ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
CUHK

SPES: Towards Optimizing Performance Resource Trade-Off for Serverless Functions

C. Lee[*], Z. Zhu[†], T. Yang[*], Y. Huo[*], Y. Su[‡], P. He[†], and M. R. Lyu[*]

*The Chinese University of Hong Kong, †The Chinese University of Hong Kong (ShenZhen), ‡ Sun Yat-sen University

5/14/2024

Serverless Computing

Serverless computing provides backend services on an as-used basis.

Open API

Services

- Operations
- Invocation

Cloud Events

Events

- Produced
- Consumed
- Correlations

Workflow Patterns

Control Flow

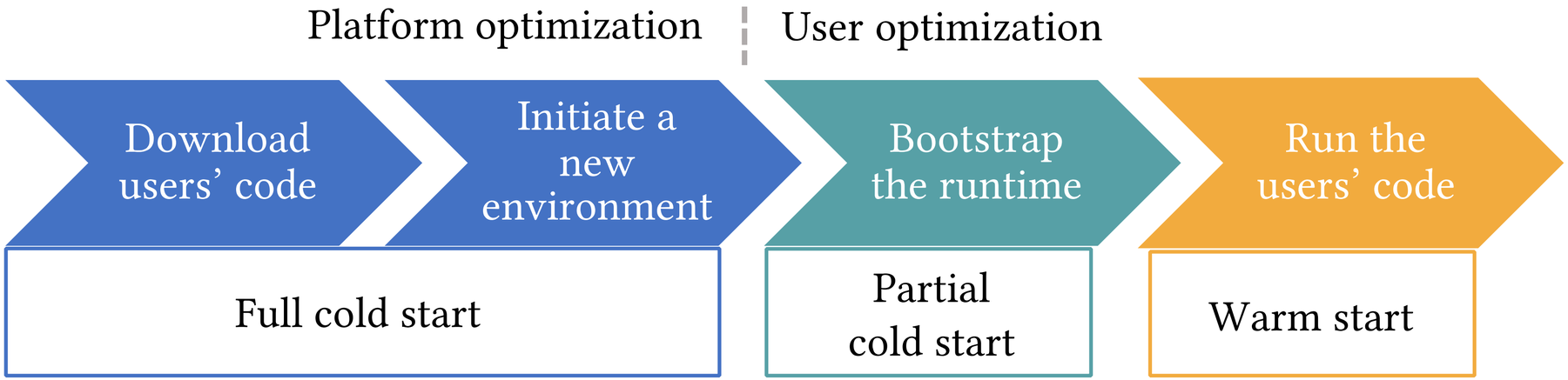
- Execution Order
- Error Handling
- Data Management

aws Lambda Google Cloud Functions Microsoft Azure Alibaba Cloud snowflake

Cold Start

Cold Start: A new function environment being initialized in response to a request, causing latency spikes

Cold-start latency can account for 80% of the total response latency



Main Goal: Design a scheduler of function instances to reduce cold starts without wasting too much memory!

Challenges

Efficiency

- Millions of available functions
- Immediate decisions for thousands of invocations every minute

Scalability

- Bursty and dynamic invocations
- Require elastic scaling to respond to invocation spikes

