SMIless: Serving DAG-based Inference with Dynamic Invocations under Serverless Computing

Chengzhi Lu, Huanle Xu*, Yudan Li, Wenyan Chen, Kejiang Ye,
Chengzhong Xu*

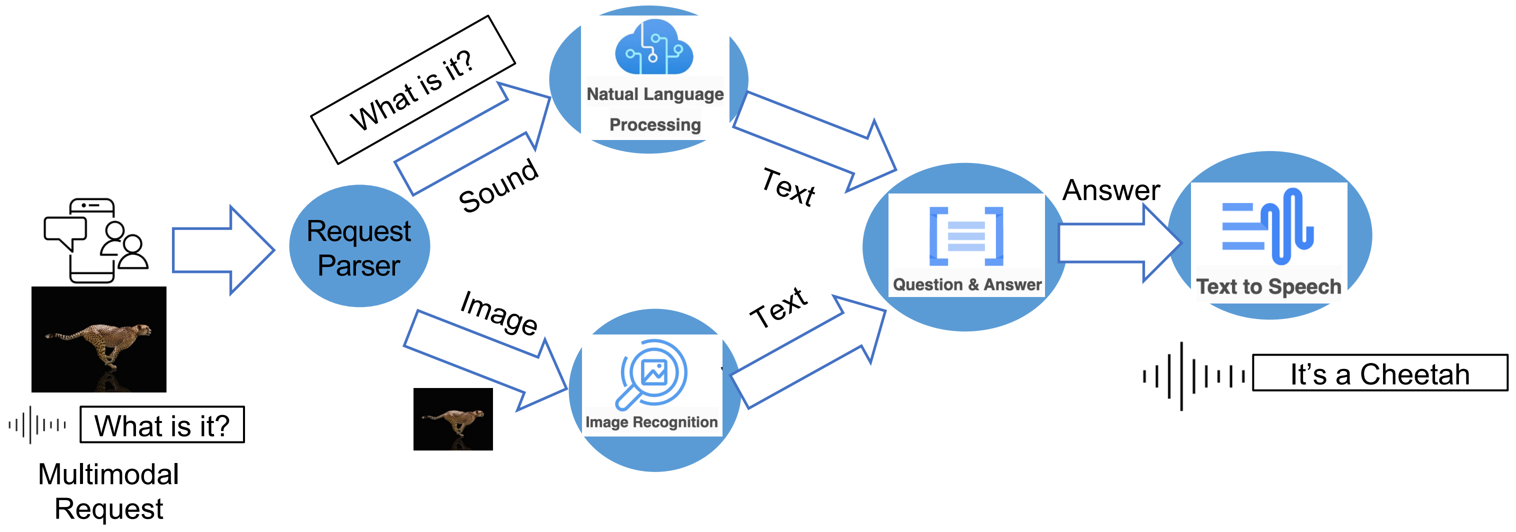
中国科学院深圳先进技术研究院
SHENZHEN INSTITUTES OF ADVANCED TECHNOLOGY
CHINESE ACADEMY OF SCIENCES

中国科学院大学
University of Chinese Academy of Sciences

澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Provide Comprehensive Services

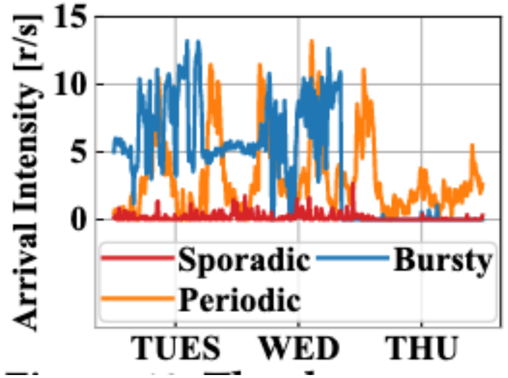
- Multi-stage ML serving application
  - By incorporating multiple inference models



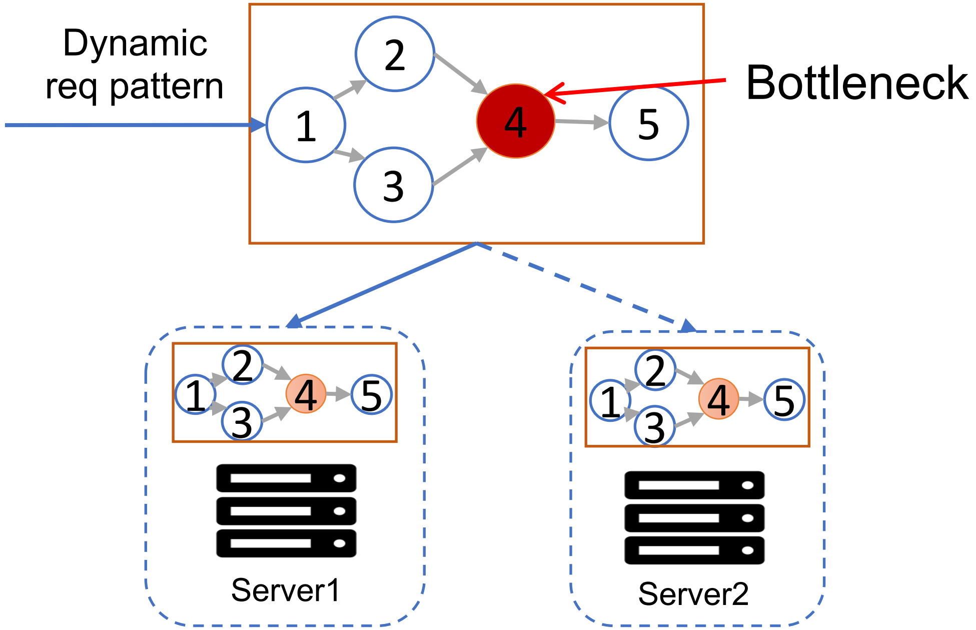
Intelligent Personal Assistant

Leverage Serverless Computing

- ML serving applications suffer from dynamic request patterns
  - Server centric deployment: resource overprovision



Highly dynamic request pattern of ML inference application in a real-world cluster [1]



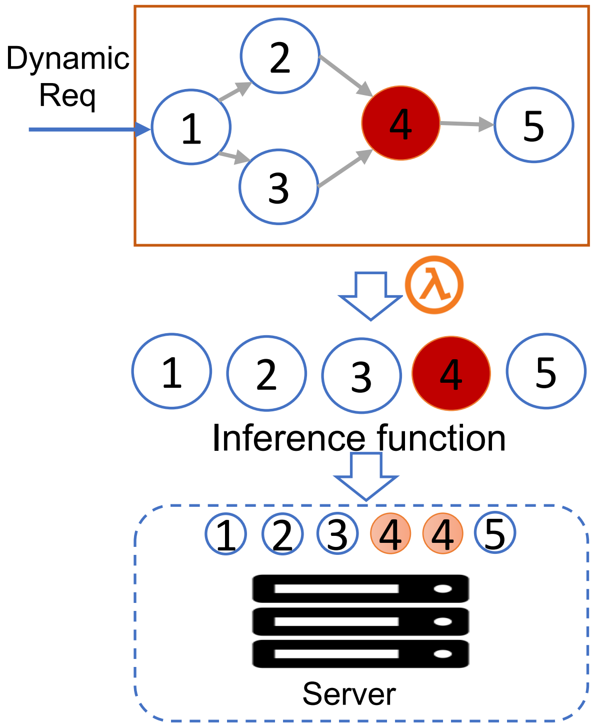
Server Centric Deployment

[1] Yang, Yanan, et al. "INFless: a native serverless system for low-latency, high-throughput inference." ASPLOS'22

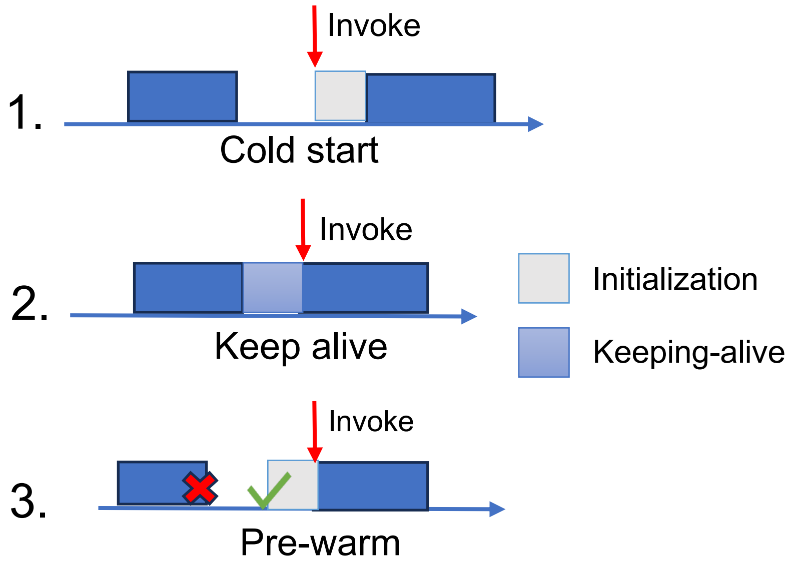
Leverage Serverless Computing

➢ ML serving applications suffer from dynamic request patterns

- Server centric deployment: resource overprovision
- Serverless computing: precisely tailor resource utilization of each function



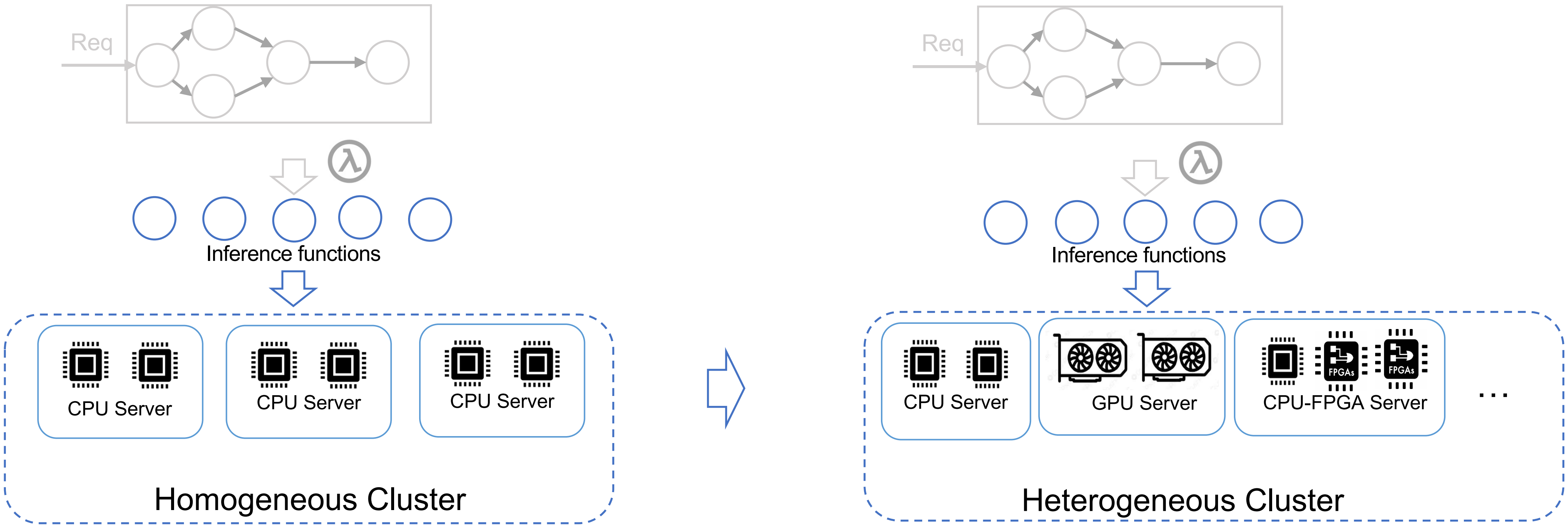
Serverless Computing



Function behavior in serverless computing

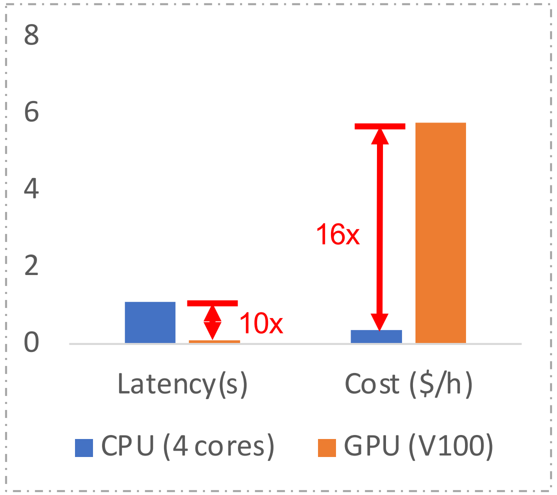
In Heterogeneous Environment

- Underlying hardware resources are undergoing heterogeneity
  - Enhance the performance of ML applications



In Heterogeneous Environment

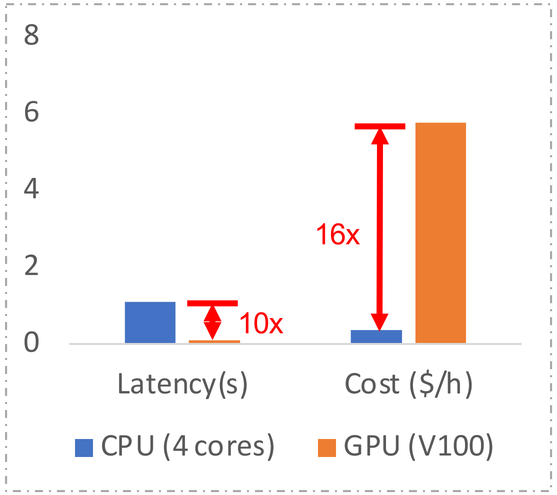
- Get trade off between performance and cost with the heterogeneous hardware for ML serving applications



