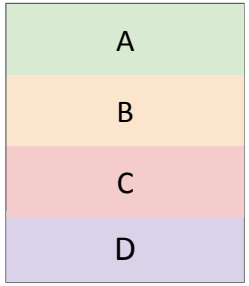
Slowpoke: End-to-end Throughput Optimization Modeling for Microservice Applications

Yizheng Xie, Di Jin, Oğuzhan Çölkesen, Vasiliki Kalavri[BU], John Liagouris[BU], Nikos Vasilakis

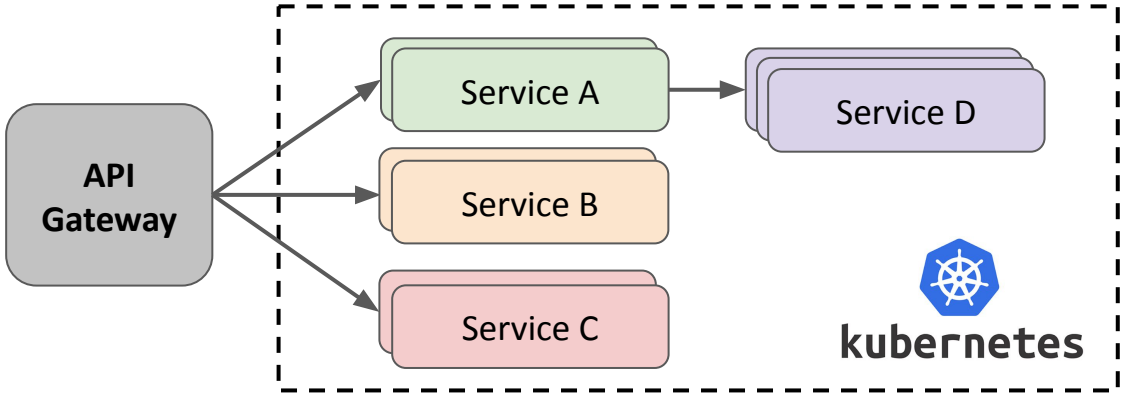

github.com/atlas-brown/slowpoke

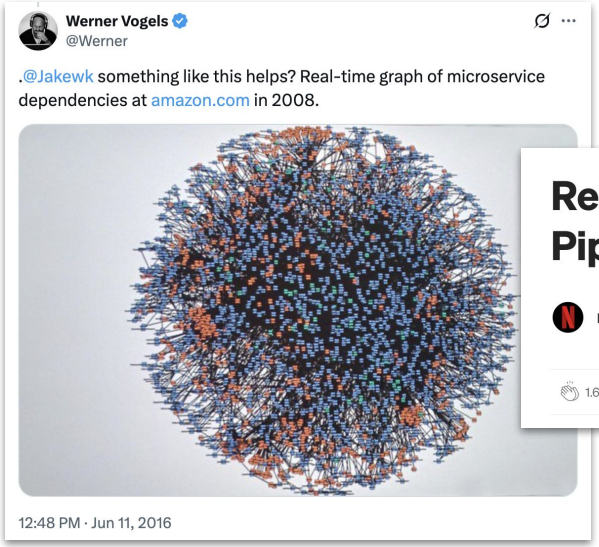
Microservices are Pervasive

Monoliths



Microservices

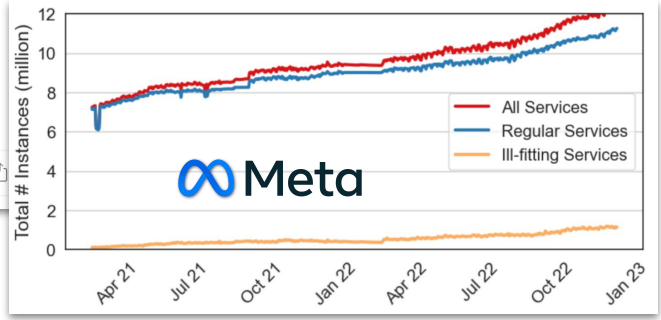




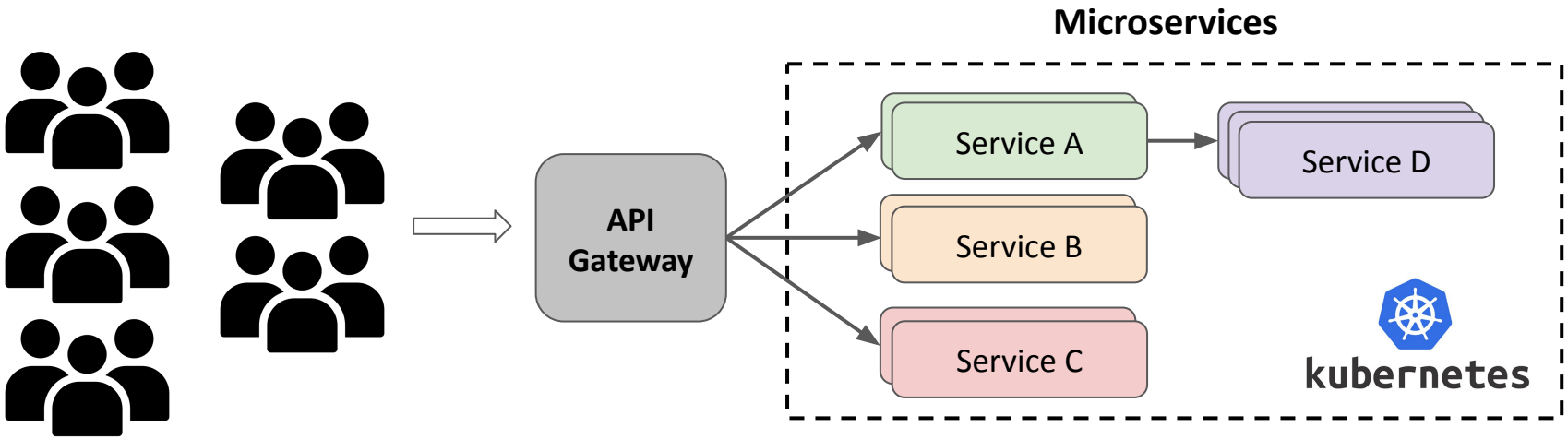
Rebuilding Netflix Video Processing Pipeline with Microservices