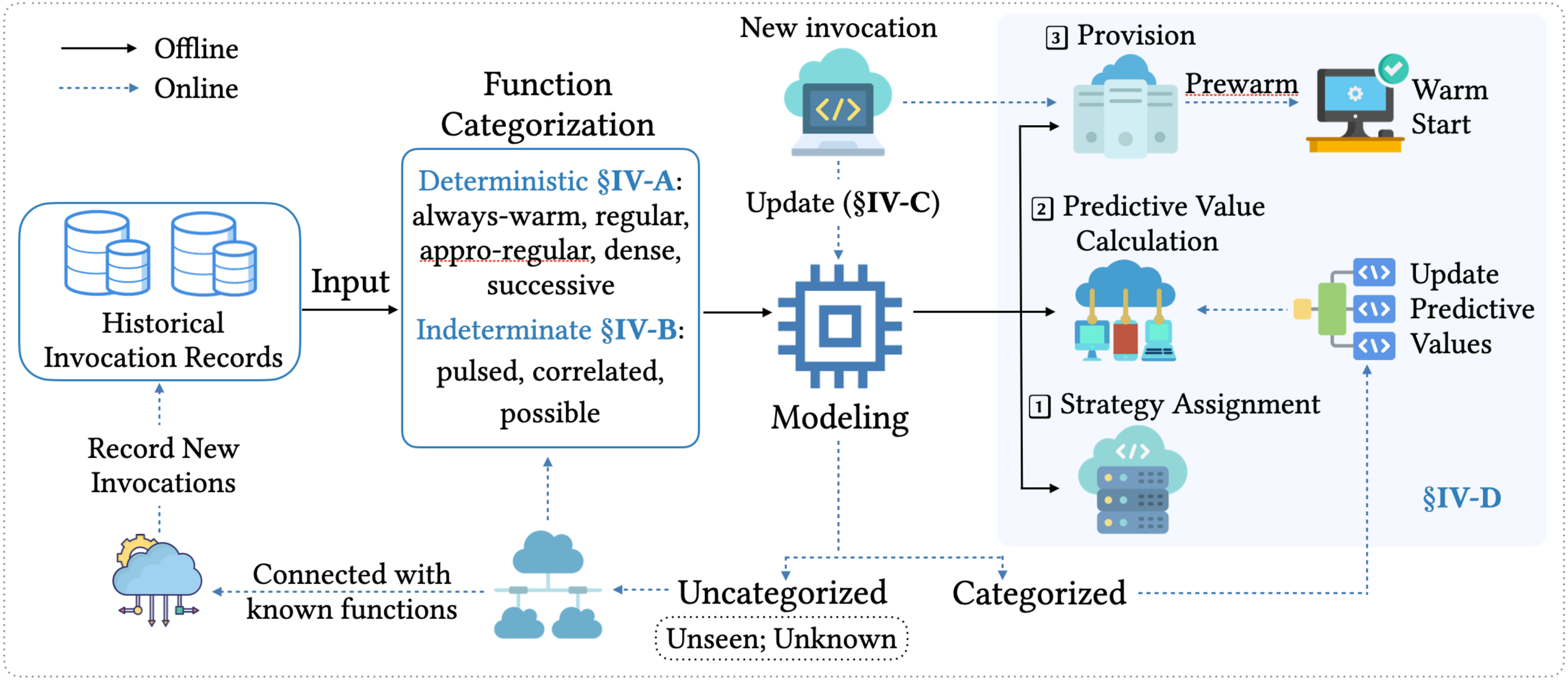
Imbalance

- Diverse implementations and events lead to an extremely uneven invocation distribution

Evolution

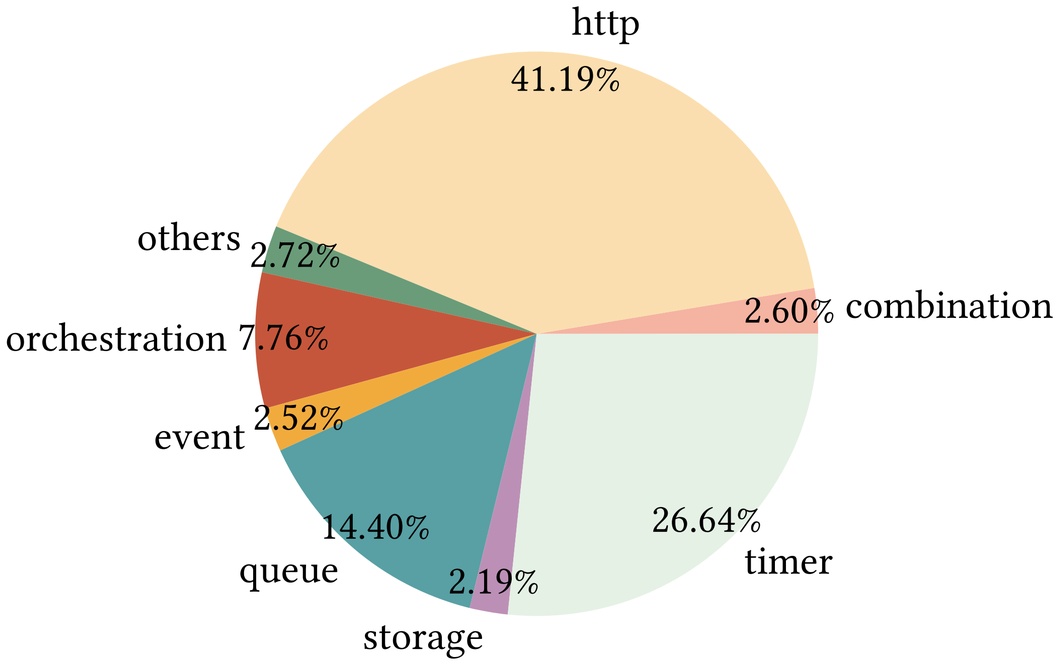
- Invocation behaviors evolve as business needs change
- Functions can be re-bound to new triggers or a calling chain

Overview


Offline
Online
New invocation
Function Categorization
Deterministic §IV-A: always-warm, regular, appro-regular, dense, successive
Indeterminate §IV-B: pulsed, correlated, possible
Update (§IV-C)
3 Provision
Prewarm
Warm Start
2 Predictive Value Calculation
Update Predictive Values
1 Strategy Assignment
§IV-D
Historical Invocation Records
Input
Modeling
Record New Invocations
Connected with known functions
Uncategorized
Unseen; Unknown
Categorized

Insight

Event triggers lead to serverless invocations. Studying triggers beyond the surface numbers of invocations can mine patterns better and model invocation behaviors.

