JustInTimeout: Enacting Dynamic Network Timeouts via Distributed Feedback

Yash Trivedi
Dylan Wilson
Nisarg Chokshi

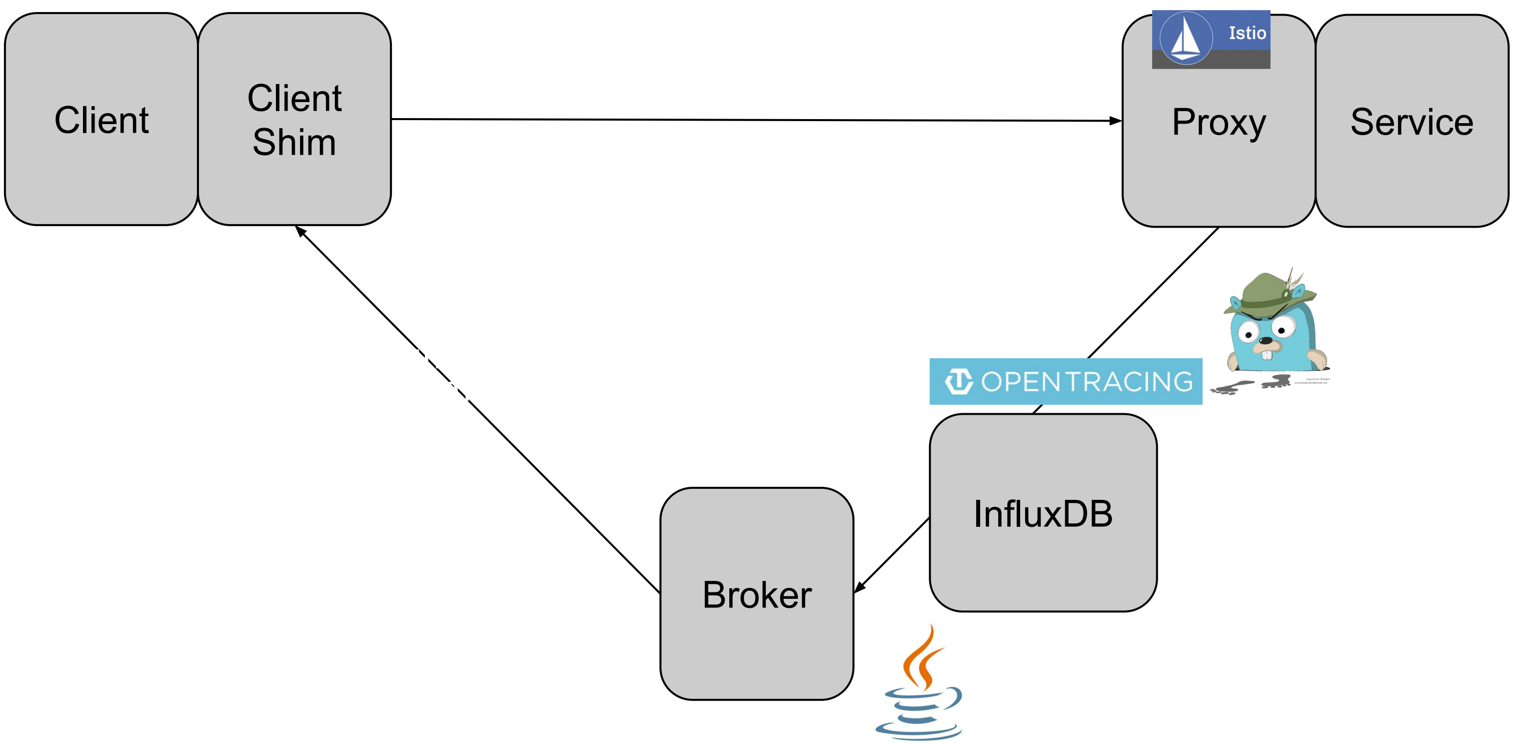
Agenda

- Big picture
  - Goals
  - Logical View
- System Overview
  - RNN Broker
  - Client Shim
  - GP Broker
- Results
- Future Directions

Goals

- Use network response data to predict near-term timeout values.
- Transparently inject appropriate timeout values into network calls.

JustInTimeout: Logical View



Recurrent Neural Network Broker

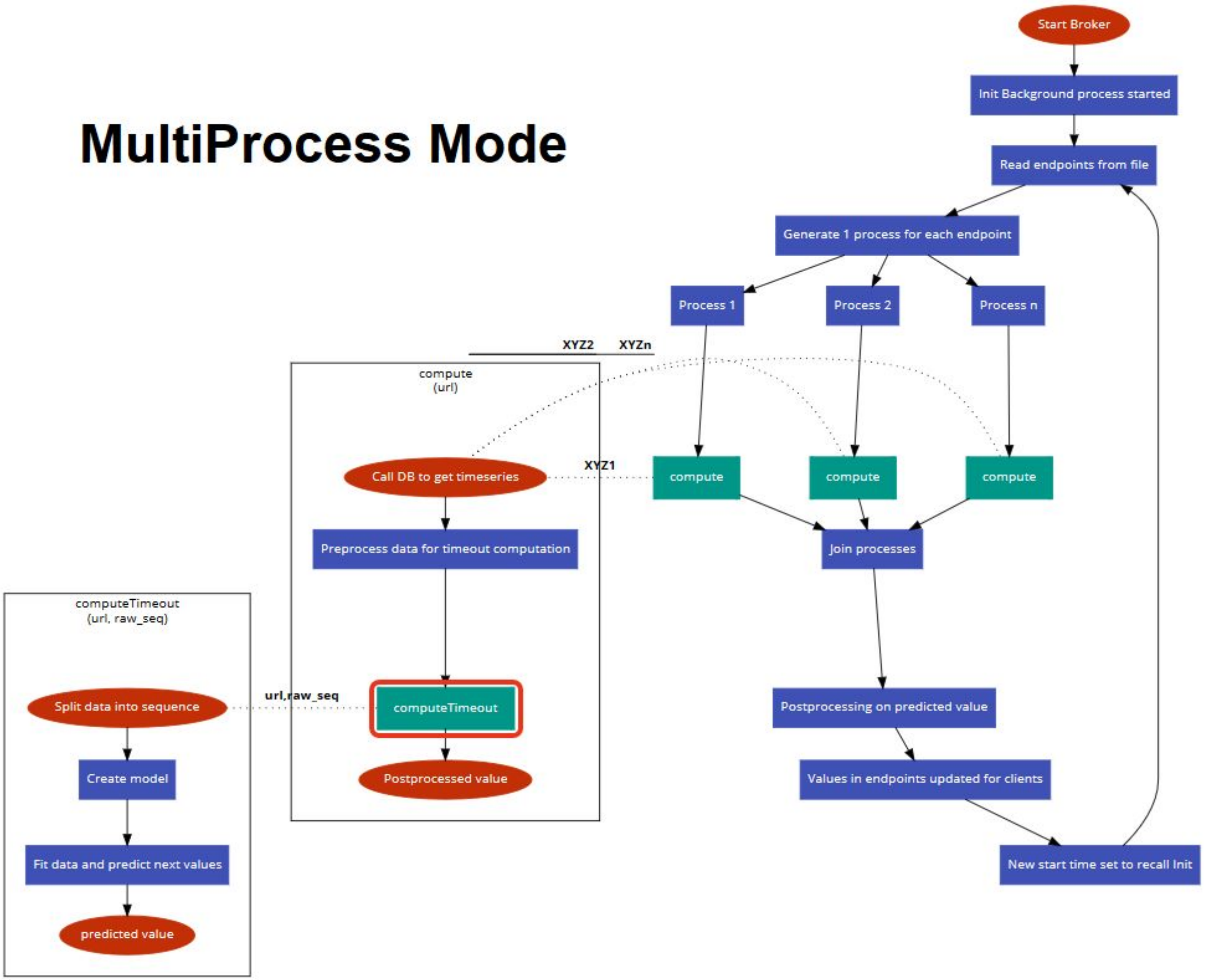
Why Recurrent Neural Network

- This problem can be simplified as a **time series prediction problem** where the previous inputs within a short window span will most likely be the best candidates to predict future value.
- **Convoluted LSTM** is optimal when you have no need for convolution operation in the input data all the while making use of CNNs along with LSTM for fast predictions.

