# AI-Assisted Performance Feedback in API Programming

Malith Jayasinghe

VP of Research & AI, WSO2 inc.

# Expecting a

PERSONALIZED

REAL-TIME

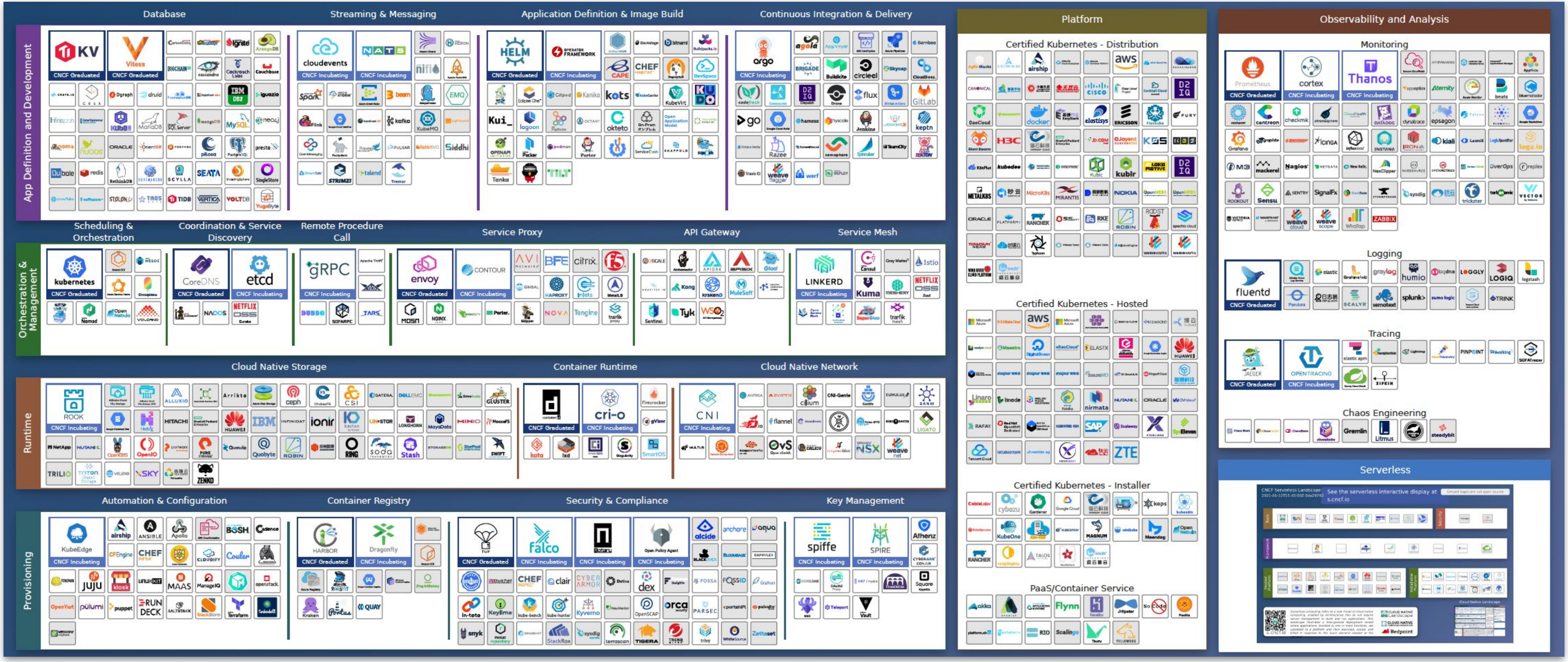
GEO-SENSITIVE

PREDICTIVE

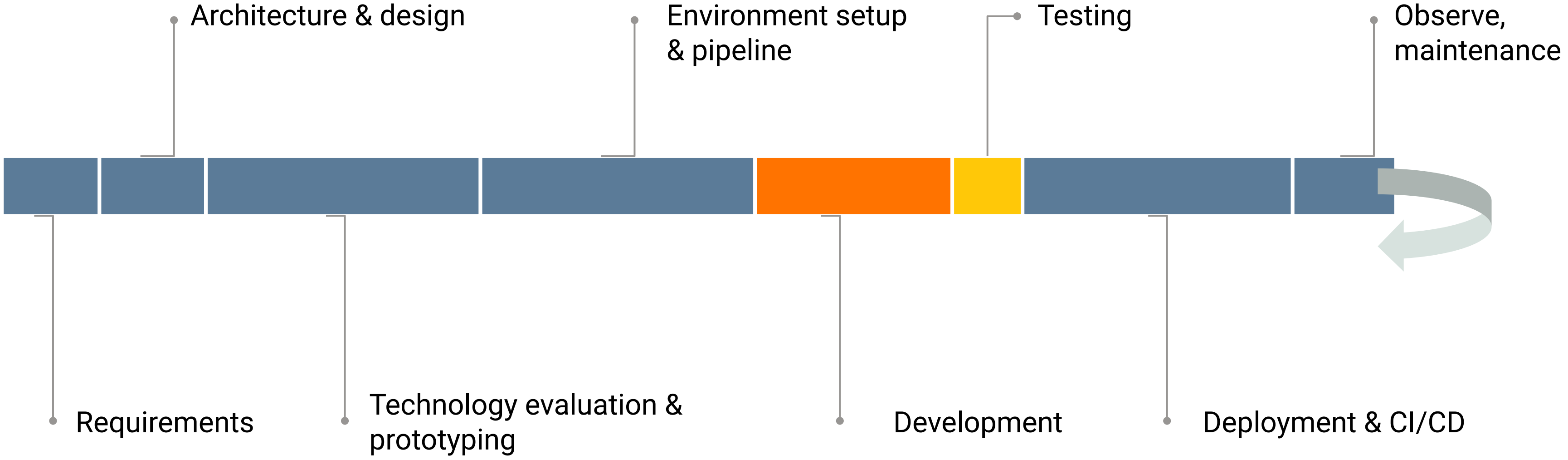
# Digital Experience

**Picture credit**: https://www.freepik.com/

# Development teams focus less time on building digital experiences


App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Serverless
CNCF Graduated
CNCF Incubating

# Application lifecycle and time spent on each stage



Enterprises need a readily available **platform for innovation** and an enhanced **engineering practice** – to do this right, we have to adopt a new engineering paradigm.

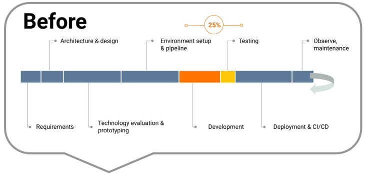
## Digital Platform

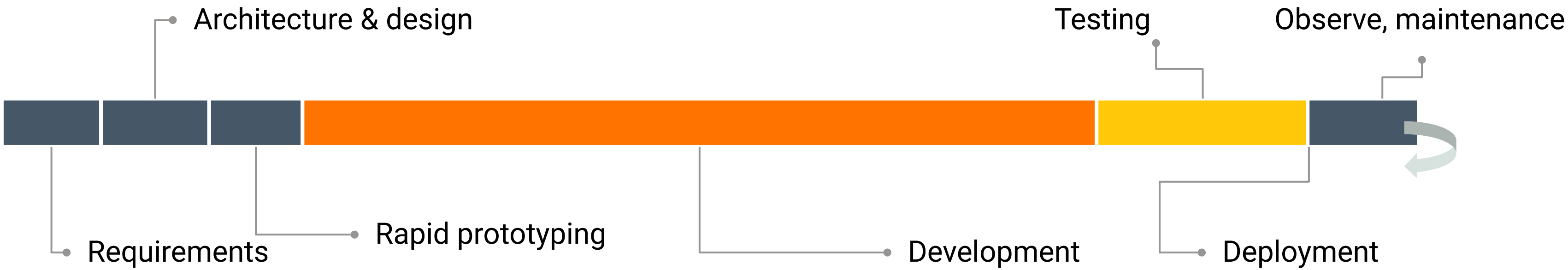
A Digital Platform provides a collection of business and technology capabilities that technologists within and beyond IT can use to deliver their own digital capabilities.

## Digital Experience Engineering

Digital Experience Engineering is how businesses create and operate new digital experiences for their stakeholders by creating digital applications.

# Application lifecycle and time spent on each stage with Choreo





Increase developer productivity with low-code, AI-assisted development, and a pipeline tuned environment.

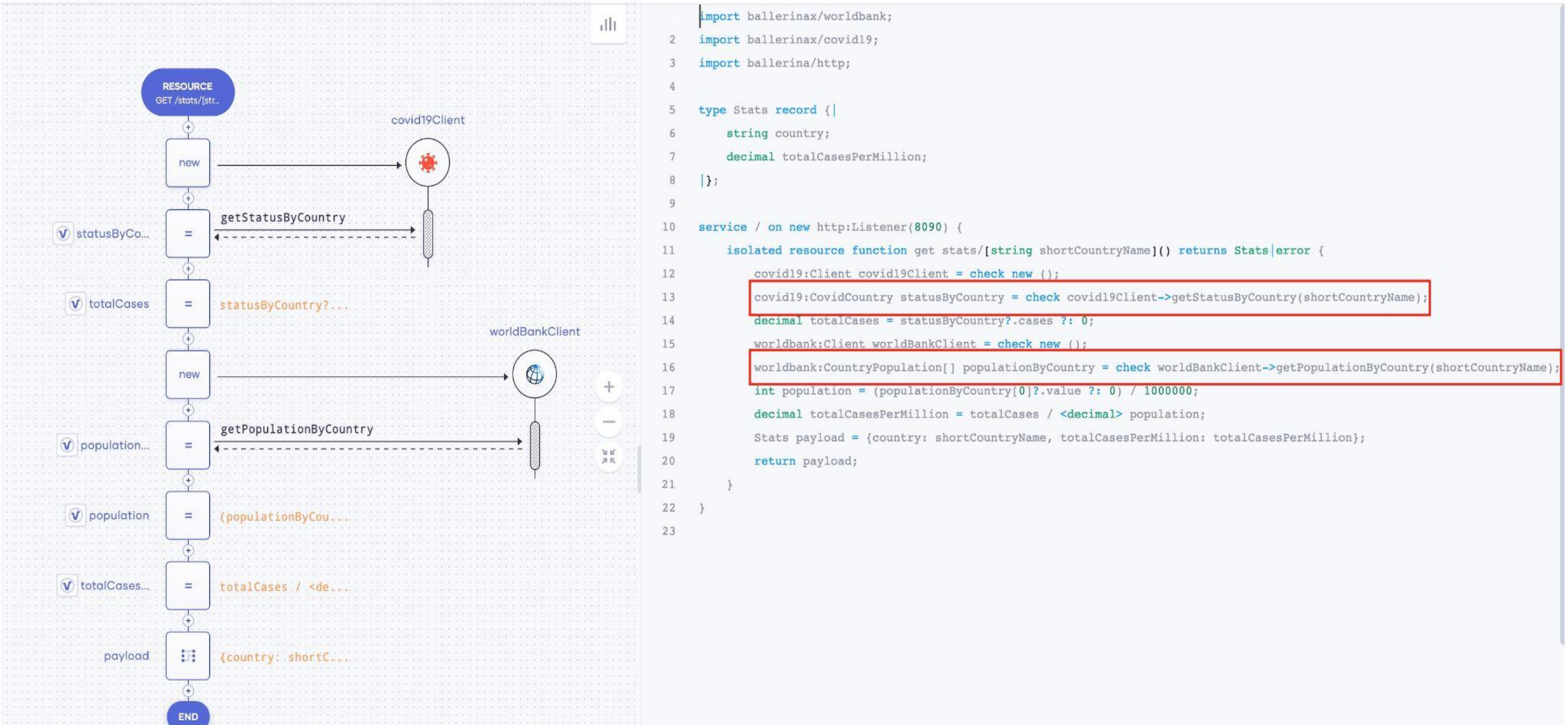
# API Programming

- API (Application Programming Interface) programming
  - APIs offer a simple way to programmatically interact with a separate software component or a resource
  - APIs hide (abstract) underlying implementation and only expose objects/actions the developer needs