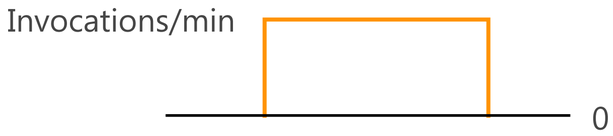


- 68.12% timer-triggered functions are invoked periodically or quasi-periodically
- 45.02% HTTP-triggered invocations follow a Poisson arrival process

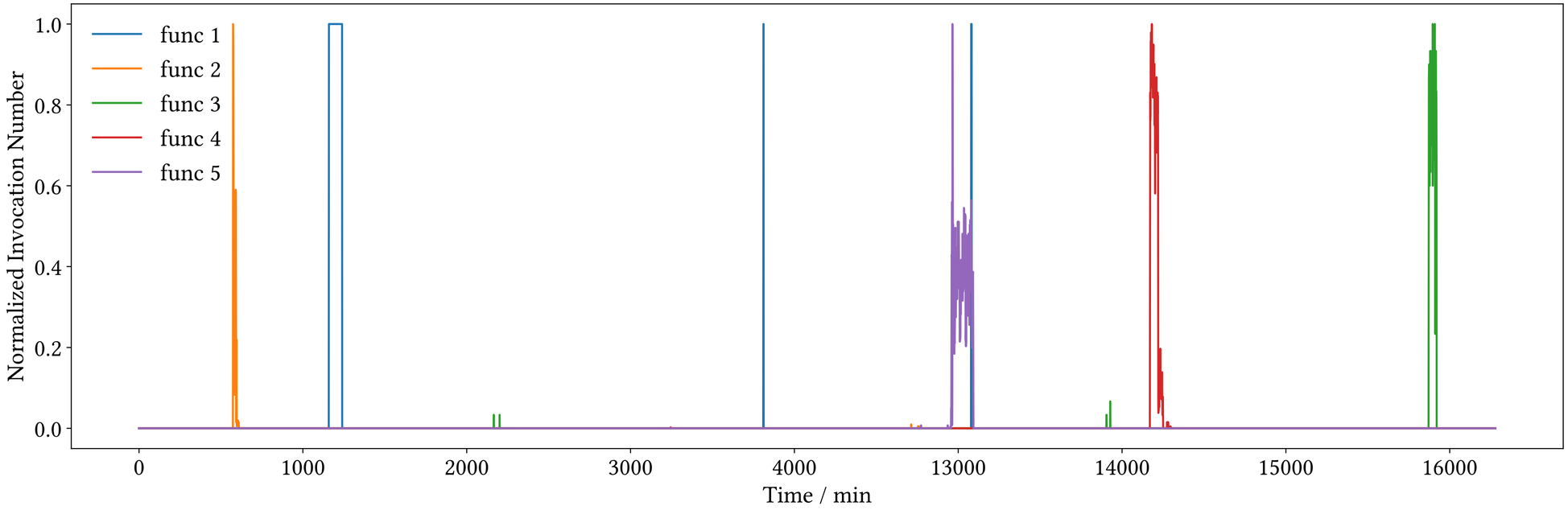
Insight

Infrequently invoked functions may experience short bursts

Temporal locality

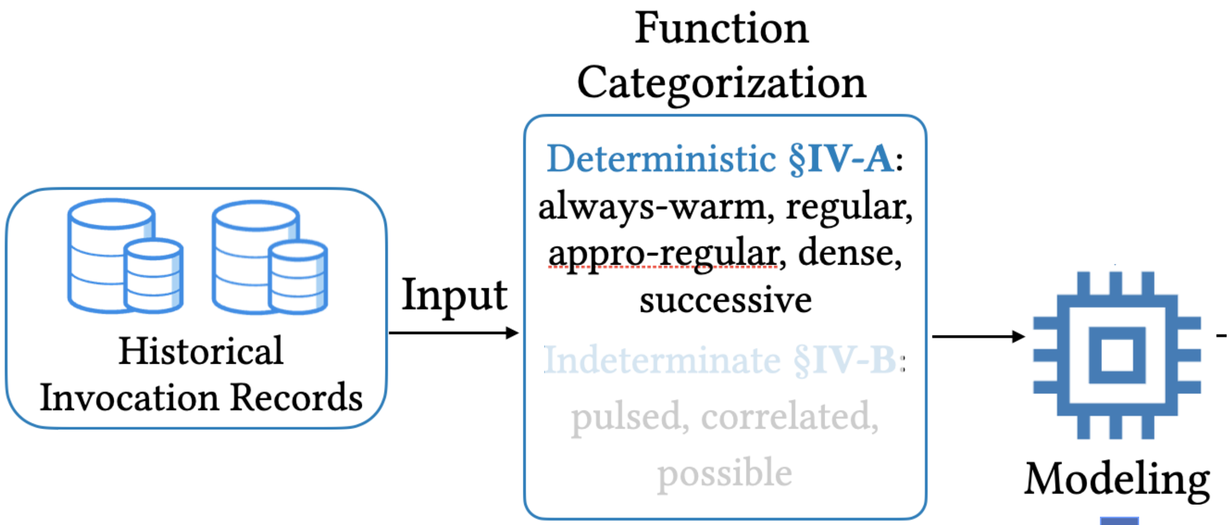


Keep for a while once invoked



Simple Keeping a loaded instance for a while can reduce such cold starts without wasting too many resources.

