Teasing Out Causal Relationships in Distributed Systems by Adding Delays to gRPCs

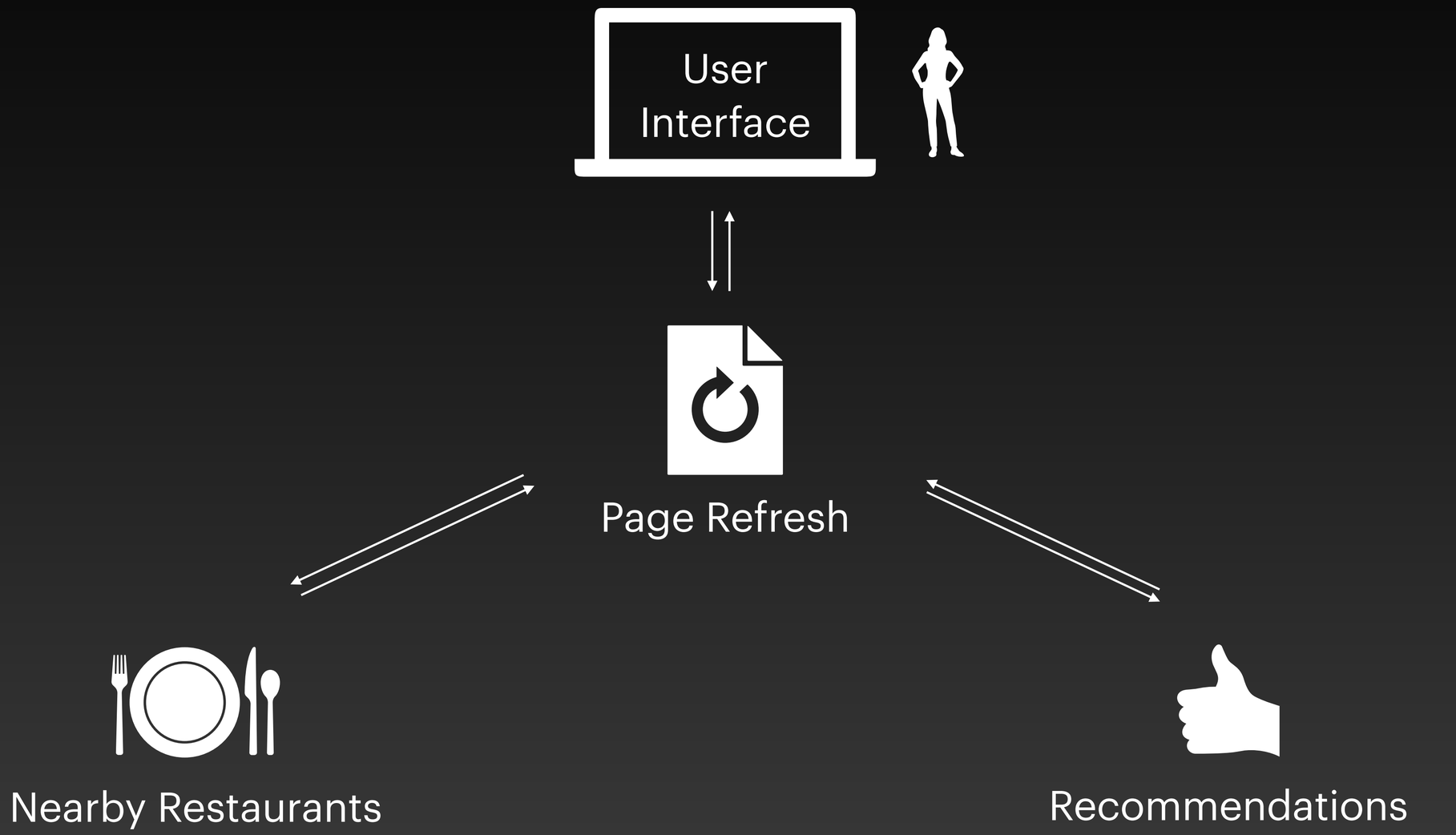
Mona Ma

May 10th 2024

Introduction

- Definitions
- Motivation

Microservices - Parent & Siblings



Execution Design

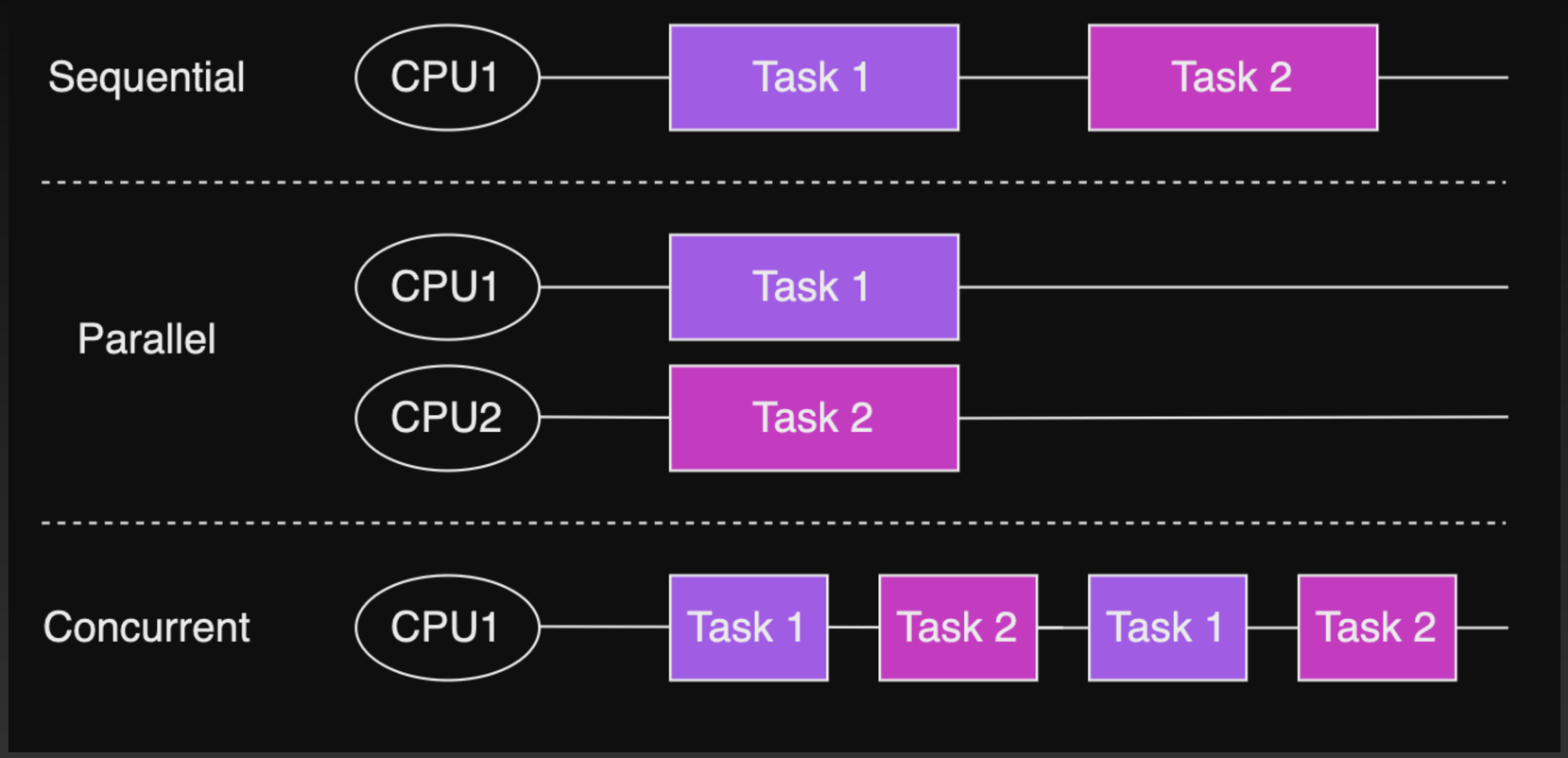


Figure 1: Visualization of Workflow

Challenge

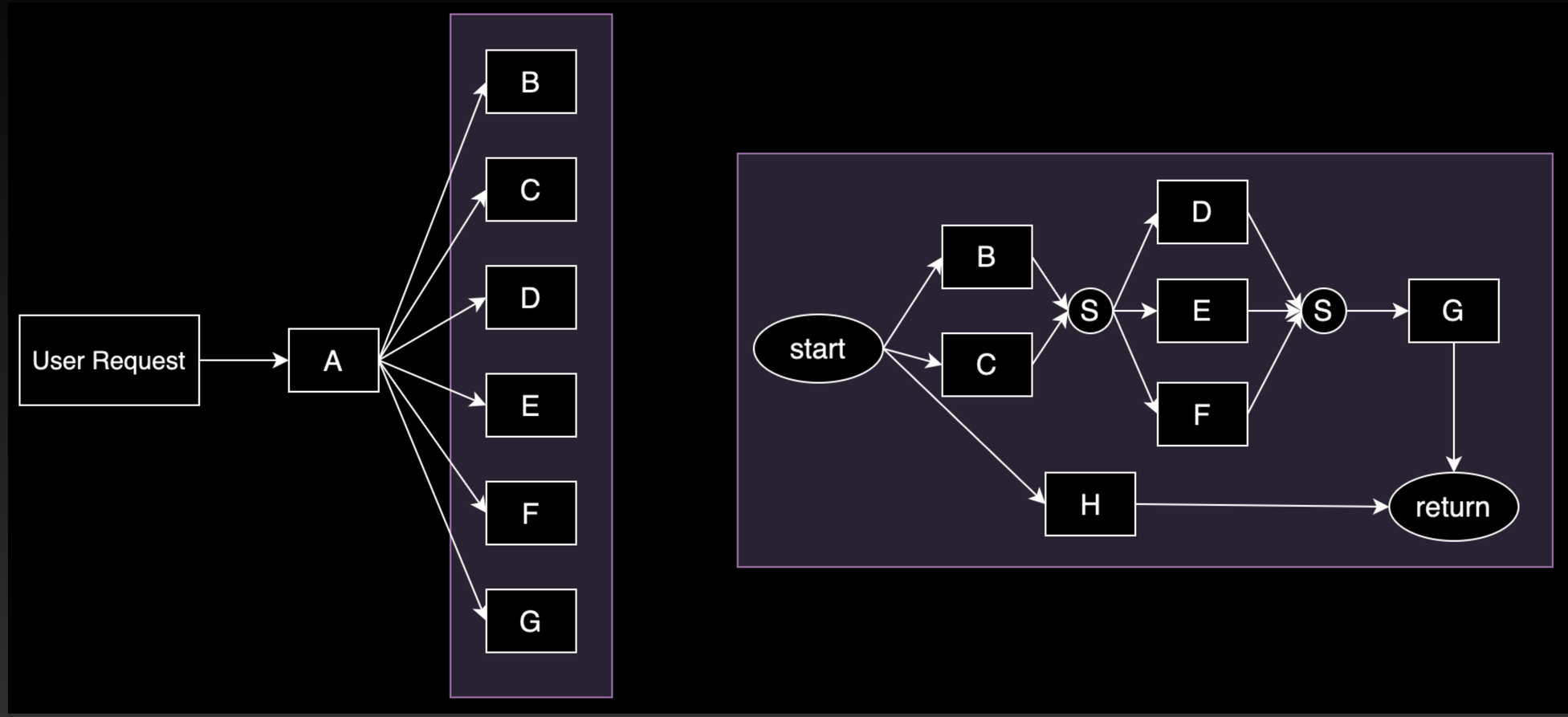


Figure 2: A Black Box within the System

Caller/Callee Versus Dependency

Problem statement

- Conduct a dependency analysis