MultiProcess vs MultiDocker

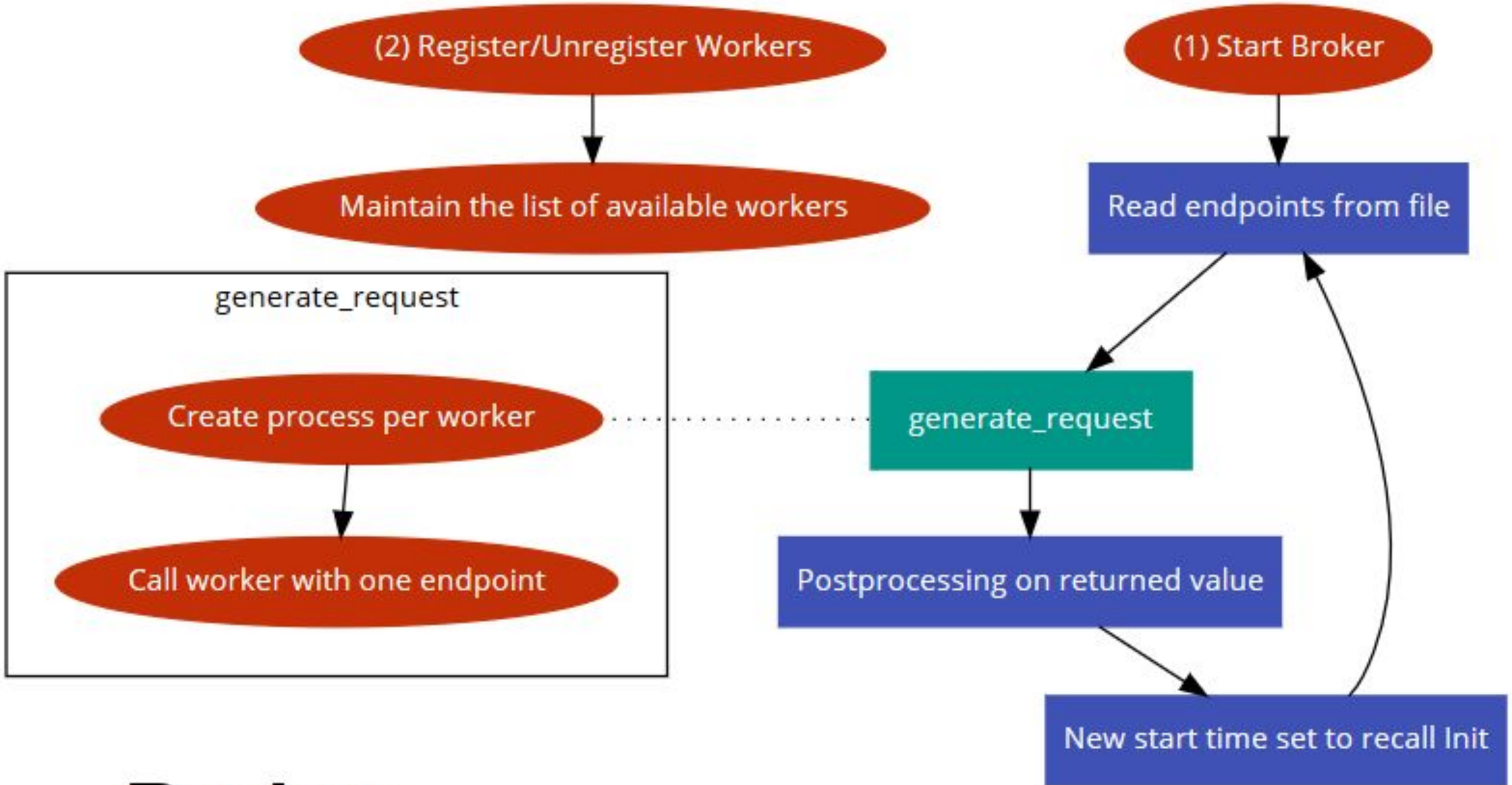
- In MultiProcess mode, there is just one server running which spawns off multiple processes which individually do the timeout computation for the url assigned to them.
- In the MultiDocker mode, this idea is taken to the extreme where there are workers pods who register themselves with the running broker and get tasks.

MultiProcess Mode

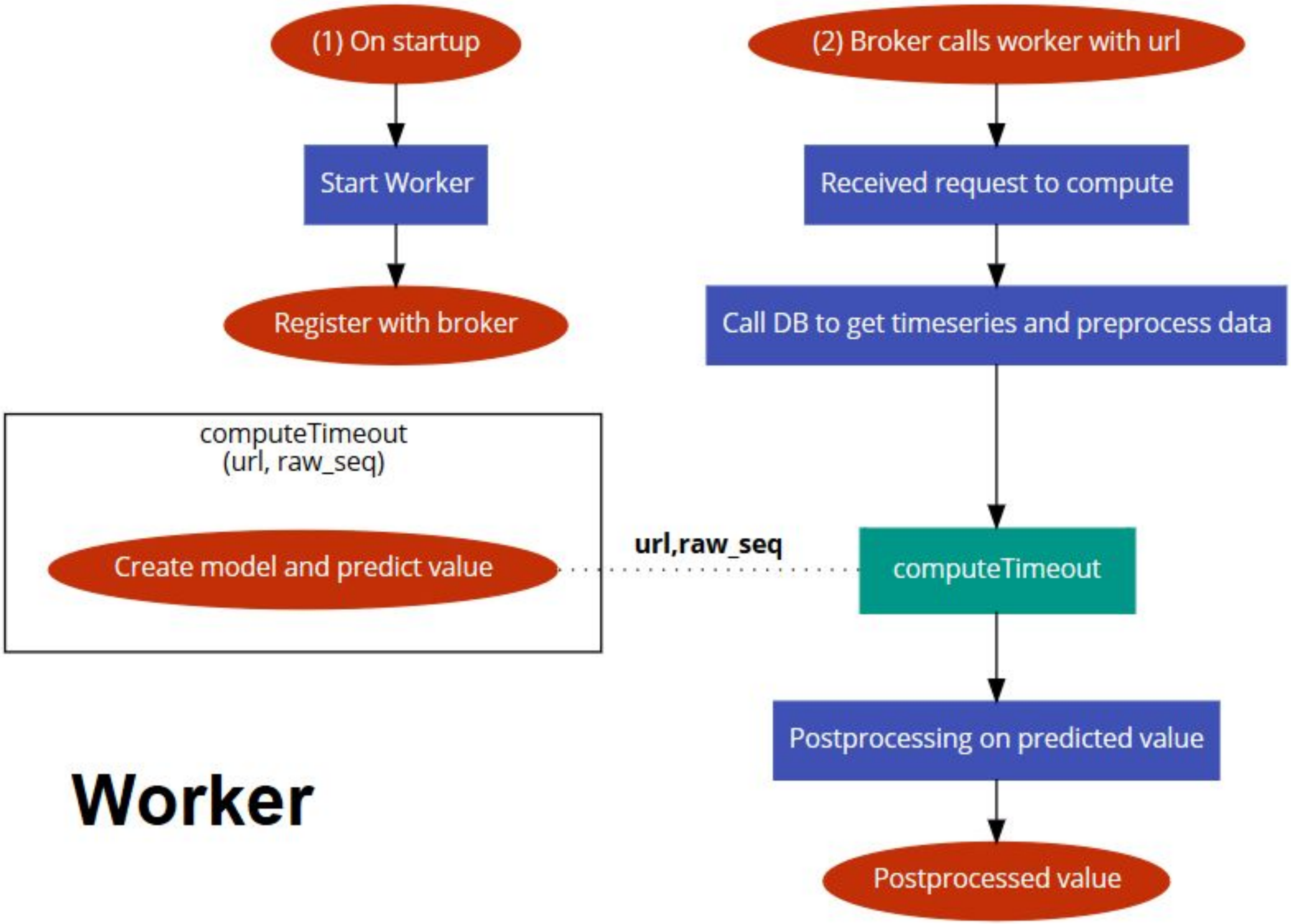


Enhancements in MultiDocker

- This approach requires workers to register with the broker.
- Two new url calls which workers can use to register and unregister from the broker.
- Once registered, they will carry out the computations.

