waste.
2. Increased cold-start overhead.

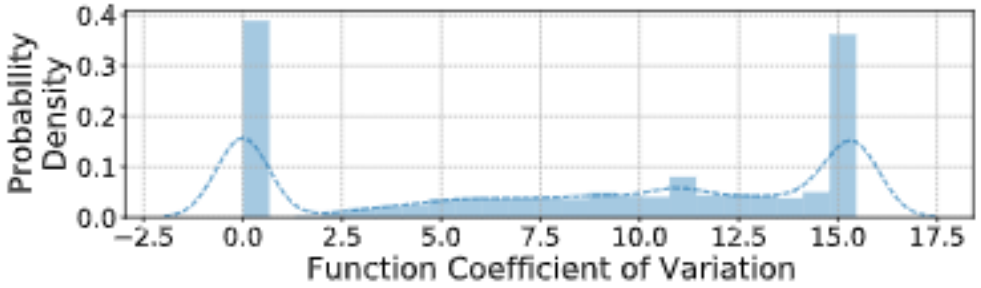
Background – Current Method

Unpredictable functions/applications

- Unpredictable functions are ubiquitous.



(a) CV Histogram of Applications



(b) CV Histogram of Functions

14% applications have CV less than 5.
32% functions have CV less than 5.

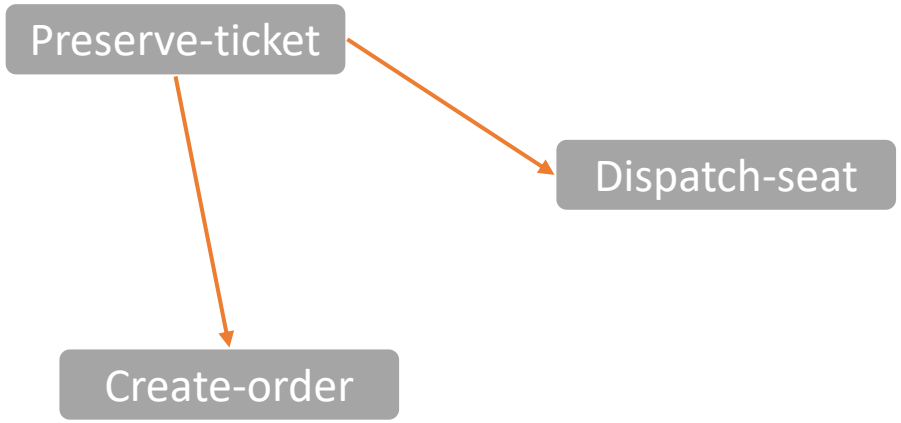
Problems:

1. Cold starts incurred by them.
2. Cannot schedule in a finer granularity.

Background – Function Dependency

Dependencies of Serverless Functions

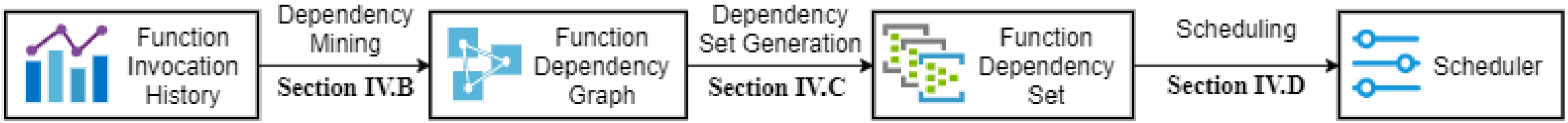
- Serverless functions are APIs.
- The usage pattern of clients -> Dependency



Implications:

1. Can be leveraged to reduce memory wastes.
2. Can be used to solve unpredictable functions.

Approach


Function Invocation History
Dependency Mining
Section IV.B
Function Dependency Graph
Dependency Set Generation
Section IV.C
Function Dependency Set
Scheduling
Section IV.D
Scheduler

Approach – Dependency Mining

Definition of Two Types of Dependencies

- **Strong Dependency:** Function f_a and function f_b have strong dependency iff. 1) they belong to the same client and 2) there is high probability of them being simultaneously invoked in a small time window.

- **Weak Dependency:** Function f_a have weak dependency on function f_b iff. 1) they belong to the same client and 2) there is high probability that f_a is invoked under the condition that f_b is invoked.

Approach – Dependency Mining

Intuitions behind Two Types of Dependencies

- **Strong Dependency:**
  - Bi-directional relationship.
  - Relationship between predictable functions.