- API Programming in the cloud
  - API Integration/automation: create an integration application using a set of APIs
    (e.g. Send an SMS notification to a specific user when a GitHub issue is assigned to the user)
  - API Composition: create a new API/Service using existing APIs that other applications can consume

# API Programming (contd.)

## Provide COVID-19 cases per million people in a specific country

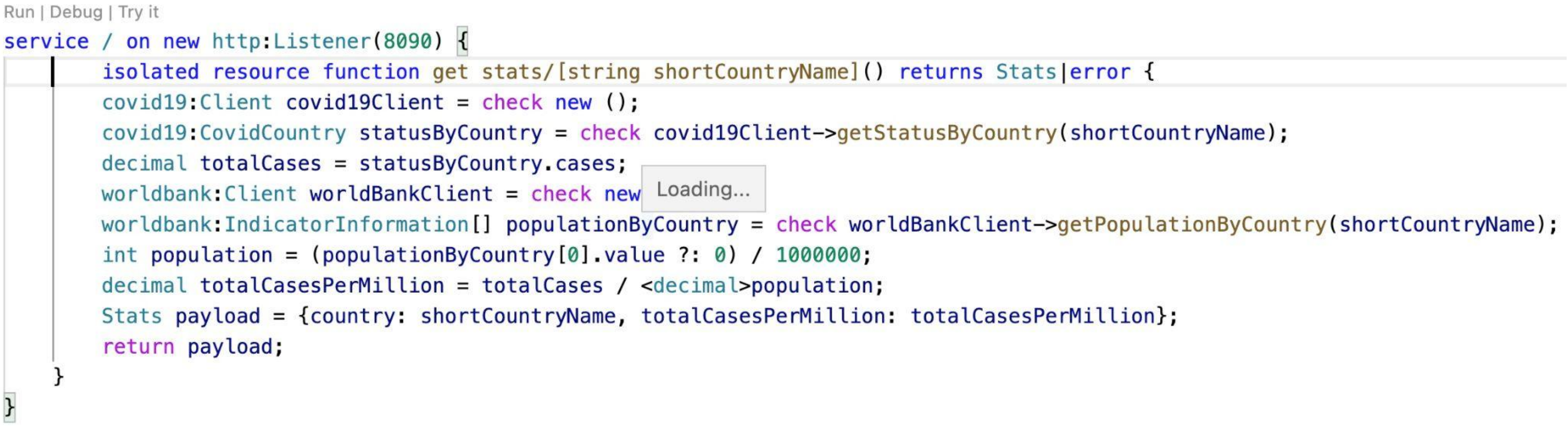


```
import ballerinax/worldbank;
import ballerinax/covid19;
import ballerina/http;

type Stats record {|
    string country;
    decimal totalCasesPerMillion;
|};

service / on new http:Listener(8090) {
    isolated resource function get stats/[string shortCountryName]() returns Stats|error {
        covid19:Client covid19Client = check new ();
        covid19:CovidCountry statusByCountry = check covid19Client->getStatusByCountry(shortCountryName);
        decimal totalCases = statusByCountry?.cases ?: 0;
        worldbank:Client worldBankClient = check new ();
        worldbank:CountryPopulation[] populationByCountry = check worldBankClient->getPopulationByCountry(shortCountryName);
        int population = (populationByCountry[0]?.value ?: 0) / 1000000;
        decimal totalCasesPerMillion = totalCases / <decimal> population;
        Stats payload = {country: shortCountryName, totalCasesPerMillion: totalCasesPerMillion};
        return payload;
    }
}
```

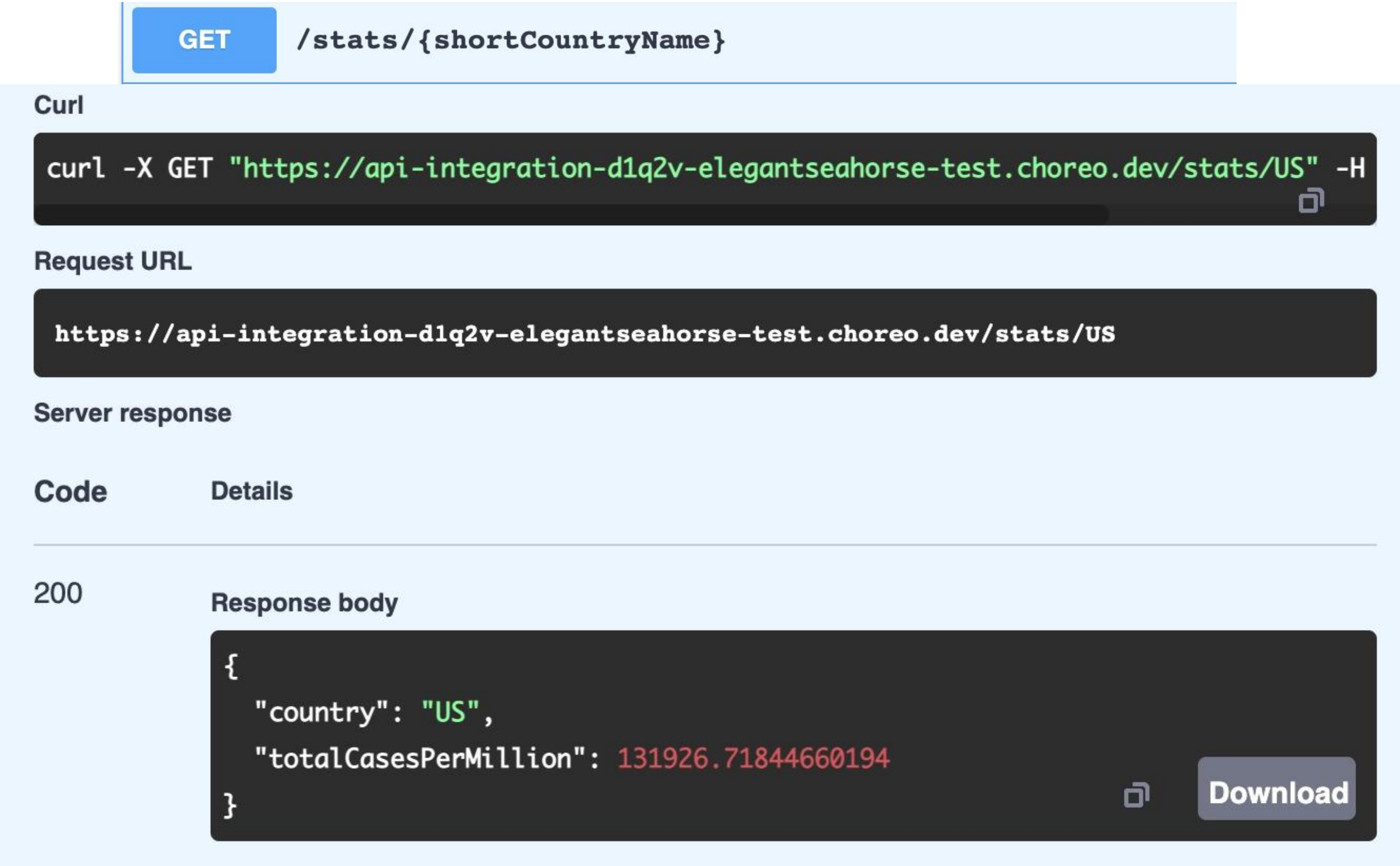
# API Programming (contd.)



Run | Debug | Try it

```
service / on new http:Listener(8090) {
    isolated resource function get stats/[string shortCountryName]() returns Stats|error {
        covid19:Client covid19Client = check new ();
        covid19:CovidCountry statusByCountry = check covid19Client->getStatusByCountry(shortCountryName);
        decimal totalCases = statusByCountry.cases;
        worldbank:Client worldBankClient = check new Loading...
        worldbank:IndicatorInformation[] populationByCountry = check worldBankClient->getPopulationByCountry(shortCountryName);
        int population = (populationByCountry[0].value ?: 0) / 1000000;
        decimal totalCasesPerMillion = totalCases / <decimal>population;
        Stats payload = {country: shortCountryName, totalCasesPerMillion: totalCasesPerMillion};
        return payload;
    }
}
```

# API Programming (contd.)



GET /stats/{shortCountryName}

Curl

```
curl -X GET "https://api-integration-d1q2v-elegantseahorse-test.choreo.dev/stats/US" -H
```

Request URL

```
https://api-integration-d1q2v-elegantseahorse-test.choreo.dev/stats/US
```

Server response

| Code | Details |
|---|---|
| 200 | Response body |

```
{
  "country": "US",
  "totalCasesPerMillion": 131926.71844660194
}
```

# Performance Characteristics of API Programs

- Performance is one of the most important non-functional requirements
    - Impacts the user experience
    - Poor performance can cause customer dissatisfaction (can lead to customer churn)
- API programs
    - Handle a large volume of requests
    - Performance depends on
        - Workload characteristics (e.g. concurrency, message sizes)
        - Program characteristics
            - Network calls (connector calls/actions)

# Why Provide Performance Estimates?

- Can use estimates to check if SLAs are met
- Minimize performance related bugs in code
- Understand performance/scalability behaviours
- Save developer’s time (minimizes the number of performance tests)
- Save resources (minimize the cost of running performance tests)

# Performance metrics

- **Throughput**

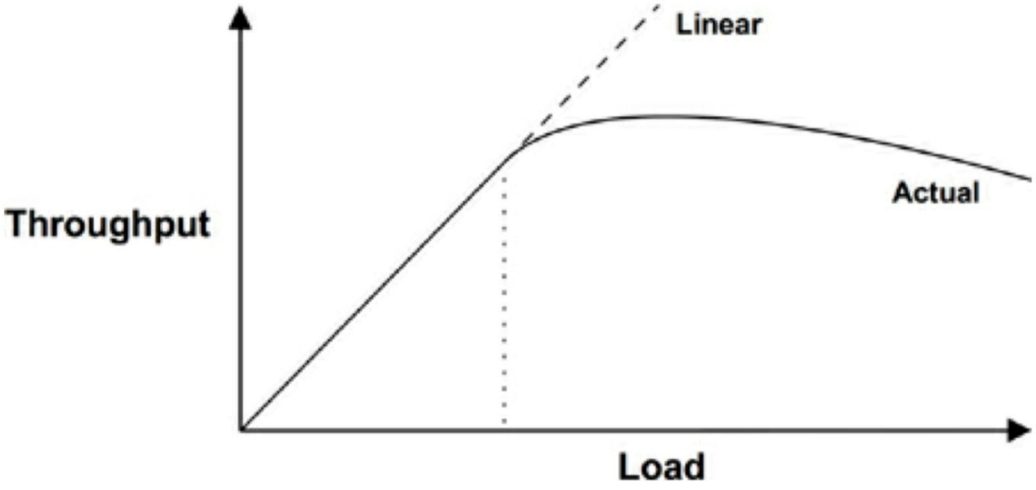
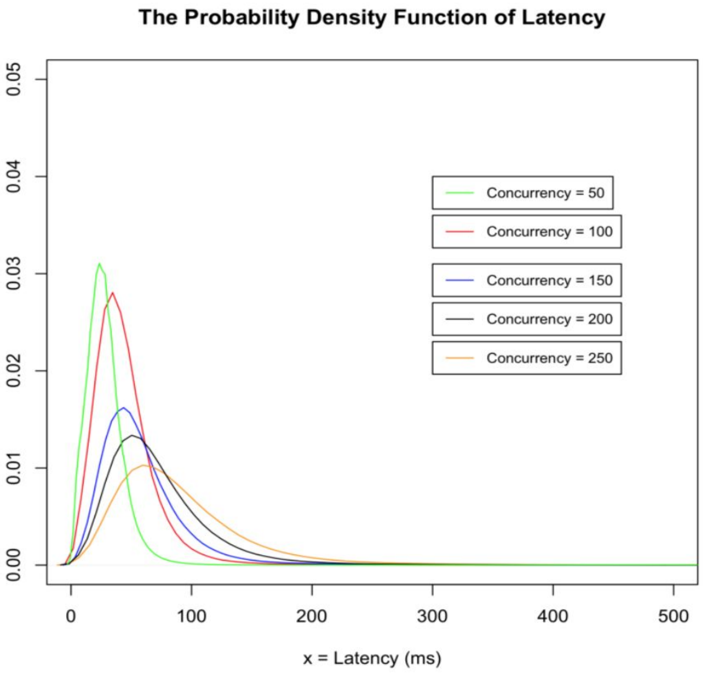