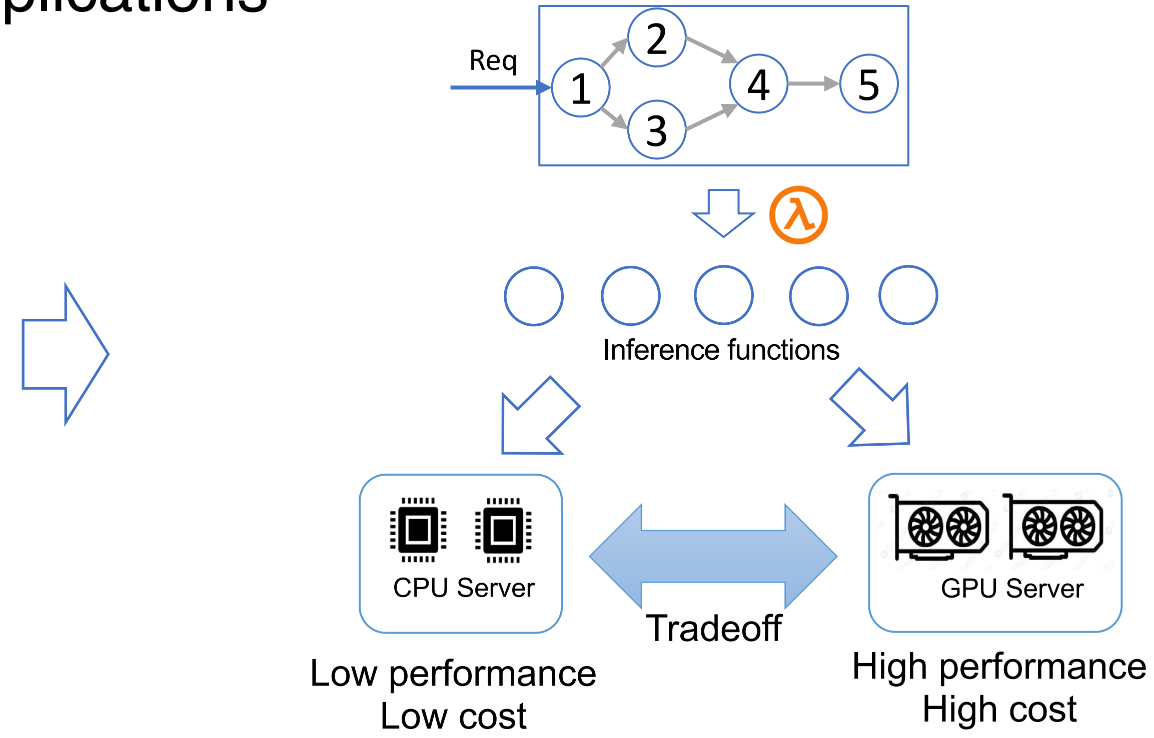
Perf. VS Cost of heterogeneous hardware in AWS serving the ResNet50 model

In Heterogeneous Environment

- Get trade off between performance and cost with the heterogeneous hardware for ML serving applications



Perf. VS Cost of heterogeneous hardware in AWS serving the ResNet50 model



Serving ML applications

- Design resource provision policy
  - for multi-stage ML serving applications
  - on a serverless platform
  - harnesses heterogeneous hardware

  to reduce cost while keeping performance stable

Q1: When start or stop instances? (cold start management)

Q2: Which and how many devices? (hardware configuration)

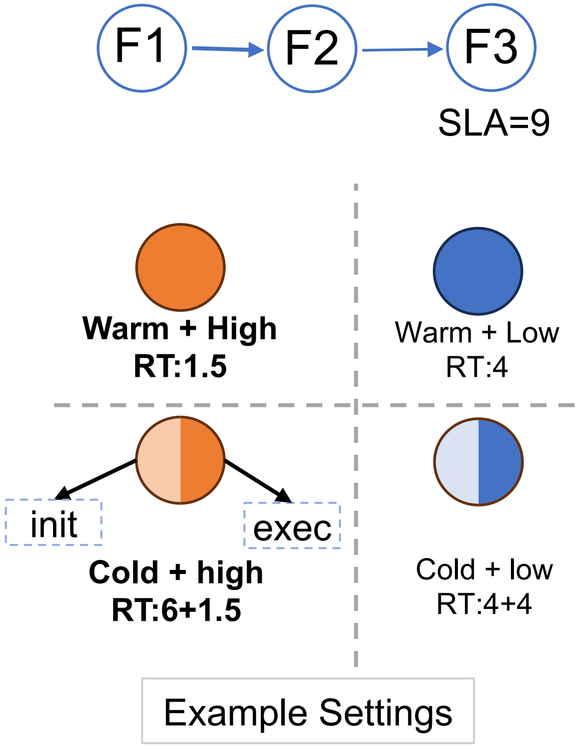
Challenge

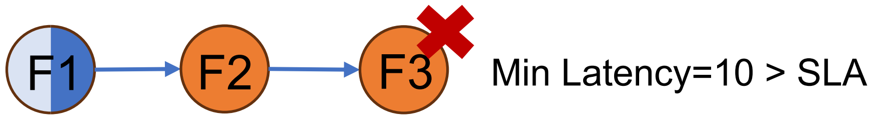
- Cascading Effect in management of serverless ML serving application
  - To satisfy E2E SLA, the policy of one function influences the selection of the policies of all succeeding functions within a DAG application

Challenge

➢ Cascading Effect in management of serverless ML serving application

- To satisfy E2E SLA, the policy of one function influences the selection of the policies of all succeeding functions within a DAG application

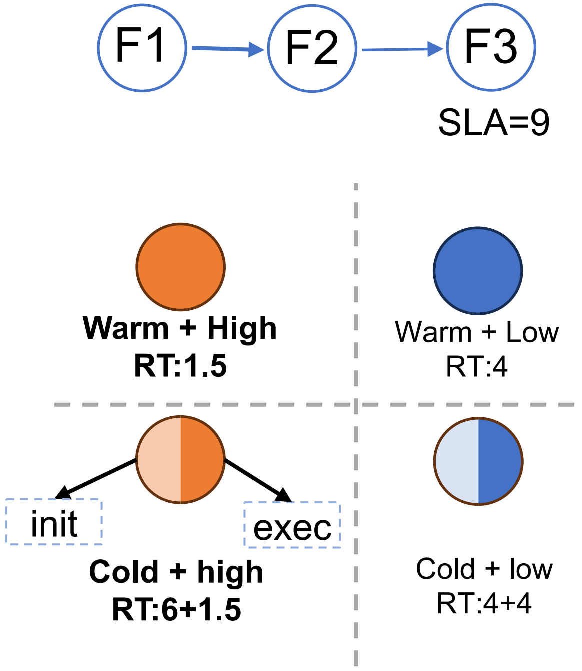




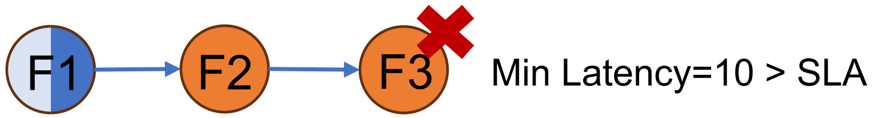
Cold and low policy of F1 leads to the inevitable SLA violation!

Challenge

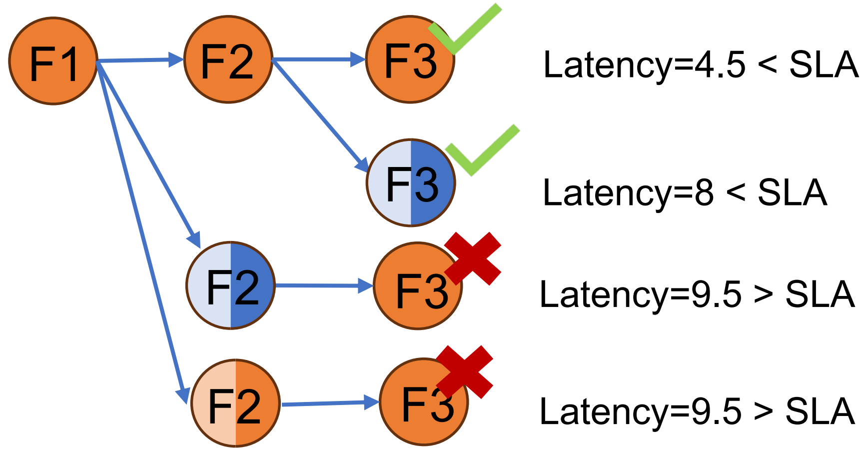
- Cascading Effect in management of serverless ML serving application
  - To satisfy E2E SLA, the policy of one function influences the selection of the policies of all succeeding functions within a DAG application



Example Settings



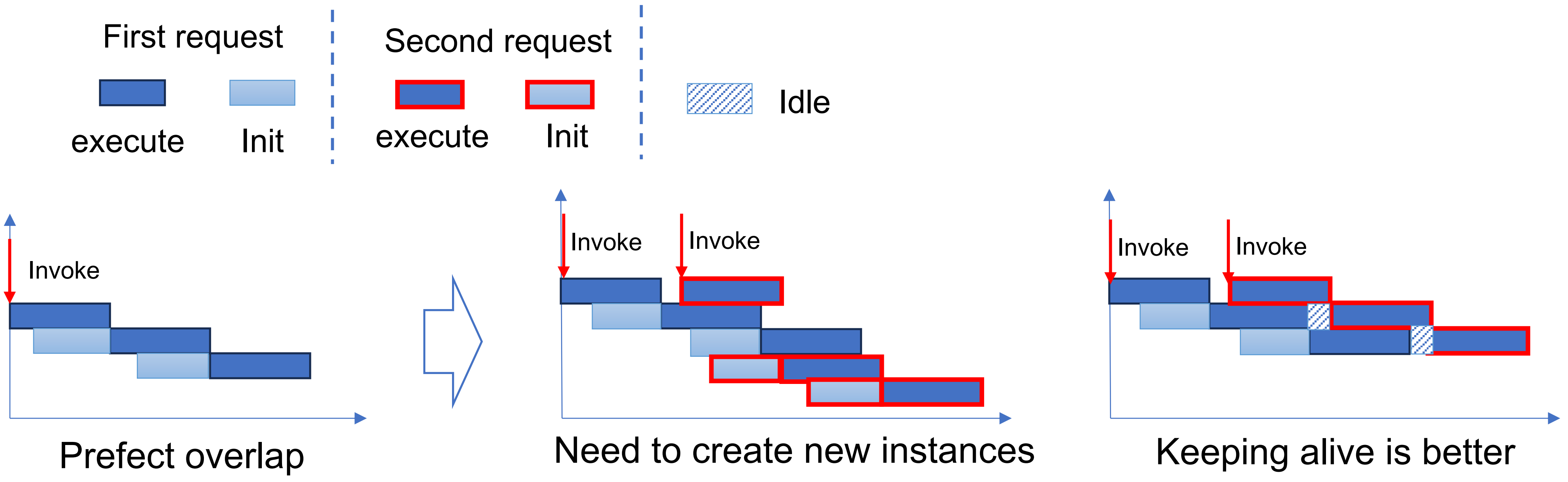
Cold and low policy of F1 leads to the inevitable SLA violation!



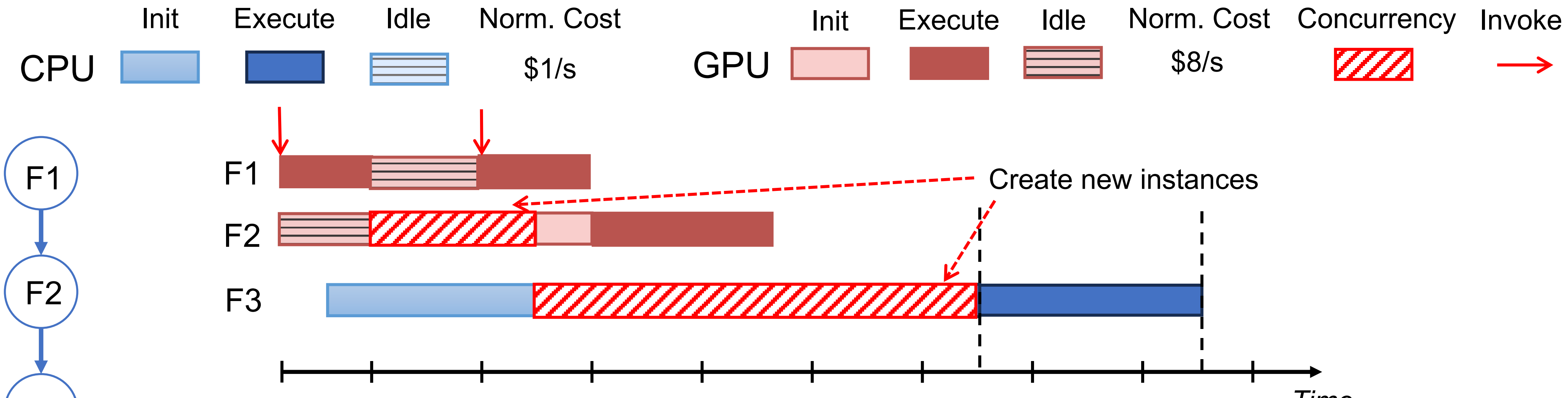
Cold start policy of F2 leads to the inevitable SLA violation!

Challenge

- ➢ Dynamic Invocation Pattern further amplifies the cascading effect
  - Policy that is optimal for a request may not be optimal for more requests in a dynamic context.