Netflix Technology Blog · Follow · 9 min read · Jan 10, 2024

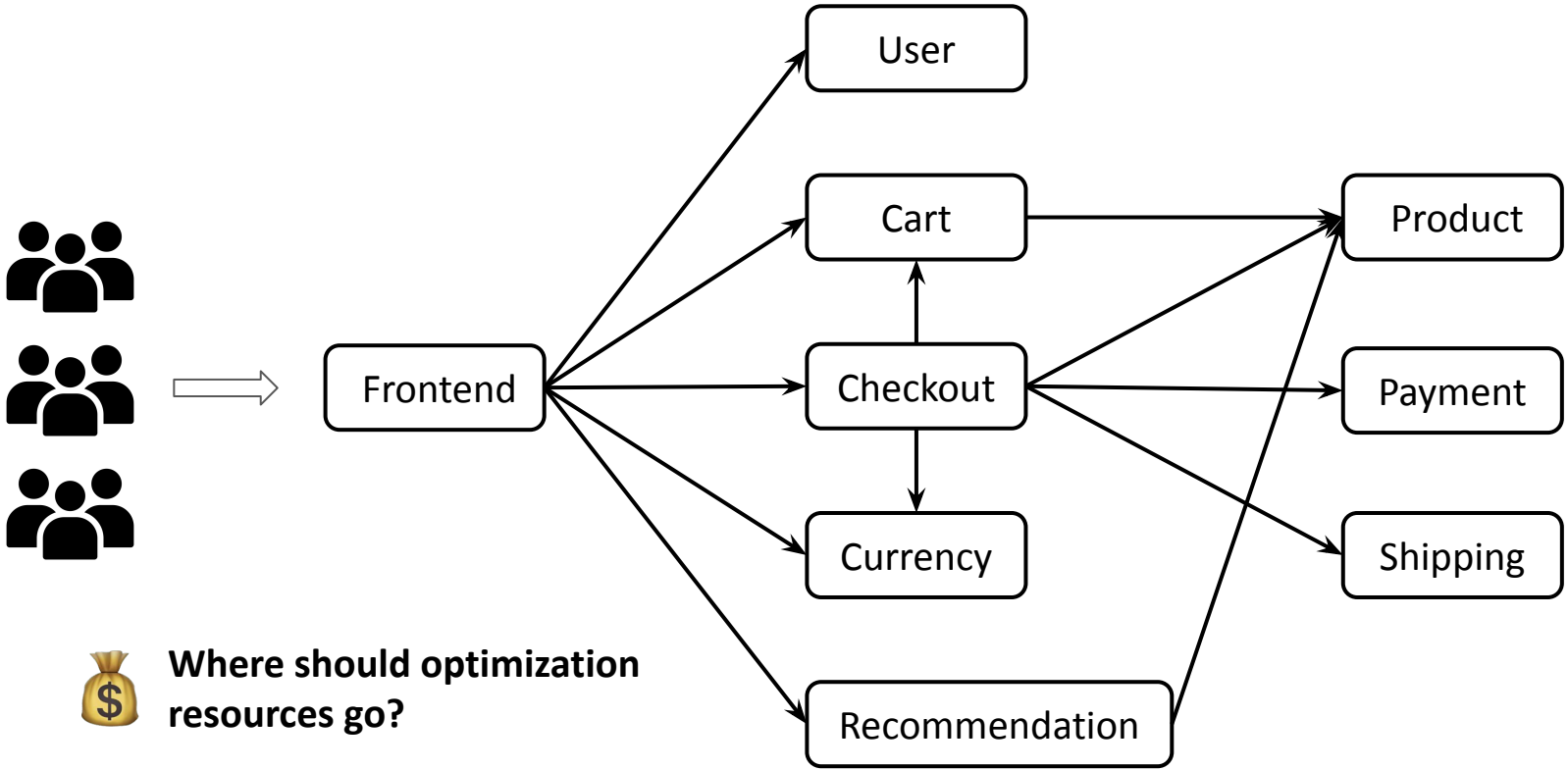
1.6K · 22



Throughput Improvement



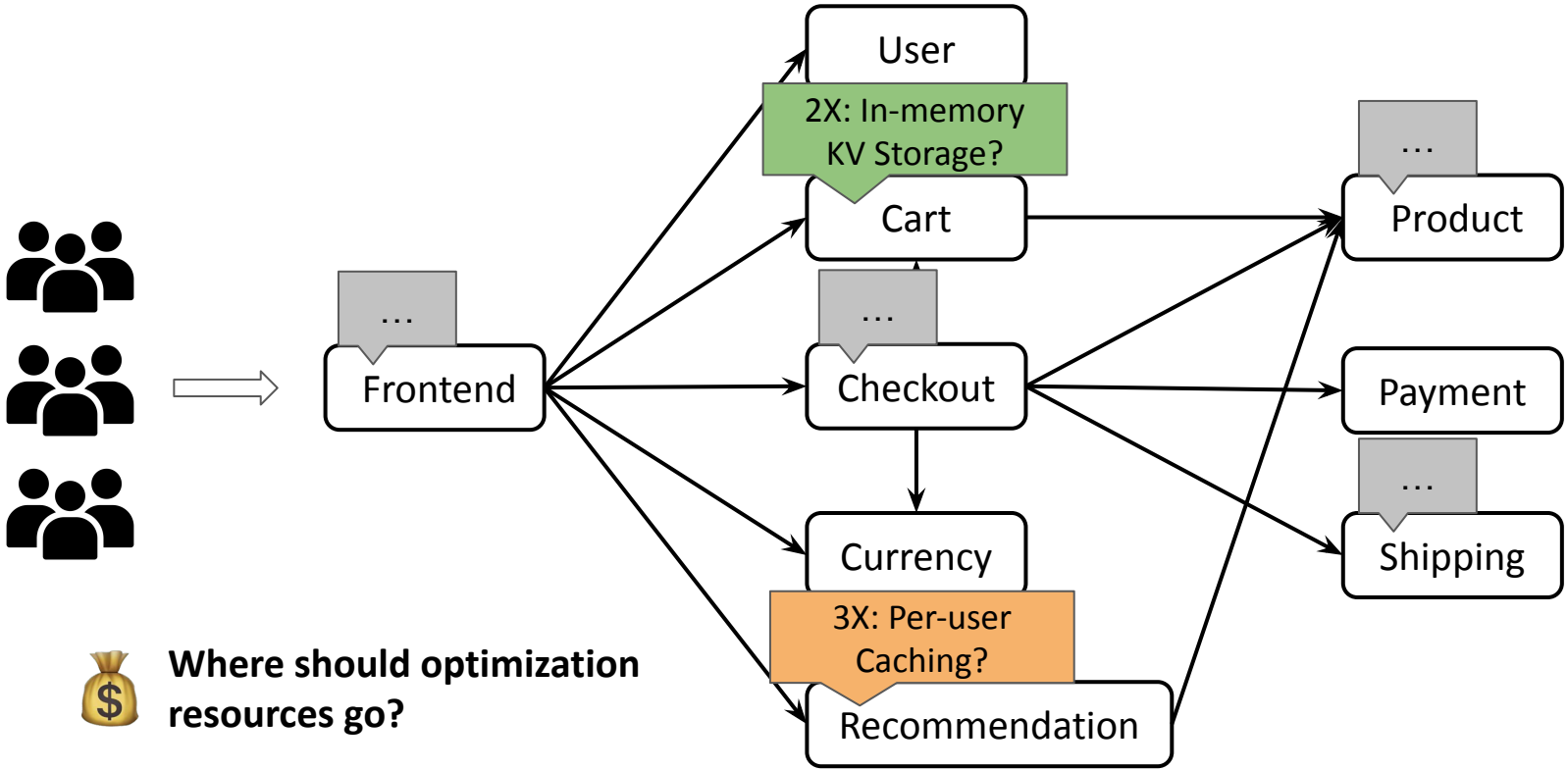
Example Shopping Application



💰 **Where should optimization resources go?**

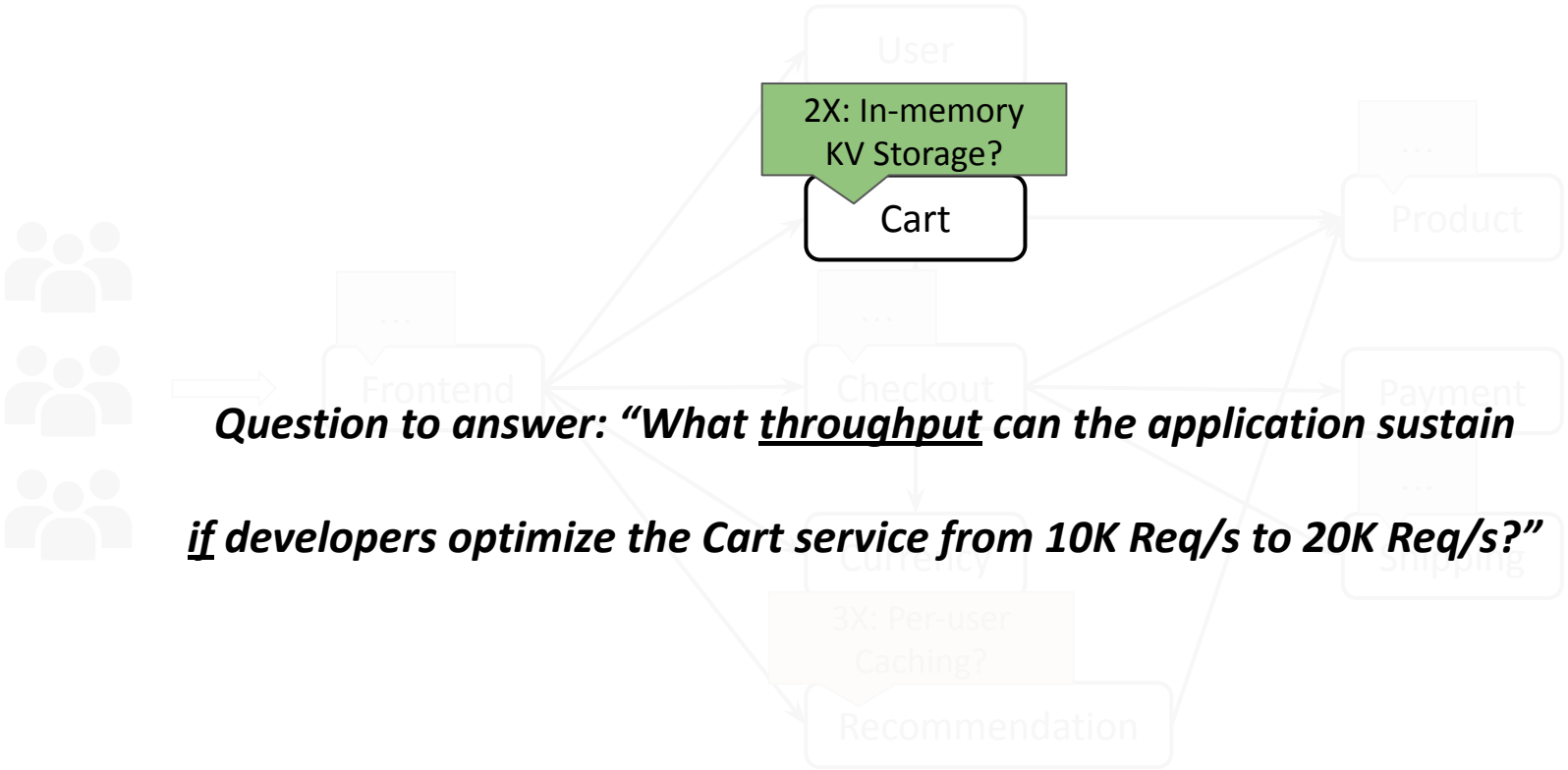
Reference: OnlineBoutique, https://github.com/GoogleCloudPlatform/microservices-demo

Identifying High-Impact Service Optimizations



Reference: OnlineBoutique, https://github.com/GoogleCloudPlatform/microservices-demo

Identifying High-Impact Service Optimizations



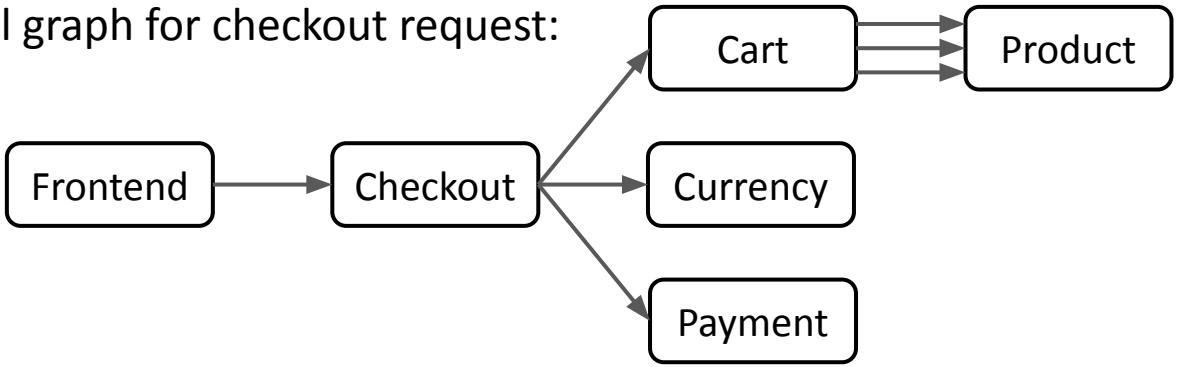
Question to answer: "What <u>throughput</u> can the application sustain