**Figure 2.6** Throughput versus load

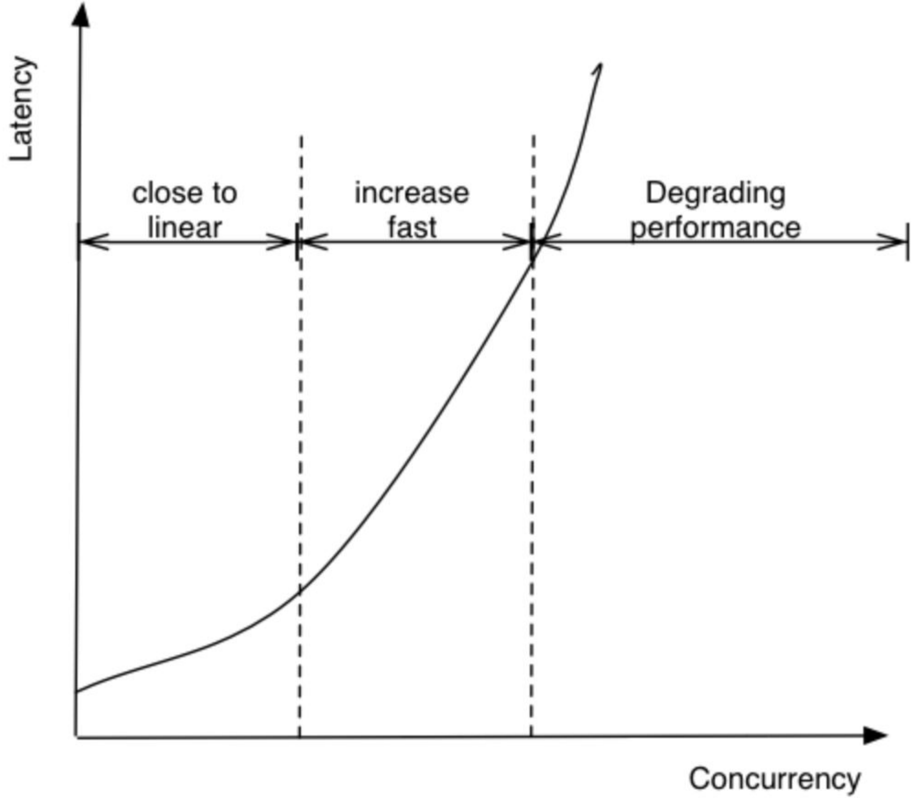
# Performance metrics

- Latency: Latency is a measure of time an operation spends waiting to be serviced (e.g. response time)

- Latency
  - Average latency
  - latency percentile (e.g. 90%, 99%, 99.99%)

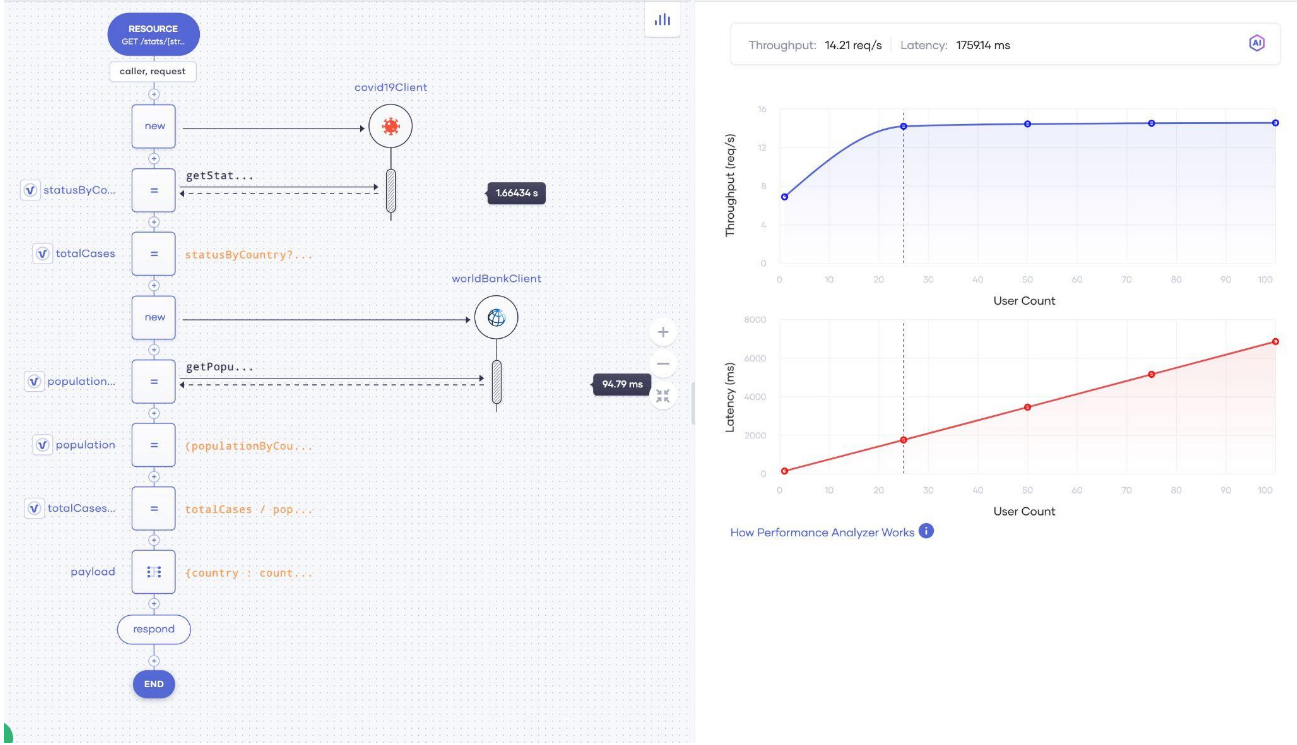


# Performance metrics



# Estimating Performance of API programs

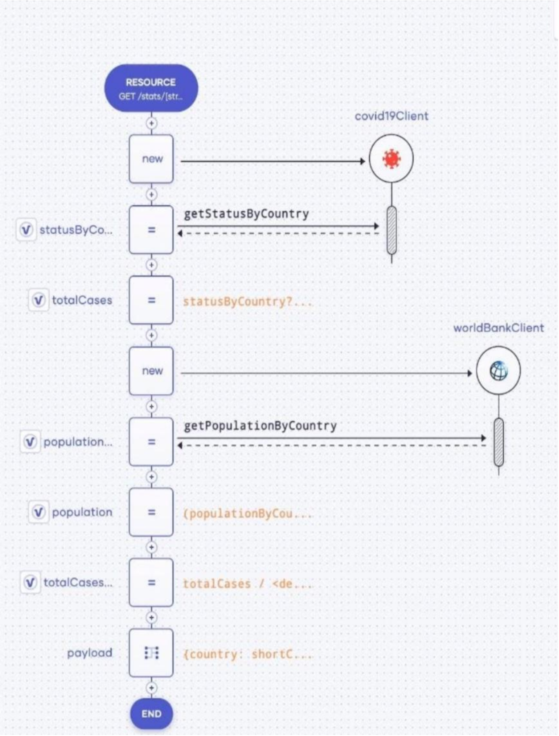
- Objective: Provide performance estimates at development time

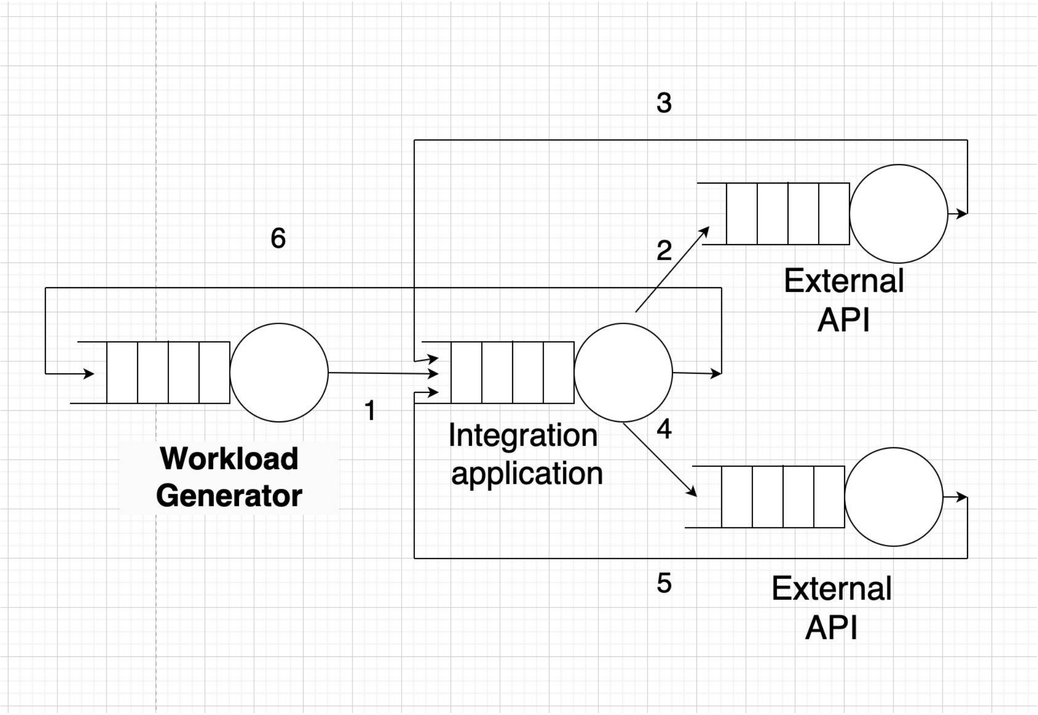


Throughput: 14.21 req/s | Latency: 1759.14 ms

# System Model

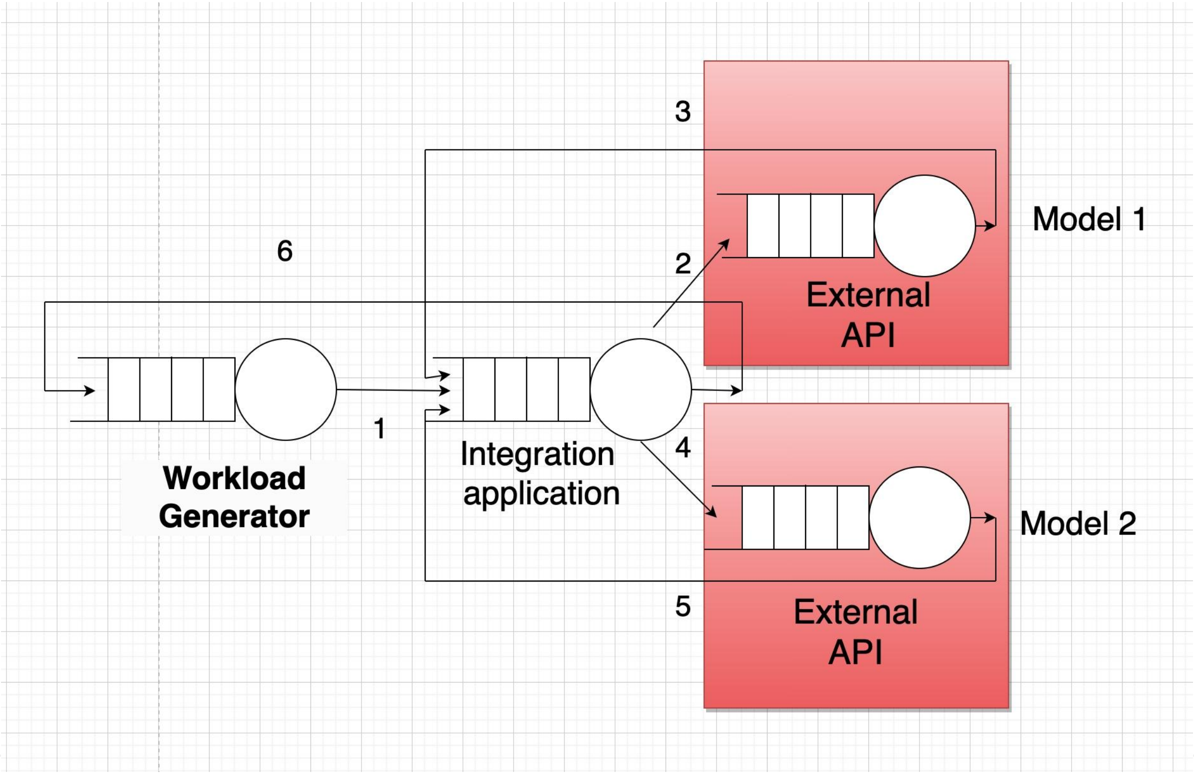
- Model program/service as a queuing network/system