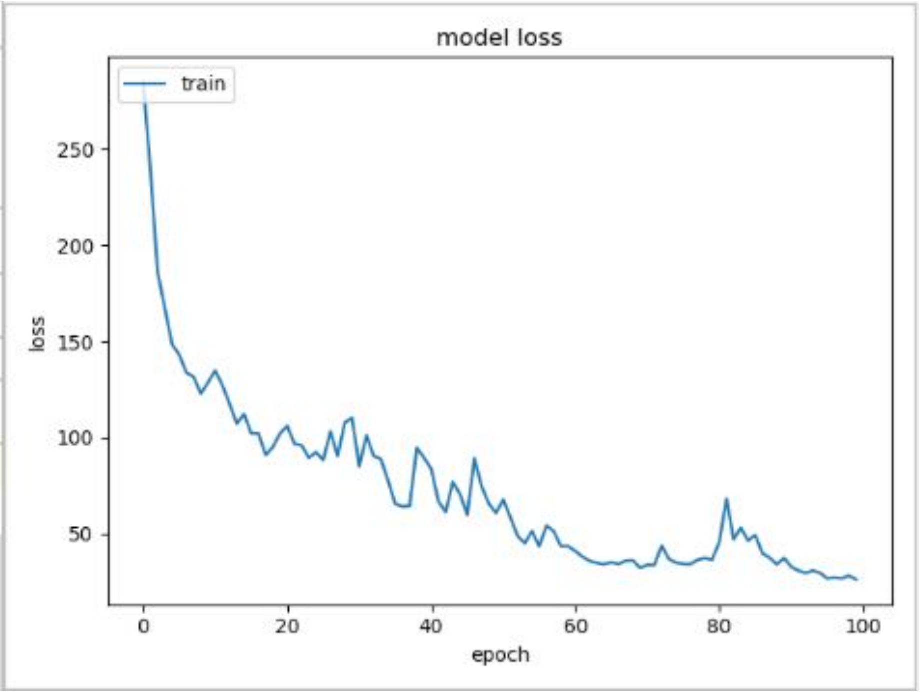

(2) Register/Unregister Workers
Maintain the list of available workers
(1) Start Broker
Read endpoints from file
generate_request
Create process per worker
Call worker with one endpoint
generate_request
Postprocessing on returned value
New start time set to recall Init

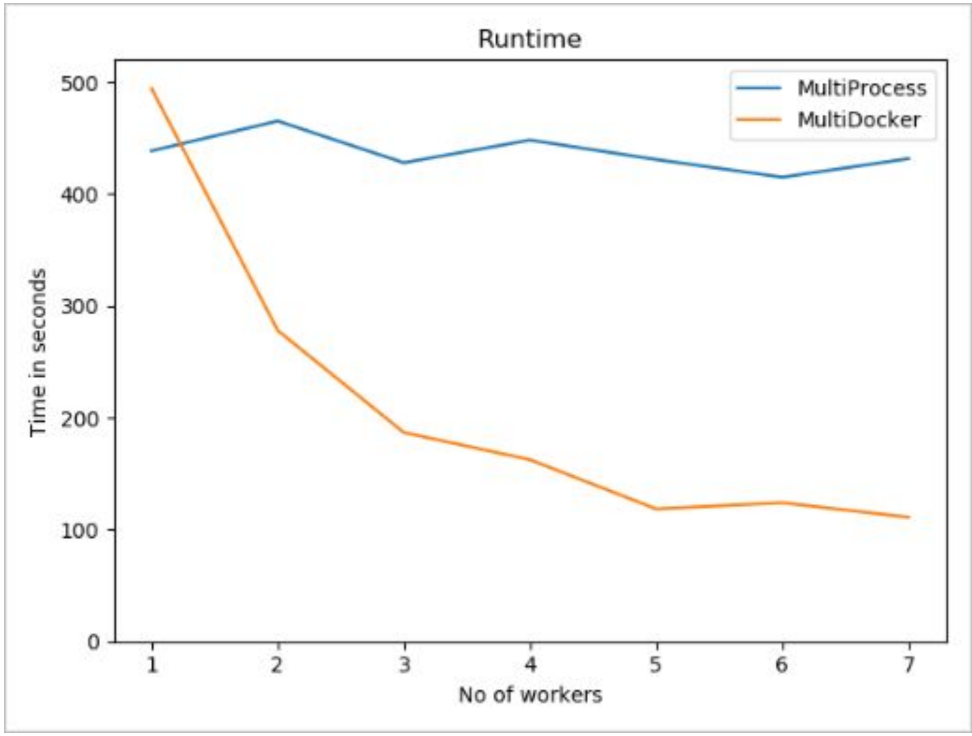
Broker


(1) On startup
Start Worker
Register with broker
(2) Broker calls worker with url
Received request to compute
Call DB to get timeseries and preprocess data
computeTimeout
(url, raw_seq)
Create model and predict value
url,raw_seq
computeTimeout
Postprocessing on predicted value
Postprocessed value

Worker

Performance



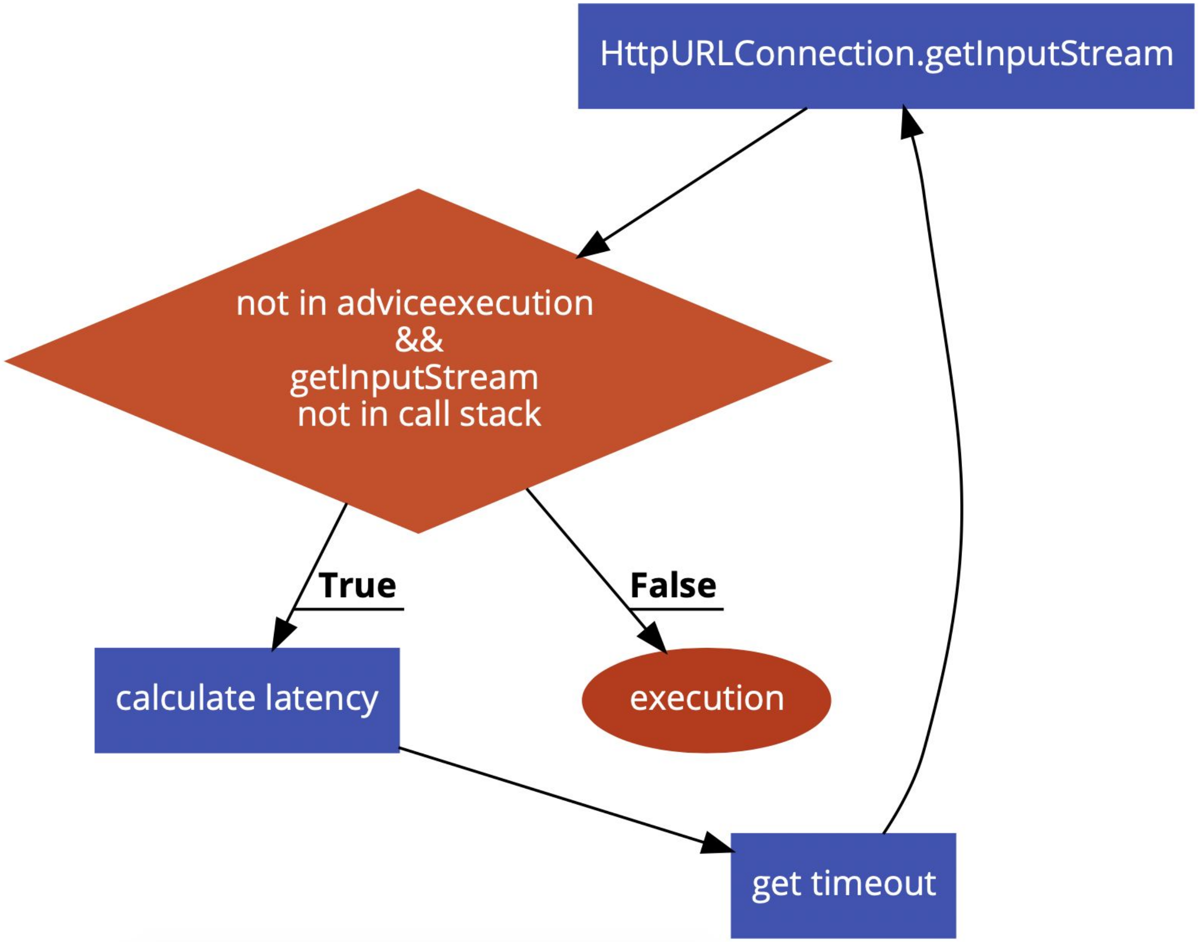


Client Shim

Aspect Oriented Timeout Injection

- Distributed feedback for timeout adjustment
  - Fetching Timeouts
  - Caching Timeouts
  - Calculating Latency
- Weaved into the JRE