<u>if</u> developers optimize the Cart service from 10K Req/s to 20K Req/s?"

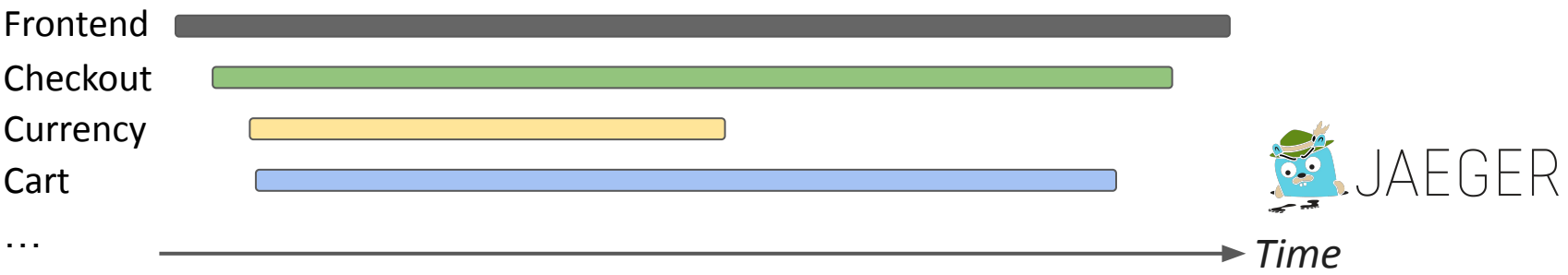
Reference: OnlineBoutique, https://github.com/GoogleCloudPlatform/microservices-demo

Distributed Tracing Only Captures Current Execution

Call graph for checkout request:



Current execution!

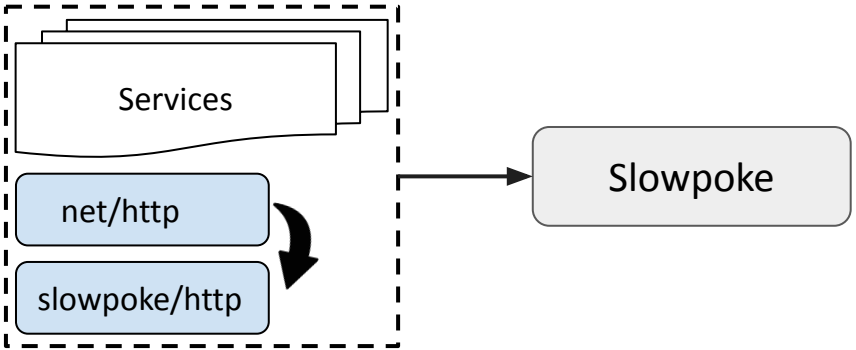


*Slowpoke is a system for **accurately quantifying the effect of a hypothetical optimization** on end-to-end **throughput** for microservice applications.*

Using Slowpoke

Inputs: (1) A target service X.
(2) A hypothetical optimization. E.g., reducing the processing time of 500ms by 50%.
(3) A workload.