# Model Training: Modelling Connector Actions (Calls)

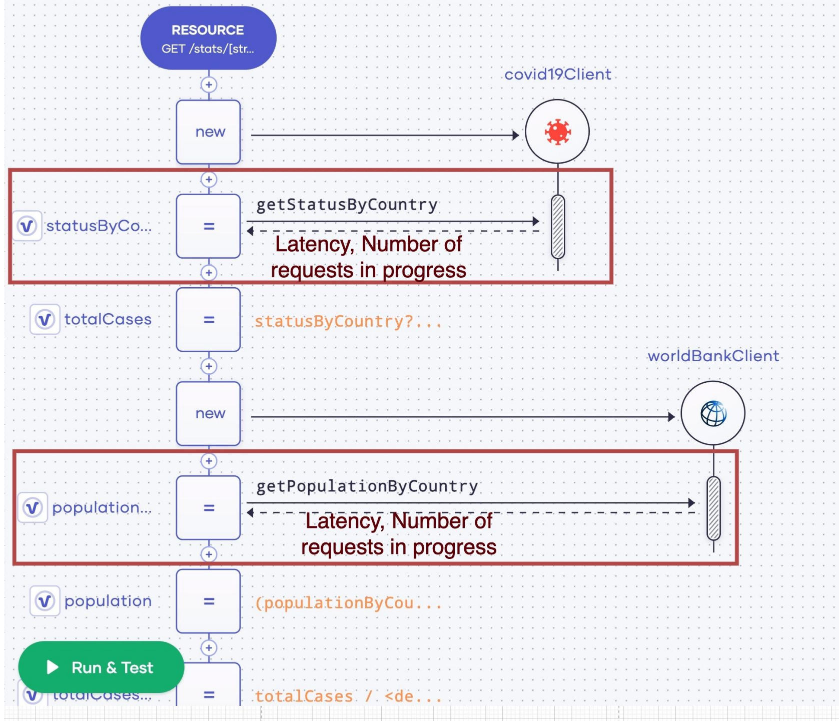
- Train a model for each connector action/call **(using historical data)**



# Model Training

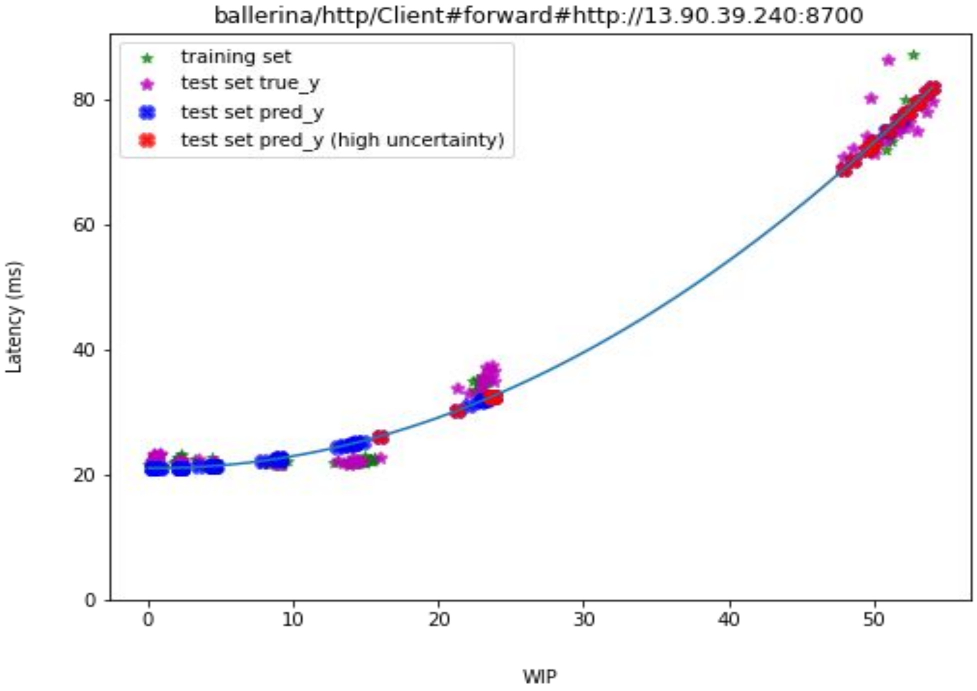
- Model will estimate the latency & throughput for a given concurrency (i.e. concurrent users)
- Data
  - Number of requests in progress (work in progress)
  - Latency of requests

- Observability platform collects the above performance metrics from running applications and stores this data in cloud storage

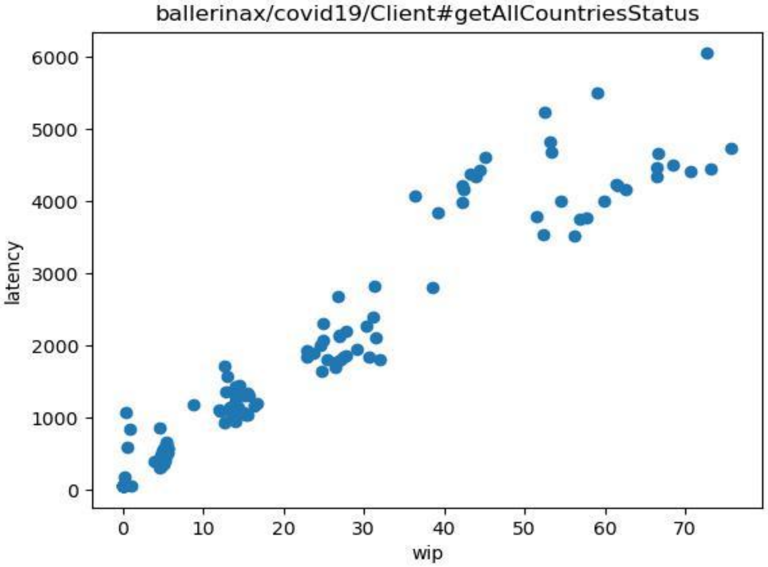


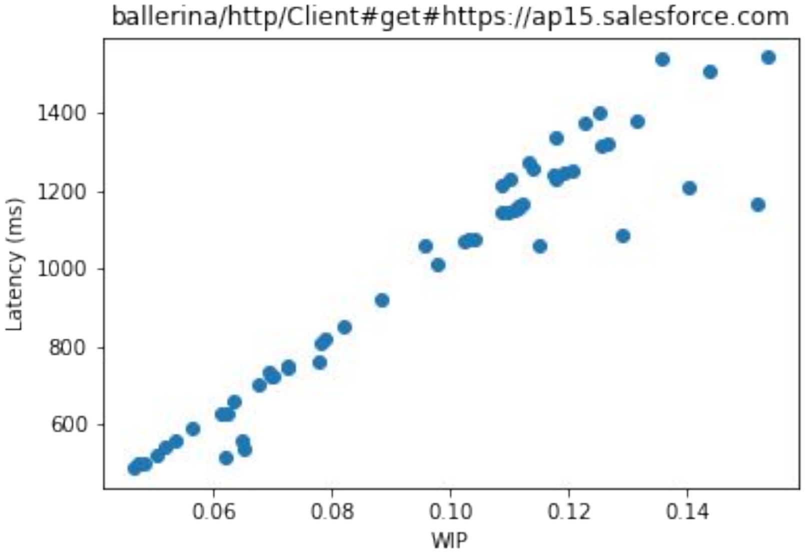
# Model Training

- Models: Analytical models (e.g. USL), Machine learning models

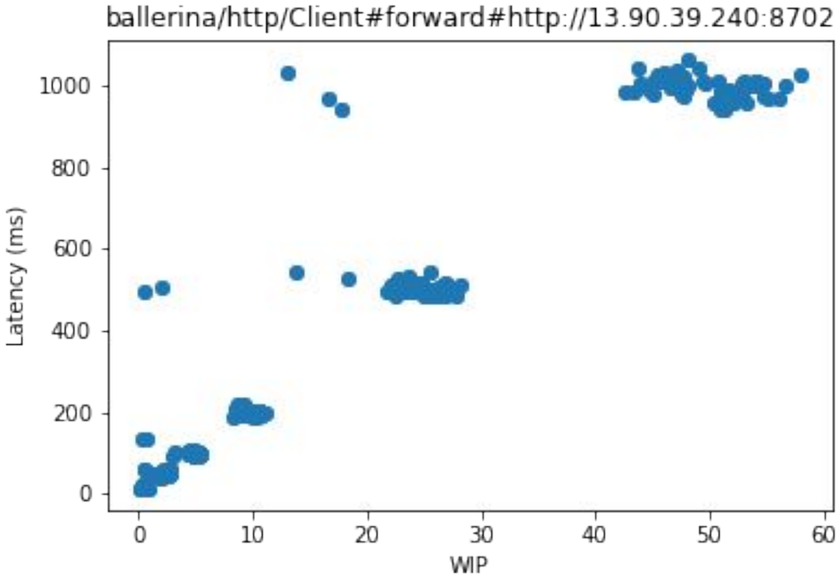


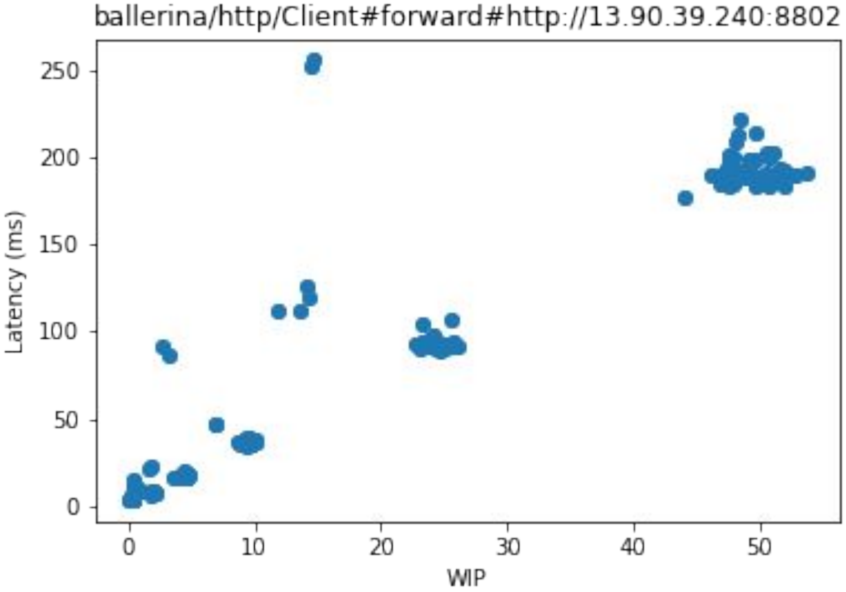
# Pattern 1


ballerinax/covid19/Client#getAllCountriesStatus
latency
0
1000
2000
3000
4000
5000
6000
wip
0
10
20
30
40
50
60
70