Advicing the Advice



Portability of the Shim

- Boot class path
- System properties
- Environment Variables
- Dynamically disable injection

Timeout Injection

```
2020-04-28 20:22:45.294 CDT java.net.SocketTimeoutException: Read timed out at sun.reflect.NativeConstructorAccessorImpl.
2020-04-28 20:23:07.668 CDT java.net.SocketTimeoutException: Read timed out at java.net.SocketInputStream.socketRead0(Nat
2020-04-28 20:23:07.668 CDT There was an error processing request number 2
2020-04-28 20:23:07.668 CDT java.net.SocketTimeoutException: Read timed out at sun.reflect.NativeConstructorAccessorImpl.
2020-04-28 20:24:36.538 CDT java.net.SocketTimeoutException: Read timed out at java.net.SocketInputStream.socketRead0(Nat
2020-04-28 20:24:36.539 CDT There was an error processing request number 10
2020-04-28 20:24:36.539 CDT java.net.SocketTimeoutException: Read timed out at sun.reflect.NativeConstructorAccessorImpl.
2020-04-28 20:25:55.664 CDT java.net.SocketTimeoutException: Read timed out at java.net.SocketInputStream.socketRead0(Nat
2020-04-28 20:25:55.665 CDT There was an error processing request number 17
2020-04-28 20:25:55.665 CDT java.net.SocketTimeoutException: Read timed out at sun.reflect.NativeConstructorAccessorImpl.
2020-04-28 20:26:05.354 CDT java.net.SocketTimeoutException: Read timed out at java.net.SocketInputStream.socketRead0(Nat
2020-04-28 20:26:05.355 CDT There was an error processing request number 18
2020-04-28 20:26:05.355 CDT java.net.SocketTimeoutException: Read timed out at sun.reflect.NativeConstructorAccessorImpl.
2020-04-28 20:26:49.899 CDT java.net.SocketTimeoutException: Read timed out at java.net.SocketInputStream.socketRead0(Nat
2020-04-28 20:26:49.902 CDT There was an error processing request number 22
2020-04-28 20:26:49.902 CDT java.net.SocketTimeoutException: Read timed out at sun.reflect.NativeConstructorAccessorImpl.
2020-04-28 20:26:58.813 CDT java.net.SocketTimeoutException: Read timed out at java.net.SocketInputStream.socketRead0(Nat
2020-04-28 20:26:58.816 CDT There was an error processing request number 23
```

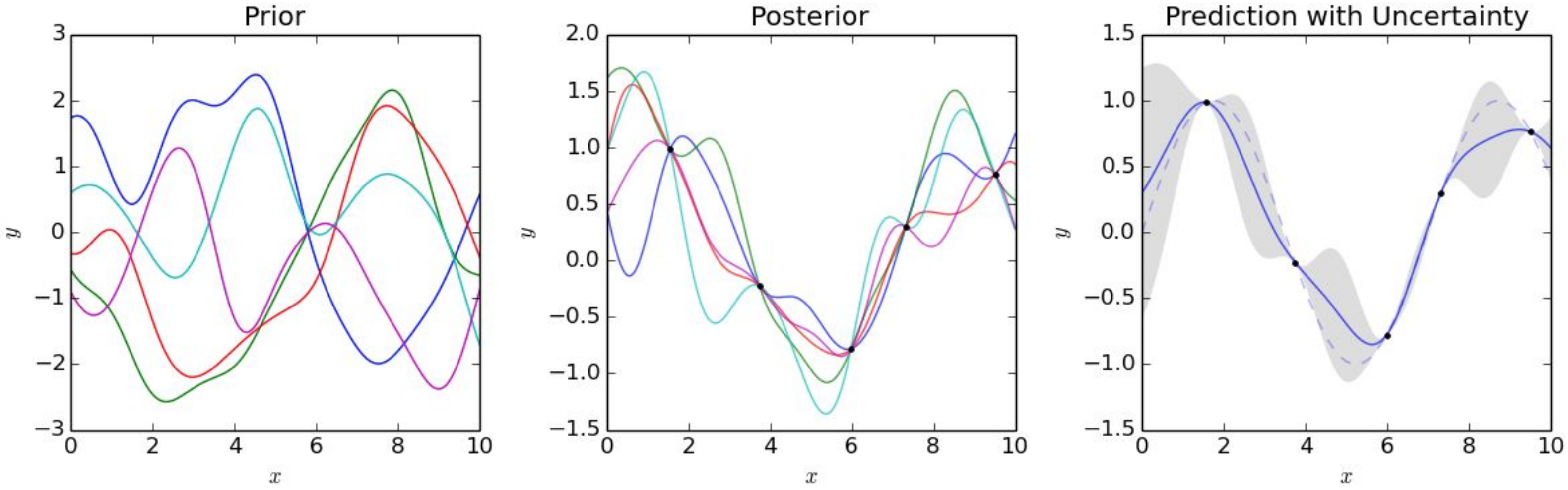
Gaussian Process Broker

Bayesian Synthesis of Gaussian Processes

Sample from the set of possible GP programs to find a set the likely generated the data.

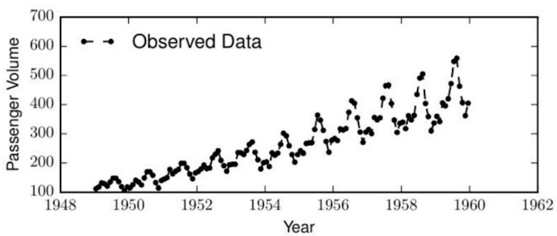
Iteratively move from a vaguely random set of programs to a likely set of programs.

Gaussian processes

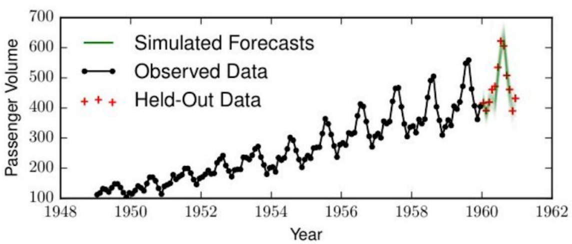


Program Synthesis

Time series structure discovery using program synthesis in Venture




```
(+ (* (WHITE-NOISE 49.5)
      (CONSTANT 250.9))
   (+ (PERIODIC 13.2 8.6)
      (+ (LINEAR 1.2)
         (LINEAR 4.9)
   (WHITE-NOISE 0.1))
```




```
// ** LOAD THE TIME SERIES DATA **
define xs_obs = get_data_xs("./data-train.csv");
define ys_obs = get_data_ys("./data-train.csv");
define xs_test = get_data_xs("./data-test.csv");
define ys_test = get_data_ys("./data-test.csv");

// ** SAMPLE ENSEMBLE OF GAUSSIAN PROCESSES FROM THE PRIOR **
resample(100);
assume dsl_code ~ generate_random_program();
assume gaussian_process_model = produce_gaussian_process(dsl_code);
observe gaussian_process_model(${xs_obs}) = ys_obs;

// ** RUN BAYESIAN SYNTHESIS **
repeat(1000, {
  infer resimulation_mh(/?hypers/*);
  infer resimulation_mh(/?structure/*)})

// ** OBTAIN FORECASTS **
sample_all(gaussian_process_model(${xs_test}$))
```

https://popl19.sigplan.org/details/POPL-2019-Research-Papers/79/Bayesian-Synthesis-of-Probabilistic-Programs-for-Automatic-Data-Modeling

GP Model Exploration

Scenarios:

1. Long Ramp
2. High Low Cycles
3. Random

Various Ensemble Counts

Various Iterations

~2000 seconds of training data

~1200 seconds forward prediction

```
CREATE CONTINUOUS QUERY "cq_20s_max" ON "tracing"
BEGIN
SELECT max("duration") INTO "max_duration" FROM "span"
GROUP BY time(20s), service_name
END
```