Output: End-to-end throughput as if optimization was implemented (Y req/s)



Example:

Input: (1) Cart service
(2) 50% processing time reduction
(3) A mix workload

Output: ~6800 Req/s (from ~3900 Req/s)

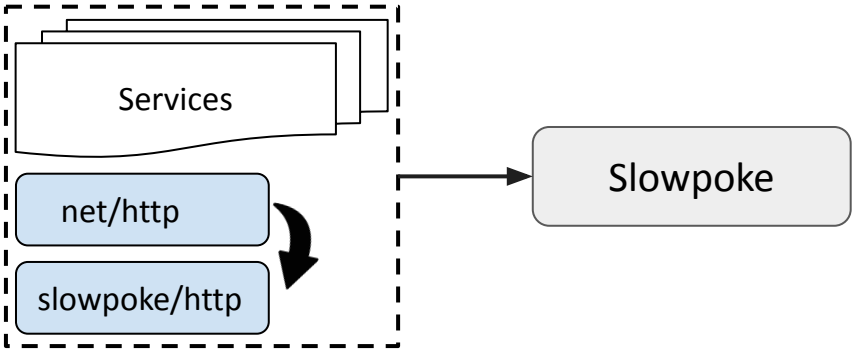
Using Slowpoke

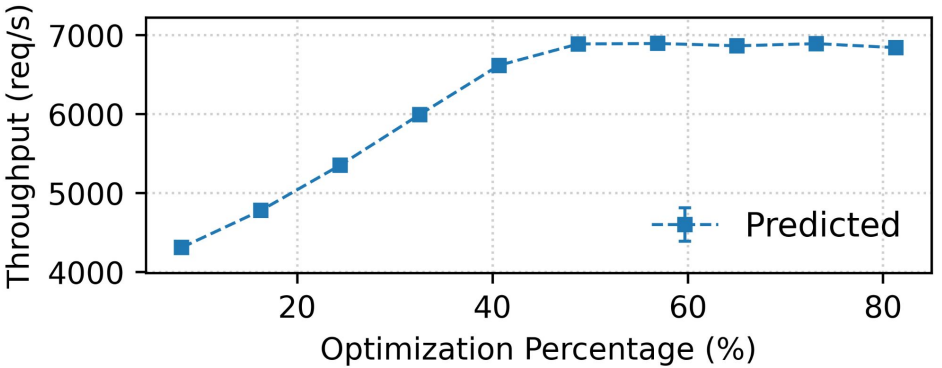
Inputs: (1) A target service X.

(2) A hypothetical optimization. E.g., reducing the processing time of 500ms by 50%.

(3) A workload.

Output: End-to-end throughput as if optimization was implemented (Y req/s)





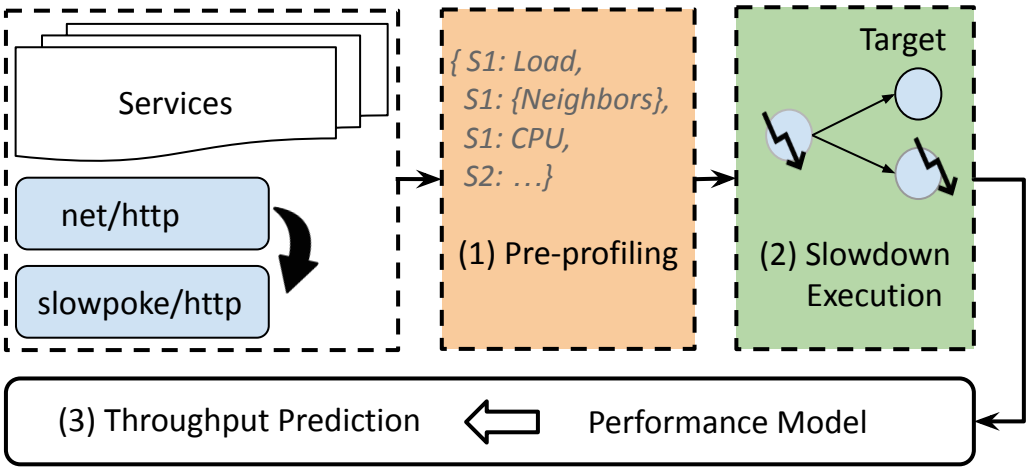
Two Executions and One Performance Model

Inputs: (1) A target service X.

(2) A hypothetical optimization. E.g., reducing the processing time of 500ms by 50%.

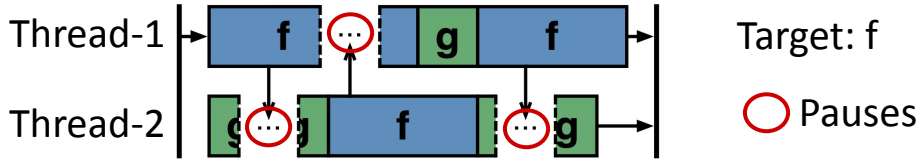
(3) A workload.

Output: End-to-end throughput as if optimization was implemented (Y req/s)

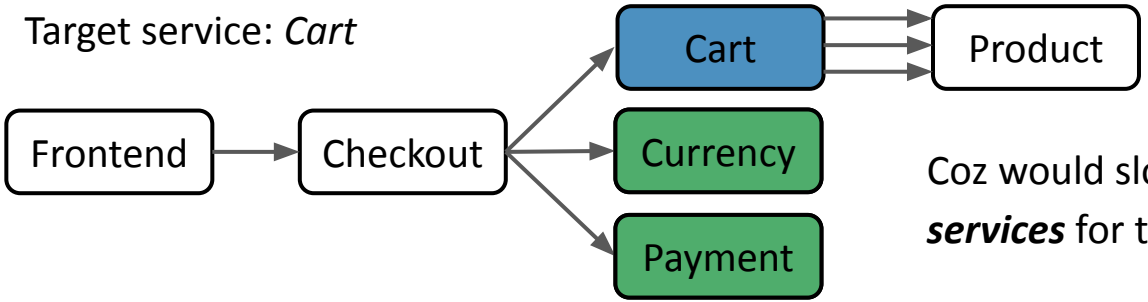


Applying Causal Profiling to Microservices is Hard

Coz (SOSP'15)



Target service: *Cart*



Coz would slow down ***threads in concurrent services*** for the same request.

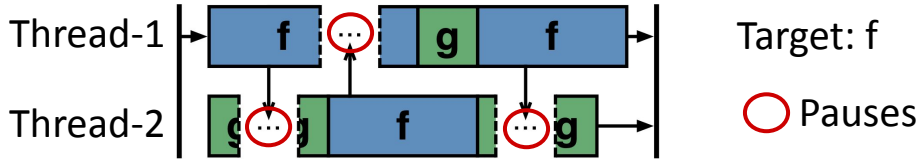
Identifying which threads to slow down incurs **high runtime overhead.**

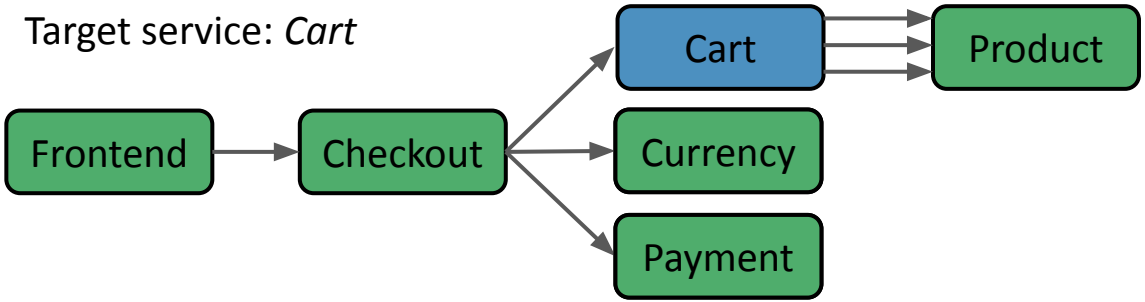
Slowing down these threads at the right time requires **careful synchronization.**

Charlie Curtsinger, and Emery D. Berger. "Coz: Finding code that counts with causal profiling." In Proceedings of the 25th Symposium on Operating Systems Principles. 2015.