Deterministic Function Categorization



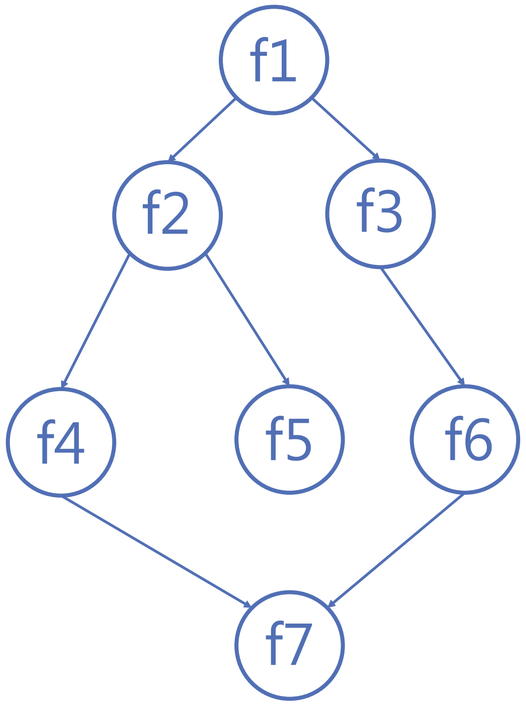
Function Categorization

Always Warm	Regular	Approximatively (Appro-) Regular	Dense	Successive
(Almost) being invoked all the time	(Almost) being invoked periodically	Invocation intervals converge to a few values	Instance won't be long-term idle	Inactive functions has short-term, consecutive invocation sequences (temporal locality)

Insight

Functions may work in a certain logic workflow → Correlated invocations ← measure **Co-occurrence rate (COR)**

$$\mathrm{COR}(f1) = \frac{\mathrm{Count}(f1, f2)}{\mathrm{Count}(f1)}$$

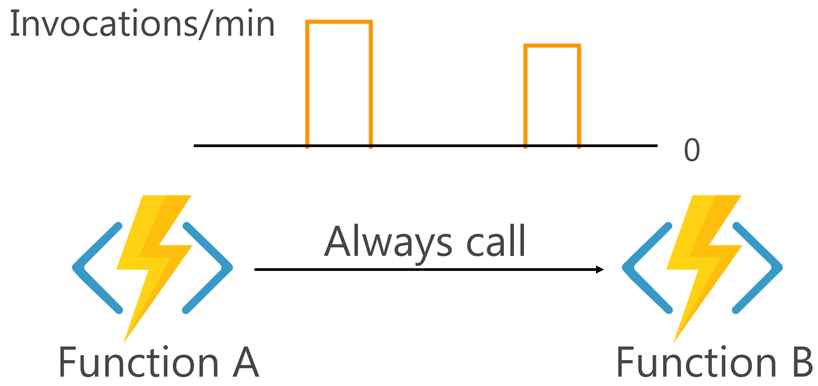


- Functions with the same user or application tend to have a higher COR.
- Among them, functions have a more significant convergence with the same trigger.

