Defuse

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

Introduction

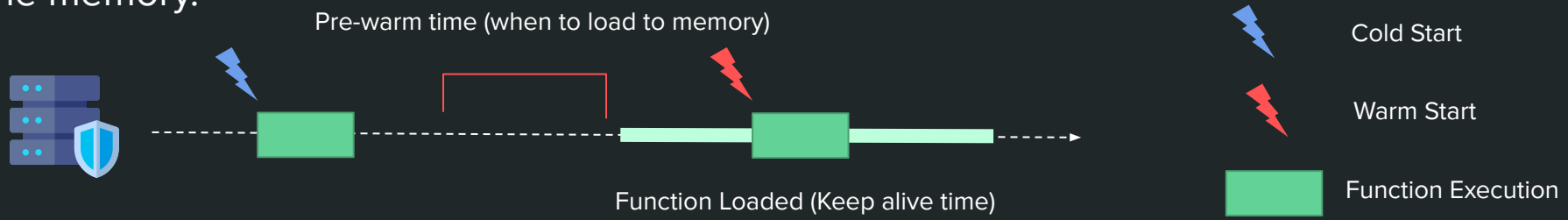
Understanding the problem. Understanding the solution’s approach.

The problem

FaaS platforms suffer from the performance degradation caused by the cold starts of serverless functions.

Cold Start

Happens when serverless functions are invoked before they have been loaded into the memory.



Trade-off: Readiness of Serverless Functions and Memory Consumption

When serverless function is invoked, if a container with the function has been initialized in memory, the function can be executed instantly. However, if the container is not loaded the latency of the function invocation will degenerate greatly because container initialization is time-consuming.

Current Reality

Current methods on cold start mitigation can be classified into two categories:

1. System-level optimizations
   Reducing the time spent on executing a cold start.

2. Serverless functions scheduling
   Focus on reducing the occurrences of cold starts according to their invocation histories

Two issues are identified by the analysis of the current scheduling methods (2):

I. Scheduling coarse granularity
   Current scheduling methods schedule all the serverless functions in an application as a whole. The idle functions, i.e. those that are loaded but not invoked, inevitably waste memory in FaaS platforms.

II. Unpredictable invocation behaviors
   Current scheduling methods cannot cope with applications without clear invocation patterns, thus deteriorating the overall performance of Faas platforms.

An observation

Clients compose serverless functions into complex applications. That composition forms dependencies among those functions. There are two aspects regarding serverless function dependency definition:

A. Global frequently invoked functions, that are usually invoked together and predictable.

B. Ubiquitous unpredictable functions.

The idea

Dependencies are the patterns when clients invoke serverless functions. What if those dependencies among serverless functions are leveraged to improve current scheduling methods?

What can be done with these dependencies?

For functions that are usually invoked together (A) we can schedule the dependent functions as a whole. Cold starts can then be reduced because these functions will naturally be invoked together.

For functions that are unpredictable (B) we can relate unpredictable functions to predictable ones. In that way, cold starts incurred by unpredictable functions can be diminished with the help of the predictable invocated patterns.

The approach of Defuse

Exploit the dependencies among serverless functions to reduce memory waste and decrease the occurrences of cold starts. Compared with loading the whole application, memory can be saved by only loading those necessary functions.

However, there are challenges in leveraging the dependencies to schedule serverless functions:

1. How to reveal the dependencies among the serverless functions ?
2. How to cope with unpredictable functions?
3. How to have comparable cold start performance while having less memory usage compared with coarse-grained scheduling methods?

Dufuse tackles the above challenges and offers a dependency-guided function scheduler to mitigate those cold starts.