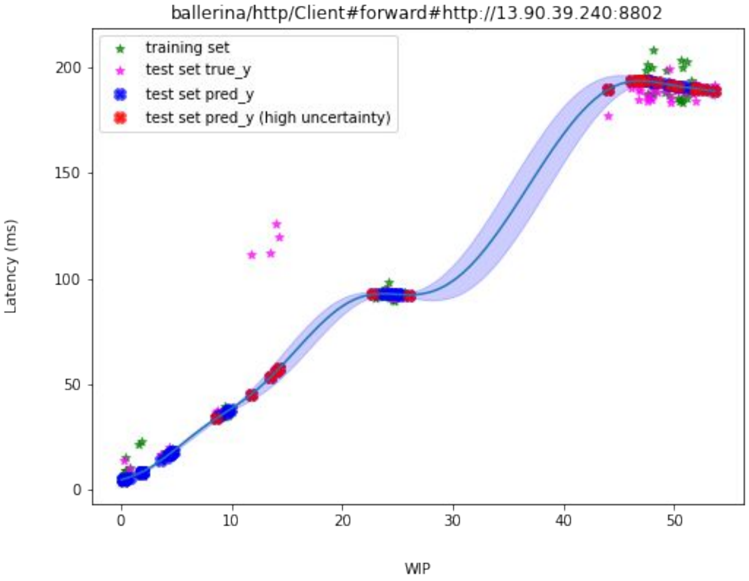

ballerina/http/Client#get#https://ap15.salesforce.com
Latency (ms)
600
800
1000
1200
1400
WIP
0.06
0.08
0.10
0.12
0.14

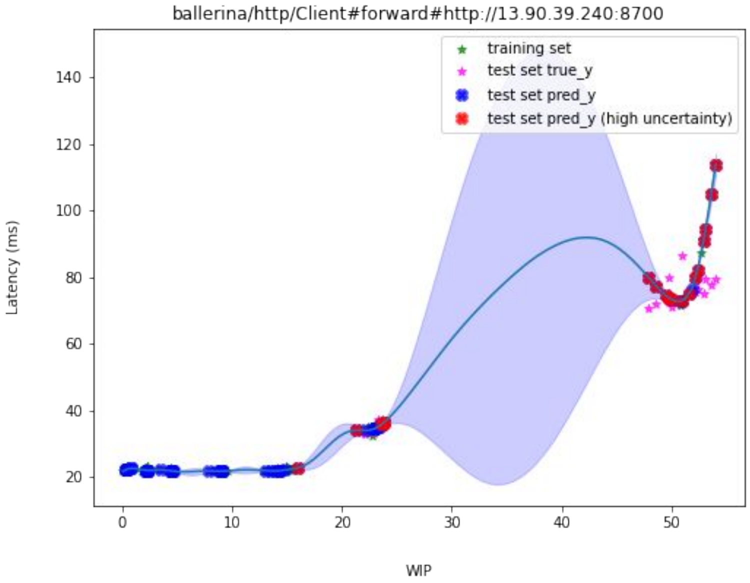
# Pattern 2 (missing data in certain regions)