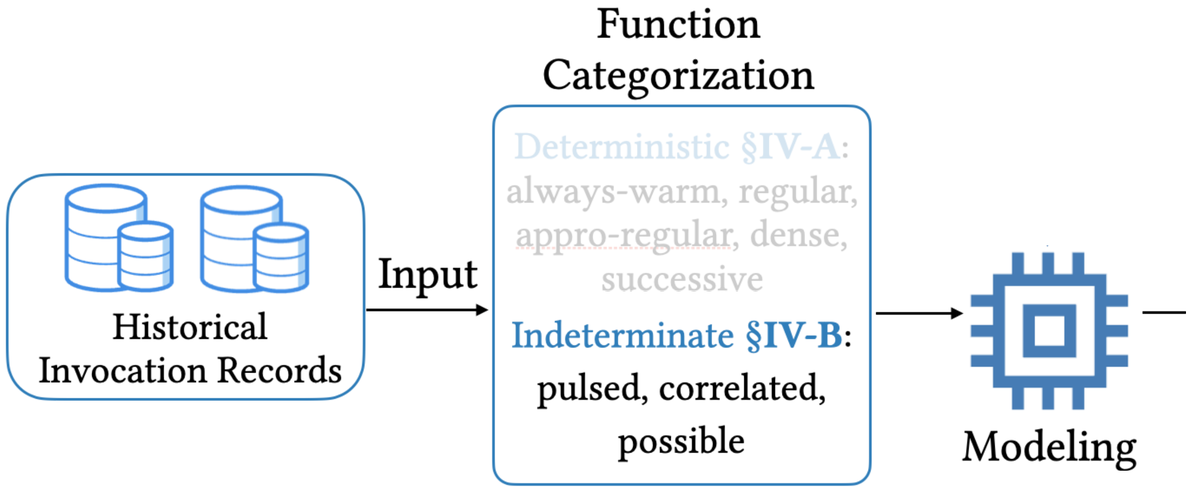
Indeterminate Function Assignment

Use validation results to assign strategies to functions

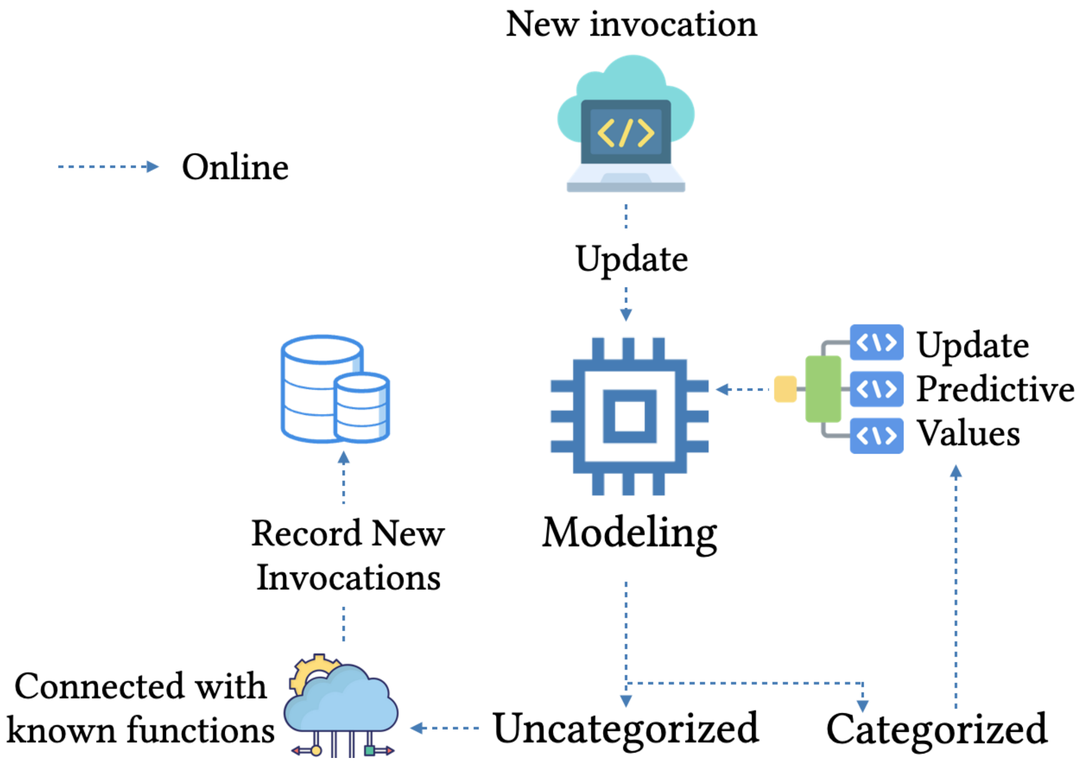
- **Pulsed:** tolerating a cold start and keeping the function warm until its ideal time reaches a pre-defined threshold. (temporal locality).
- **Correlated**: if function A is always invoked before function B, then B's invocations can be predicted by the occurrence of A's invocations. (measured by co-occurrence rate, COR)
- **Possible**: provisions infrequently invoked functions by regarding its interval mode as a fixed interval.



Unknown: functions that have never been invoked during the validation period.