Key Insight: Slowpoke Slows Down All Non-target Services

Coz (SOSP'15)

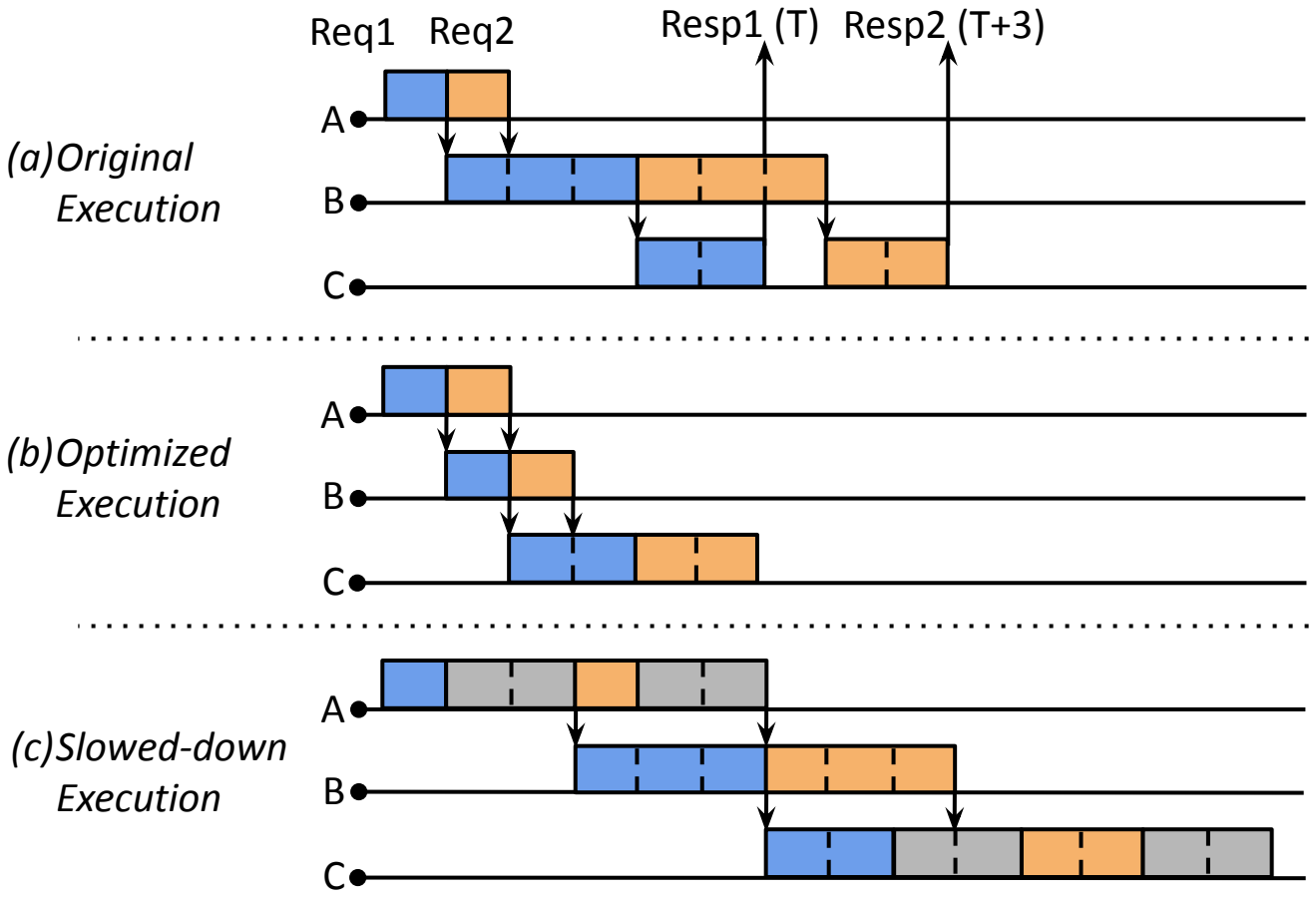




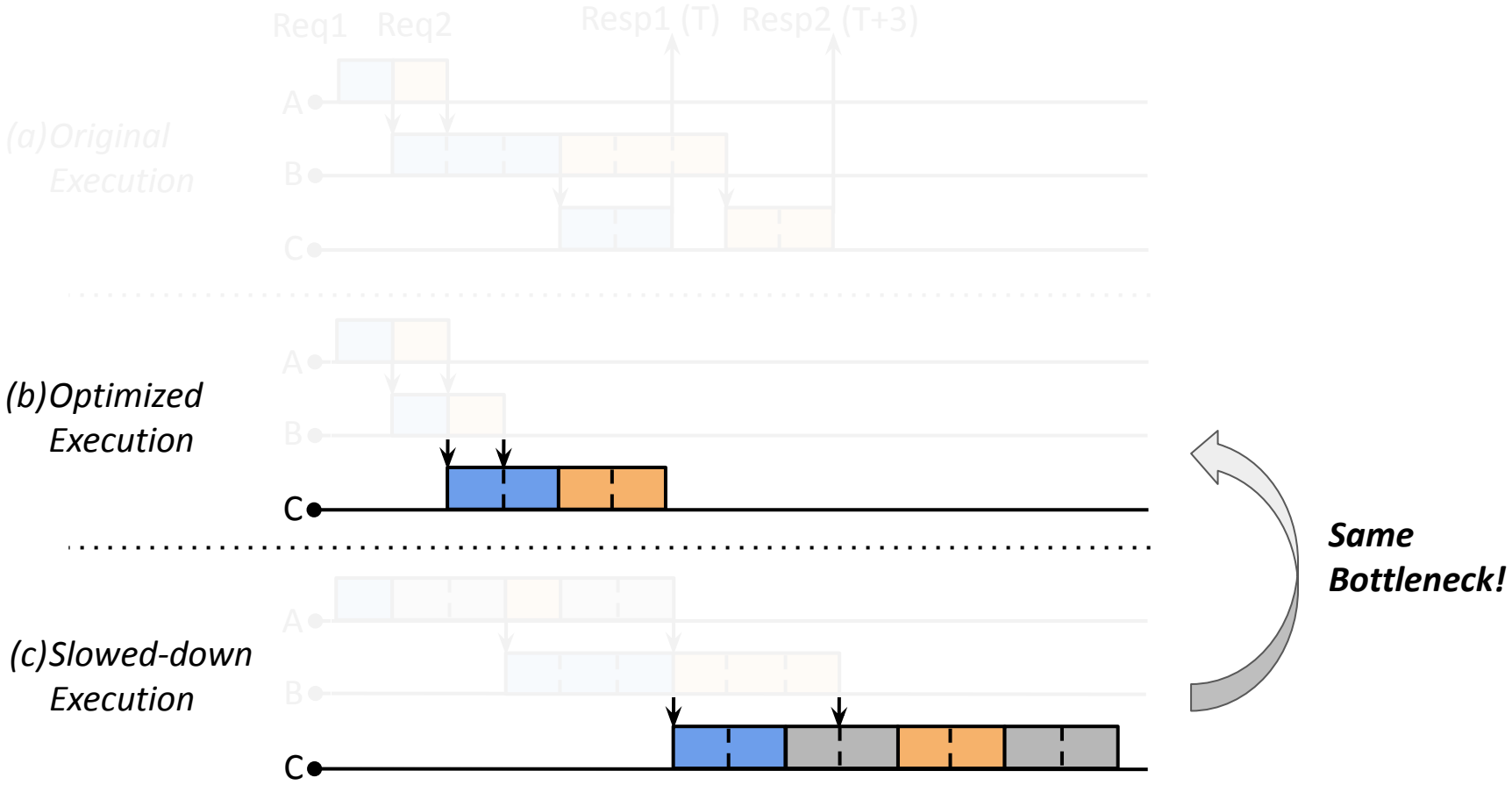
Slowpoke slows down ***all non-target services*** with distributed slowdown ***coordination***!