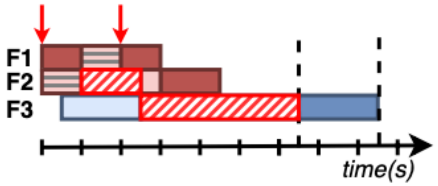


Limitation of Existing Works

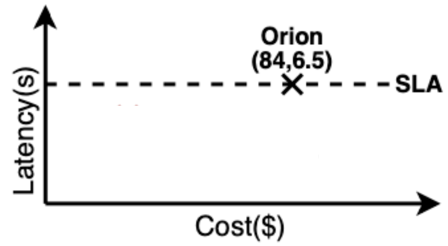


Orion[1]: Ignore dynamic request rate

Latency = 6.5, One Request Cost:42, Total Cost = 84



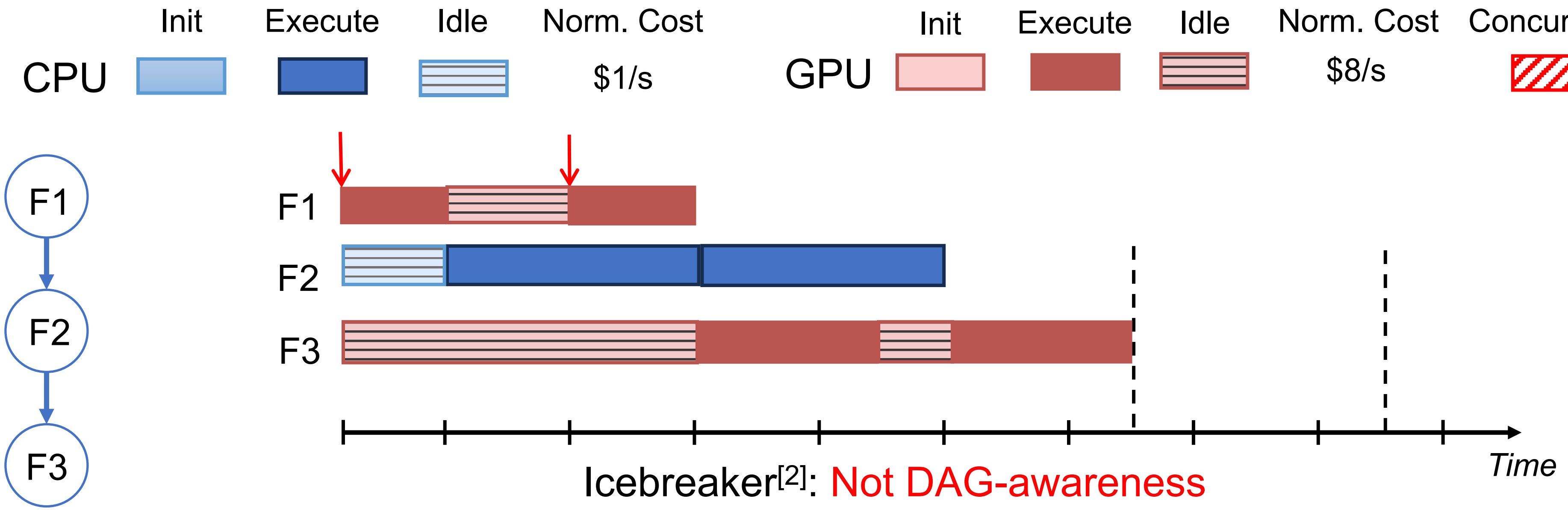
(a) Orion



(d) Cost-latency trade-off

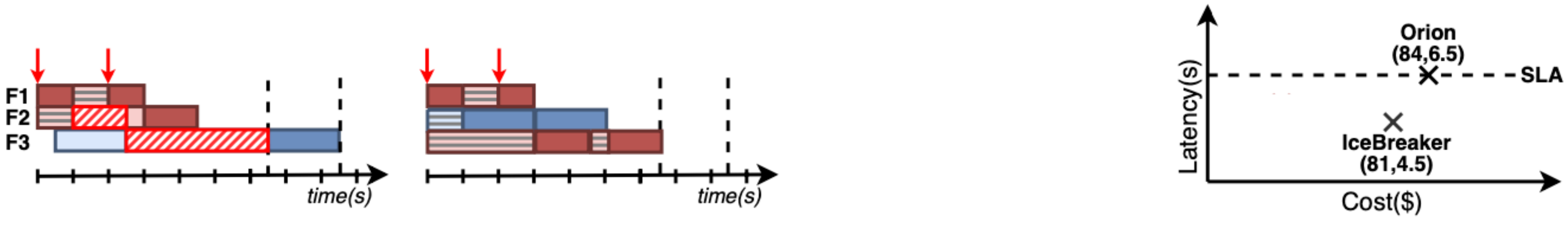
[1] Mahgoub A, Yi E B, Shankar K, et al. ORION and the three rights: Sizing, bundling, and prewarming for serverless DAGs(OSDI’22).
[2] Roy R B, Patel T, Tiwari D. Icebreaker: Warming serverless functions better with heterogeneity (ASPLOS’22)

Limitation of Existing Works



Icebreaker[2]: Not DAG-awareness

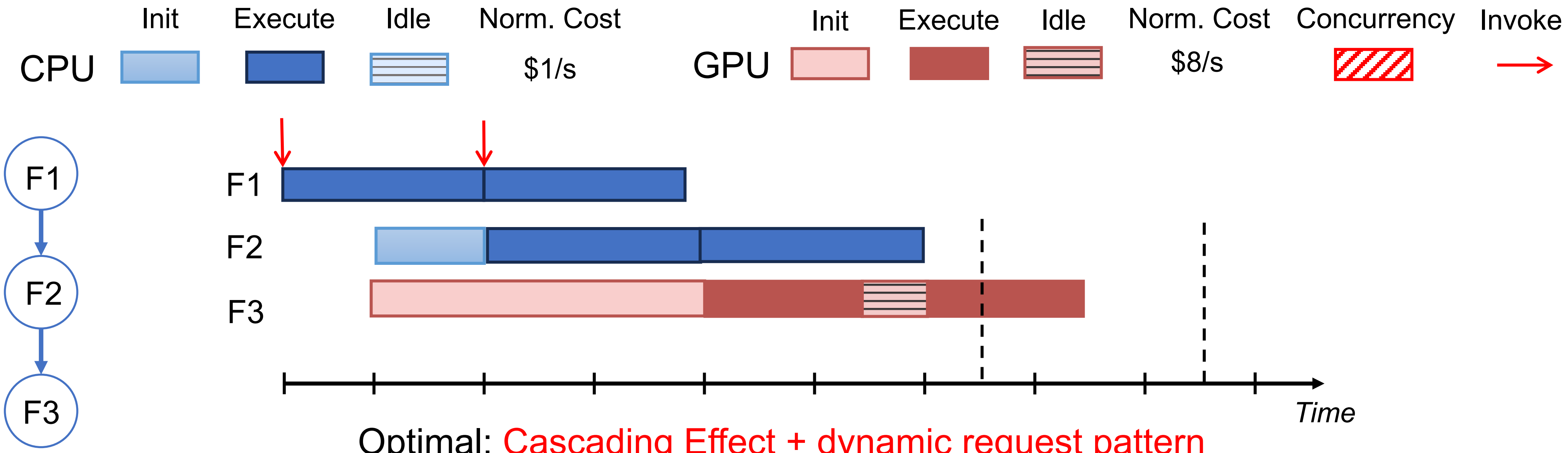
Latency = 4.5, One Request Cost:47, Total Cost = 81



(a) Orion (b) IceBreaker (d) Cost-latency trade-off

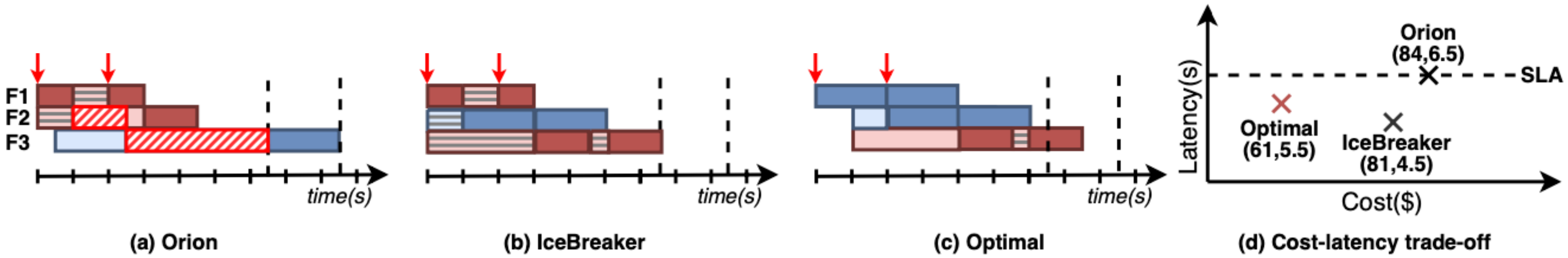
[1] Mahgoub A, Yi E B, Shankar K, et al. ORION and the three rights: Sizing, bundling, and prewarming for serverless DAGs(OSDI'22).
[2] Roy R B, Patel T, Tiwari D. Icebreaker: Warming serverless functions better with heterogeneity (ASPLOS'22)

Limitation of Existing Works



Optimal: Cascading Effect + dynamic request pattern

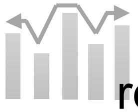
Latency = 5.5, One Request Cost:45,Total Cost = 61



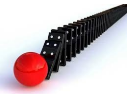
[1] Mahgoub A, Yi E B, Shankar K, et al. ORION and the three rights: Sizing, bundling, and prewarming for serverless DAGs(OSDI'22).
[2] Roy R B, Patel T, Tiwari D. Icebreaker: Warming serverless functions better with heterogeneity (ASPLOS'22)

Our Solution

Optimal



Dynamic request pattern



Cascading effect

SMIless:a serverless framework for optimizing the execution of ML serving applications with highly dynamic invocations

Adaptive Cold-Start Management
Leveraging adaptive pre-warming to manage dynamic invocations

Key Idea1

Path Search Based Co-optimization Framework
Addressing the cascading effect through path search formulation

Key Idea2

Low cost
Stable performance

System Design-SMIless

➢ Offline Profiler ❶

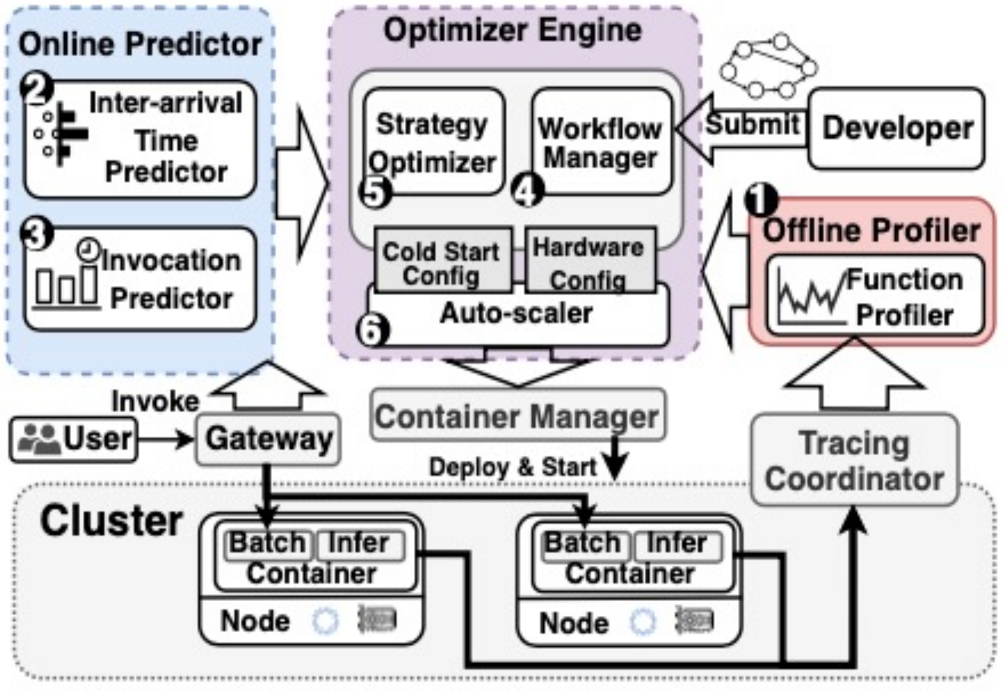
- Profiles inference and initialization time

➢ Online Predictor

- Predicts Inter-arrival time with **Inter-arrival Time Predictor** ❷
- Predicts invocation number by **Invocation Predictor** ❸

➢ Optimizer Engine

- Parsing the workload and merging the result in **Workflow Manager** ❹
- Generate the optimized initialization and execution strategies with **Strategy Optimizer** ❺
- Auto-scaling the function instance for high request rate with **Auto-scaler** ❻