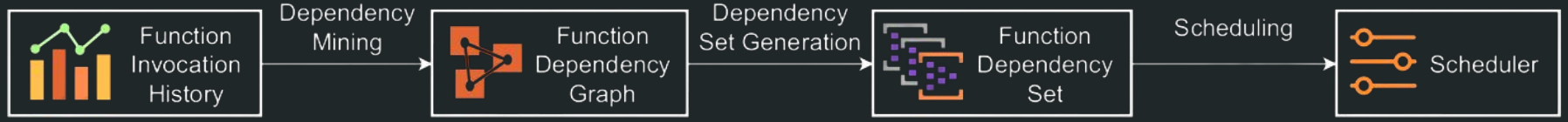
But, will that work ?

Experimental results show that compared with baseline methods, Defuse reduces 35% of function. Cold-start rate while cutting 22% of memory usage.

Design & Implementation

How Dufuse works internally ?

The Defuse Workflow



The above figure shows the overview of Defuse workflow, which is divided into 3 steps:

1. Dependency Mining: Defuse takes the invocation histories of serverless functions as the input and conducts dependency mining on these invocation records, constructing a Function Dependency Graph.

2. Dependency Set Generation: Based on the Function Dependency Graph, defuse generates Function Dependency Sets.

3. Dependency-Guided Scheduling Policy: Given the input of the Function Dependency Sets, all serverless functions in these set are scheduled as a whole.

FaaS platforms can then decide when to load a dependency set and how long to keep in the memory based on the scheduling policy

1. Dependency Mining

In this step, Defuse discovers the dependencies among serverless functions. There two types:

- Strong Dependency: When two functions belong to the same client and there is high probability of them being simultaneously invoked in a small time window. It is bidirectional dependency and describes frequently invoked functions that are predictable.

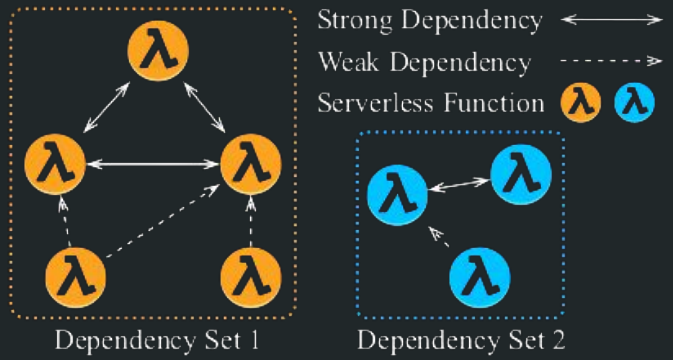
  Discovery:
  Frequent Pattern Mining (given a set of transactions, find all the itemsets with frequency greater than a given threshold) with the help of FP-Growth algorithm.

- Weak Dependency: When two functions belong to the same client and there is high probability that a function is invoked under the condition that another function is invoked. It is a single directional relationship and describes dependencies between unpredictable and predictable functions.

  Discovery:
  Distinguishing unpredictable functions with predictables ones with the Coefficient Variation (CV) of Idle Time (IT) histograms. Costructs co-occurrence matrix. Calculates Positive Point-Wise Mutual Information (PPMI). Assigns top-k predictable functions to be weekly dependent on the unpredictable functions.