- **Goal:** determine whether the siblings of a parent service are executed sequentially or in parallel
- **Outcome:** what can be parallelized -> performance optimization

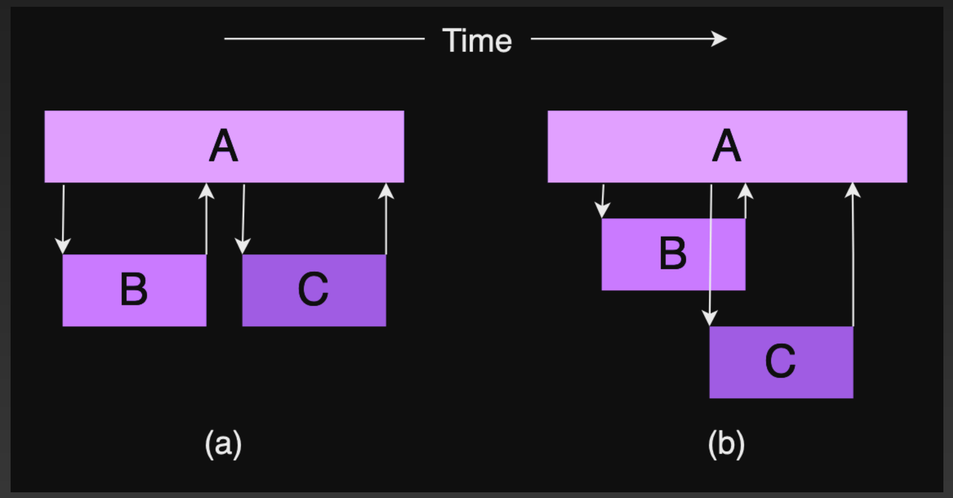


Figure 3: Sibling Services in a Simple Scenario

Deferred/Asynchronous Computation

Programming Languages	Functionality
Python	concurrent.futures aync-io
JavaScript	promises, async/await
Java	java.util.concurrent, CompletableFuture
Go	Goroutines, Channels
...	...