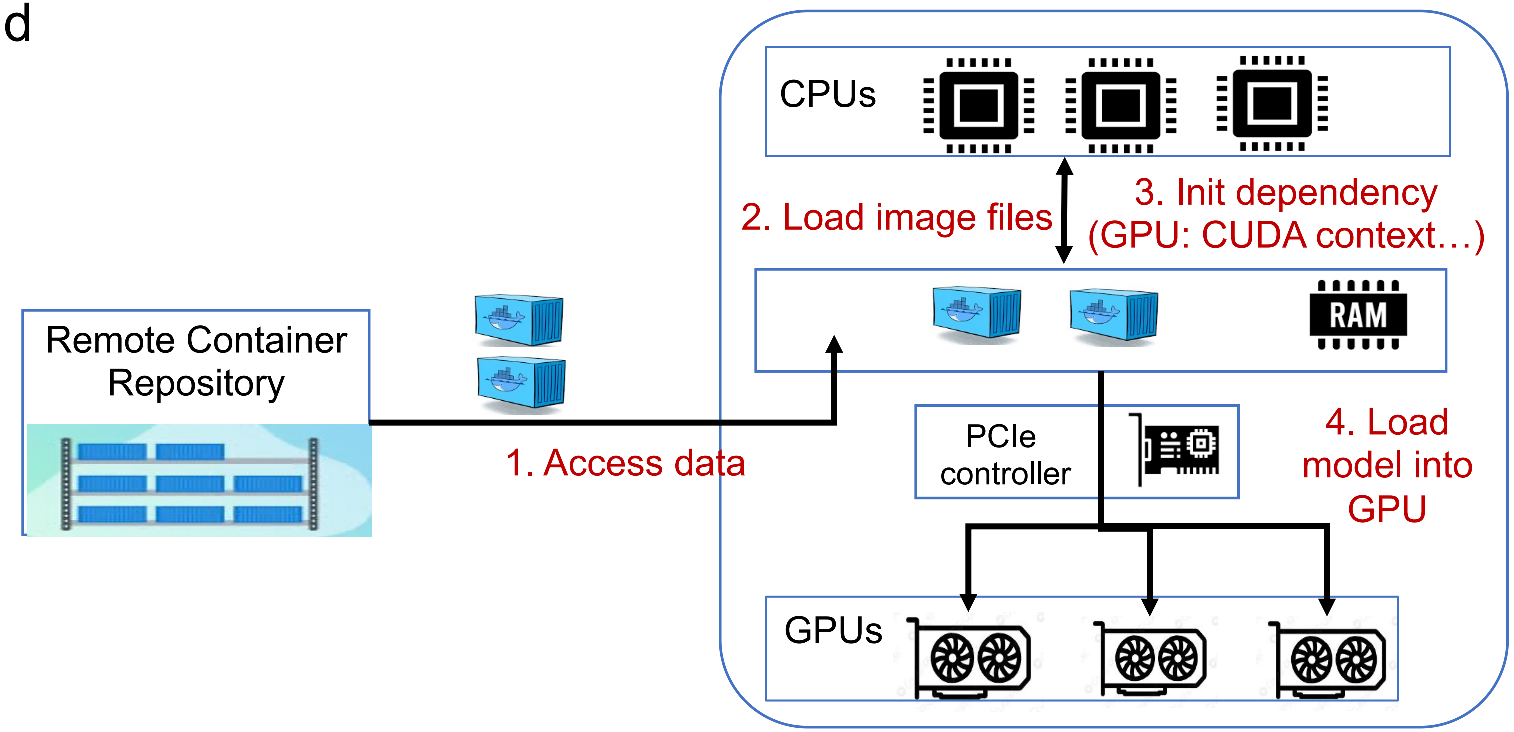
Offline Profiler-SMIless

- Profiling initialization time
  - Support the design of the pre-warming policy of the function

Offline Profiler-SMIless

➢ Profiling initialization time

- Support the design of the pre-warming policy of the function
- Initialization of the function involves in three main steps for the CPU backend and four for the GPU backend



Typical Function Initialization Process

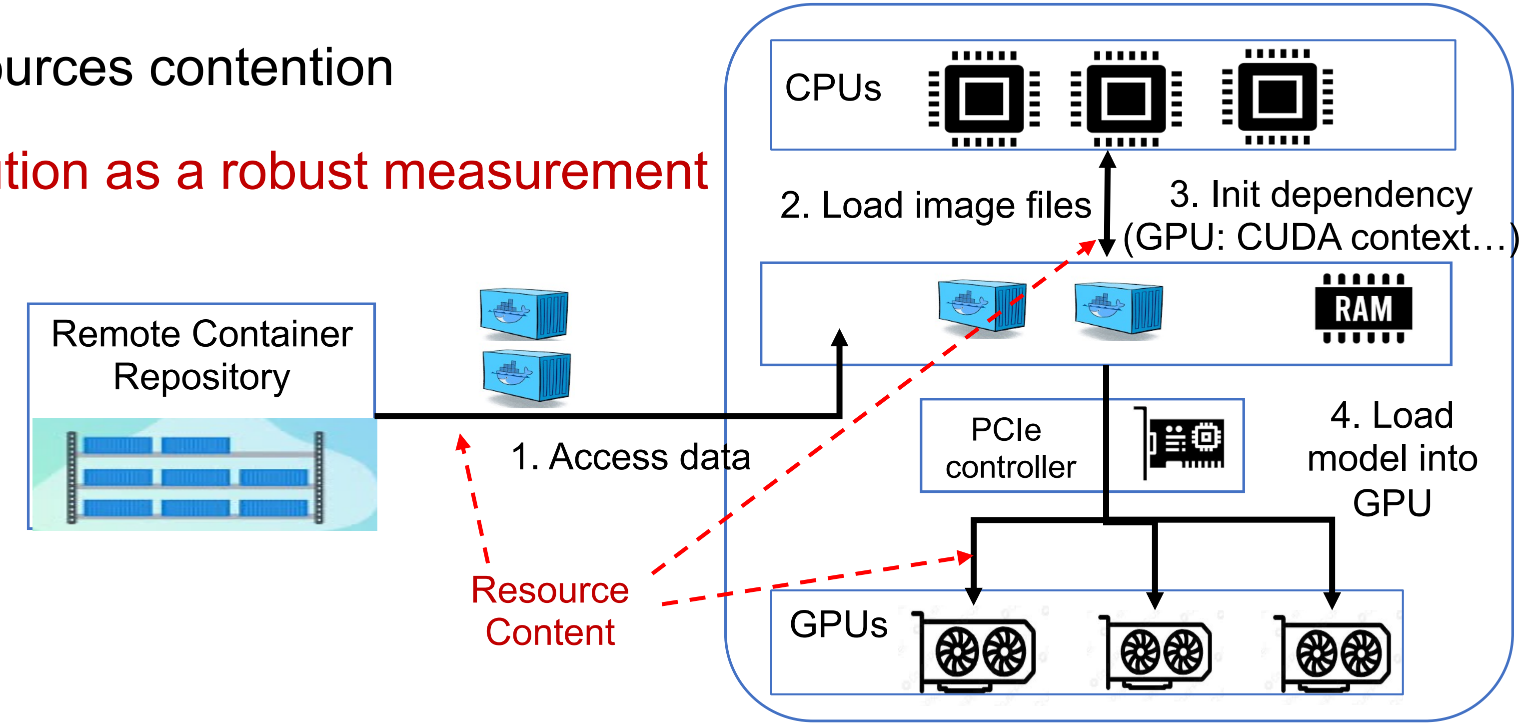
Offline Profiler-SMIless

➢ Profiling initialization time

- Support the design of the pre-warming policy of the function
- Initialization of the function involves in three main steps for the CPU backend and four for the GPU backend
- Fluctuate due to shared resources contention
- Based on the normal distribution as a robust measurement

$$\mu + n\sigma$$

Uncertainty



Typical Function Initialization Process

Offline Profiler-SMIless

➢ Profiling inference time

- Avoid to profile a huge number of configurations
  - ❑ Influenced by both hardware configuration and input batch size.
- Based on Amdahl’s Law:
  - ❑ Capture the acceleration effect due to the excellent parallelism offered by deep learning frameworks

Offline Profiler-SMIless

➢ Profiling inference time

- Avoid to profile a huge number of configurations
  - Influenced by both hardware configuration and input batch size.
- Based on Amdahl's Law:
  - Capture the acceleration effect due to the excellent parallelism offered by deep learning frameworks
  - Independently obtained the λ, α ,β and γ for different types of hardware through curve-fitting

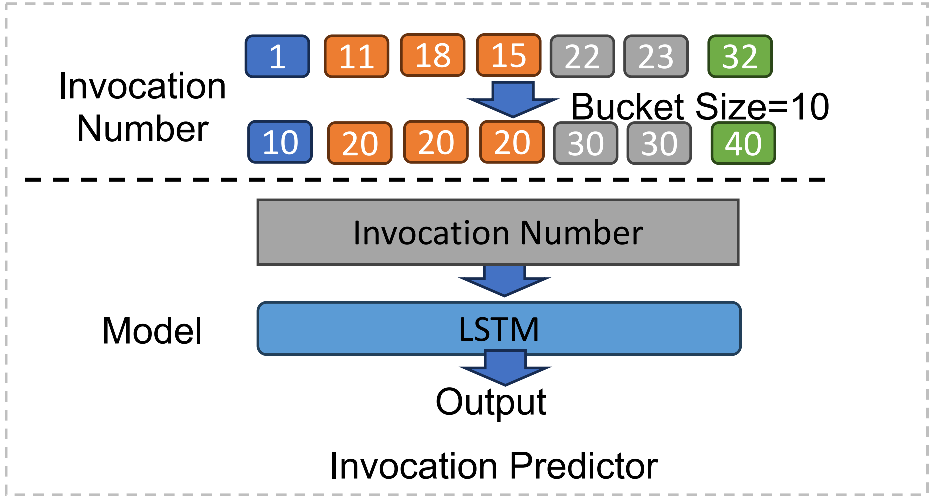
$$\lambda \times B \times \left(\frac{\alpha}{resource\ volume} + \beta\right) + \gamma$$

- $\lambda \times B$: Perf. Degradation coefficient and batch size
- $\frac{\alpha}{resource\ volume}$: Accelerated inference time
- β: Non-accelerated time (Code fetching, data transmission...)
- γ: Network connection time

Online Predictor-SMIless

➢ Predicting invocation number

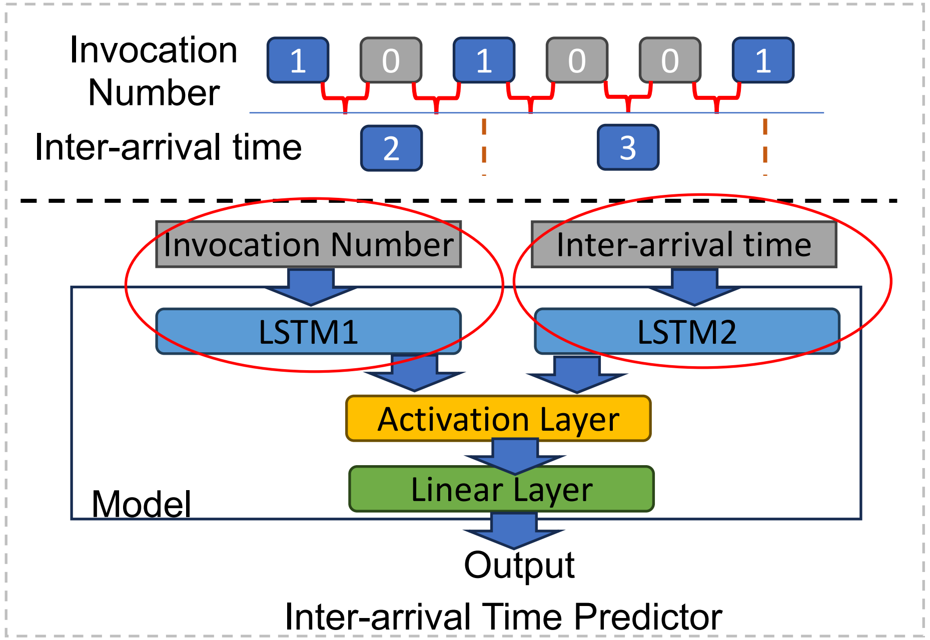
- Divide invocation number into buckets
- Transform into classification problem
  - ❑ Avoid under-estimation for SLA



Online Predictor-SMIless

➢ Predicting inter-arrival time

- Input both inter-arrival time and invocation number
  - ❑ Improve the prediction accuracy
  - ❑ Avoid the overestimation
- Consist of two individual LSTM modules



Optimization-SMIless

➢ Co-optimization Framework

- Minimize the overall execution cost of the application, while satisfying SLA requirements

$$\min_{\{\vec{\chi},\vec{\varphi}\}} \sum_{k=1}^{N} C_k(\star_k,\Delta_k)\,,\mathrm{s.t.}\,\mathcal{L}atency(\vec{\chi},\vec{\varphi}) \leq SLA,$$

$$C_k(\star_k,\Delta_k) = E_k(\star_k,\Delta_k) \cdot U(\star_k)$$

Optimization-SMIless

➢ Co-optimization Framework

- Minimize the overall execution cost of the application, satisfying SLA requirements

$\star_k$ is the hardware config of function k and $\vec{\chi}$ are that of all functions

Δ_k is the cold start policy of function k and $\vec{\varphi}$ are that of all functions

$$\min_{\{\vec{\chi},\vec{\varphi}\}} \sum_{k=1}^{N} C_k(\star_k, \Delta_k)\ , \text{s.t.}\ \mathcal{L}atency(\vec{\chi}, \vec{\varphi}) \leq SLA,$$

$$C_k(\star_k, \Delta_k) = E_k(\star_k, \Delta_k) \cdot U(\star_k)$$

Optimization-SMIless

➢ Co-optimization Framework

- Minimize the overall execution cost of the application, satisfying SLA requirements

$\star_k$ is the hardware config of function k and $\vec{\chi}$ are that of all functions

Δ_k is the cold start policy of function k and $\vec{\varphi}$ are that of all functions

$$\min_{\{\vec{\chi},\vec{\varphi}\}} \sum_{k=1}^{N} C_k(\star_k, \Delta_k)\text{ , s.t. } Latency(\vec{\chi}, \vec{\varphi}) \leq SLA,$$

$$C_k(\star_k, \Delta_k) = E_k(\star_k, \Delta_k) \cdot U(\star_k)$$

Cost of function k with hardware config $\star_k$ and cold start policy Δ_k

Execution time of function k with hardware config $\star_k$ and cold start policy Δ_k

Unit execution cost of function k with hardware config $\star_k$

Optimization-SMIless

➢ Co-optimization Framework

- Minimize the overall execution cost of the application, satisfying SLA requirements

$$\min_{\{\vec{\chi},\vec{\varphi}\}} \sum_{k=1}^{N} C_k(\star_k, \Delta_k),$$
$$\text{s.t. } \mathcal{L}atency(\vec{\chi}, \vec{\varphi}) \leq SLA,$$
$$C_k(\star_k, \Delta_k) = E_k(\star_k, \Delta_k) \cdot U(\star_k)$$

⇨ Minimize overall cost

⇨ Satisfying the SLA requirement

⇨ Cost of each function with given configuration

⇩

Constrained Shortest Path Problem

NP-Hard!