Model Exploration: Random

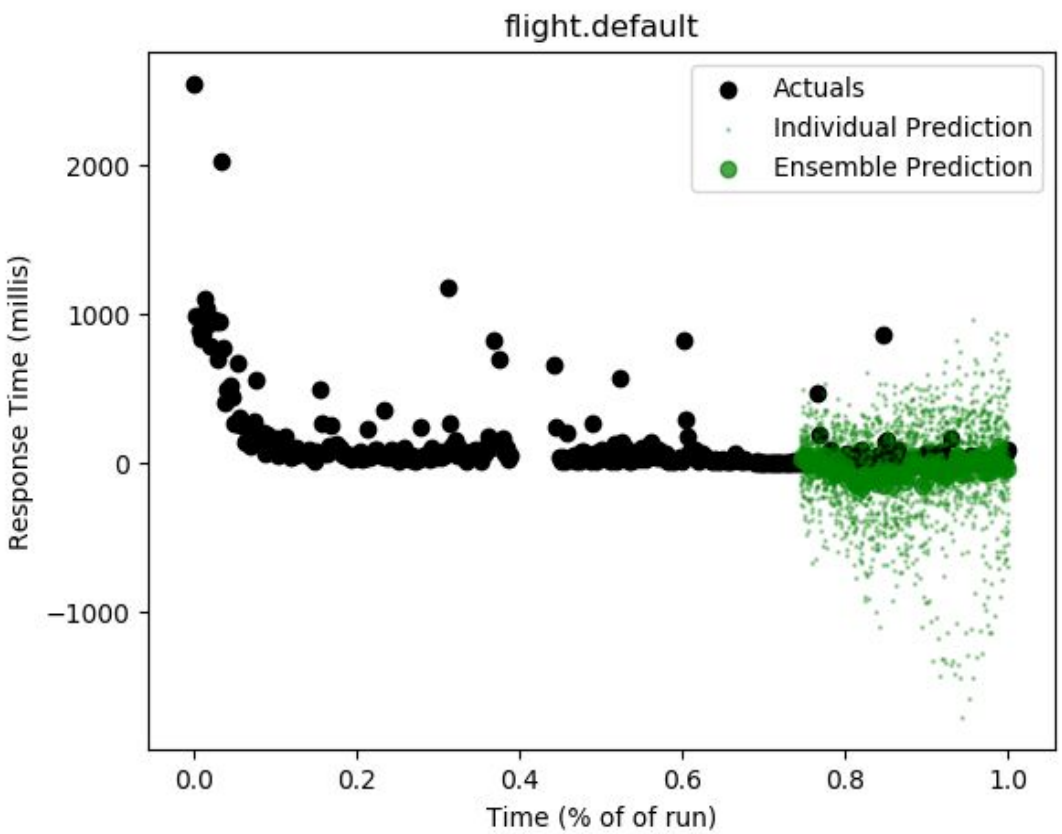
Ensemble: 100

Iterations: 1000

Train Time: 5544 s

Train Data: 90 m

Under Predict: 81%



Model Exploration: Cyclical

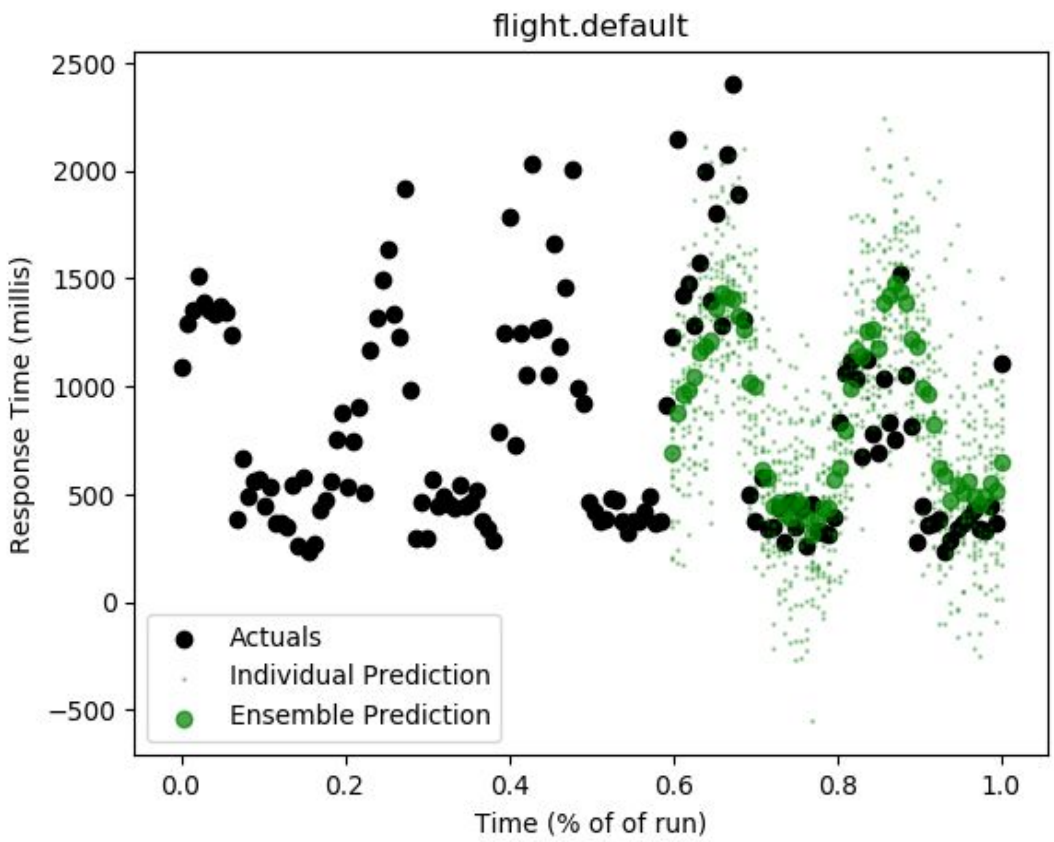
Ensemble: 20

Iterations: 1000

Train Time: 576 s

Train Data: 31 m

Under Predict: 22%



Model Exploration: Ramp

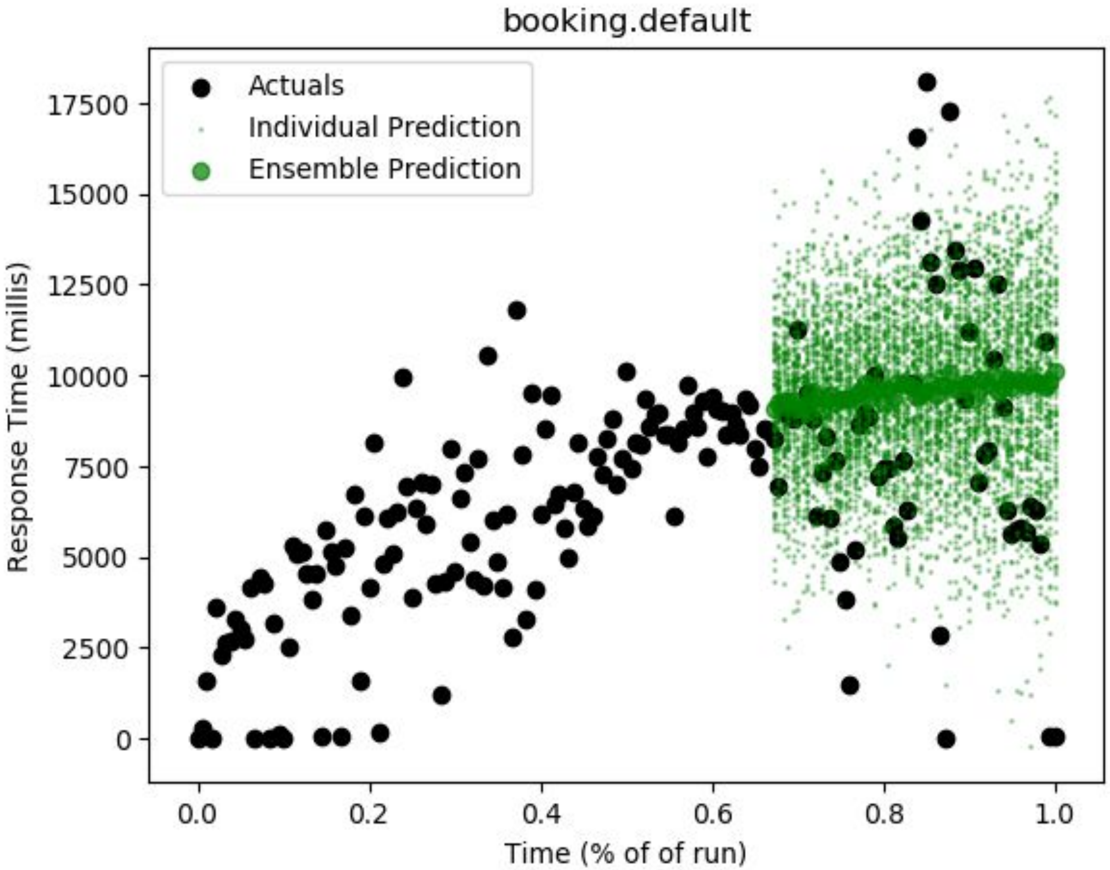
Ensemble: 100

Iterations: 100

Train Time: 359 s

Train Data: 40 m

Under Predict: 28%



GP Setup For Broker Use

- 20 second granularity using max
- (Train) on <= last 10 minutes - Ensemble Count = 1
- Predict ~7 minutes
- Timeout Recommended = 150% of
  - 99th Percentile of entire predicted next 7 minutes