Adaptive Strategies for Concept Shifts



Update predictive values

Record online invocations to update the predictive values

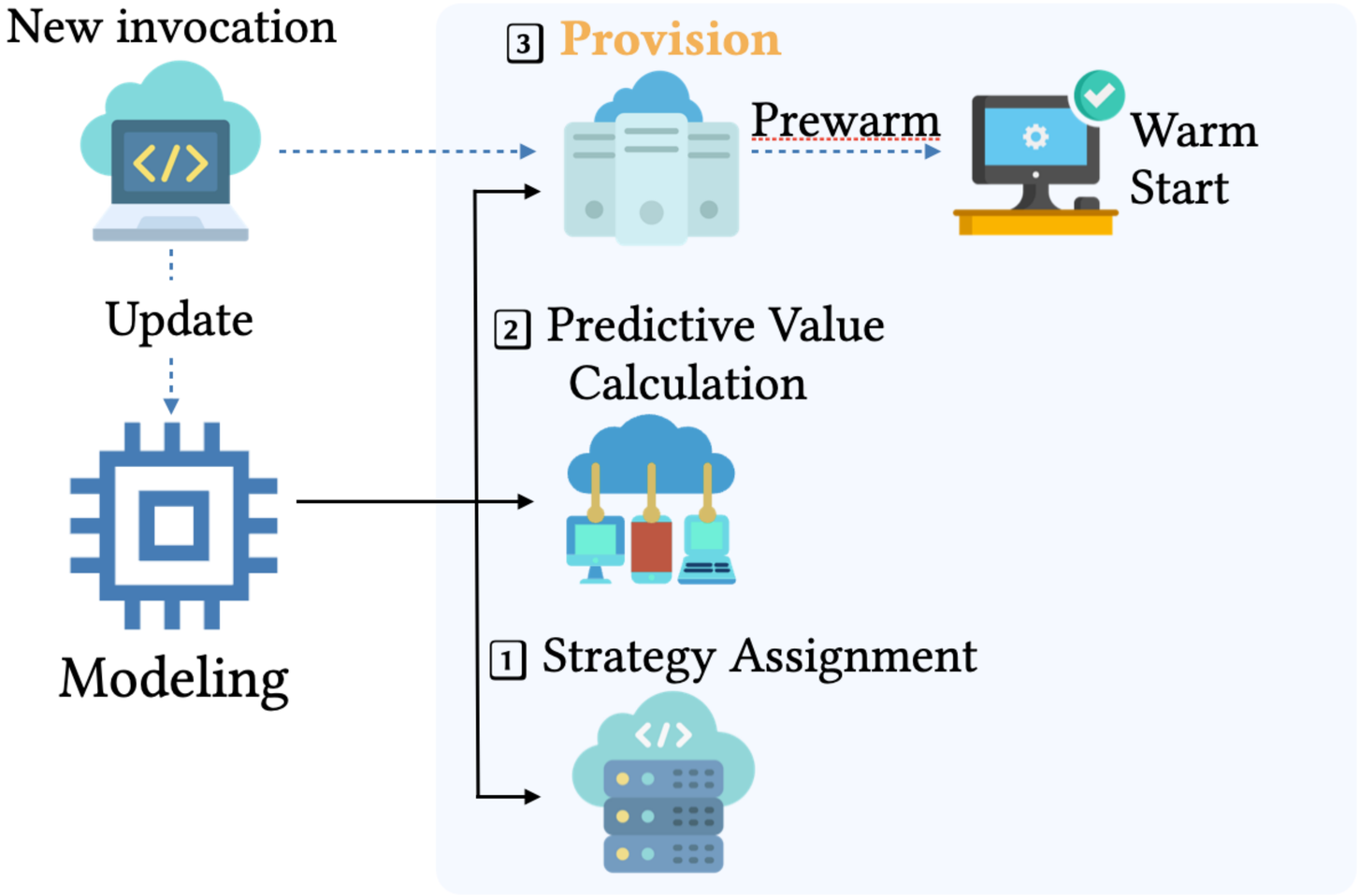
Update categorization

Categorize unknown functions according to online WTs

Update correlation

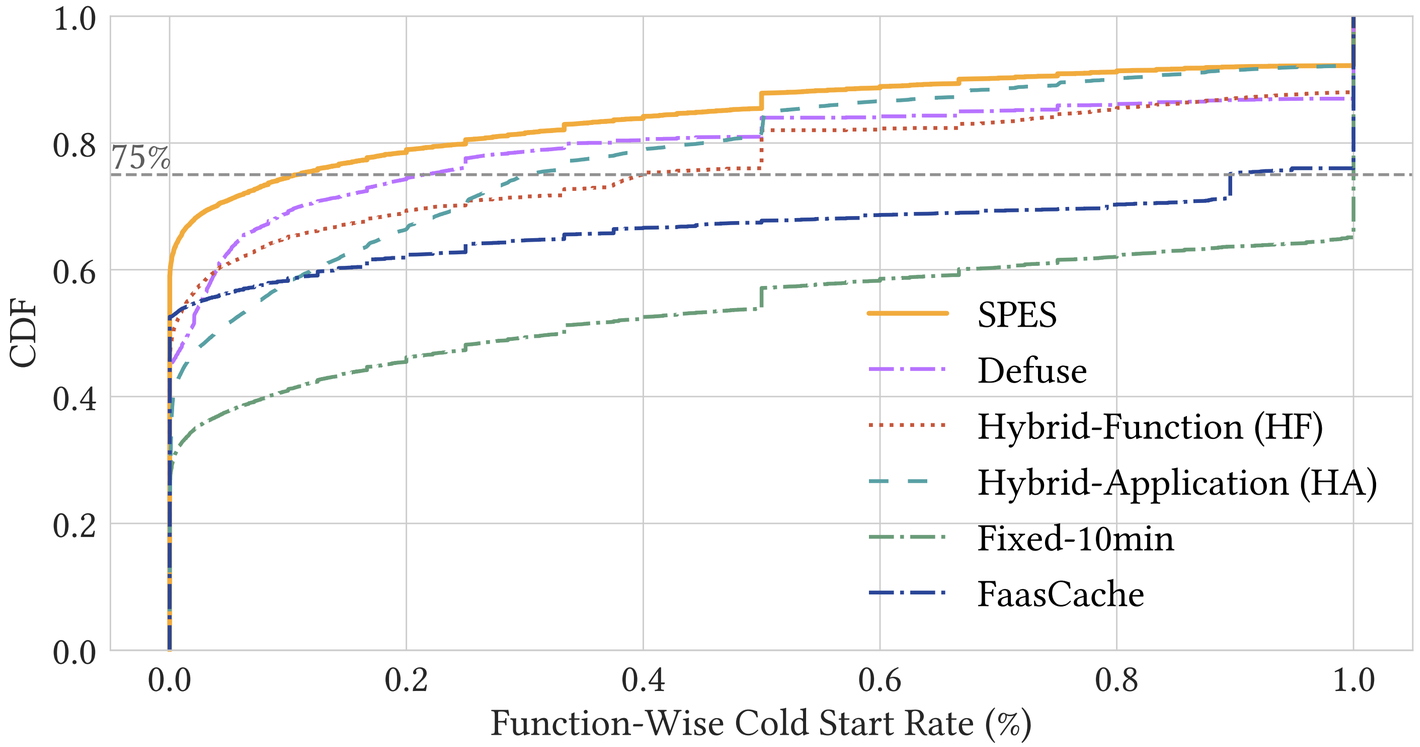
Correlate unknown functions with known, categorized functions using COR