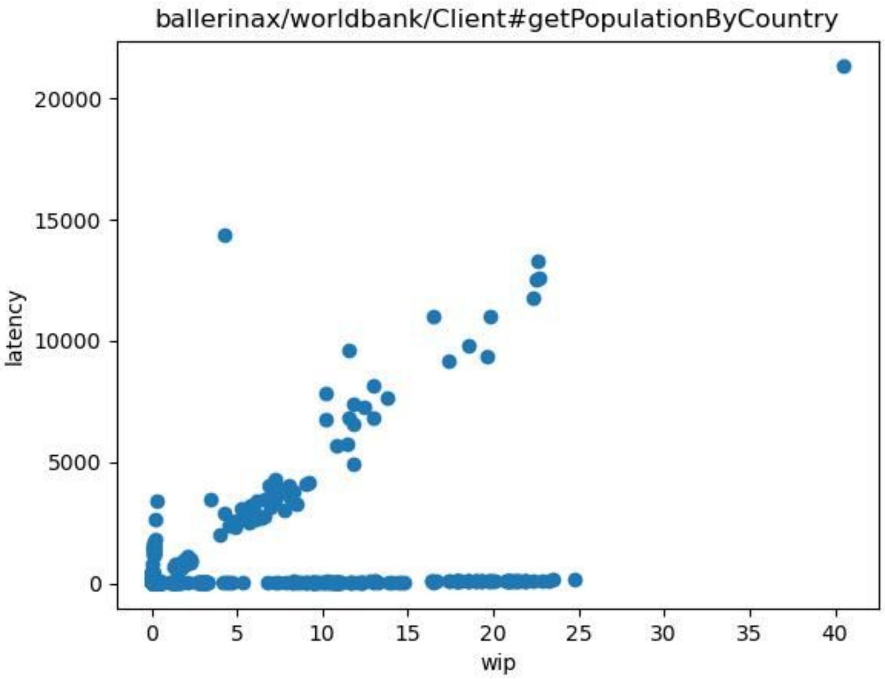


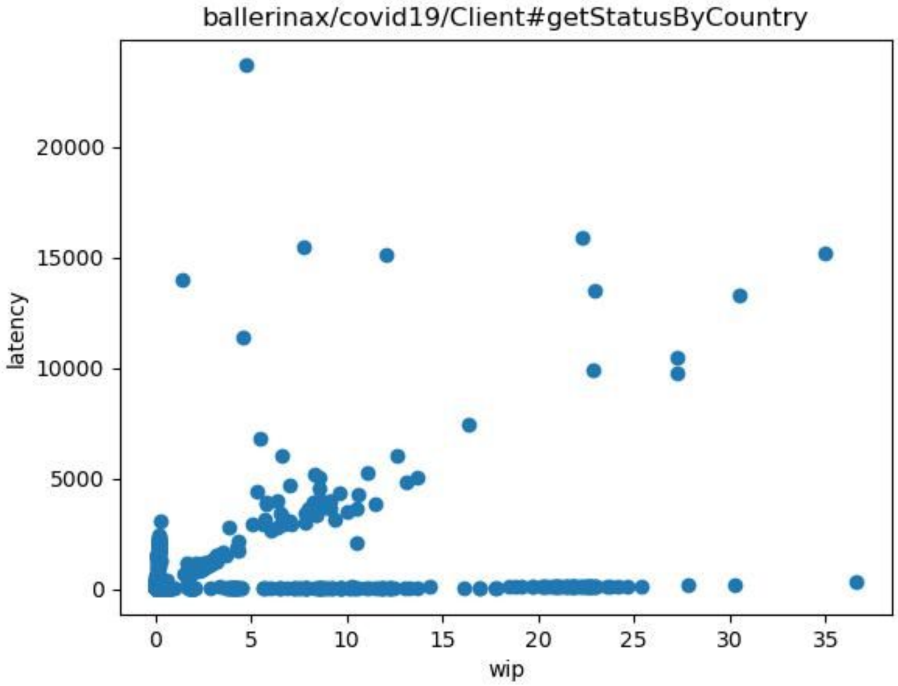
# Bayesian Fit



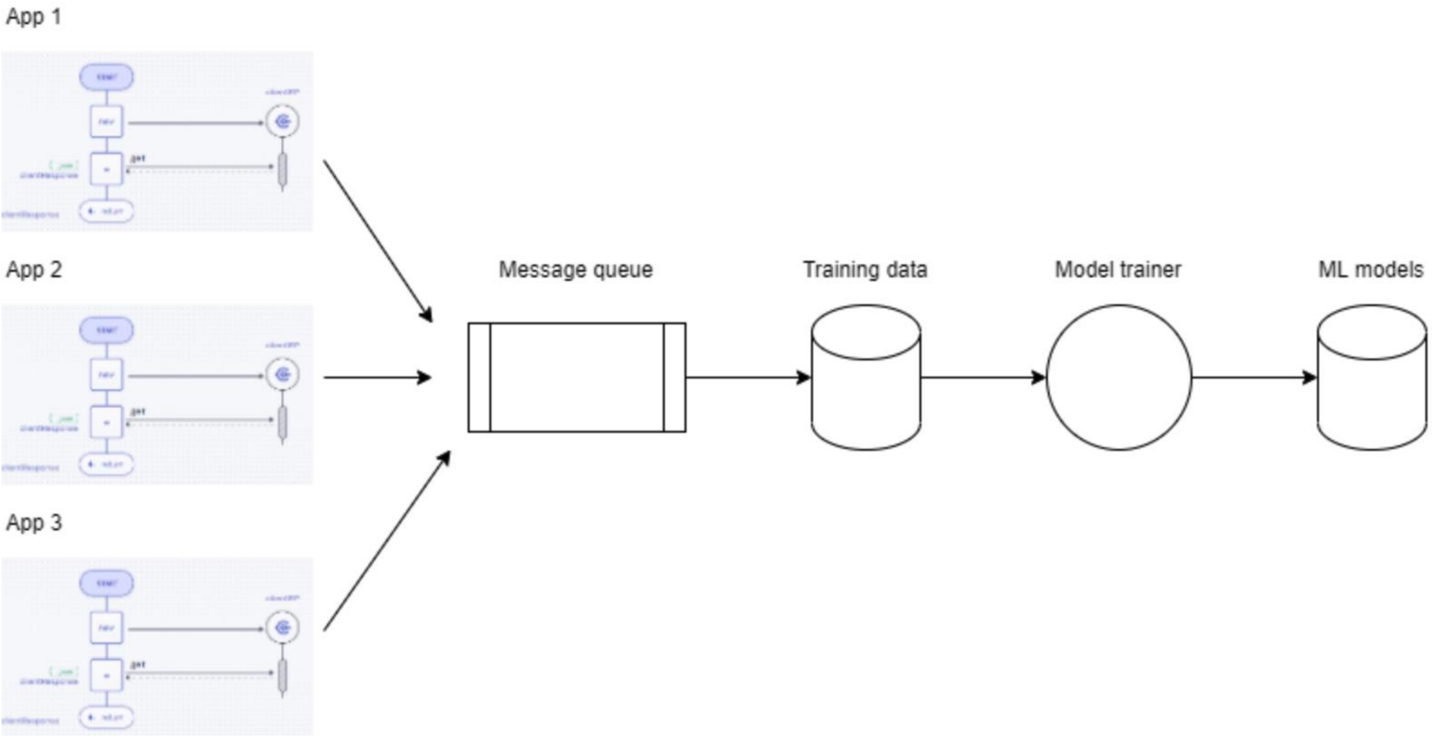


# Pattern 3 (Caching)

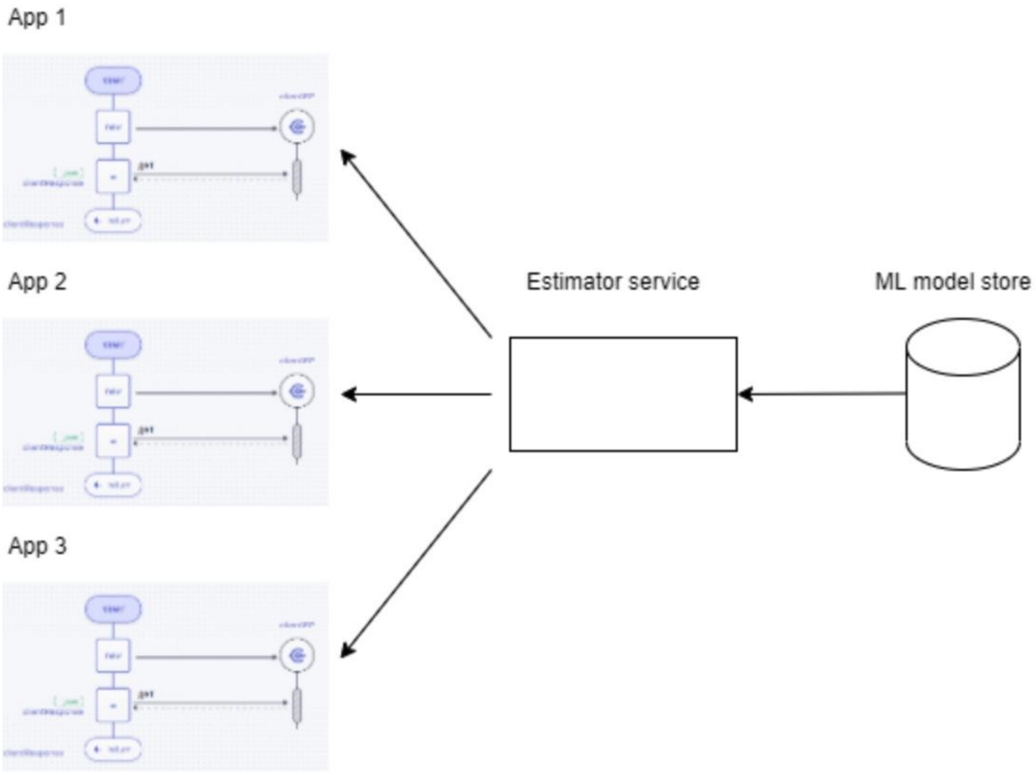




# Data Collection & Model Training



# Estimating performance



App 1

App 2

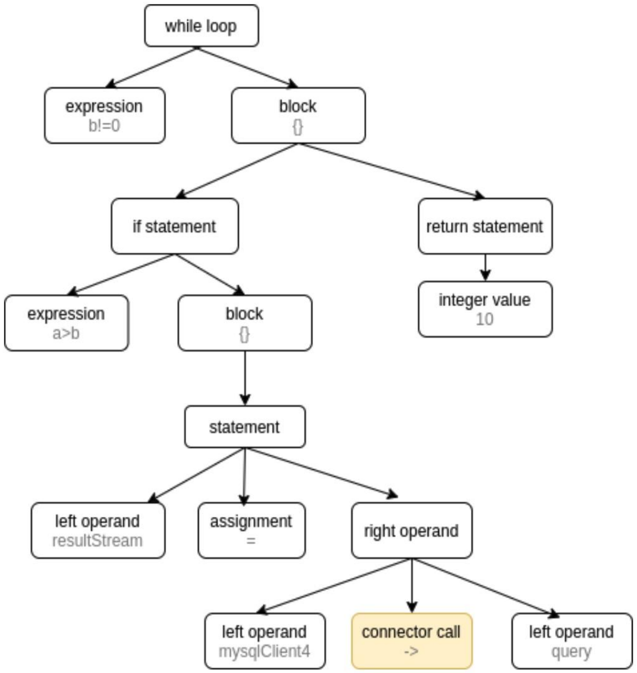
App 3

Estimator service

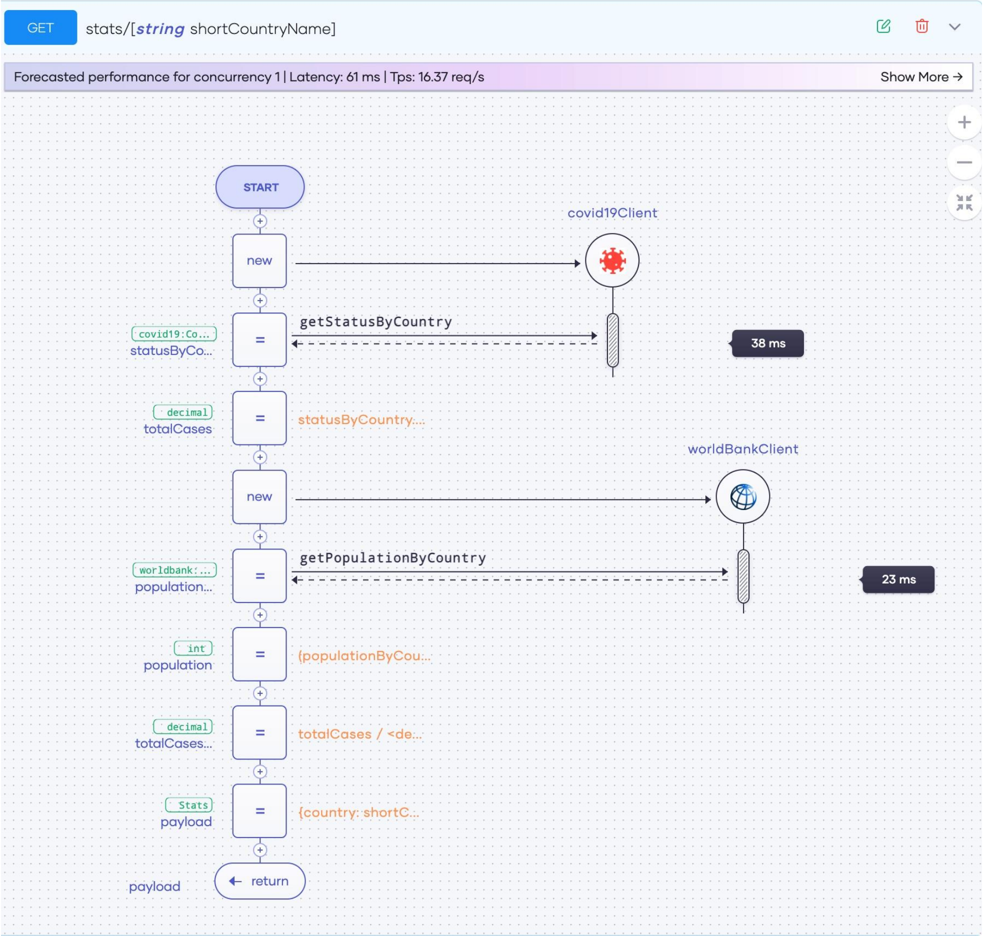
ML model store

# Providing performance estimates (at development time)

- Analyze the integration program and extract connector calls, loops, etc. (can parse the AST)
- Construct a message with this data and send it to the estimator serviced
- Estimator service computes the TPS & Latency (under different number of concurrent users) using an analytical model. This model is based on individual models (described in previous slides)
- Show the estimates to the user on Web UI or on IDE
- Each time user changes the program, performance estimates are updated



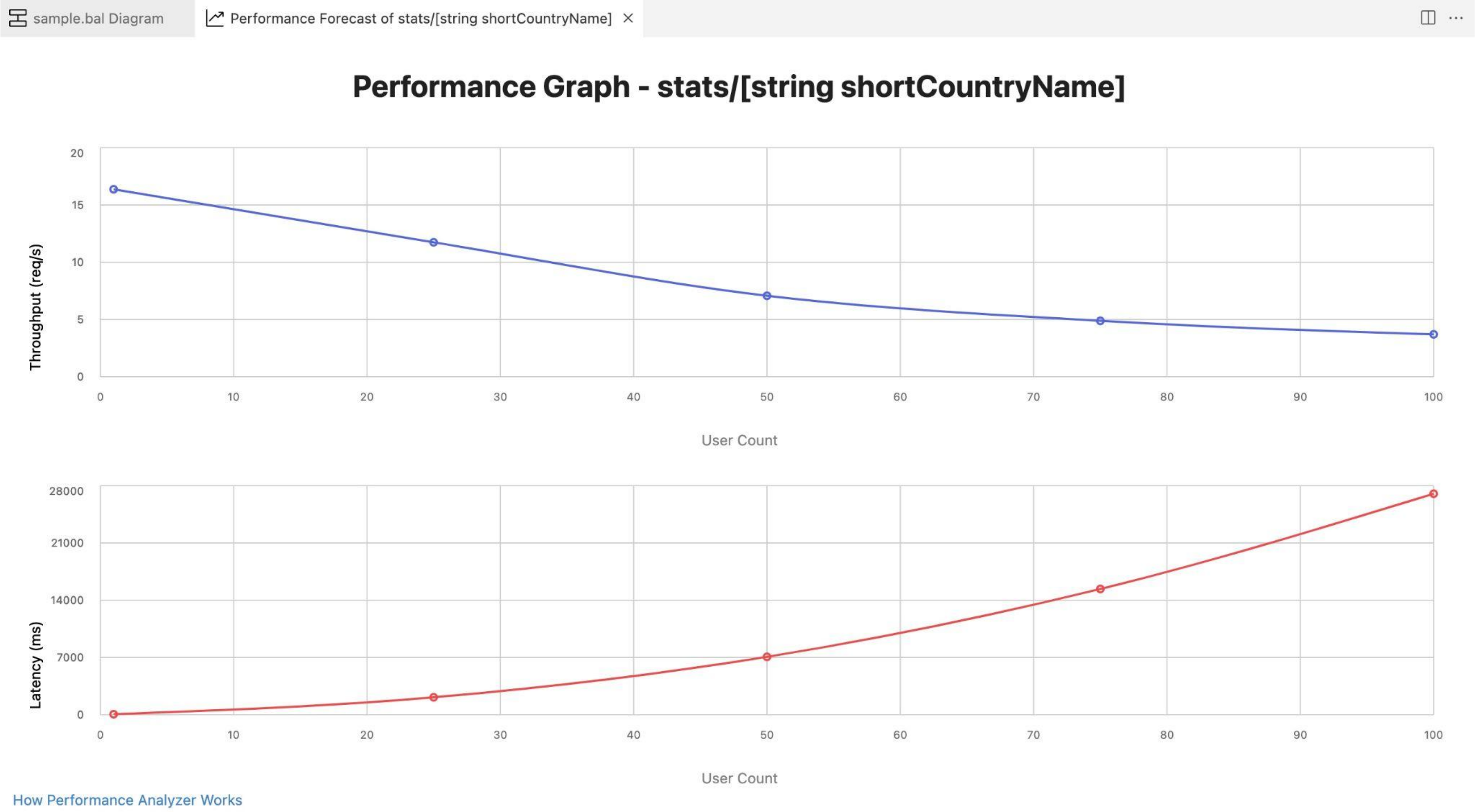
# DEMO



GET stats/[*string* shortCountryName]

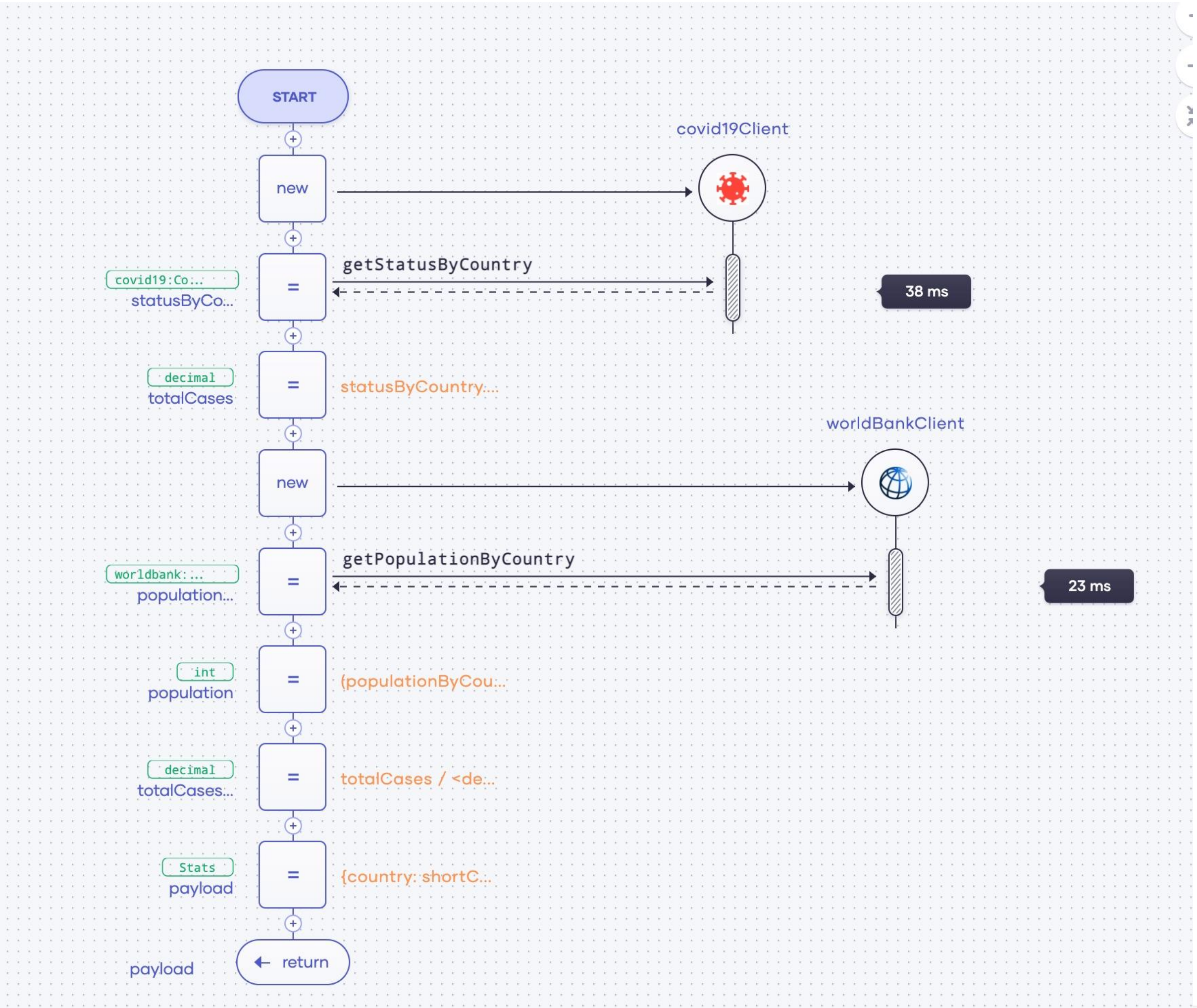
Forecasted performance for concurrency 1 | Latency: 61 ms | Tps: 16.37 req/s