Table 1: Different Programming Languages and Their Related Functions for Implementing Concurrency

Approach

- Active perturbation
- Add delays to selective sibling’s return gRPC call

Mechanism

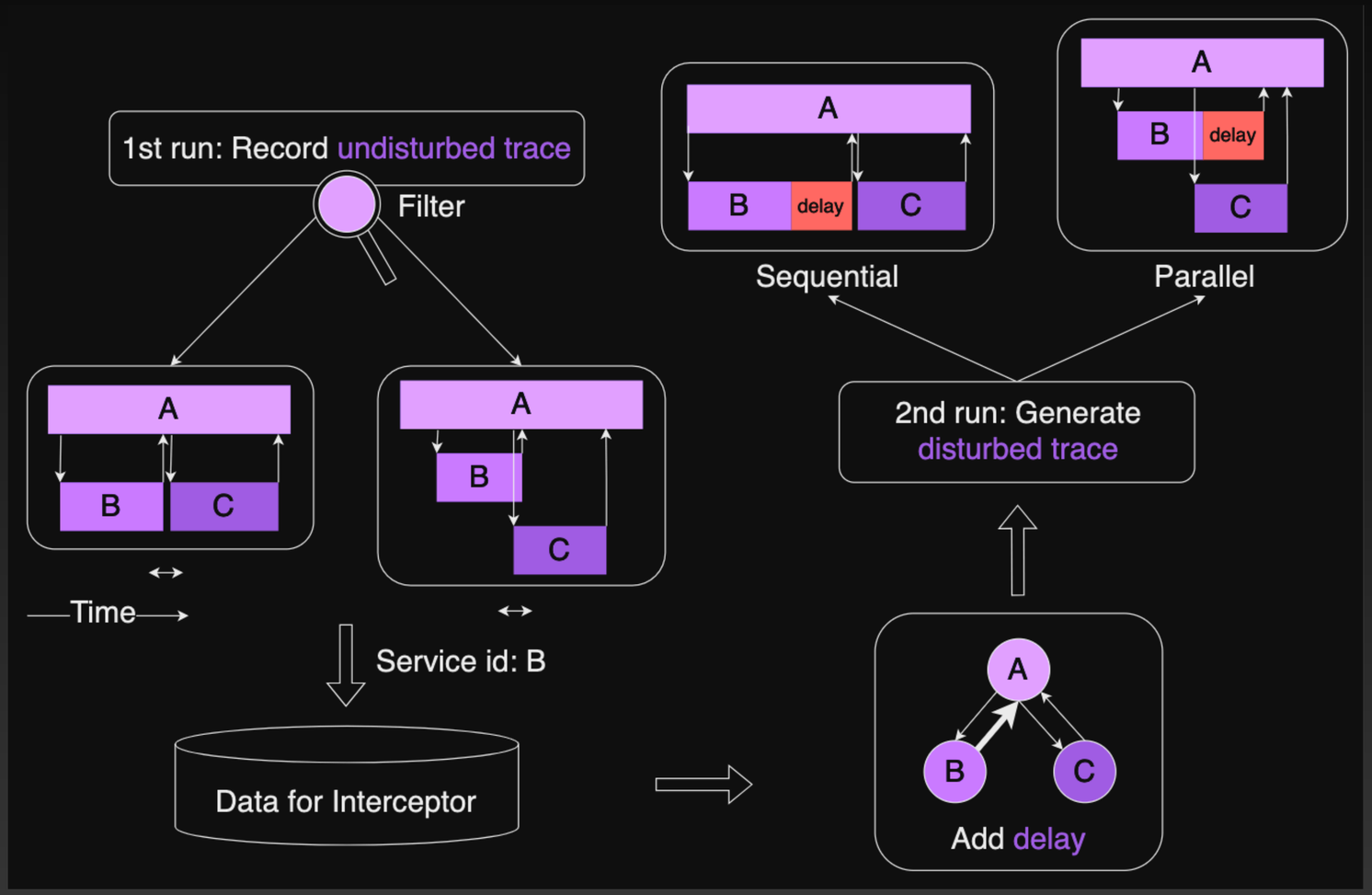


Figure 4: Project Workflow

- 2 Executions in total
- 1st Run:
  - Identify locality
  - Record interested services & timestamps
- 2nd Run:
  - Delay specific retuned response
  - Record timestamps
- Compare the results to summarize happen-before relationships