Optimization-SMIless

➢ Co-optimization Framework

- Minimize the overall execution cost of the application, satisfying SLA requirements

$$\min_{\{\vec{\chi},\vec{\varphi}\}} \sum_{k=1}^{N} C_k(\star_k, \Delta_k),$$

$$\text{s.t.}\ \mathcal{L}atency(\vec{\chi}, \vec{\varphi}) \leq SLA,$$

$$C_k(\star_k, \Delta_k) = E_k(\star_k, \Delta_k) \cdot U(\star_k)$$

⇒ Minimize overall cost

⇒ Satisfying the SLA requirement

⇒ Cost of each function with given configuration

⇓

NP-Hard!

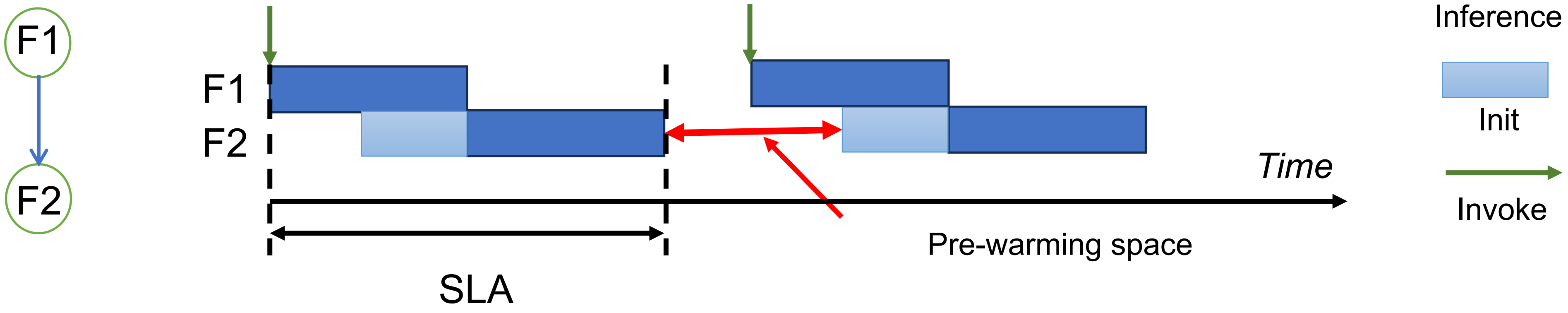
⇓

Adaptive cold-start management + Path Search Based Co-optimization Framework

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 1: Low invocation arrival rate

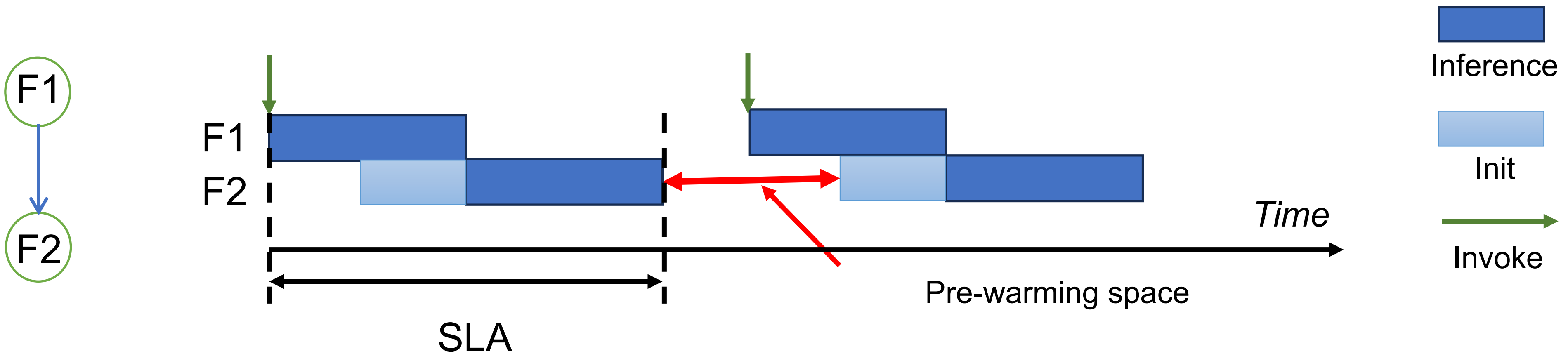


The initialization and inference time of the function can be perfectly overlapped with the inter-arrival time.

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 1: Low invocation arrival rate



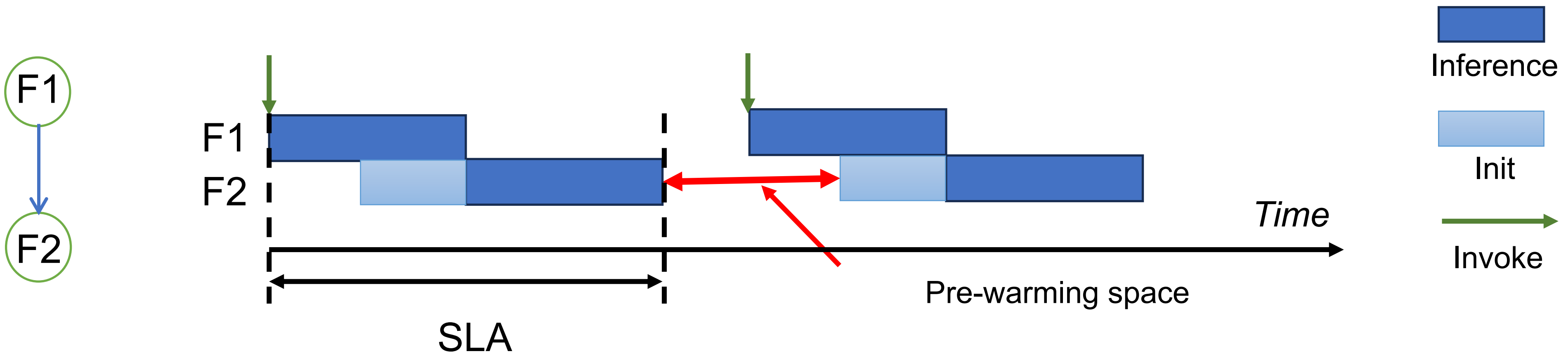
The initialization and inference time of the function can be perfectly overlapped with the inter-arrival time.

- Terminating and pre-warming the function to reduce cost
- The latency is the sum of the inference time of all functions
- The cost equals the product of the execution time and the unit cost $U(\star)$ of the function

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 1: Low invocation arrival rate



The initialization and inference time of the function can be perfectly overlapped with the inter-arrival time.

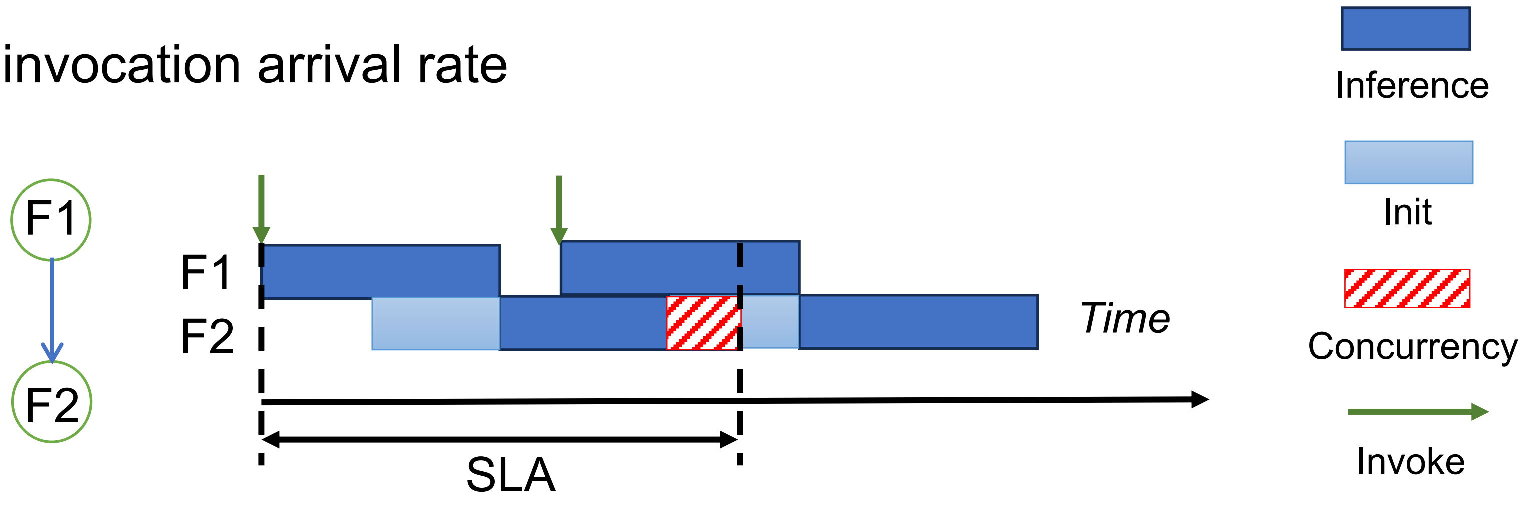
- Terminating and pre-warming the function to reduce cost
- The latency is the sum of the inference time of all functions
- The cost equals the product of the execution time and the unit cost $U(\star)$ of the function

Theorem 1: When I_2+I_1< SLA and $T_2 + I_2 < IT$, the warming-up policy guarantees the minimum overall execution cost.

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 2: High invocation arrival rate

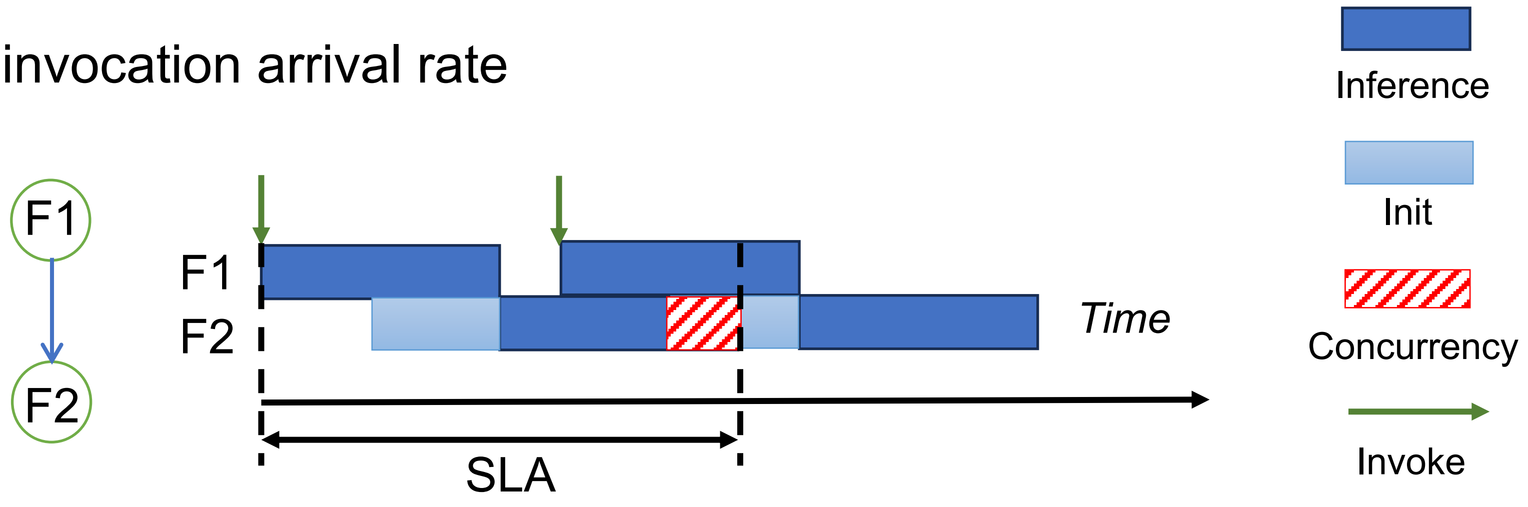


The inter-arrival time cannot overlap the initialization but can overlap the inference of the function

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 2: High invocation arrival rate