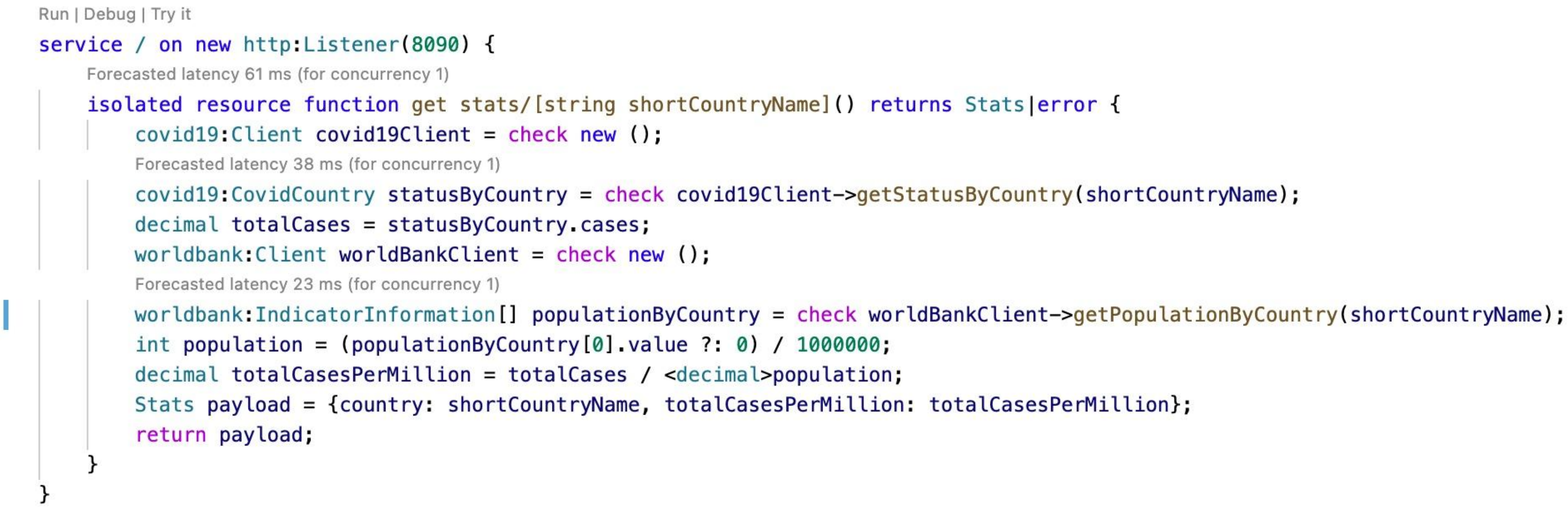
Show More →



# Performance Graph - stats/[string shortCountryName]

How Performance Analyzer Works


START
new
covid19Client
covid19:Co...
statusByCo...
=
getStatusByCountry
38 ms
decimal
totalCases
=
statusByCountry....
worldBankClient
new
worldbank:...
population...
=
getPopulationByCountry
23 ms
int
population
=
{populationByCou...
decimal
totalCases...
=
totalCases / <de...
Stats
payload
=
{country: shortC...
payload
return

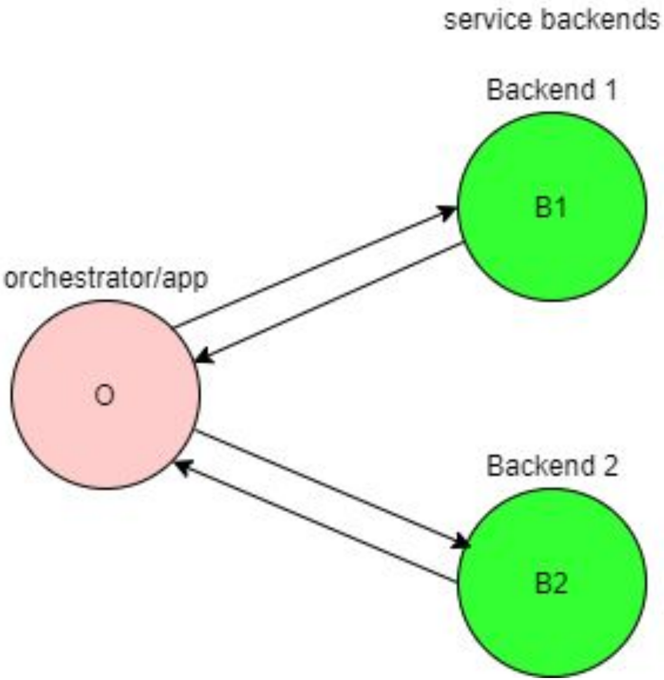


```
Run | Debug | Try it
service / on new http:Listener(8090) {
    Forecasted latency 61 ms (for concurrency 1)
    isolated resource function get stats/[string shortCountryName]() returns Stats|error {
        covid19:Client covid19Client = check new ();
        Forecasted latency 38 ms (for concurrency 1)
        covid19:CovidCountry statusByCountry = check covid19Client->getStatusByCountry(shortCountryName);
        decimal totalCases = statusByCountry.cases;
        worldbank:Client worldBankClient = check new ();
        Forecasted latency 23 ms (for concurrency 1)
        worldbank:IndicatorInformation[] populationByCountry = check worldBankClient->getPopulationByCountry(shortCountryName);
        int population = (populationByCountry[0].value ?: 0) / 1000000;
        decimal totalCasesPerMillion = totalCases / <decimal>population;
        Stats payload = {country: shortCountryName, totalCasesPerMillion: totalCasesPerMillion};
        return payload;
    }
}
```

# Evaluating accuracy of estimates

- Two methods
  - Discrete event simulation (simulator code)
    - Build a simulation model
    - Compare simulation results with model results
  - System level testing
    - Run tests and populate data
    - Compare the results with the model results