RPC & gRPC

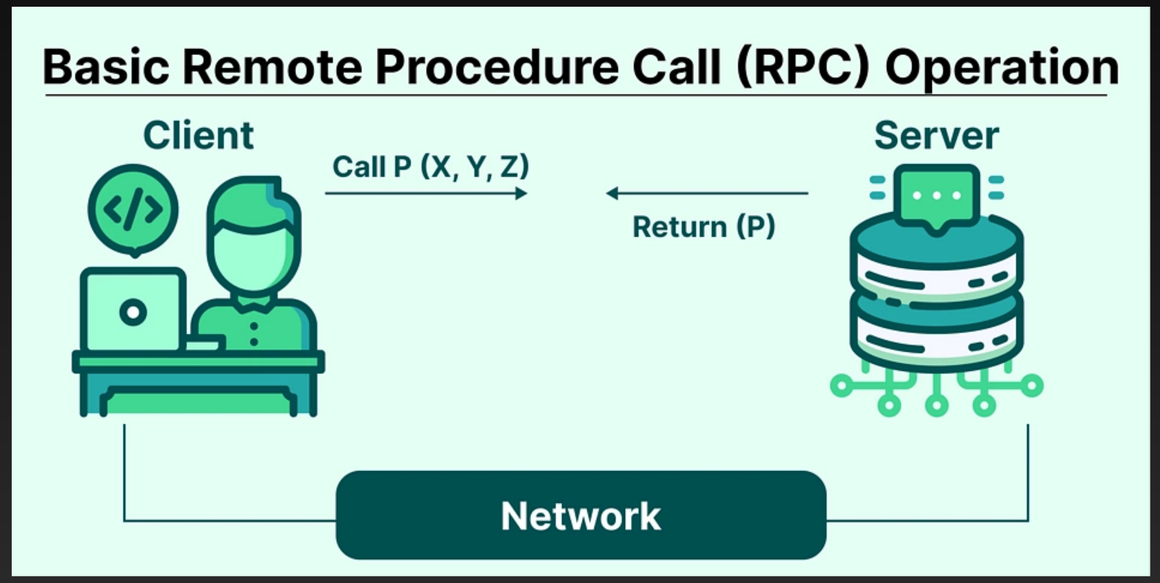


Figure 5: RPC

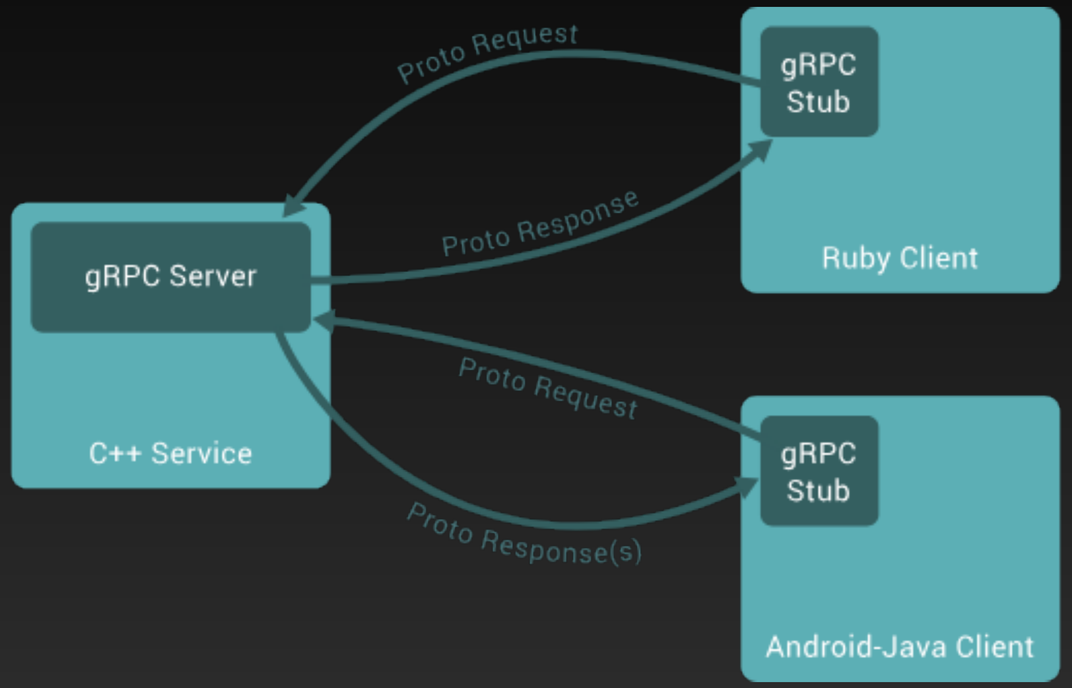


Figure 6: gRPC

gRPC Interceptor

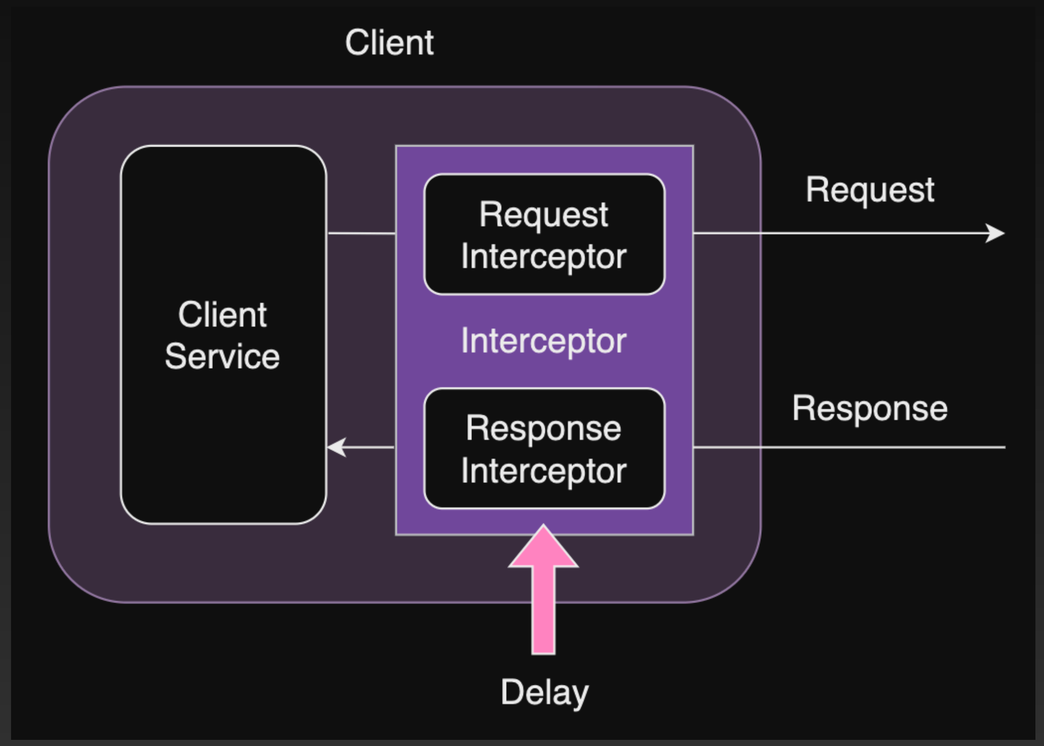


Figure 7: Client Side Interceptors

- Filters/middleware
- Popular use cases:
  - Metadata handling
  - Logging
  - Authentication & Authorization
- Add delays (e.g. 0.001 seconds)

Implementation

- Reusable gRPC interceptor class & functions
- Input parameters:
  - delay_ms: int
  - service_ids: Optional[list[str]]
- Output:
  - Timestamps