Charlie Curtsinger, and Emery D. Berger. "Coz: Finding code that counts with causal profiling." In Proceedings of the 25th Symposium on Operating Systems Principles. 2015.

Slowing Down All Non-target Services



Bottleneck Equivalence



Modeling Bottleneck Equivalence Using Linear Programming

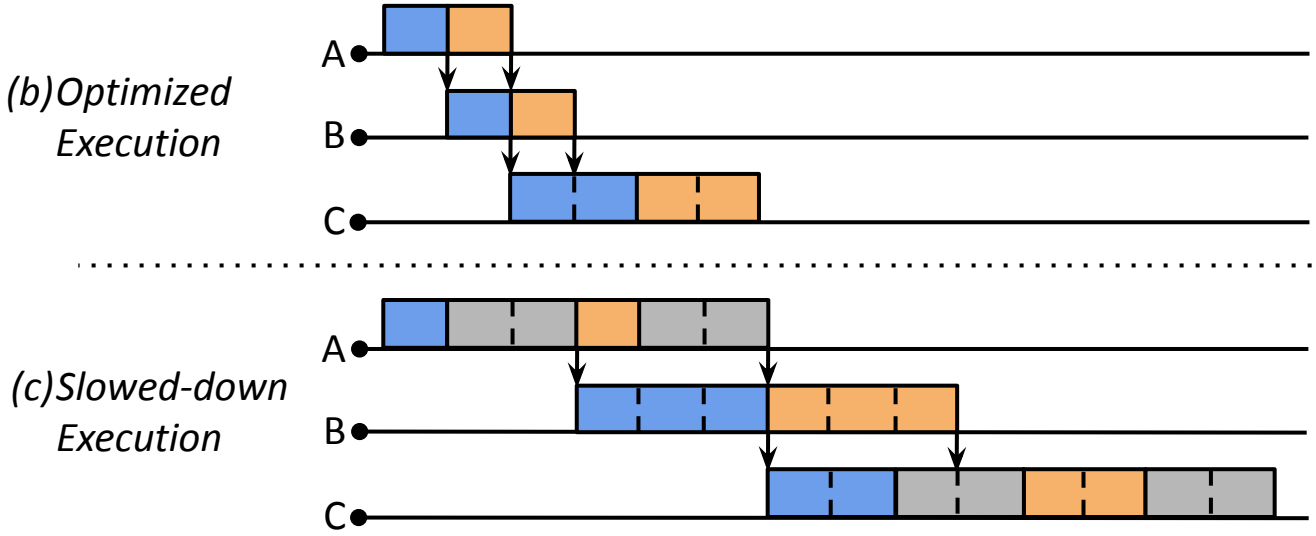
Slowpoke generalizes to:

- Multithreaded services
- Dynamic call graphs
- Different types of bottlenecks, e.g., mutexes.

Maximize end-to-end throughput

Subject to { Constraint - A, Constraint - B, Constraint - C }

E.g., CPU usage <= CPU quota

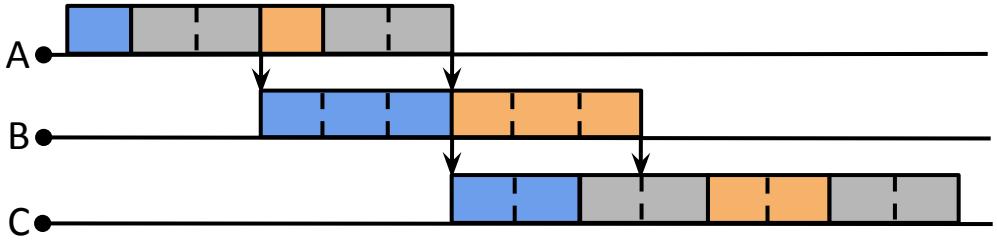


Linear program (opt)

Constraint equivalence

Linear program (slowdown)