# Simulation vs Model

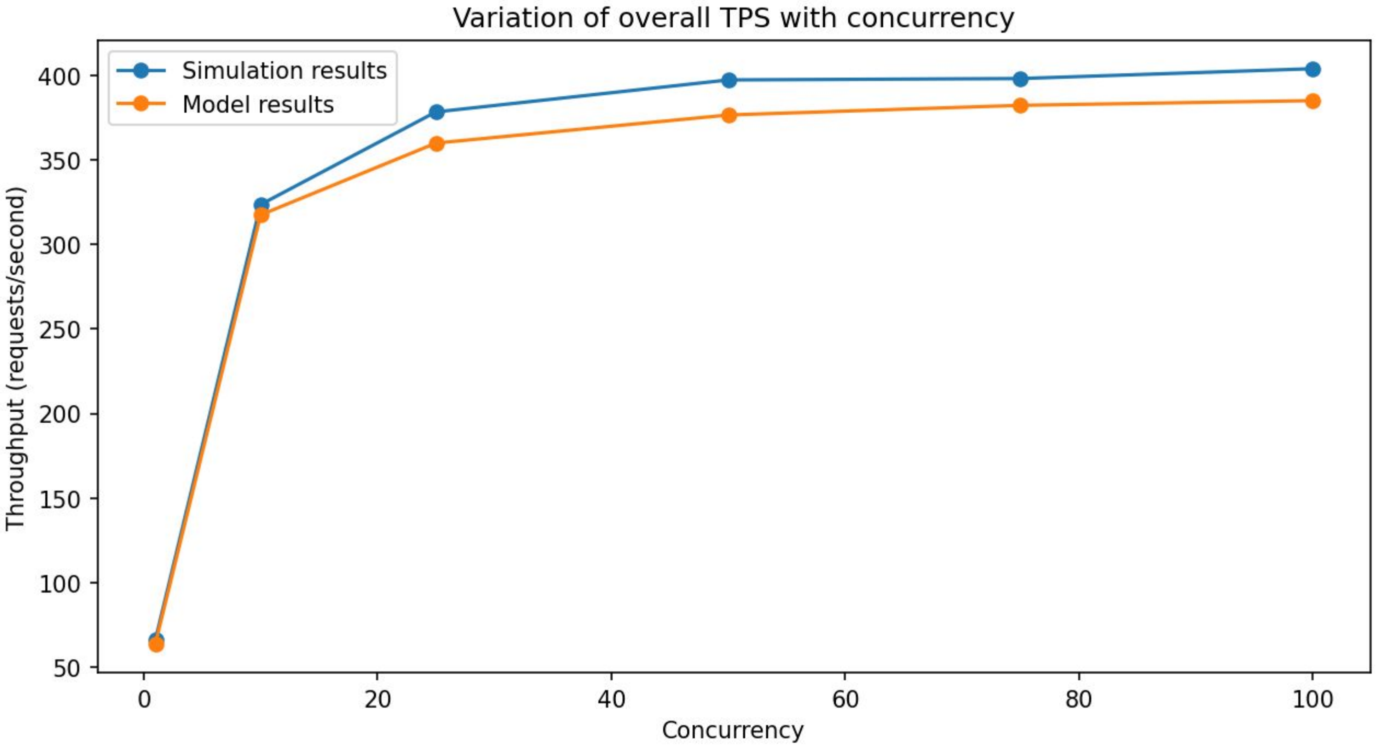


# Simulation vs Model

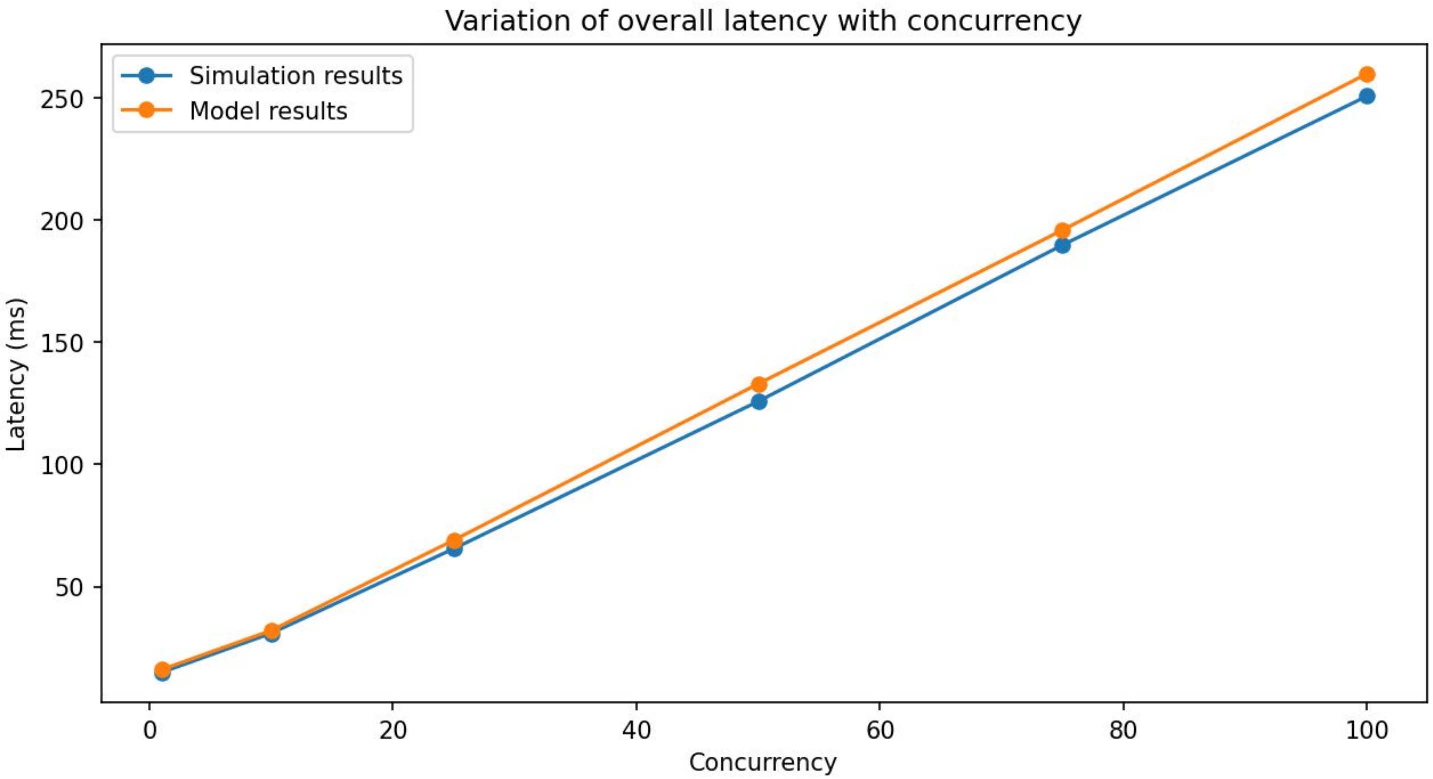
- Orchestrator with 2 (back-end) services

| | Average processing time (ms) | Number of Cores | Thread pool size |
|---|---|---|---|
| Service 1 | 5 | 2 | 10 |
| Service 2 | 10 | 4 | 10 |

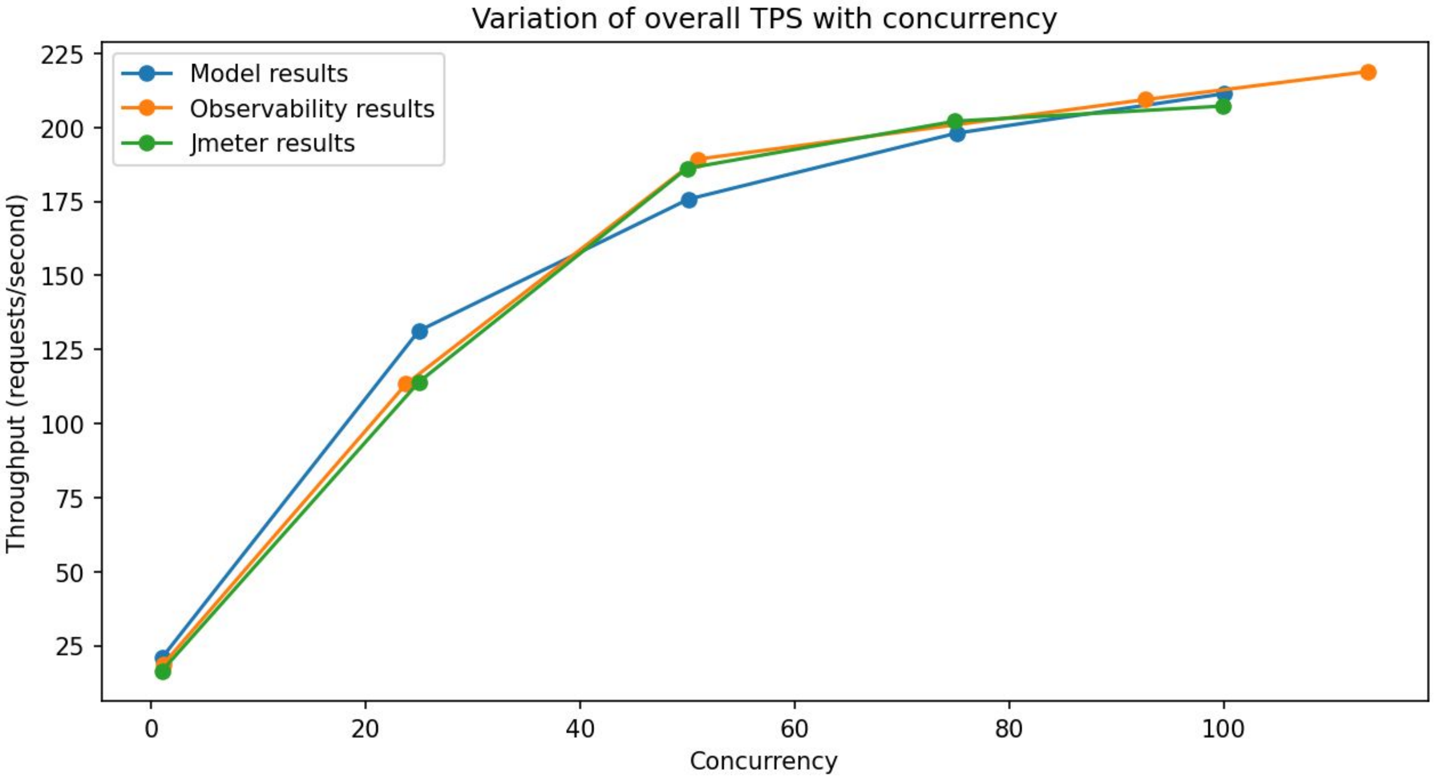
# Simulation vs Model



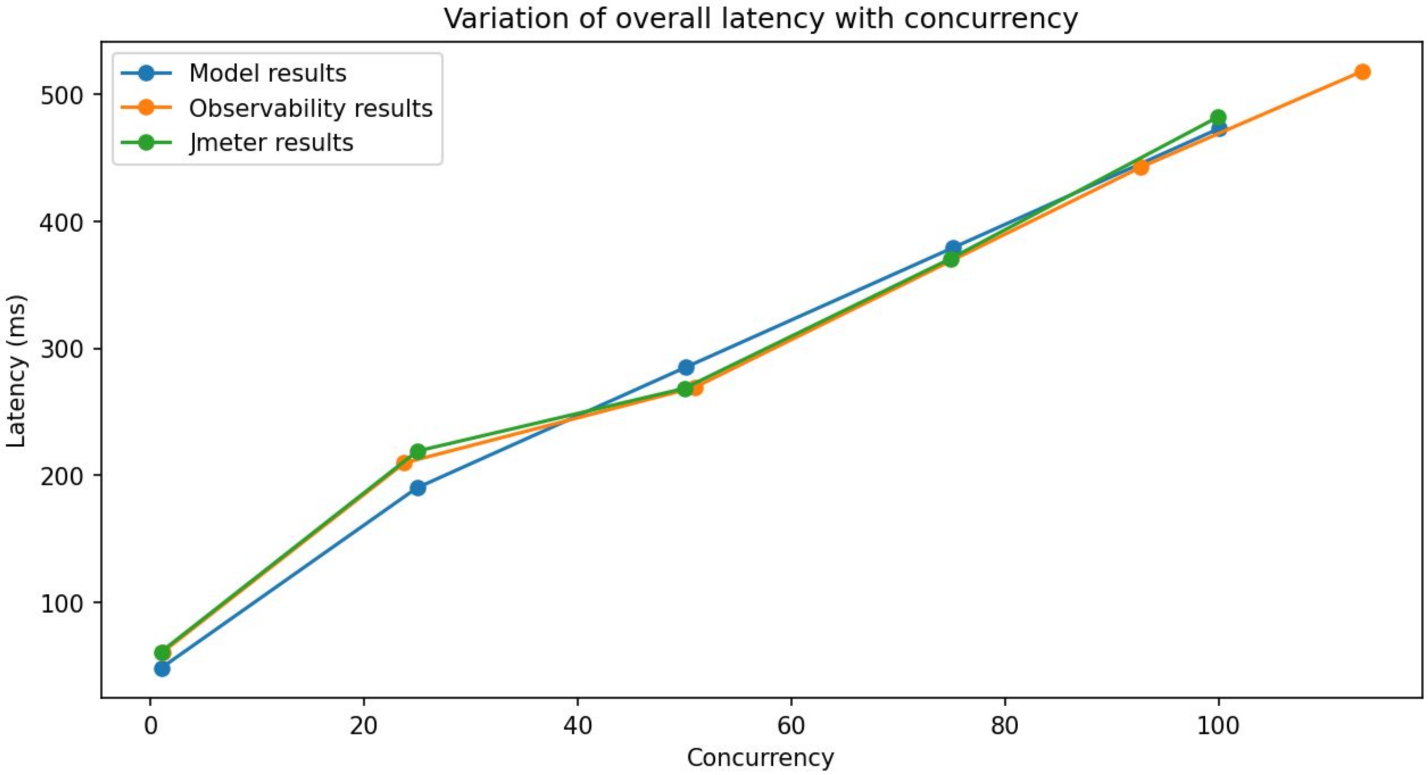
# Simulation vs Model



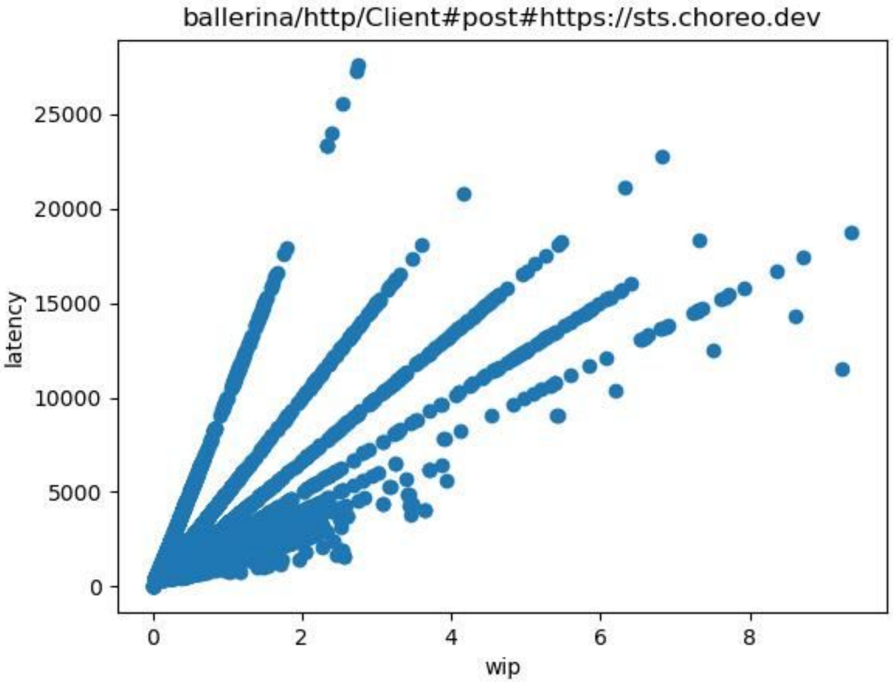
# Comparison with actual (observability) results



# Comparison with actual (observability) results



# Pattern 4



ballerina/http/Client#post#https://sts.choreo.dev
latency
25000
20000
15000
10000
5000
0
0
2
4
6
8
wip

# Summary

- Presented a way to provide performance feedback for API programs (at development time)
- How does this help developers? Ensure SLAs, avoid performance bugs, understand scalability behaviours and minimize the number of performance tests
- Modeled API program as a queuing network/system
- Trained a model for each type of network interaction (i.e. connector action) using historical data (collected by the observability framework) and compute the overall performance using an analytical model

# THANK YOU