- Toy Demo: Total 270 lines of code
- Interceptor: 15 lines of code

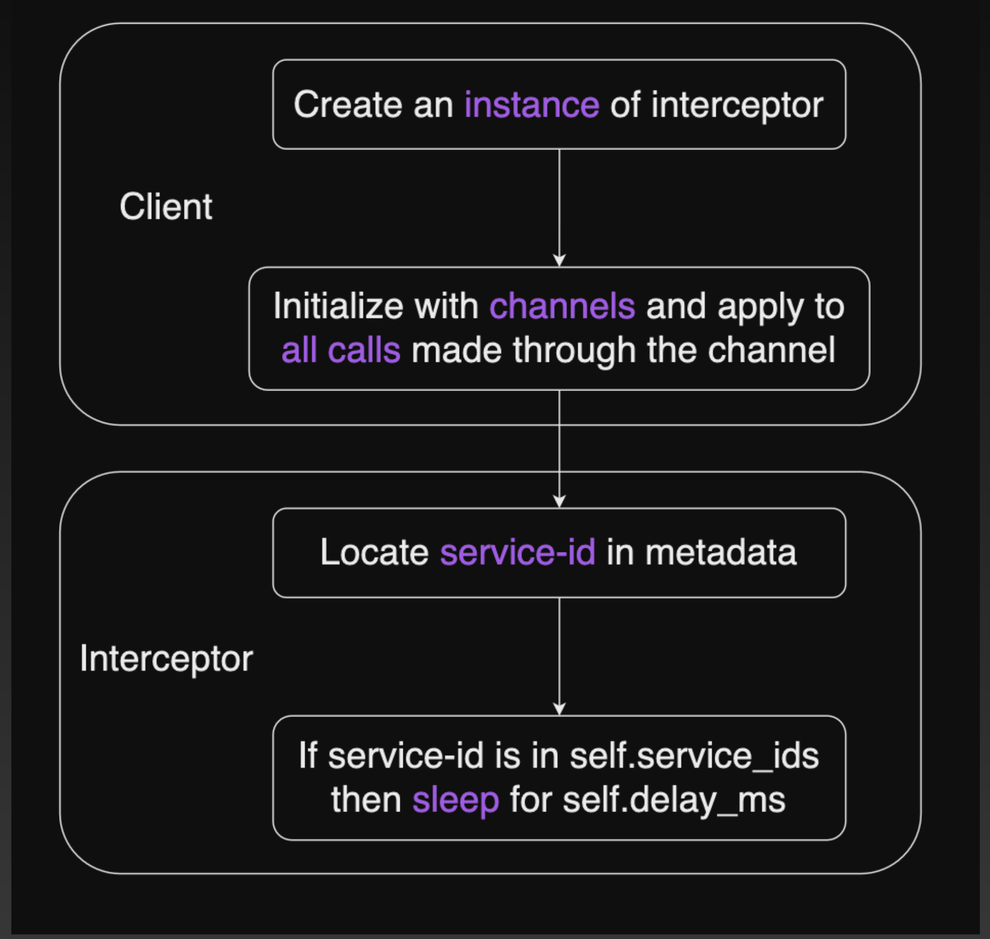


Figure 8: Implementation Steps

Toy Demo

- Using Python async.io
- 2 Microservices
  - Restaurant
  - Recommendation
- 4 gRPC calls
  - 2 requests & 2 responses
- 3 Modes
  - Concurrent
  - Sequential
  - Multiple