GP Future Directions

- Unify the Broker version with the experiments. Run parameter optimization experiments.
- Predict over multiple time granularities and combine
- Use uncertainty measures to drive multipliers of timeout values
- Run syntactic analysis of the generated programs to find change points and trigger alarms or auto-scale resources

Integrated System Evaluation

Repeating cycles of lower and higher load over ~ 90 minutes.

4 GKE Nodes

3 Prob Broker

1 RNN Broker

Client Data Collector - “YClient” sampling the endpoints and the brokers.

Results

Table 1: Performance Metrics of Each Technique by Endpoint

	RNN		GP	
Endpoint	Under-predict %	RMSE (millis)	Under-predict %	RMSE (millis)
auth	17%	1755	30%	968
customer	32%	751	33%	665
flight	19%	978	15%	5604
booking	21%	1577	11%	5697

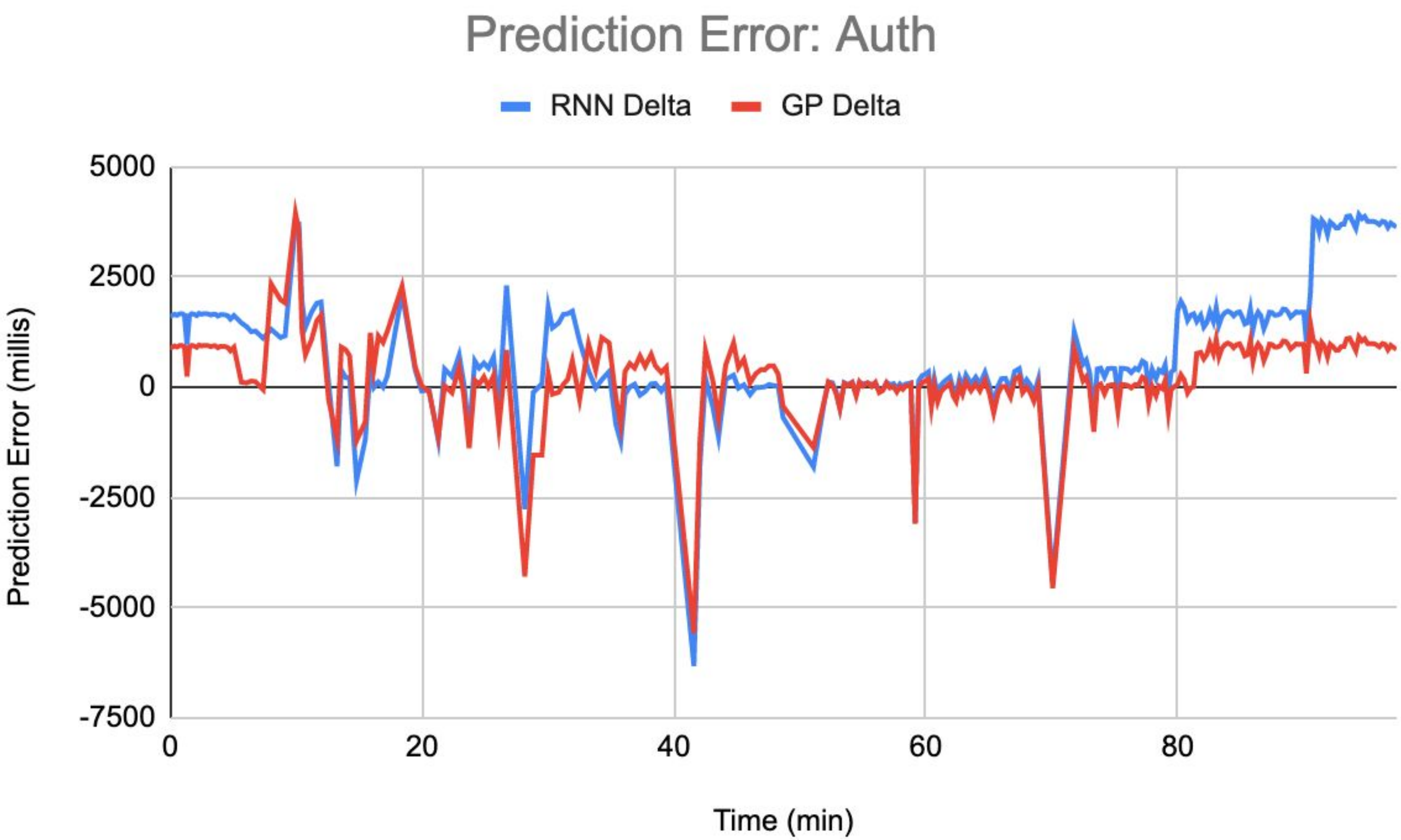


Table 1: Performance Metrics of Each Technique by Endpoint

	RNN		GP	
Endpoint	Under-predict %	RMSE (millis)	Under-predict %	RMSE (millis)
auth	17%	1755	30%	968
customer	32%	751	33%	665
flight	19%	978	15%	5604
booking	21%	1577	11%	5697

Table 1: Performance Metrics of Each Technique by Endpoint

	RNN		GP	
Endpoint	Under-predict %	RMSE (millis)	Under-predict %	RMSE (millis)
auth	17%	1755	30%	968
customer	32%	751	33%	665
flight	19%	978	15%	5604
booking	21%	1577	11%	5697