- **Weak Dependency:**
  - Uni-directional relationship.
  - Relationship between unpredictable and predictable functions.

Approach – Dependency Mining

Strong Dependency Mining

- **Frequent Pattern Mining**
  - Divide invocation records of functions of a client into time bins.
  - For each time bin:
    - Construct a transaction as all the functions invoked during that bin.
  - Conduct frequent pattern mining on the constructed transactions.
  - Repeat for each user.

- **Frequent itemset**
  - Functions will be invoked with probability greater than the support.

Approach – Dependency Mining

Weak Dependency Mining

- **Positive Point-wise Mutual Information**
  - Suppose the probability of function f_a and f_b being invoked individually is P_a and P_b.
  - The probability of them being invoked together is P_{ab}.
  - $PPMI(f_a, f_b) = \max(0, PMI(f_a, f_b))$
  - $PMI(f_a, f_b) = \log_2 \frac{P_{ab}}{P_a \cdot P_b}$

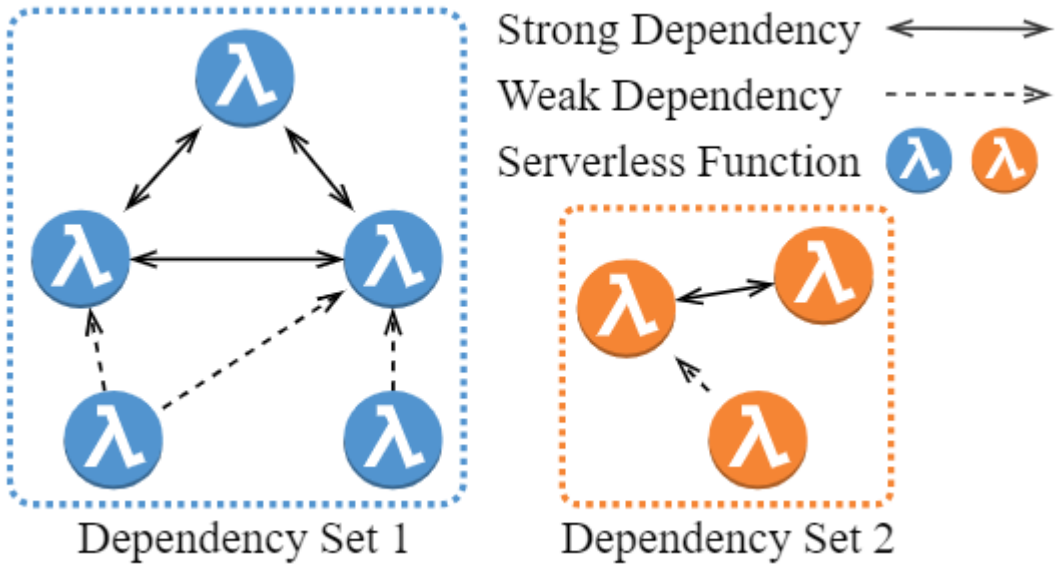
- **Intuition:**
  - $PMI(f_a, f_b) > 0$ -> $P_{ab} > P_a * P_b$ -> f_a depends on f_b.

Approach – Dependency Mining

Weak Dependency Mining

- Use Coefficient of Variation to distinguish unpredictable function from predictable ones.
- Construct a co-occurrence matrix.
- Calculate PPMI for each pair of functions.
- Select top-k as weak dependency.