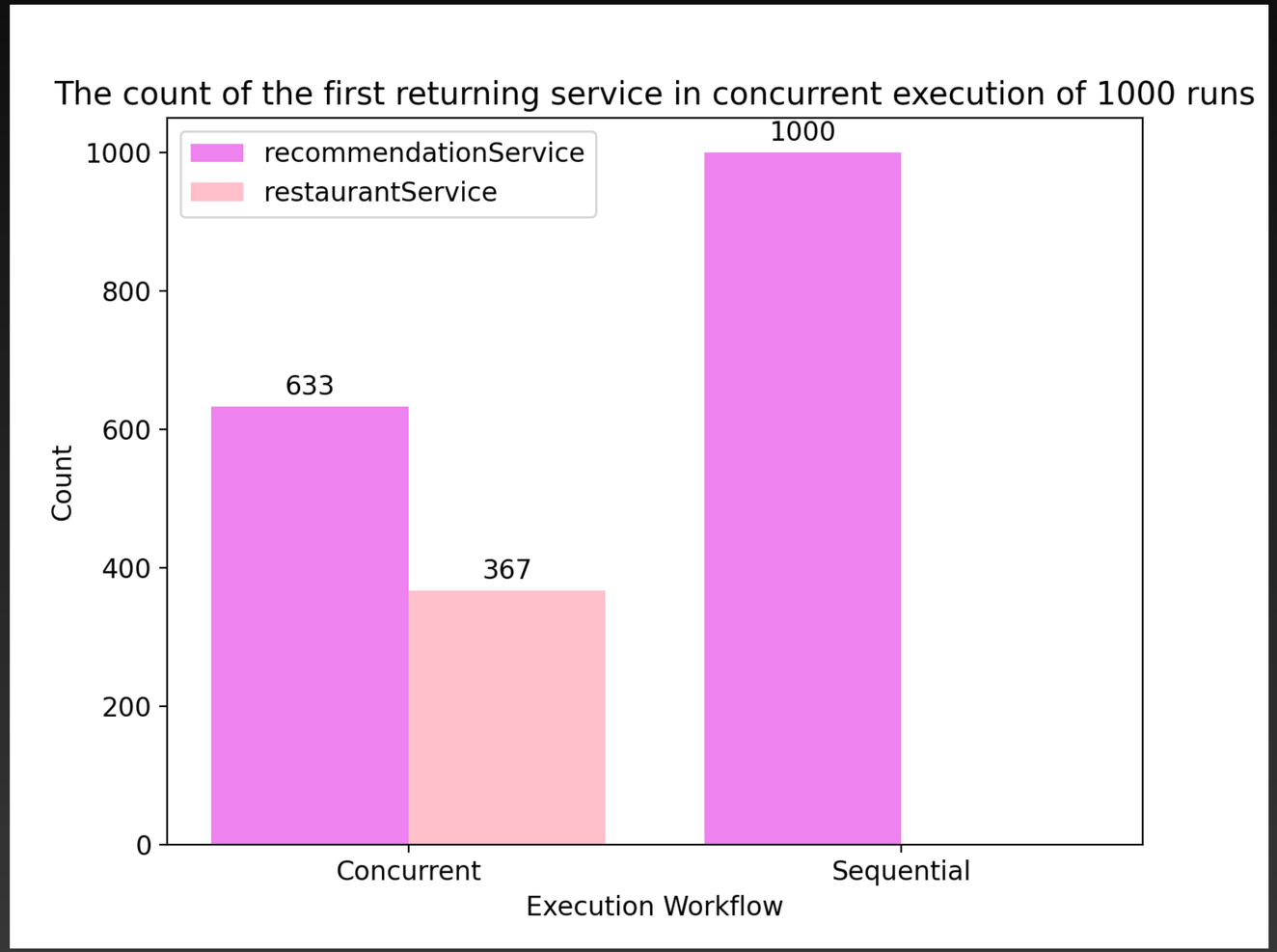


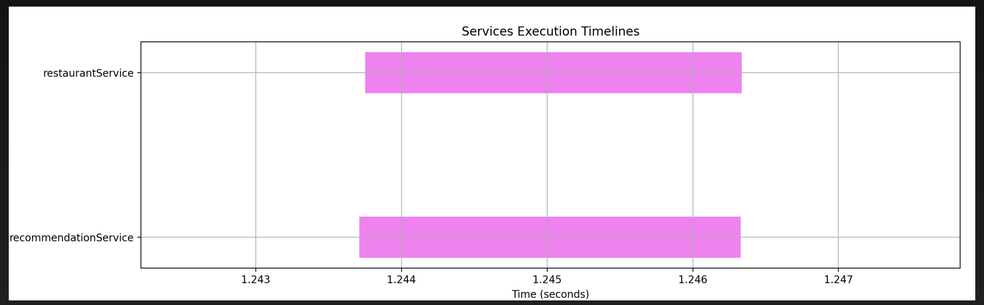
Figure 9: Toy Demo Validation

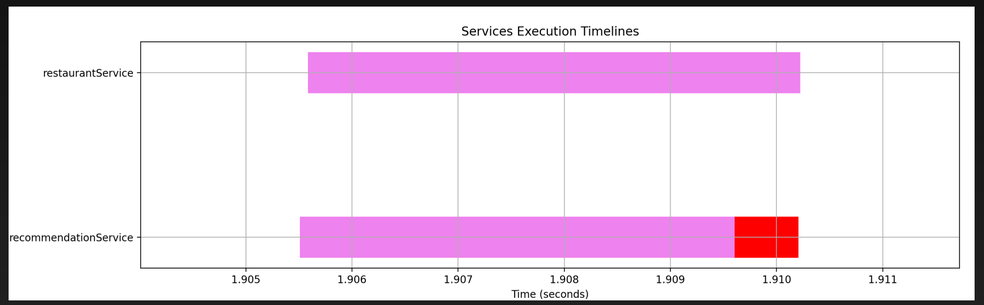
Toy Demo Output

Without Delay

With Delay

Concurrent

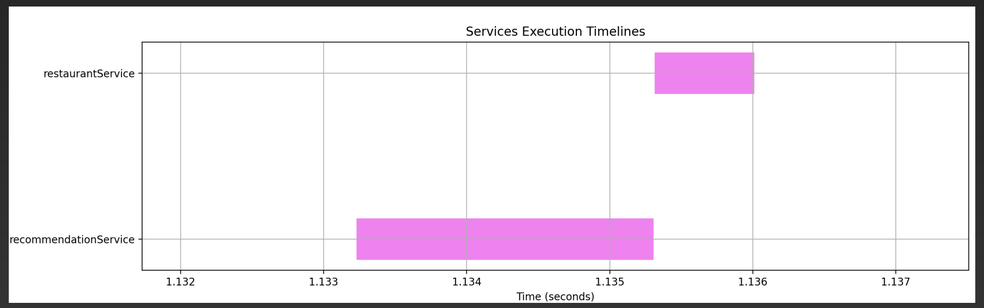


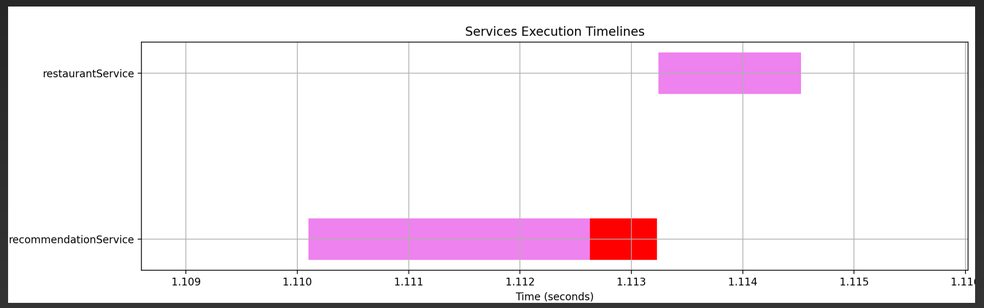


Recommendation Service usually returns first (6:4) -> only Restaurant Service returns first

Figure 10: Toy Demo Output

Sequential





Recommendation Service always returns first

Evaluations

- Add Server Interceptor to OTel Demo

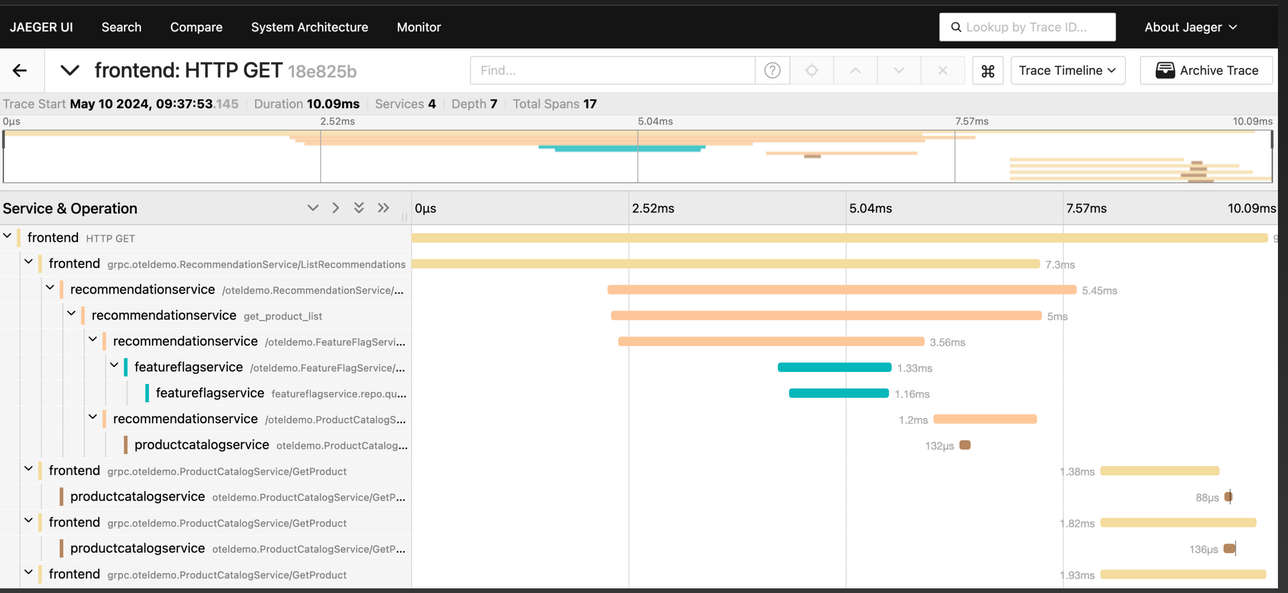
- Overheads

-Precision

Is the Result Valuable/Significant?

- Effective dependency analysis
- Construct the true structure -> validated with Jeager visual
- A working proof of concept