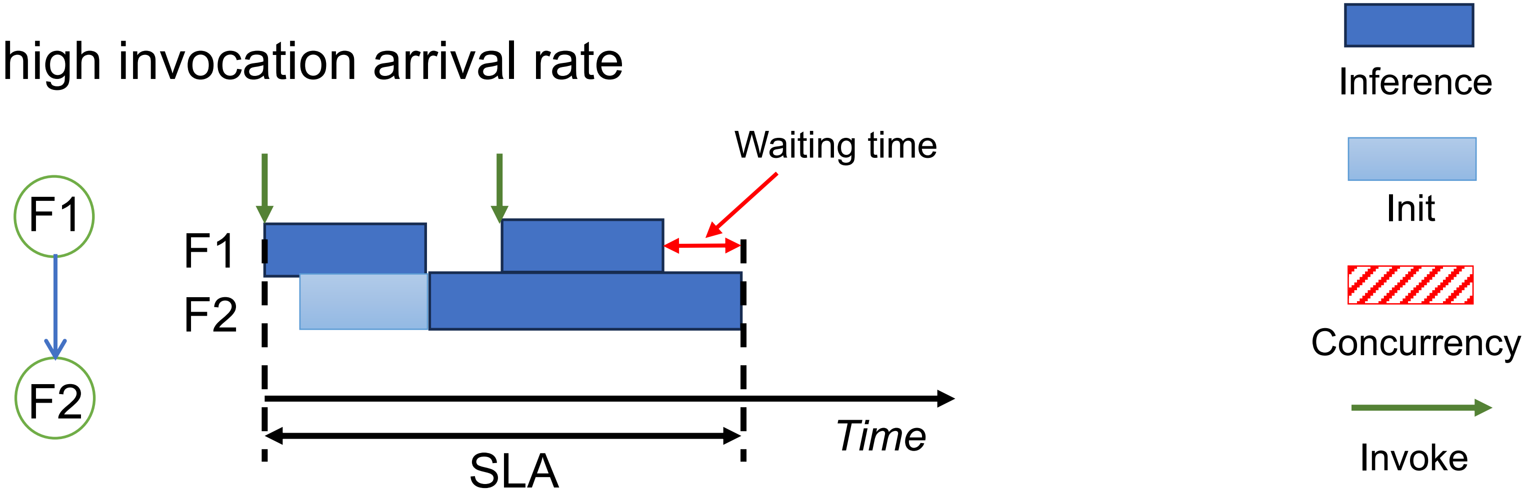
The inter-arrival time cannot overlap the initialization but can overlap the inference of the function

- Keeping alive the function to reduce cost
- The latency is the sum of the inference time of each function
- The cost equals the product of the inter-arrival time and the unit cost $U(\star)$

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 3: Very high invocation arrival rate

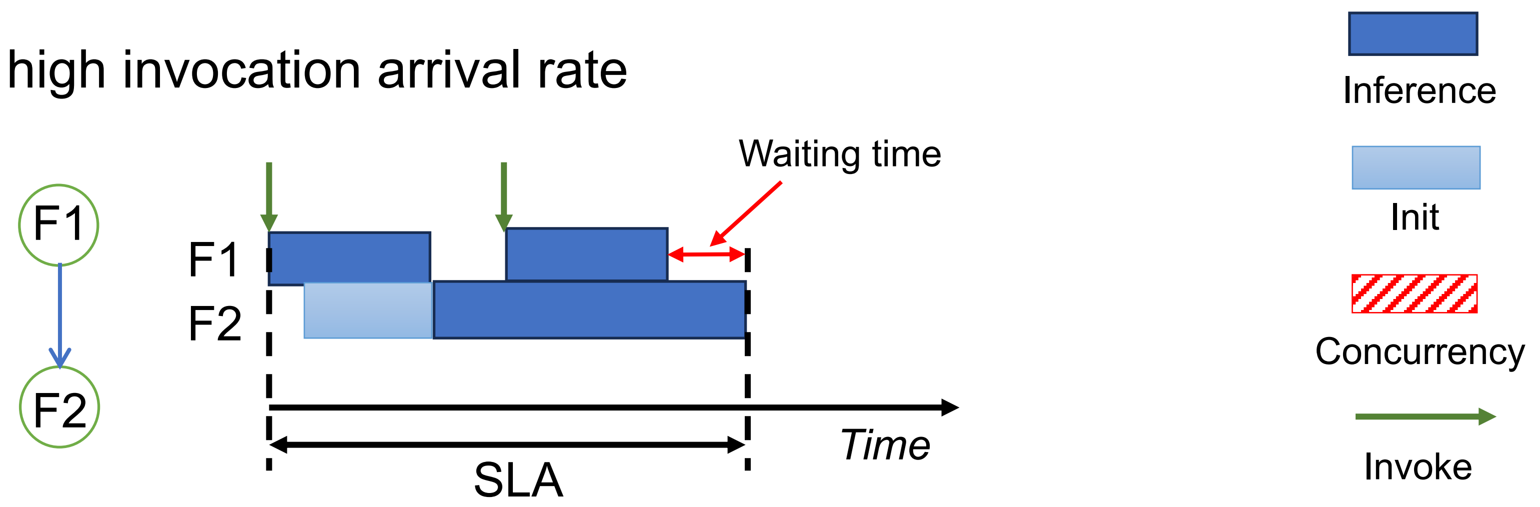


The inter-arrival time cannot overlap the inference of the function.

Optimization-SMIless

➢ Adaptive Cold-Start Management

- Case 3: Very high invocation arrival rate



The inter-arrival time cannot overlap the inference of the function.

- Batching invocations, using high-performance hardware and launching multiple instances to reduce the inference time of the function to avoid the SLA violation
- The latency is the sum of the inference time of each function
- The cost equals the product of the inter-arrival time and the unit cost $U(\star)$

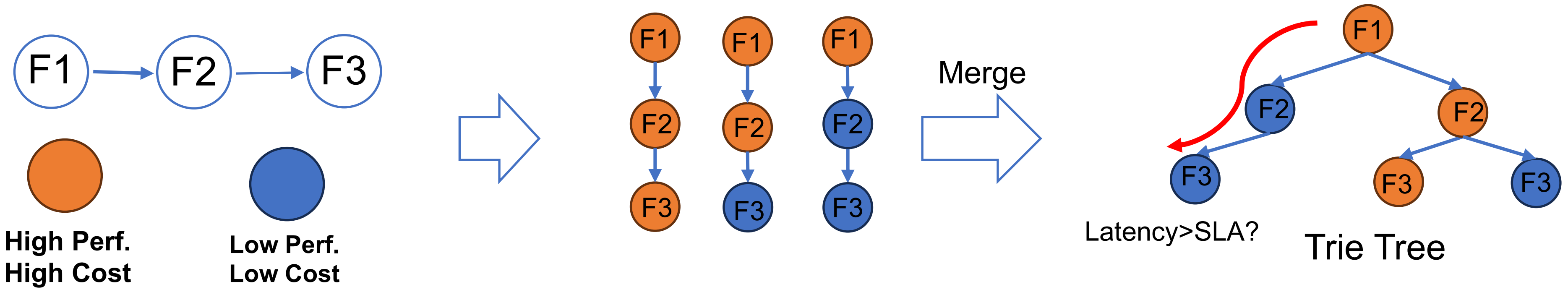
Optimization-SMIless

- Path Search Based Co-optimization Framework
  - Convert the optimization problem to a path search problem

Optimization-SMIless

➢ Path Search Based Co-optimization Framework

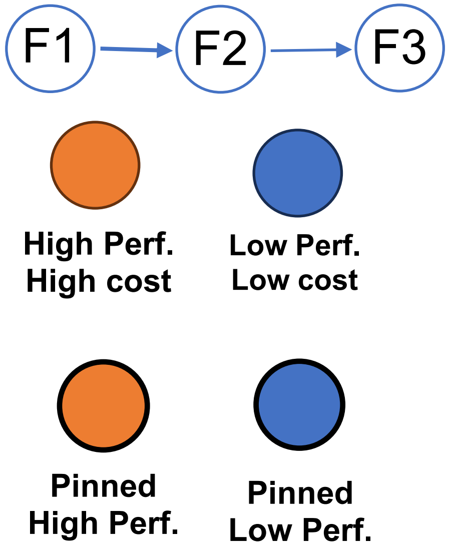
- Convert the optimization problem to a path search problem

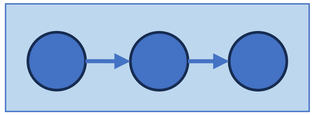


Can only check the policy combination until reaching the leaf node, **high overhead**

Optimization-SMIless

- Path Search Based Co-optimization Framework
  - Convert the optimization problem to a path search problem
  - Provide opportunity to prune the tree before traverse to the leaf node, **reduce overhead**





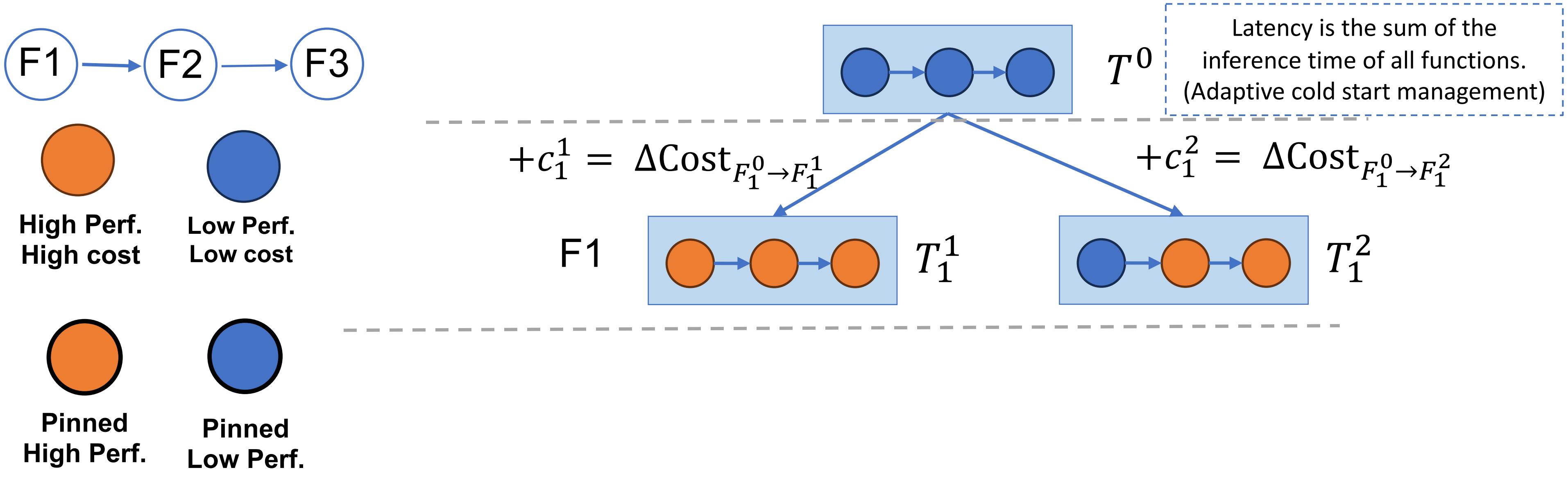
T^0

Latency is the sum of the inference time of all functions. (Adaptive cold start management)

Optimization-SMIless

➢ Path Search Based Co-optimization Framework

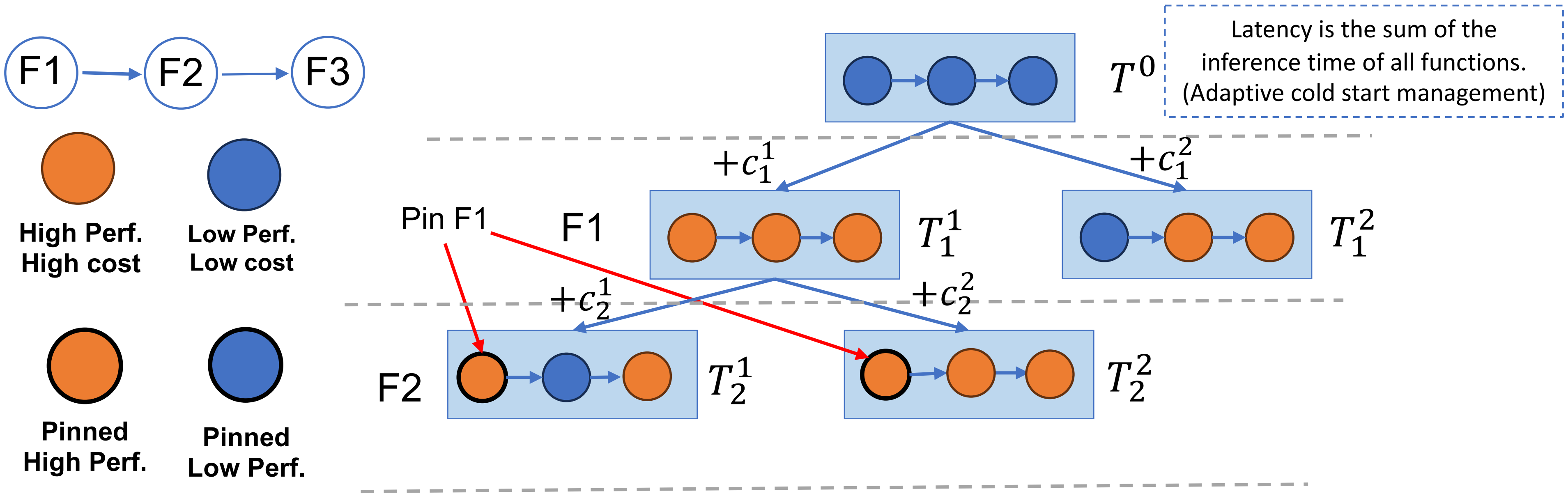
- Convert the optimization problem to a path search problem
- Provide opportunity to prune the tree before traverse to the leaf node, **reduce overhead**