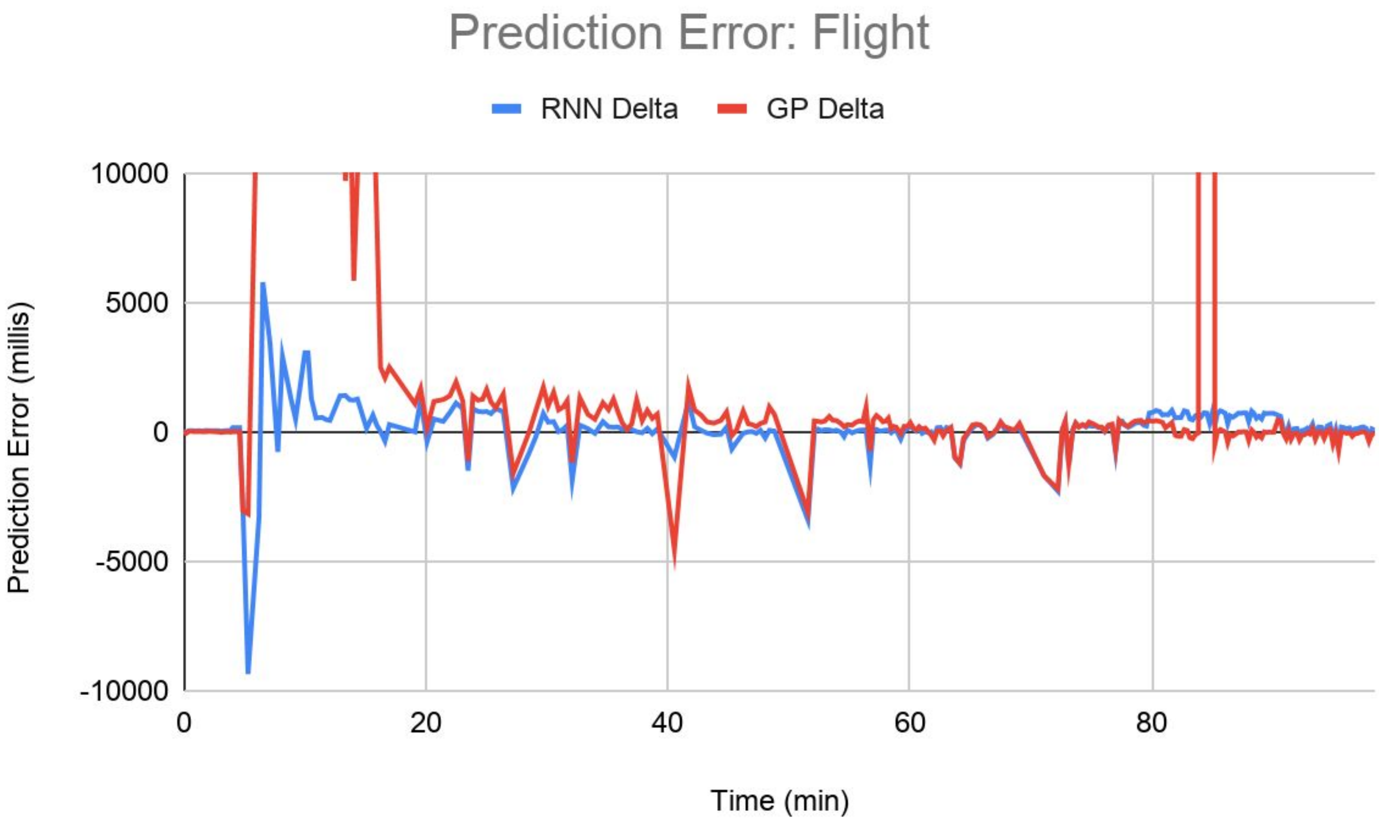


Table 1: Performance Metrics of Each Technique by Endpoint

	RNN		GP	
Endpoint	Under-predict %	RMSE (millis)	Under-predict %	RMSE (millis)
auth	17%	1755	30%	968
customer	32%	751	33%	665
flight	19%	978	15%	5604
booking	21%	1577	11%	5697

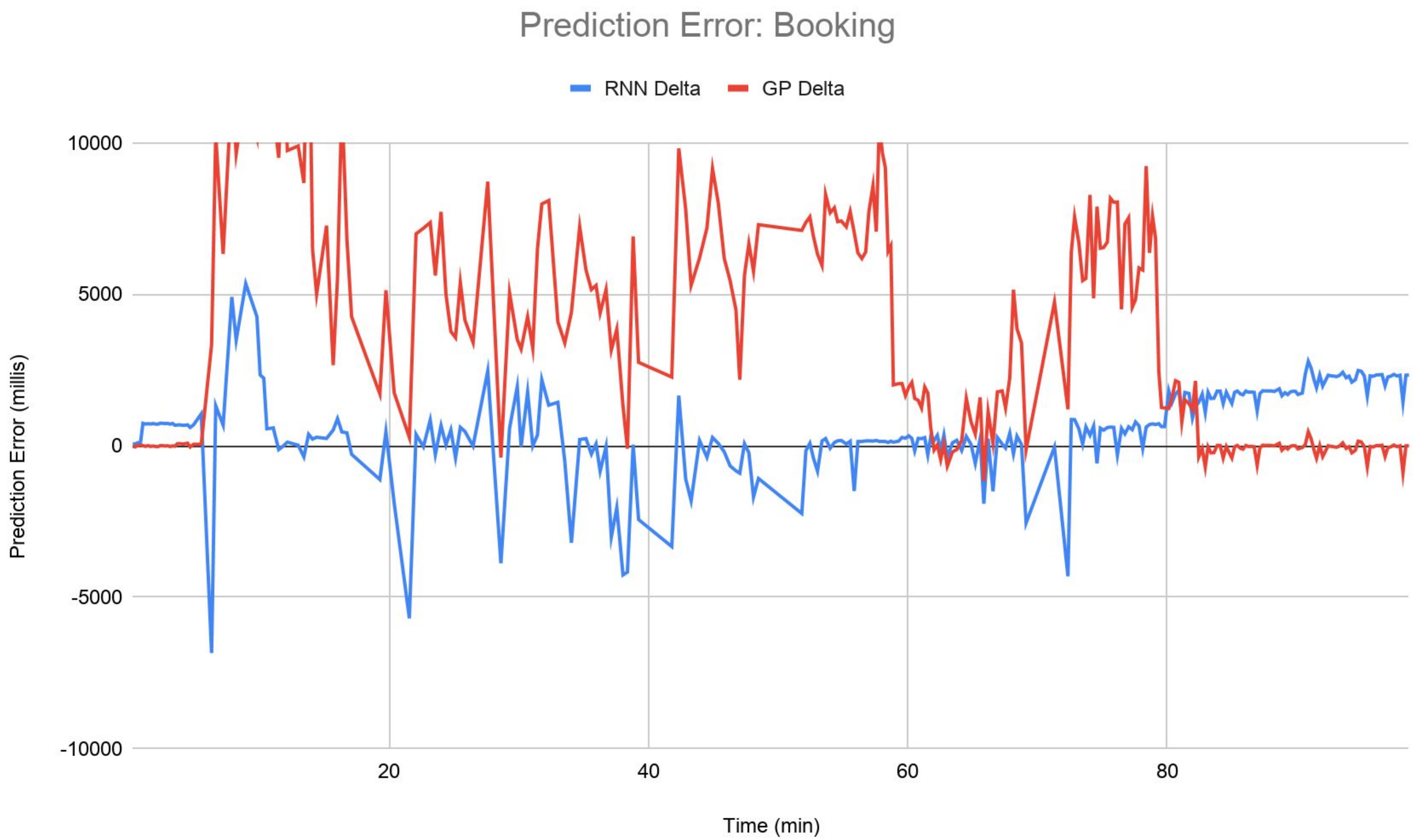
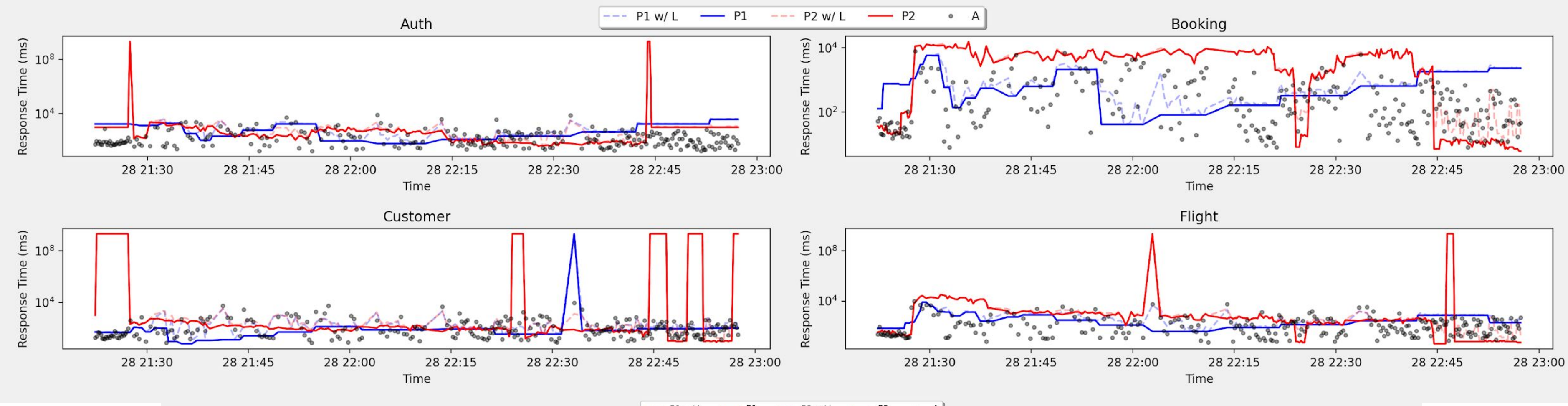
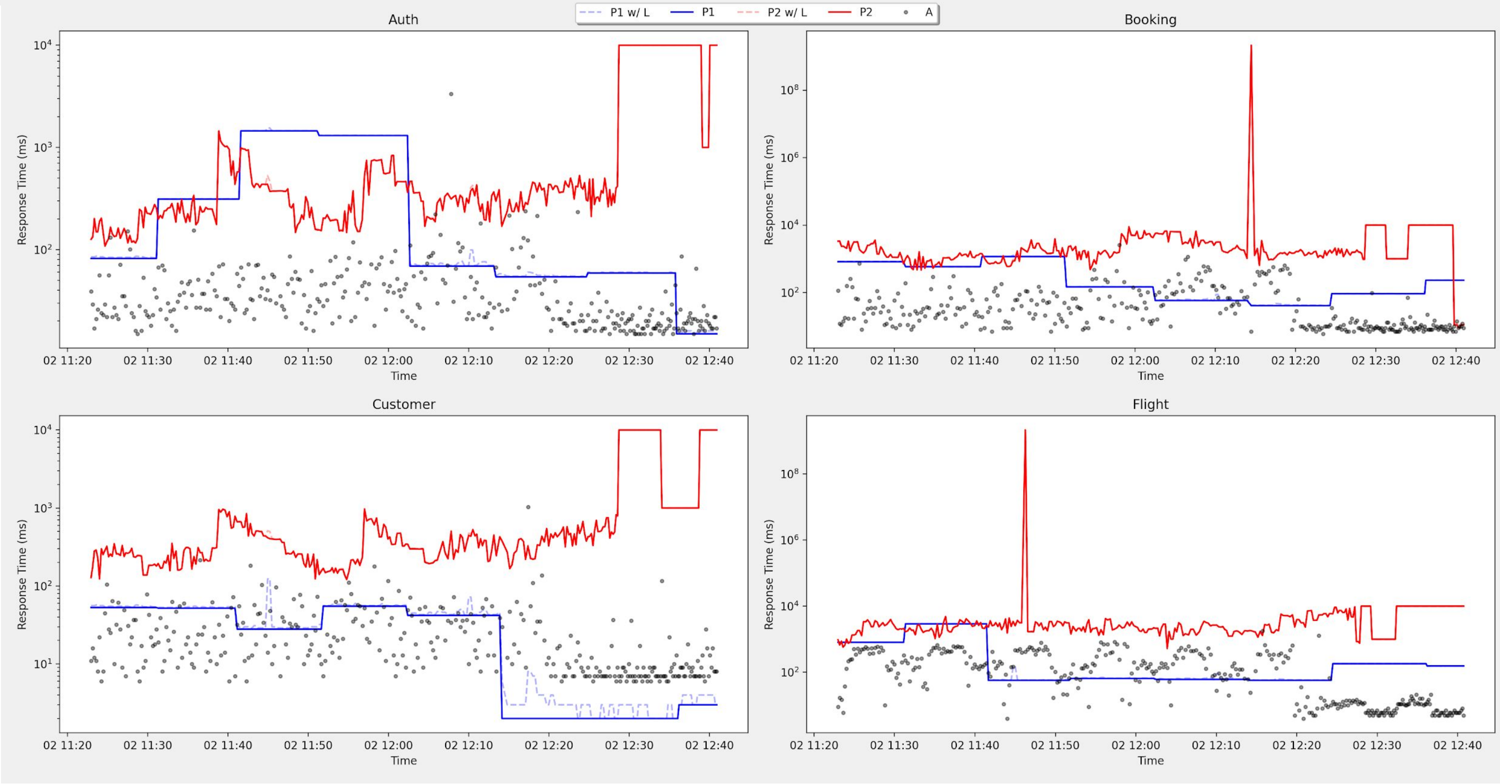