Undisturbed



Disturbed using server-side Interceptor

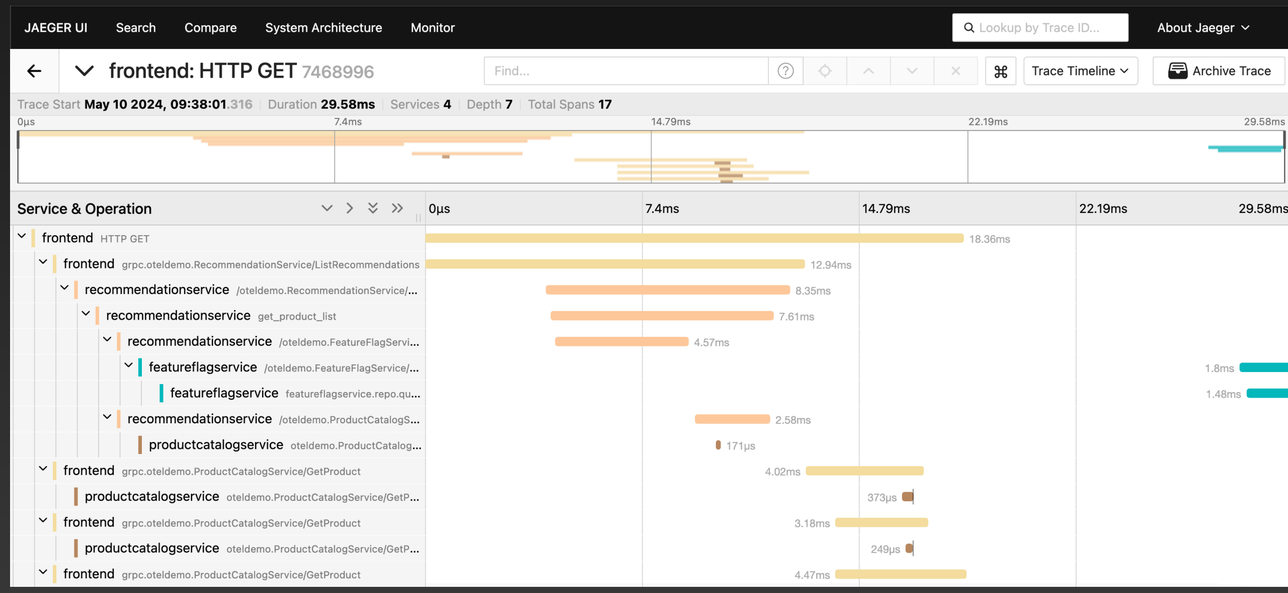


Figure 11: Jeager Trace Comparison

OpenTelemetry Demo

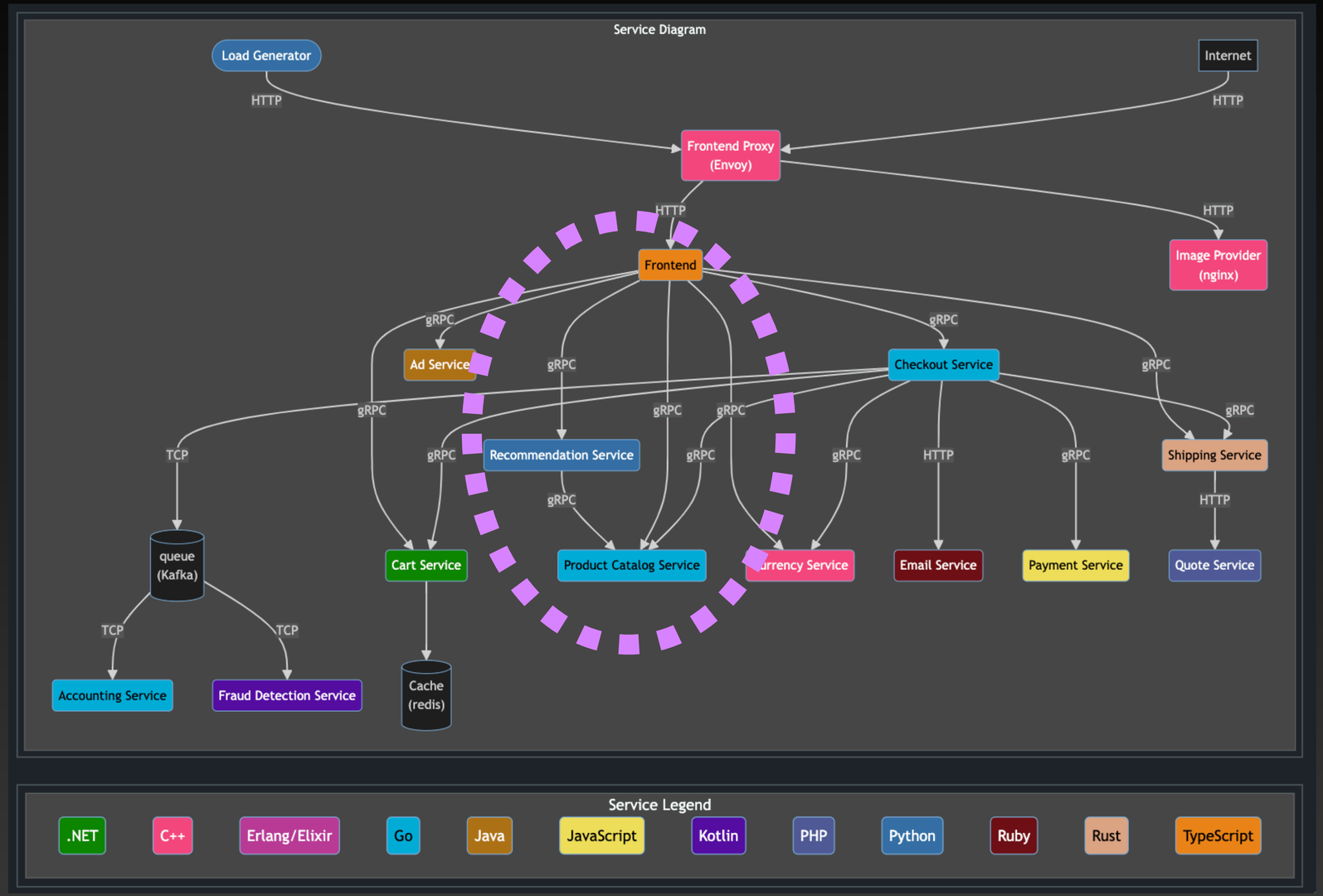


Figure 12: OpenTelemetry Astronomy Shop Microservices Architecture

- 15 lines of code
- Retrieve service name from docker-compose.yml
- 5 seconds delay before returning back to frontend
- Reveal the true dependency: sequential workflow

What is the Sensitivity/Ripple Effect?

- Overheads versus delays (0 to 0.1 with step 0.001)

Concurrent

Sequential

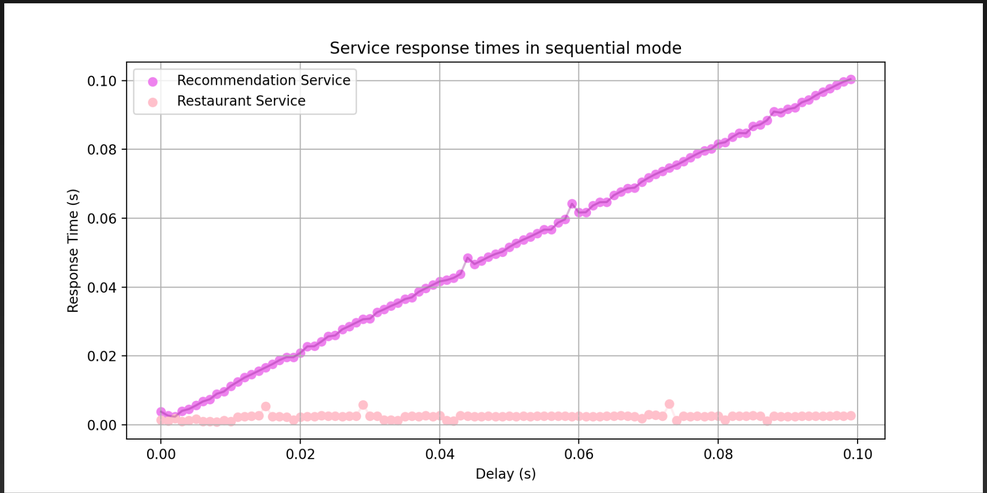


Figure 13: Measuring overheads

- Undisturbed sibling service in concurrent mode experience overhead

What is the Precision: Expected Versus Actual Delay?