2. Dependency Set Generation

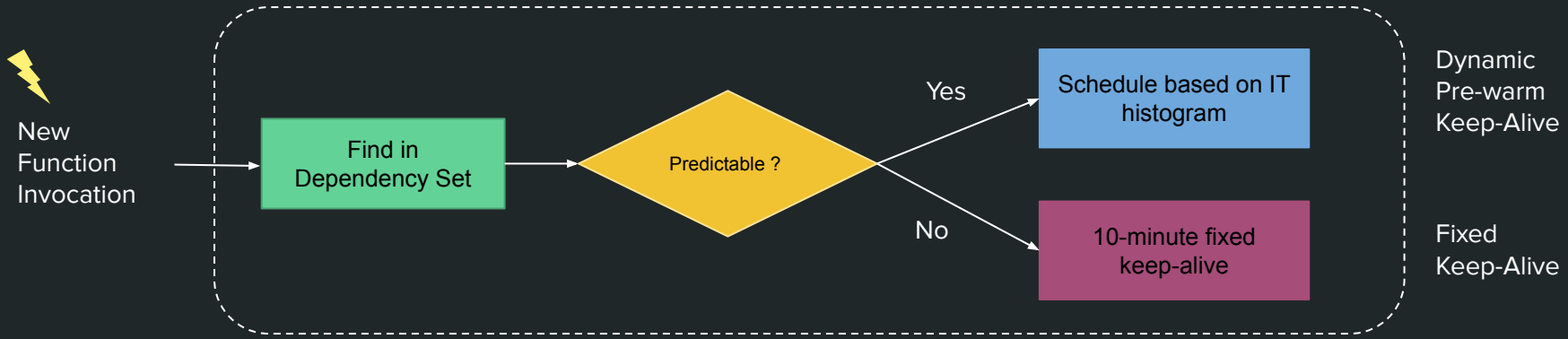
Mined dependencies are relationships among serverless functions, which cannot be directly exploited. Defuse constructs a Function Dependency Graph, where each vertex is a serverless function and each edge represents either a strong or weak dependency. Then, uses union-find to extract all these connected components and groups them as dependency sets.



Dependency sets imply that connected functions will have a high probability of being jointly invoked, thus scheduling functions - that are in a dependency set - together reduces the occurrences of cold starts .

3. Scheduling

For each Dependency Set, the scheduler needs to decide 1) when to pre-warm it by loading it into the memory (pre-warm time) and 2) how long to keep it in memory (keep-alive time). Each Dependency Set is distinguished to predictable and unpredictable sets based on their CVs.



The scheduling based on IT histogram is utilized with the Cumulative Distribution Function (CDF), by setting the 5th percentile of the IT histogram as the pre-warm time and the time between the 5th and 95th percentile as the keep-alive time. Defuse, can also uses more aggressive timeout settings and different policies for different case.

Experimentation & Results

Answering research questions with statistics.

Experiment Settings

To evaluate the cold-start performance of Defuse, a number of simulations were conducted on the Azure Public Dataset, which contained a 14-day function invocation record. The evaluation metrics included the memory usage, the cold-start performance and the cost of cold starts of different scheduling policies (Defuse, Hybrid-Application, Hybrid-Function).

Effectiveness of Defuse

Compared with the two baseline methods, that of Hybrid-Application and Hybrid-Function, Defuse exhibits the best cold-start performance. Compared with the Hybrid-Application method, the memory usage of Defuse is reduced. The memory consumption of Defuse exceeds that of the Hybrid-Function method. This happens because Hybrid-Function schedules at a finer granularity. But the improvement in function cold-start rate outweighs the increase in the memory usage.

When it comes to overhead, compared with the Hybrid-Application method, Defuse reduces the average number of loading functions by 79%. Defuse utilizes the memory more efficiently compared with the Hybrid-Application scheduling method. The efficiency comes from the fact that combining strong and weak dependencies can guide memory usage.

Conclusion

Summing up.

To sum up

Cold start of serverless functions is a server problem on FaaS platforms. Defuse takes the dependencies among serverless functions into consideration in the scheduling process. This utilization of dependencies makes it possible to schedule functions in a finer granularity while reducing the cold-start rates.

Defuse is complementary to existing col-start mitigation approaches and is compatible with arbitrary scheduling policies as well as system improvements that reduce the cold-start time.

Experiments show that Defuse can effectively reduce memory consumption while having smaller function cold-start rates compared with the baseline methods.

THANK YOU!

Fair Use Notice & Disclaimer

The current presentation falls under fair usage for educational purposes only and was conducted for a paper presentation (Authorized licensed use limited to: Athens University of Economics and Business).

Paper Reference

Defuse: A Dependency-Guided Function Scheduler to Mitigate Cold Starts on FaaS Platforms
Jiacheng Shen∗ , Tianyi Yang∗ , Yuxin Su∗ , Yangfan Zhou †‡ , and Michael R. Lyu∗
Department of Computer Science and Engineering,
The Chinese University of Hong Kong, Hong Kong, China.