Optimization-SMIless

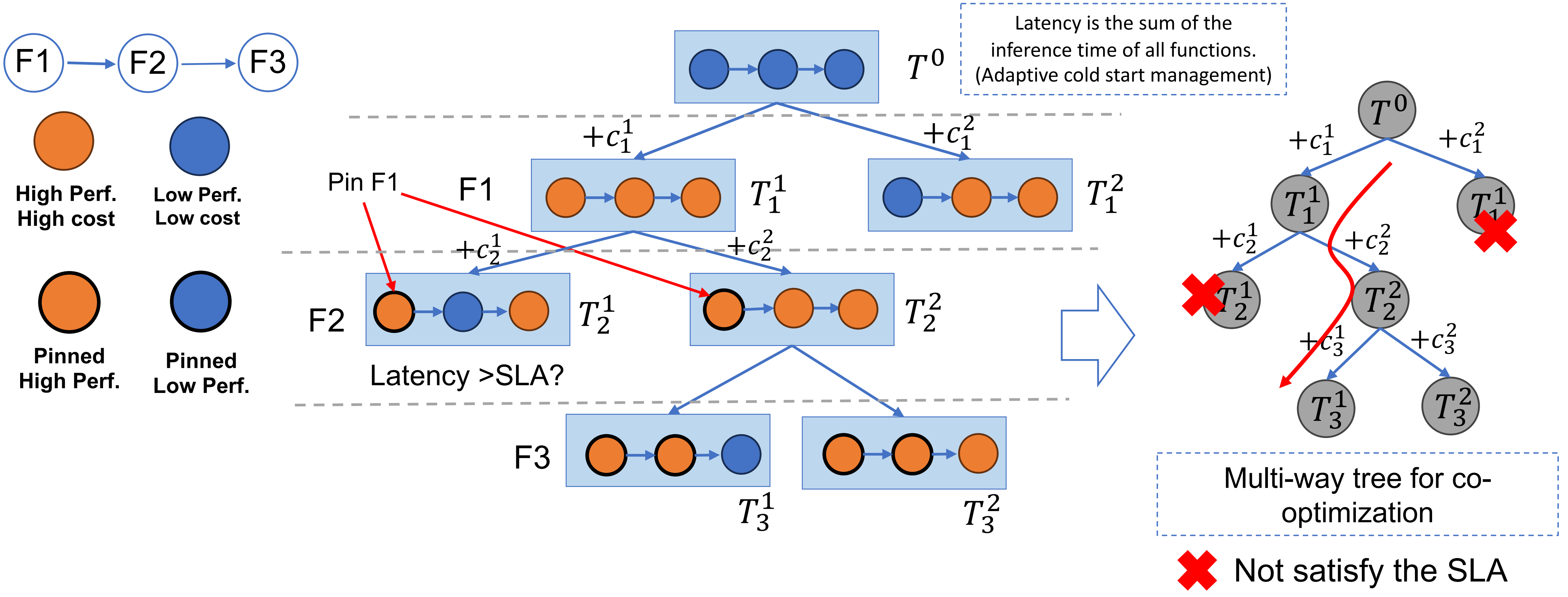
➢ Path Search Based Co-optimization Framework

- Convert the optimization problem to a path search problem
- Provide opportunity to prune the tree before traverse to the leaf node, **reduce overhead**



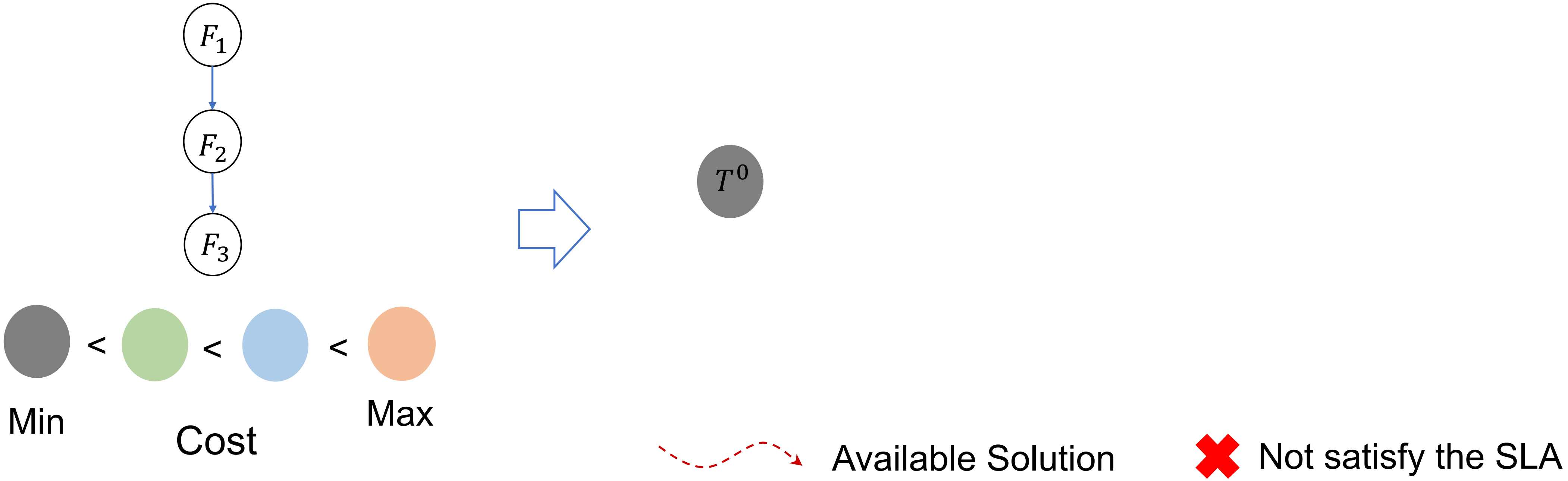
Optimization-SMIless

- Path Search Based Co-optimization Framework
  - Convert the optimization problem to a path search problem
  - Provide opportunity to prune the tree before traverse to the leaf node, **reduce overhead**



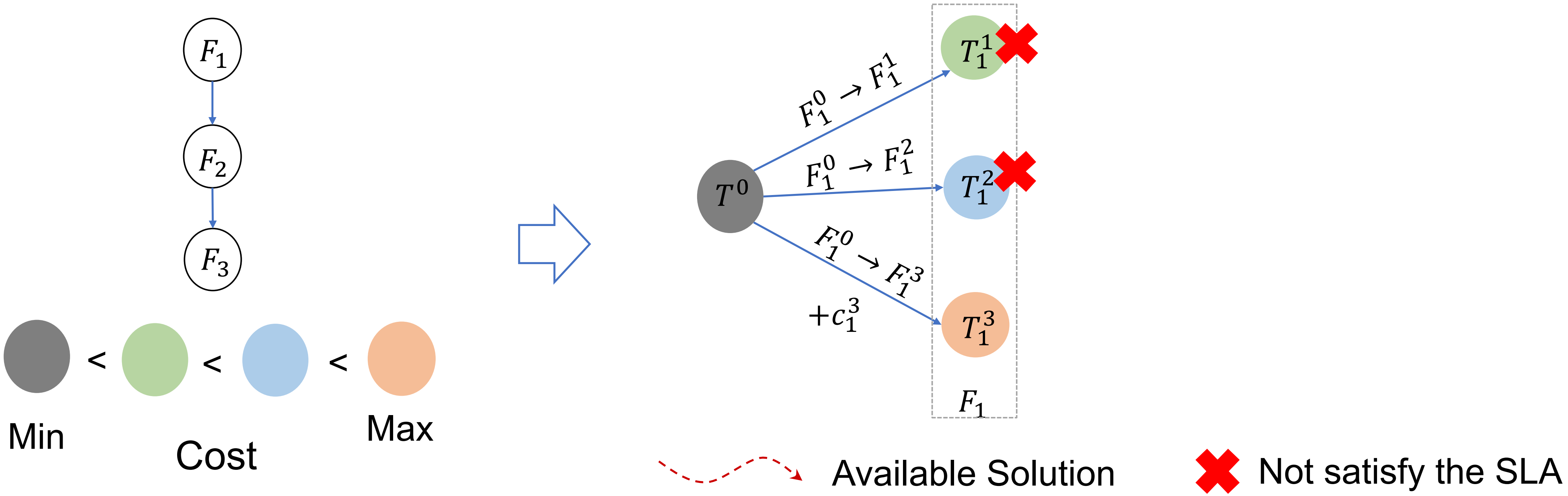
Optimization-SMIless

- Optimization for Simple Applications
  - Use a path search process to solve it
    - Combined BFS (Breadth-First Search) and DFS (Depth-First Search)
    - With Top-K (Top-1 in SMIless) path search to balance the overhead and the effectiveness



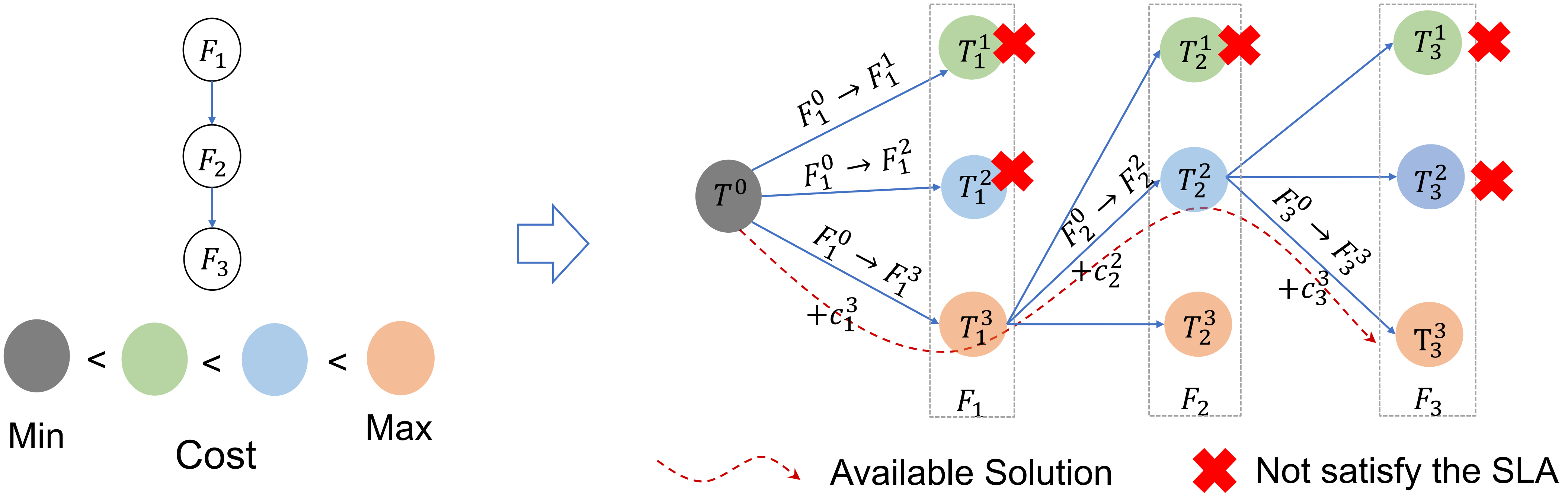
Optimization-SMIless

- Optimization for Simple Applications
  - Use a path search process to solve it
    - Combined BFS (Breadth-First Search) and DFS (Depth-First Search)
    - With Top-K (Top-1 in SMIless) path search to balance the overhead and the effectiveness



Optimization-SMIless

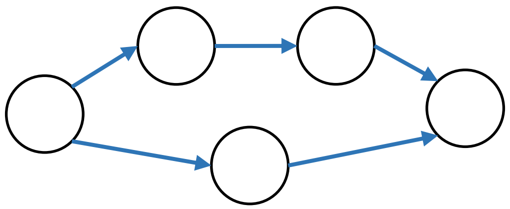
- Optimization for Simple Applications
  - Use a path search process to solve it
    - Combined BFS (Breadth-First Search) and DFS (Depth-First Search)
    - With Top-K (Top-1 in SMIless) path search to balance the overhead and the effectiveness



Optimization-SMIless

➢ Optimization for Complex Applications

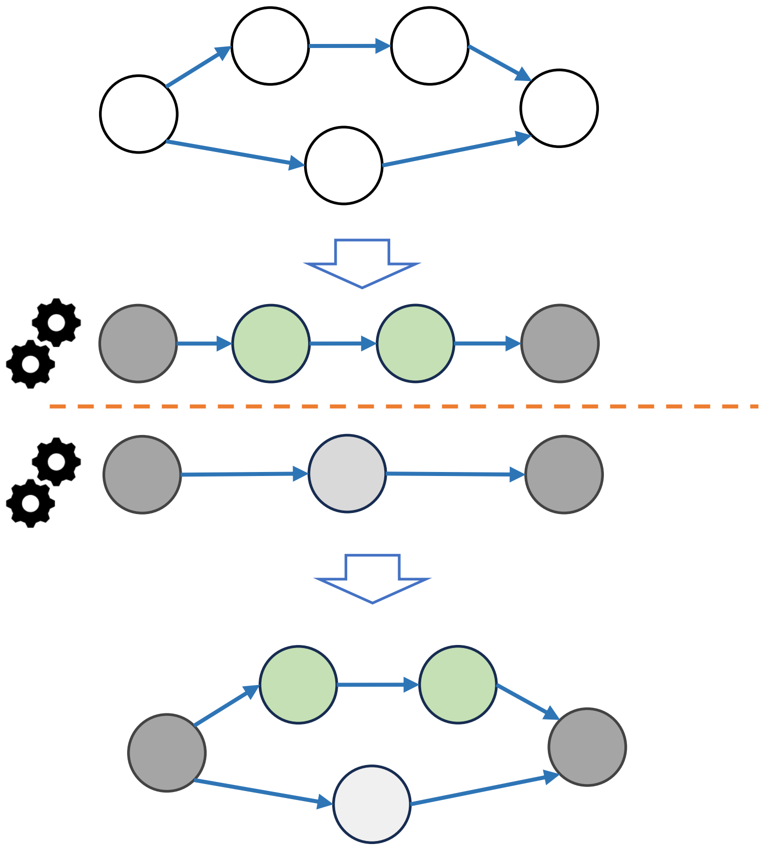
- High efficient for dynamic invocation patterns
- Heuristic strategy