- Actual latency is less than intended amount of delay (0.01 seconds)

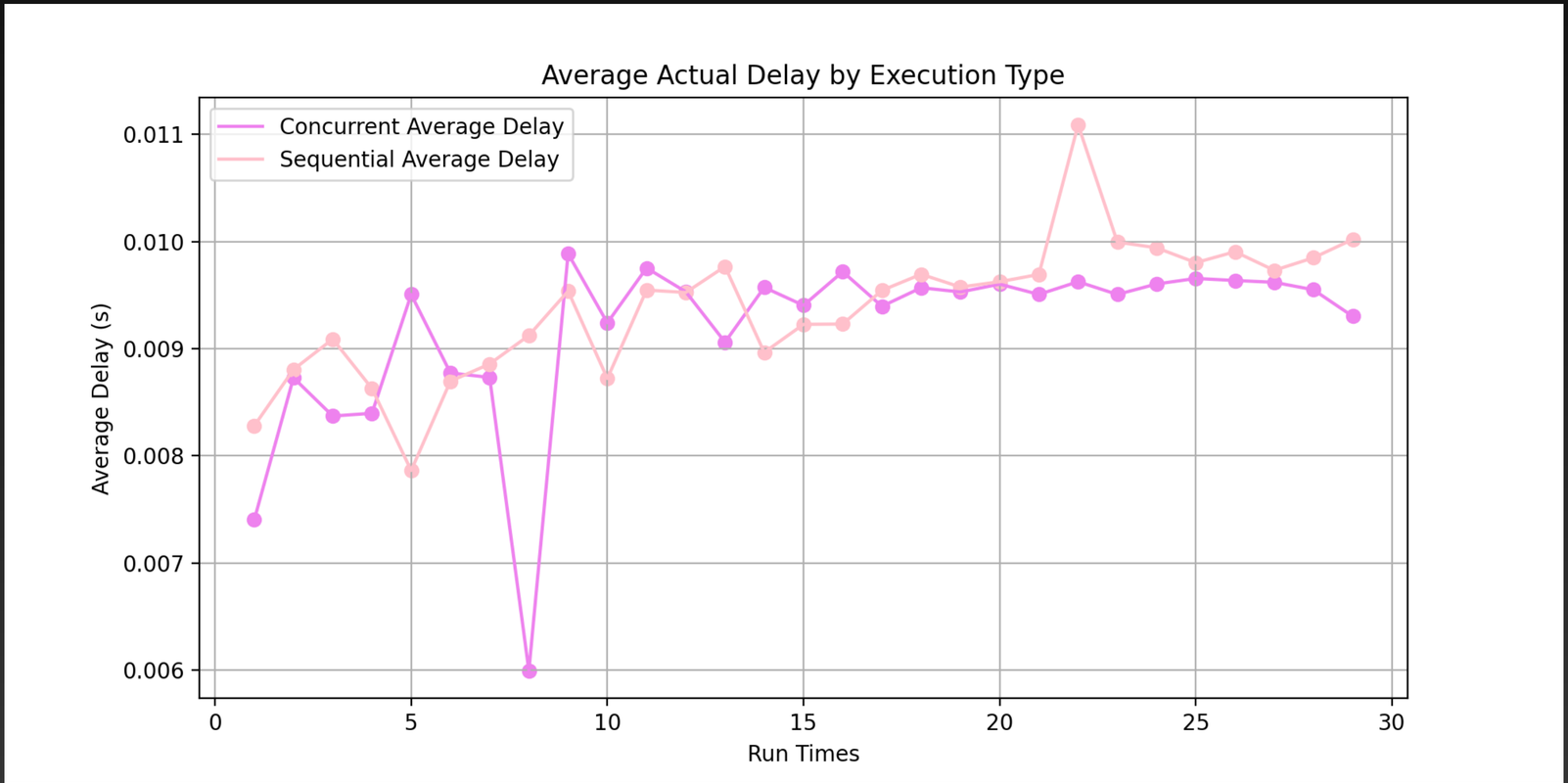


Figure 14: Actual vs expected delays

Future Work

gRPC Interceptors are Limited -> Envoy

Language Support

Language	Example
C++	C++ example
Go	Go example
Java	Java example
Python	Python example

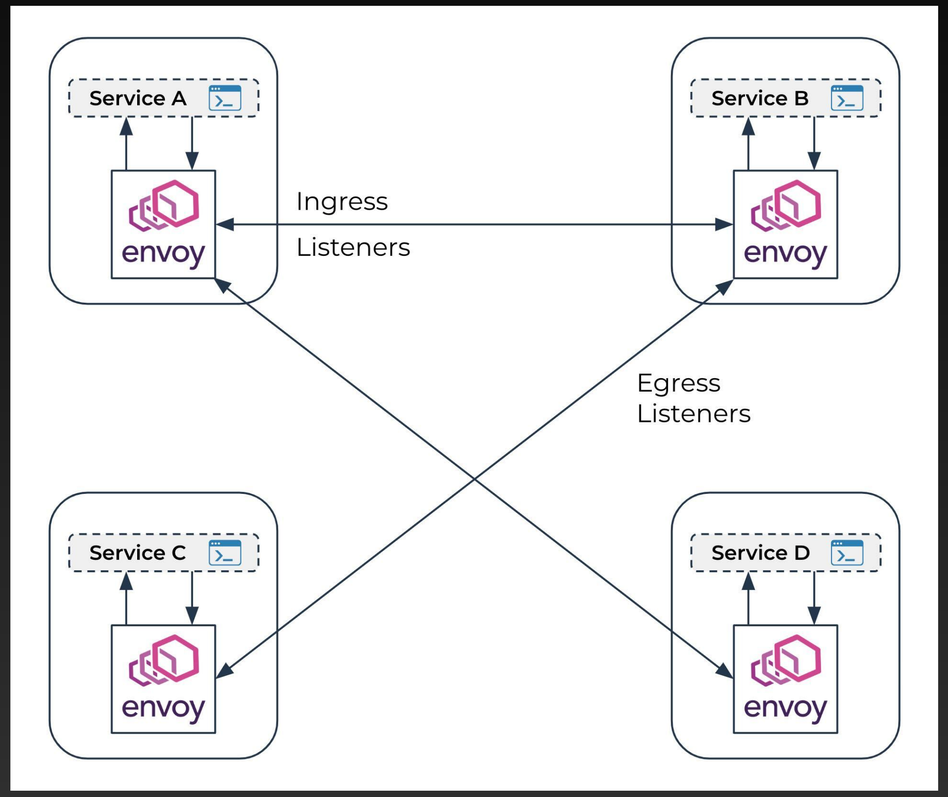


Figure 15: Envoy Architecture

- Open-source service proxy
- Language agnostic
- Observability
- Extensible

- Resource intensive
- Complexity in debugging
- **Hard to compile**

Reference:

Figure 5: https://www.coingecko.com/learn/crypto-rpc-best-rpc-providers
Figure 6: https://grpc.io/docs/what-is-grpc/introduction/
Figure 12: https://opentelemetry.io/docs/demo/architecture/
Figure 15: https://www.solo.io/topics/envoy-proxy/


Thank you for listening!