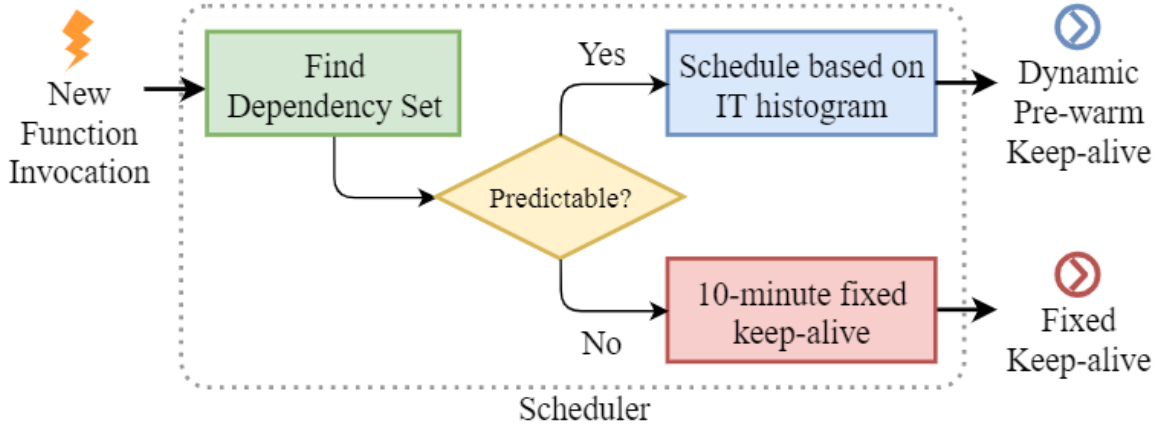
Approach – Dependency Set Generation

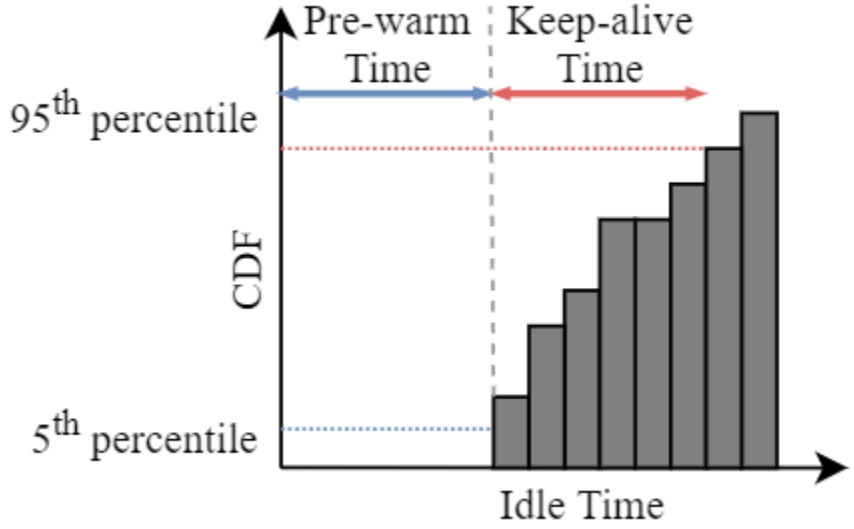


- Generate sets of functions to facilitate scheduling.
- Construct a function dependency graph based on the mined relationships.
- Conduct union-find to generate dependency sets.

Approach – Scheduling

- Similar to [2], use histograms of dependency set invocation intervals.
- Predictable sets:
  - Pre-warm: 5th percentile.
  - Keep-alive: 95th percentile.
- Unpredictable sets:
  - 10-minute fixed timeout.





Evaluation

Evaluation Method:

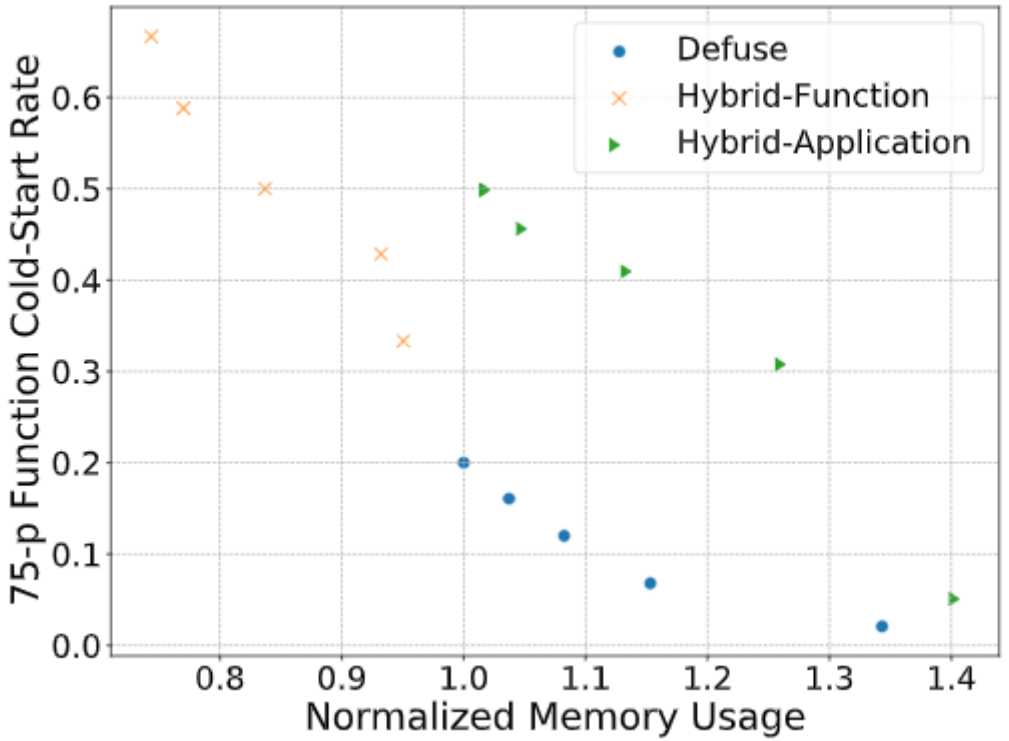
- Conduct simulation on Azure function dataset released by [2].