Table 1: Performance Metrics of Each Technique by Endpoint

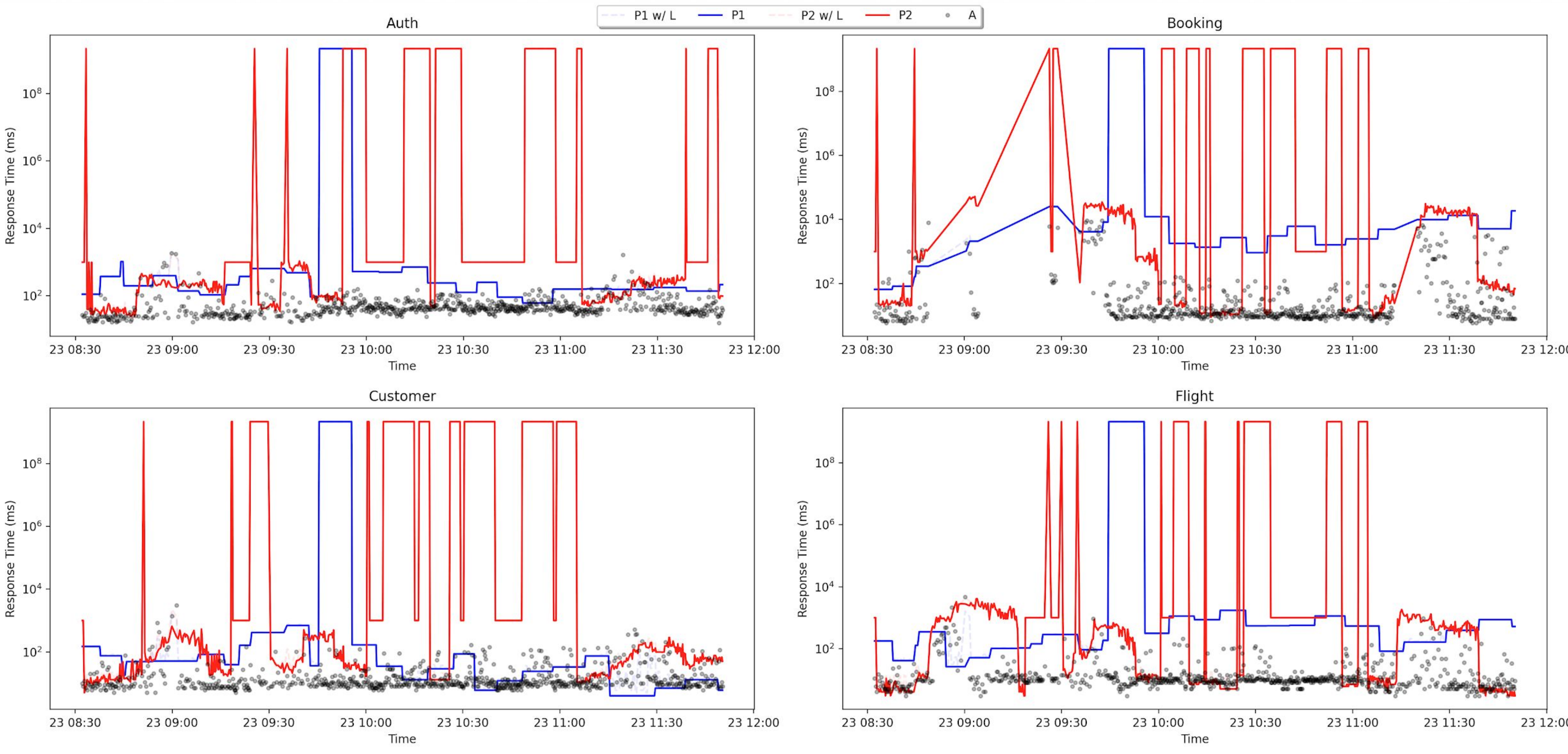
	RNN		GP	
Endpoint	Under-predict %	RMSE (millis)	Under-predict %	RMSE (millis)
auth	17%	1755	30%	968
customer	32%	751	33%	665
flight	19%	978	15%	5604
booking	21%	1577	11%	5697


P1 w/ L
P1
P2 w/ L
P2
A
Auth
Booking
Customer
Flight
Response Time (ms)
10^8
10^4
10^2
28 21:30
28 21:45
28 22:00
28 22:15
28 22:30
28 22:45
28 23:00
Time


P1 w/ L
P1
P2 w/ L
P2
A
Auth
Booking
Customer
Flight
Response Time (ms)
10^1
10^2
10^3
10^4
10^6
10^8
02 11:20
02 11:30
02 11:40
02 11:50
02 12:00
02 12:10
02 12:20
02 12:30
02 12:40
Time

Future Direction

- Better prediction models
- Differentiating types of clients
- Granular endpoint
- Auto endpoint detection in broker
- Auto scaling resources
- Add more configurability
- Eliminate the additional network call?


P1 w/ L
P1
P2 w/ L
P2
A
Auth
Booking
Customer
Flight
Response Time (ms)
10^2
10^4
10^6
10^8
Time
23 08:30
23 09:00
23 09:30
23 10:00
23 10:30
23 11:00
23 11:30
23 12:00