Ideal Slowdown Scenarios

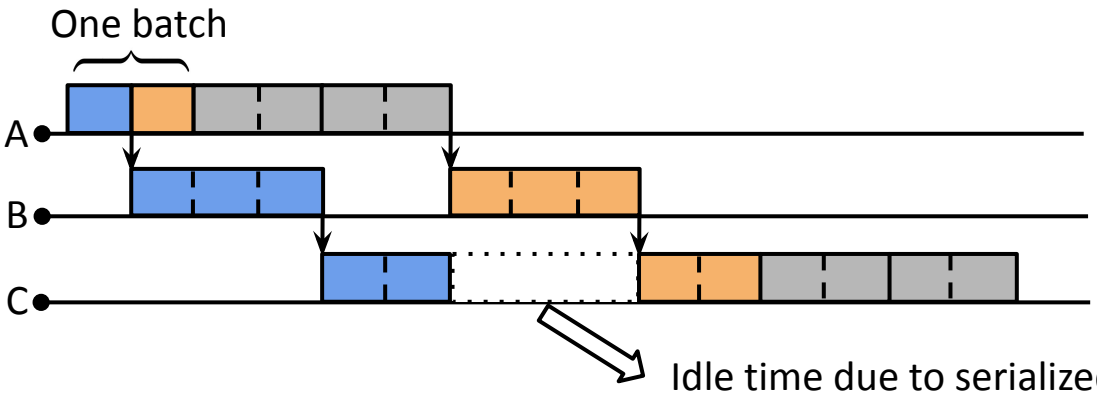


Pause: signal

Ideal slowdown

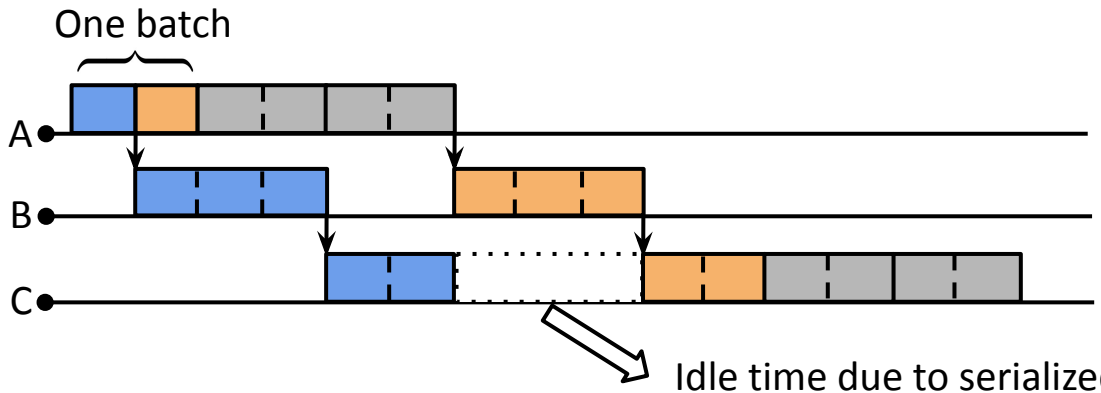
High runtime overhead

Naïve Batching Introduces Idle Time

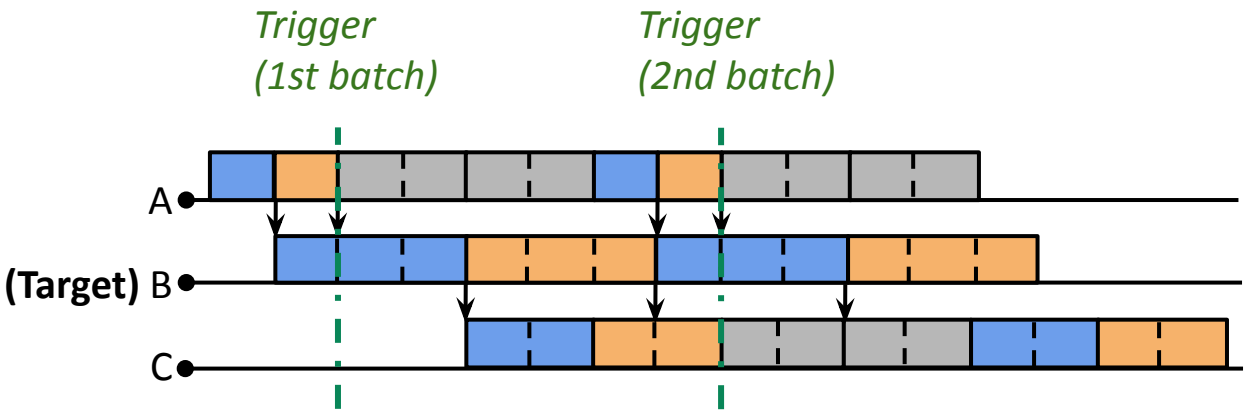


Naïve batching
(E.g., with size=2 req)

Slowpoke Uses Distributed Slowdown Coordination



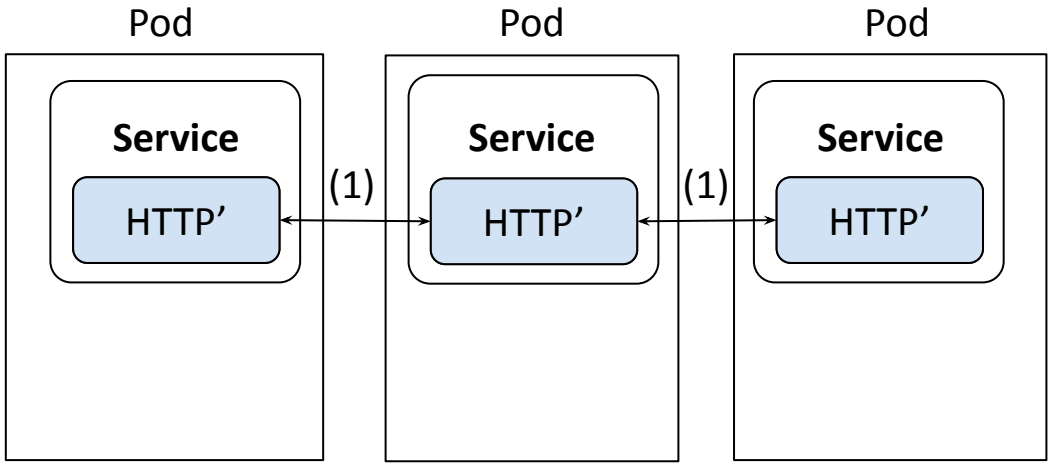
Naïve batching
(E.g., with size=2 req)



Batching with **coordination**

- Triggered by target service
- Gossip-style propagation
- Paused for batched slowdowns

Data Plane: Instrumented Libraries

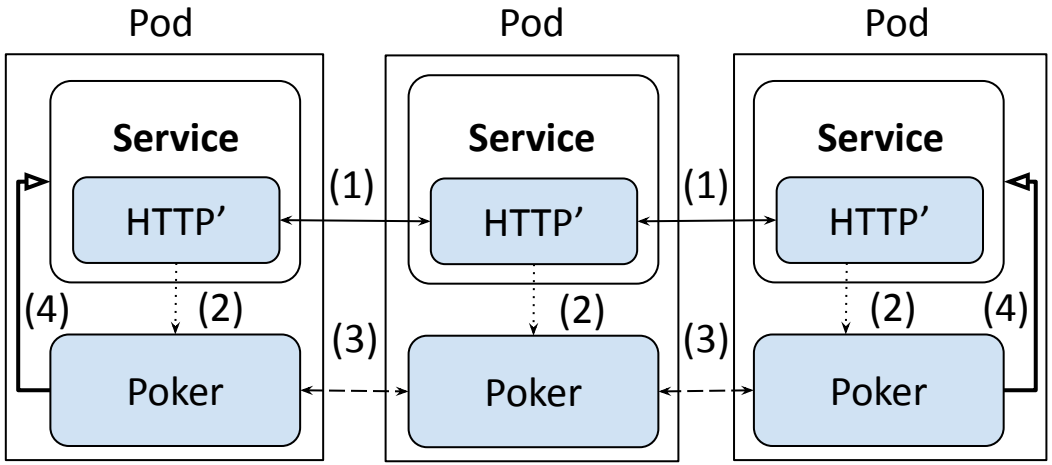


Libraries:

- Track batched slowdowns

⟷ (1) Data channel: HTTP/gRPC request/response

Data Plane: Poker



(1) Data channel: HTTP/gRPC request/response

(2) Unix pipe

(3) Control channel—TCP

(4) SIGSTOP/SIGCONT signals

Libraries:

- Track batched slowdowns

Poker:

- Collect batched slowdowns
- Coordinate slowdowns
- Pause services

Evaluation

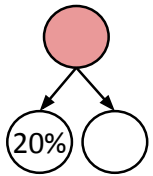
How accurate is Slowpoke's throughput prediction?

Evaluation Overview

	Benchmarks	Services	LOC
4 applications	OnlineBoutique	9	1088
	HotelReservation	6	608
	SocialNetwork	6	532
	MovieReview	12	913
108 synthetic applications →	Hydragen[1]	3–43	389