Baseline Methods

- Hybrid-Application[2]
- Hybrid-Function: Directly apply [2] to the function level.

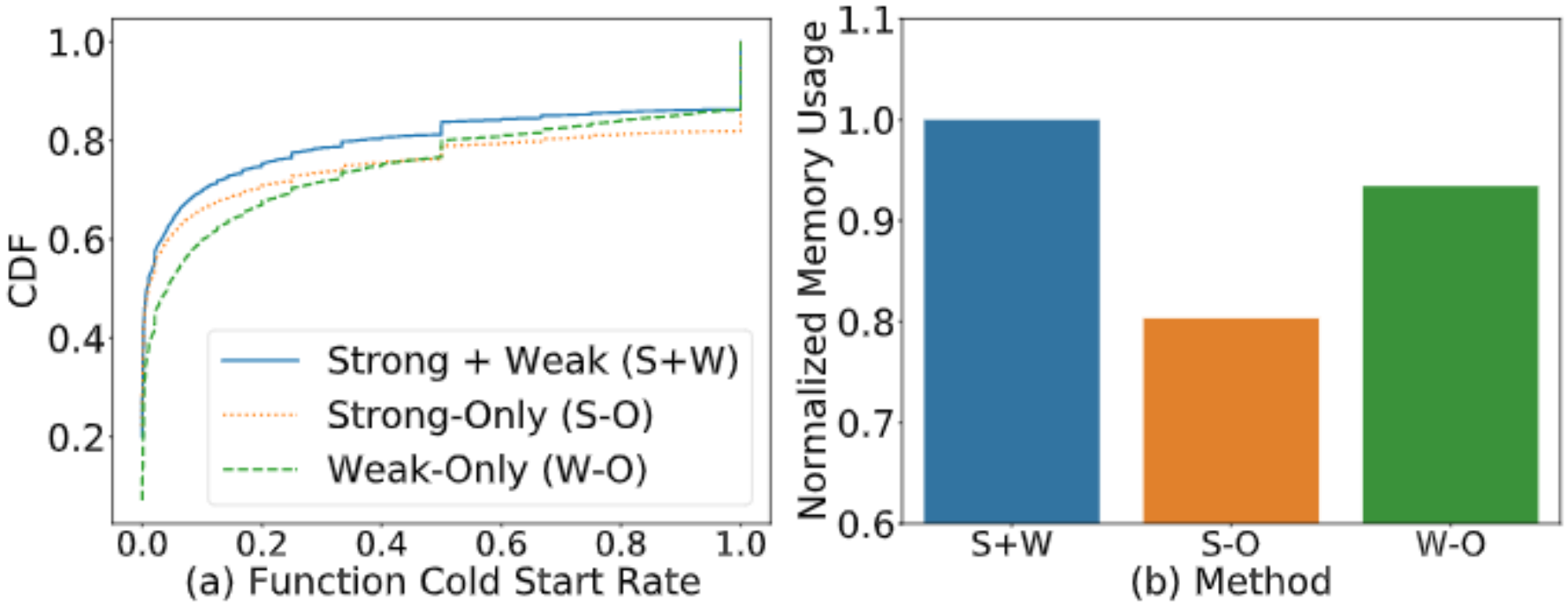
Evaluation

How effective is Defuse compared with other scheduling methods?



Evaluation

What are the contributions of weak and strong dependency?



Conclusion

- The first method to schedule serverless functions based on their dependencies.
- Add another dimension to the cold-start mitigation on FaaS platforms.
- Compatible with current scheduling methods.

Thank you!

香港中文大學
The Chinese University of Hong Kong