Optimization-SMIless

➢ Optimization for Complex Applications

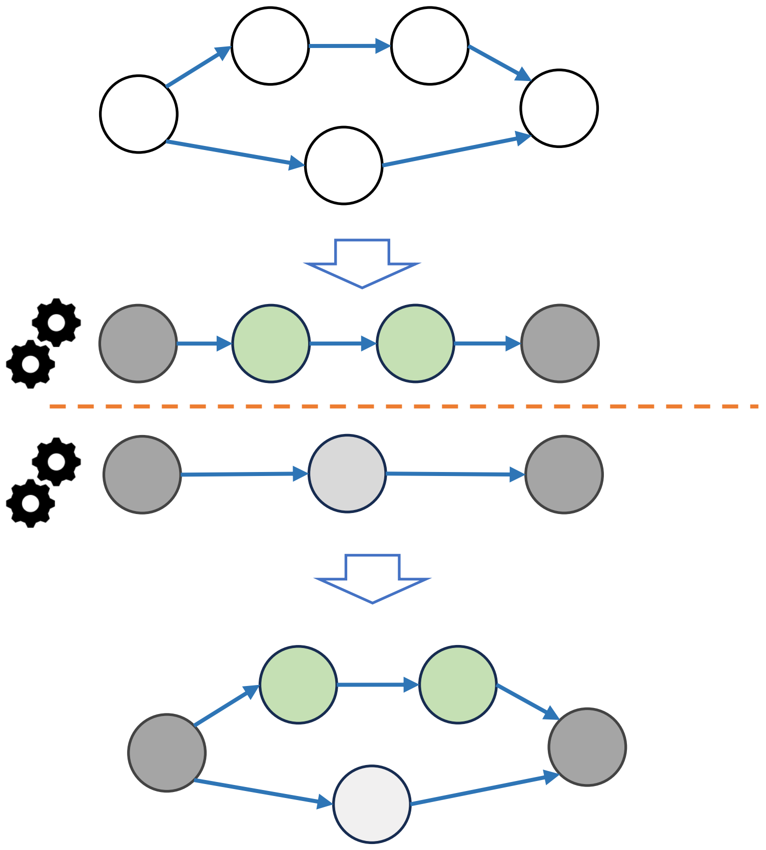
- High efficient for dynamic invocation patterns
- Heuristic strategy
  - ❑ Decompose the complex DAG into multiple subgraphs with simple DAG (by workflow manager, offline)
  - ❑ Path search for each subgraph in parallel (online)
  - ❑ Merge the results from all subgraphs with shortest inference time



Optimization-SMIless

➢ Optimization for Complex Applications

- High efficient for dynamic invocation patterns
- Heuristic strategy
  - Decompose the complex DAG into multiple subgraphs with simple paths (by workflow manager, offline)
  - Path search for each subgraph in parallel (online)
  - Merge the results with shortest inference time
- Time complexity
  - $O(N \cdot M \cdot \log(M))$, N is the number of the functions of the longest path, M is the number of hardware configuration candidates



Optimizer Engine-SMIless

➢ Auto-scaler

- Keep the inference time stable when suffering high request rate

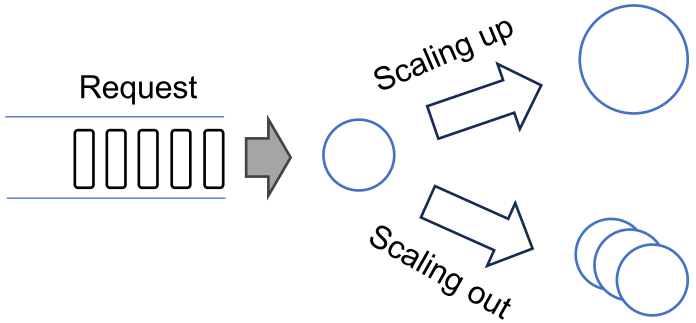
G/B: instance number
G: predicted invocation number

I_s: required inference time obtained from the co-optimization algorithm

$$\min_{\{\star_k, B\}} \frac{G}{B} \cdot IT \cdot U(\star_k)$$

$$s.t. \lambda \times B \times \left(\frac{\alpha}{resource\ volume} + \beta\right) + \gamma \leq I_s$$

- Select $\star_k$ and B with minimal overall cost
- Use a Bisection method to efficiently determine the optimal solution



Evaluation-SMIless

➢ Experimental Setup

- Applications
  - AMBER Alert, Image-Query, Voice Assistant
  - Real world and widely used
- Load generator
  - Azure Function Dataset[1]
  - Scale down the interval from 1 min to 2s, spans 2h
- Baselines
  - GrandSLAm[2], Icebreaker[3],Orion[4], Aquatope[5]

Type	Spec
Machine number	8
CPU/Mem	52-core Intel x86 Xeon Gold 5320 * 2/ 128GB
GPU	Nvidia 3090*1
CPU container	1,2 ...16 CPU cores
GPU container	10%, 20% ... 100% GPU(with MPS)
CPU container price	$x*0.034
GPU container price	$x*3.06

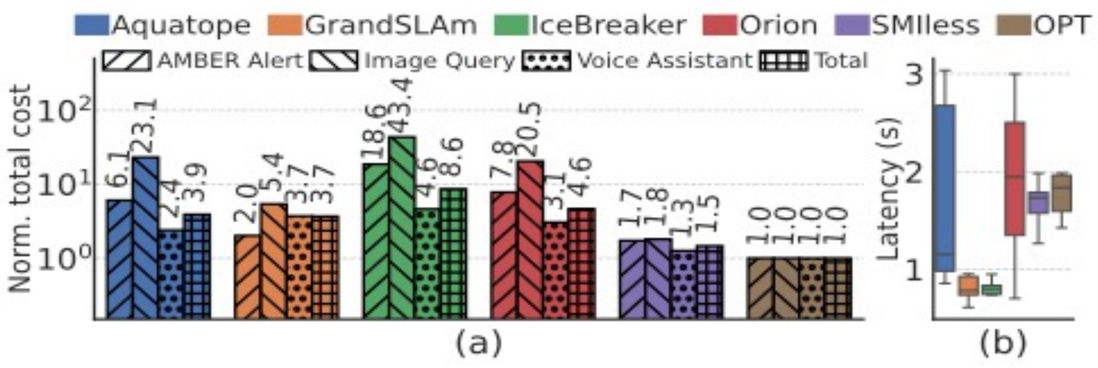
System Settings

[1] Shahrad M, Fonseca R, Goiri I, et al. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider;(ATC'20)
[2] Kannan R S, Subramanian L, Raju A, et al. Grandslam: Guaranteeing slas for jobs in microservices execution frameworks(EuroSys'19)
[3] Roy R B, Patel T, Tiwari D. Icebreaker: Warming serverless functions better with heterogeneity(ASPLOS'22)
[4] Mahgoub A, Yi E B, Shankar K, et al. {ORION} and the three rights: Sizing, bundling, and prewarming for serverless {DAGs} (OSDI'22)
[5] Zhou Z, Zhang Y, Delimitrou C. Aquatope: Qos-and-uncertainty-aware resource management for multi-stage serverless workflows(ASPLOS'22)

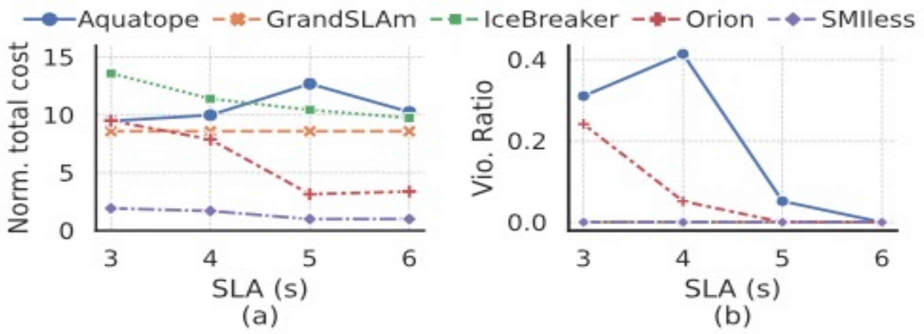
Evaluation-SMIless

➢ End-to-end Performance

- Almost no SLA violation, reduce SLA violation ratio by up to 40% compared to baseline
- Reduce cost by up to 5.73× to Icebreaker
- Achieve the lowest cost and SLA violation ratio under different SLA settings



End to end result



E2E performance under different SLA settings

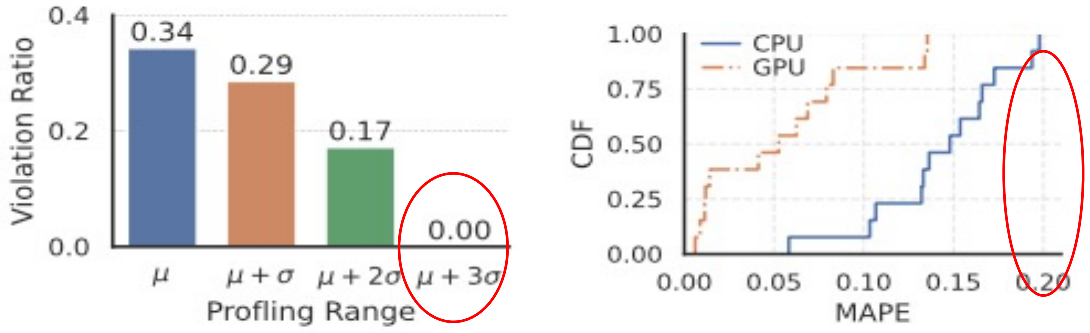
Evaluation-SMIless

➢ Offline Profiling

- SLA violations can be completely avoided with 3x uncertainty
- High accuracy of profiling inference time

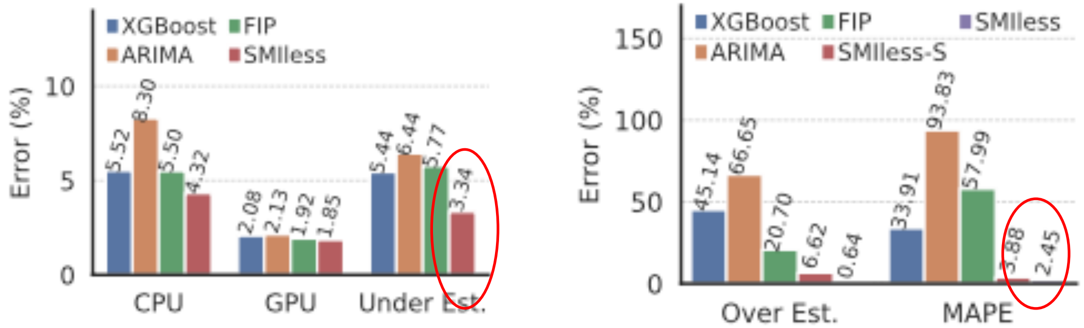
➢ Online Prediction

- Both low estimation error for invocation number and inter-arrival time



(a) The influence of profiling initialization time (b) The accuracy of profiling inference time

Offline profiling results under SMIless



(a) Invocation number (b) Inter-arrival time

Online prediction on invocation number and inter-arrival time

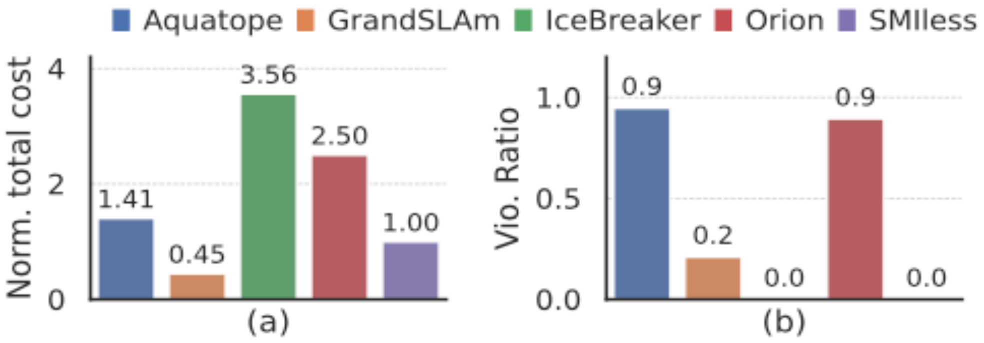
Evaluation-SMIless

➢ Adaptation to Bursty Arrivals

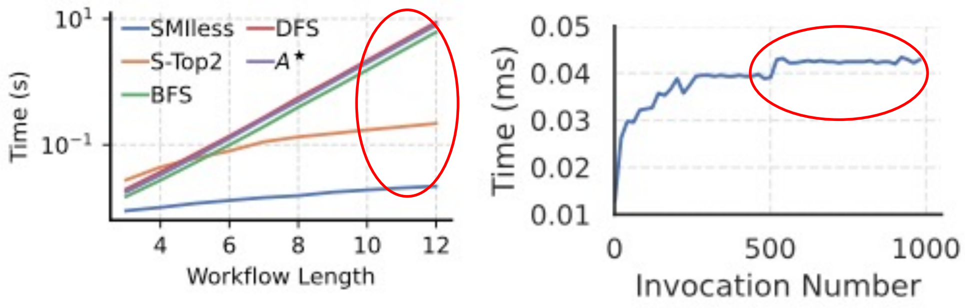
- Reduce the cost up to 3.56x while avoiding the SLA violation

➢ System Overhead

- 10x~100x time cost reduction compared with other path search methods for co-optimization
- Auto-scaling within less than 0.1ms for 1000 invocation numbers



Auto-scaling performance



(a) Co-optimization overhead (b) The overhead of auto-scaler

System overhead

Conclusion

Serving ML Inference with Dynamic Invocations under Serverless Computing

- Propose a new policy to adaptively manage the pre-warming with dynamic request pattern
- Design an efficient path search algorithm to co-optimize the performance and cost
- Achieve up to 5.73x reduction in the cost with stable application performance

Thanks & QA

cz.lu@siat.ac.cn