Provision

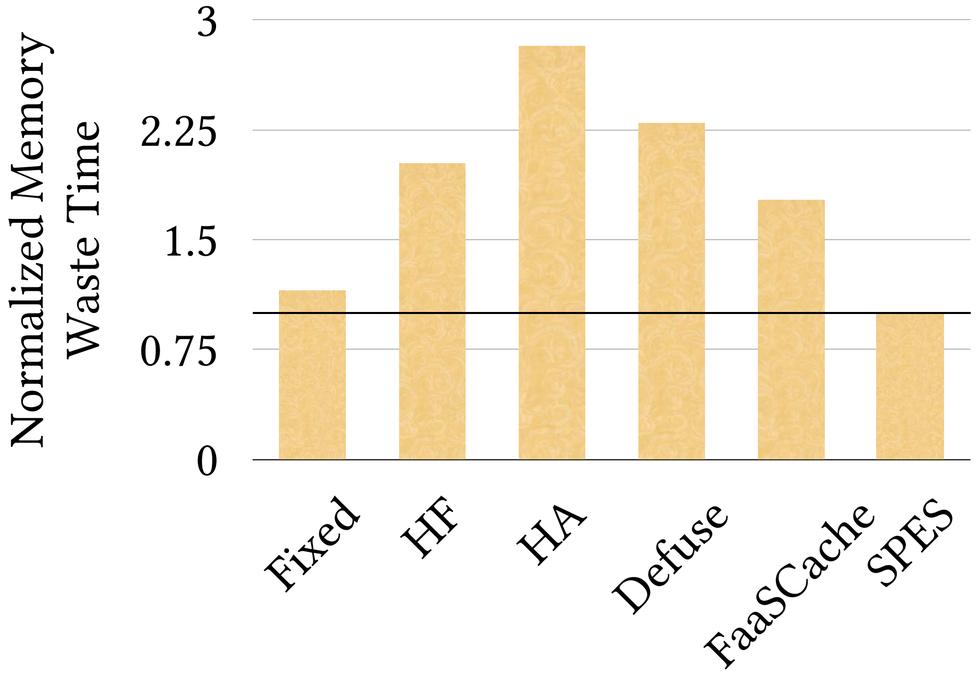

New invocation
Update
Modeling
3 Provision
Prewarm
Warm Start
2 Predictive Value Calculation
1 Strategy Assignment

SPES optimizes the trade-off

SPES reduces the 75th percentile cold-start rate by 49.77%~89.20%
and reduces wasted memory time by 10.89%~63.50%