$$\text{Error} = \frac{\text{Predicted - Ground Truth}}{\text{Ground Truth}}$$

9 topologies

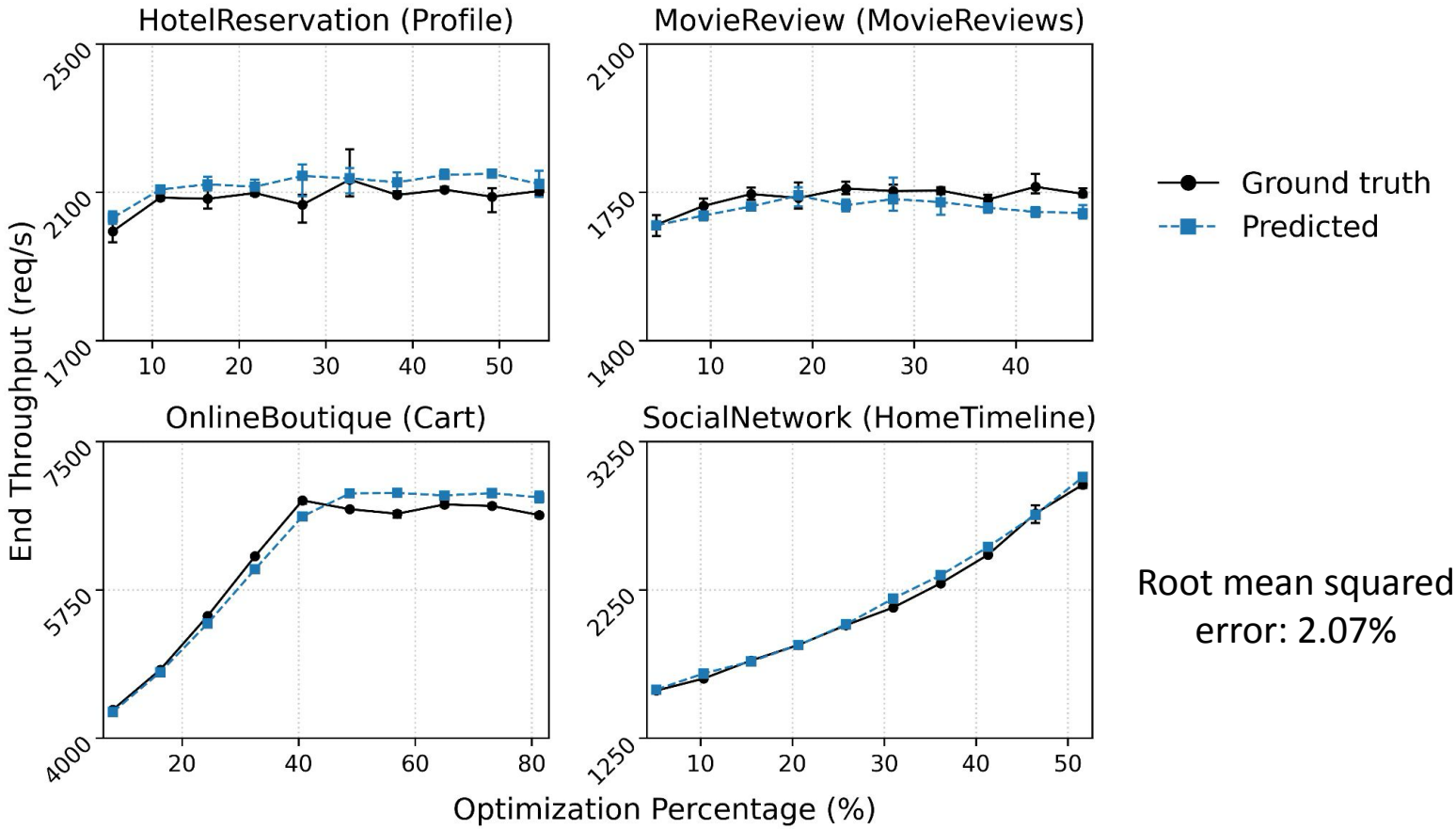


3 configurations

- Protocol: e.g., HTTP
- Request concurrency: e.g., sequential
- Target service position: e.g., root in a tree

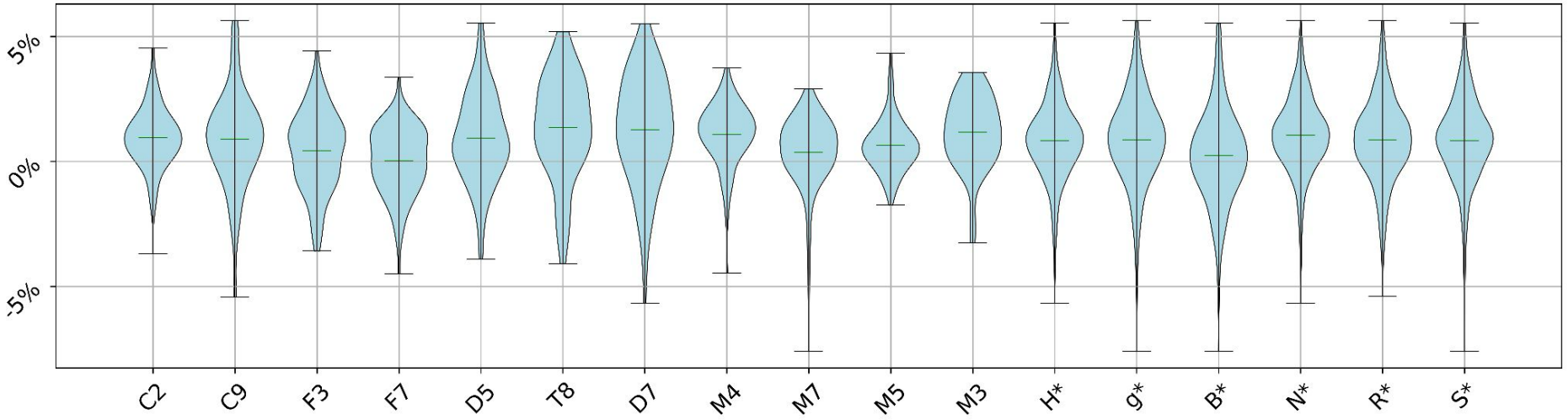
[1] Mohammad Reza Saleh Sedghpour, Aleksandra Obeso Duque, Xuejun Cai, Björn Skubic, Erik Elmroth, Cristian Klein, and Johan Tordsson. "Hydragen: A microservice benchmark generator." In 2023 IEEE 16th International Conference on Cloud Computing (CLOUD), pp. 189-200. IEEE, 2023.

Prediction Results for Four Applications



Root mean squared error: 2.07%

Prediction Results for Synthetic Benchmarks

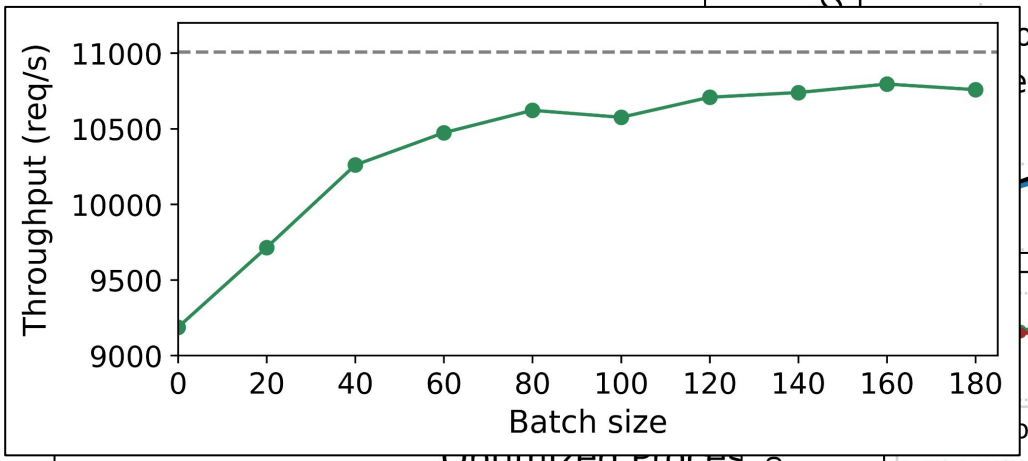


C2—Chain (length 2)
C9—Chain (length 9)
F3—Fan-out (width 3)
F7—Fan-out (width 7)
D5—DAG (5 services)
T8—Unbalanced tree
D7—DAG (7 services)
M4—Dynamic chain
M7—Dynamic tree
M5—Dynamic cycle
M3—Dynamic cache
H*—HTTP, g*—gRPC
B*—target is bottleneck
N*—target is not bottleneck
R/S*—concurrent/sequential

108 Benchmarks × 10 Optimization Percentages

Root mean squared error: 1.89%

More Results in the Paper



- Realistic scaling optimizations
- Mutex-induced bottlenecks
- Large-scale deployment
- A set of microbenchmarks

Slowpoke: End-to-end Throughput Optimization Modeling for Microservice Applications

github.com/atlas-brown/slowpoke

Yizheng Xie, Di Jin, Oğuzhan Çölkesen, Vasiliki Kalavri[BU], John Liagouris[BU], Nikos Vasilakis

atlas.cs.brown.edu sites.bu.edu/casp