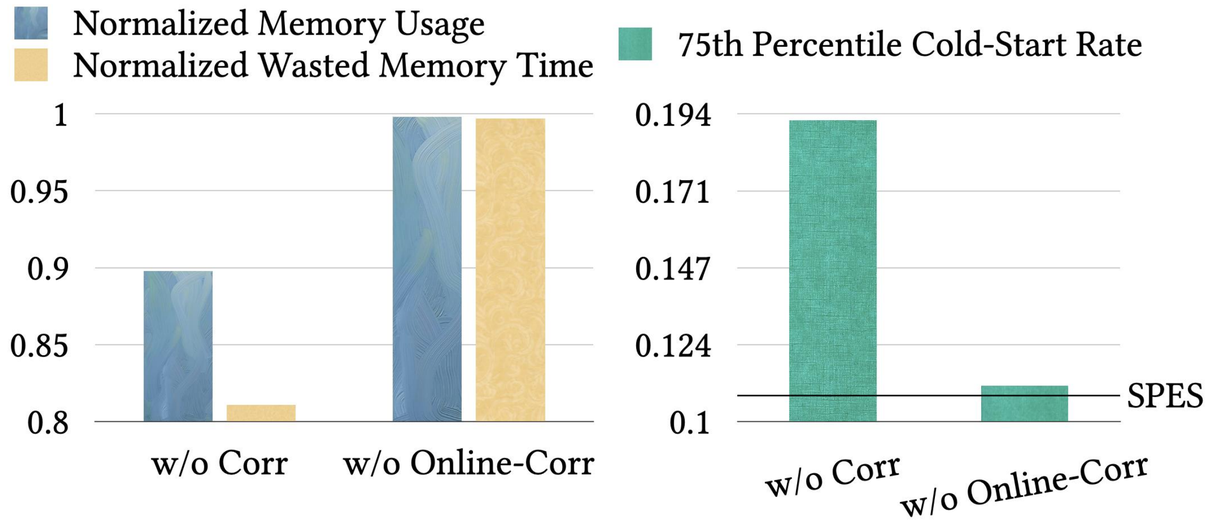
The cumulative distribution (CDF) of function-wise cold-start rates under SPES and baselines



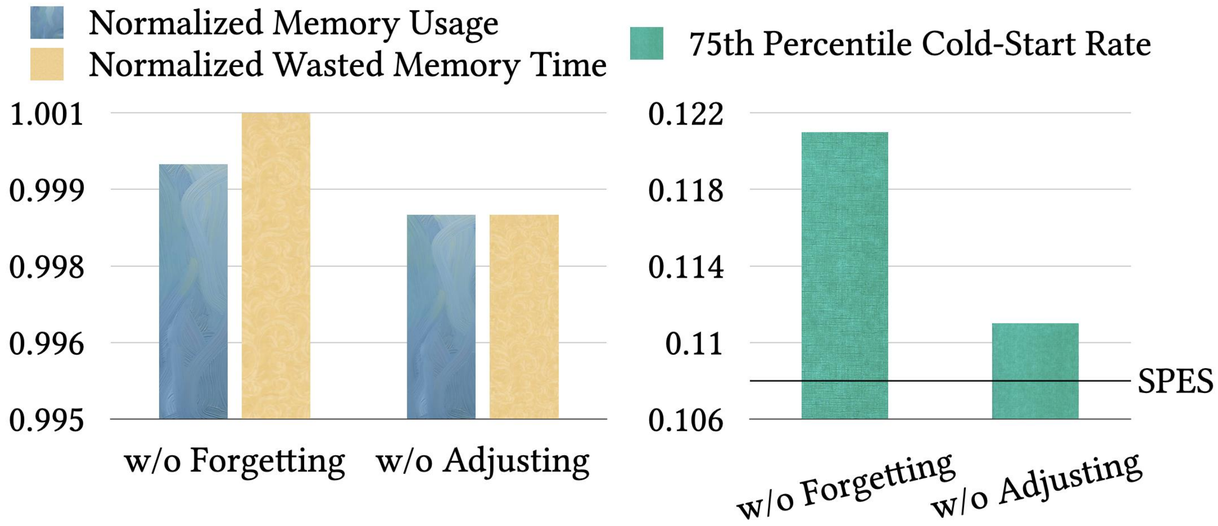
The normalized wasted memory time under SPES and baselines

Ablation Studies

The correlation design and adjusting strategies contribute to SPES



The correlation strategy contributes to the cold-start and memory-waste reduction



The adaptivity of SPES benefits cold-start reduction whereas incurring a little memory waste

CONCLUSION

Insight

- The event-driven architecture makes triggers decide serverless invocations.
- The limited trigger types lead to predictive invocation behaviors.

Solution

- SPES is the first differentiated runtime serverless function provision method.
- SPES categorizes functions and predicts invocations to pre-load/unload instances.

Evaluation

- SPES significantly improves both sides of the trade-off: economizing memory and reducing cold starts.
- Rule-based SPES has remarkable efficiency, scalability, and adaptability.

香港中文大學
CUHK

SPES: Towards Optimizing Performance Resource Trade-Off for Serverless Functions

THANK YOU

Presenter: Cheryl Lee

C. Lee, Z. Zhu, T. Yang, Y. Huo, Y. Su, P. He, and M. R. Lyu

HOMEPAGE

FULL PAPER

ARTIFACT

Email: cheryllee@link.cuhk.edu,hk WeChat: BillionareBTC Mastodon: cheryllee