Distributed FaaSRunner: Enabling Reproducible Multi-node, Multi-threaded Function-as-a-Service Endpoint Testing

Tomoki Kondo
tkondo2@uw.edu

2025 IEEE International Conference on Cloud Engineering
(IC2E '25)
September 25, 2025

School of Engineering and Technology
University Of Washington, Tacoma

Outline

Background

FaaS Performance Evaluation
Applications built on FaaS must meet the required response times under expected operating conditions.

Geographically Distributed Testing
Because FaaS endpoints can receive requests from clients worldwide, reproducible load testing from geographically distributed clients is essential for realistic performance evaluation.

Workload Trace
A workload trace is historical request data observed from a real system or service, enabling realistic and reproducible load tests.

Challenges

Temporal Reproducibility
FaaS provisions resources on demand, so performance must be measured with cold start effects included. Preserving original event timings in a workload trace ensures that endpoints are exposed to the same workload in repeated tests. The test itself should not introduce variability.

Spatial Reproducibility
Network conditions vary by client location — latency, packet loss, and bandwidth can differ significantly. Characterizing these parameters and accounting for them is required to accurately reproduce distributed workload traces.

Throughput and Scalability Requirements
Workload traces often contain large volumes of requests, sometimes peaking at very high rates. Reproducing such workloads requires client tools capable of generating high throughput and scaling horizontally across multiple nodes.

Motivation

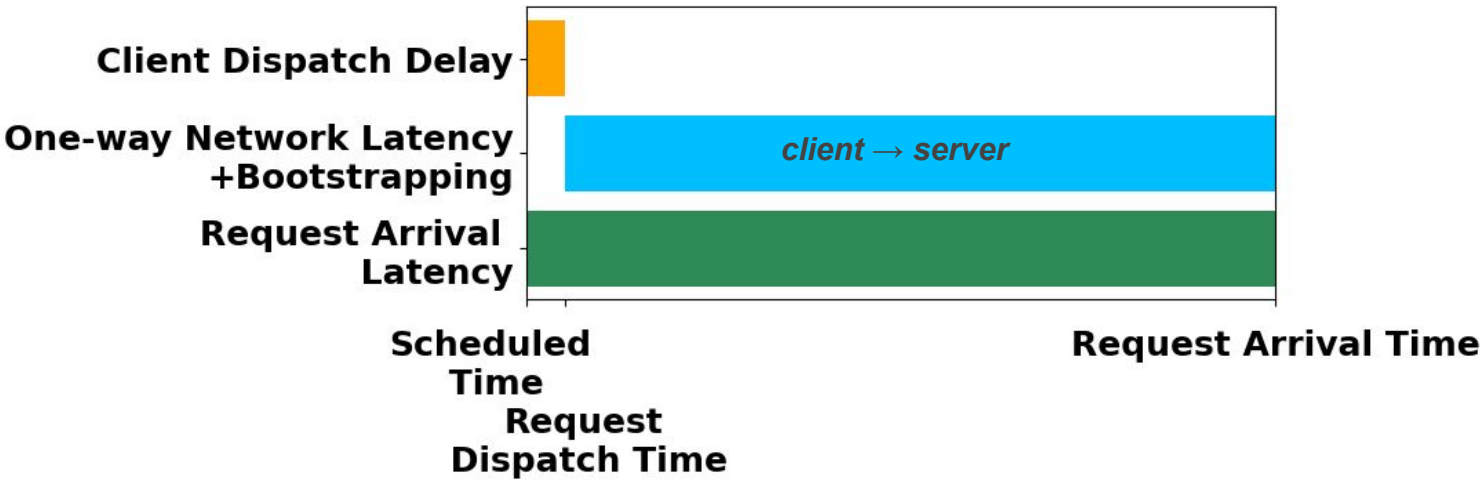
Tool	Replay of Workload Traces	Distributed Execution	Geographically Distributed Client Replay
Jmeter	×	✓	×
k6	✓	✓	×
FaaSProfiler	✓	×	×
Distributed FaaSRunner	✓	✓	✓

Outline

- Background and Motivation
- Research Questions
- Methodology
- Results
- Conclusions

Distributed Events - Timing Metrics

- **Client Dispatch Delay**: delay at the client before a scheduled event is dispatched
- **Request Arrival Latency**: delay between scheduled event time and request arrival at the server



Research Questions

- **RQ-1 (Workload Event Latency Characterization)**: What is our ability to reproduce distributed workload traces using distributed test clusters consisting of nodes dispersed across multiple cloud regions at continental vs. global levels?
- **RQ-2 (Adjustment Capability)**: By adjusting request dispatch times of scheduled events based on expected network latency, to what extent can we reduce request arrival latency when reproducing distributed event traces?

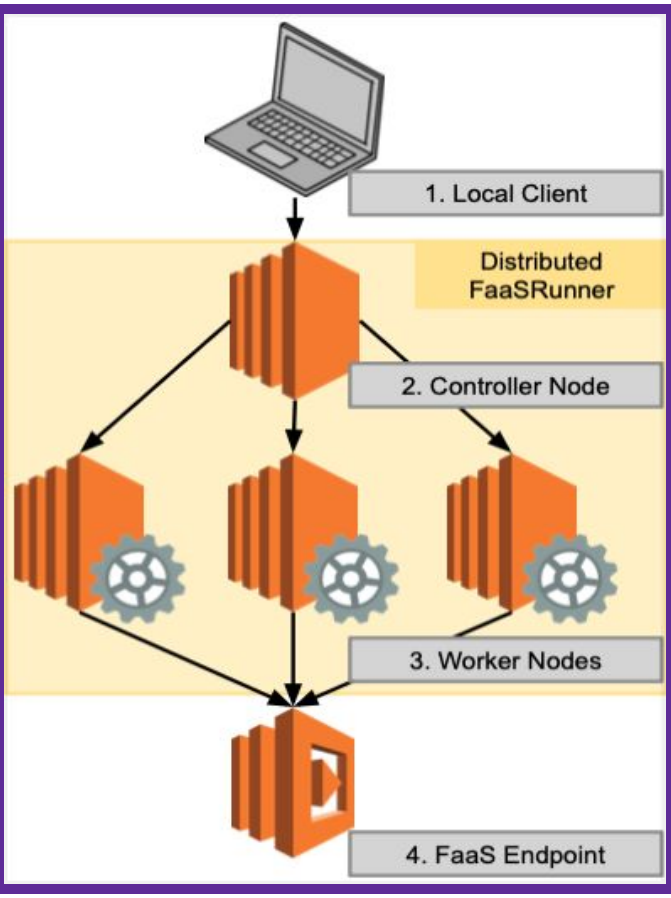
Outline

- Background and Motivation
- Research Questions
- Methodology
  - Design of Distributed FaaSRunner
  - SETGen
  - SAAF
  - Experimental Design
- Results
- Conclusions

Outline

- Background and Motivation
- Research Questions
- Methodology
  - Design of Distributed FaaSRunner
  - SETGen
  - SAAF
  - Experimental Design
- Results
- Conclusions

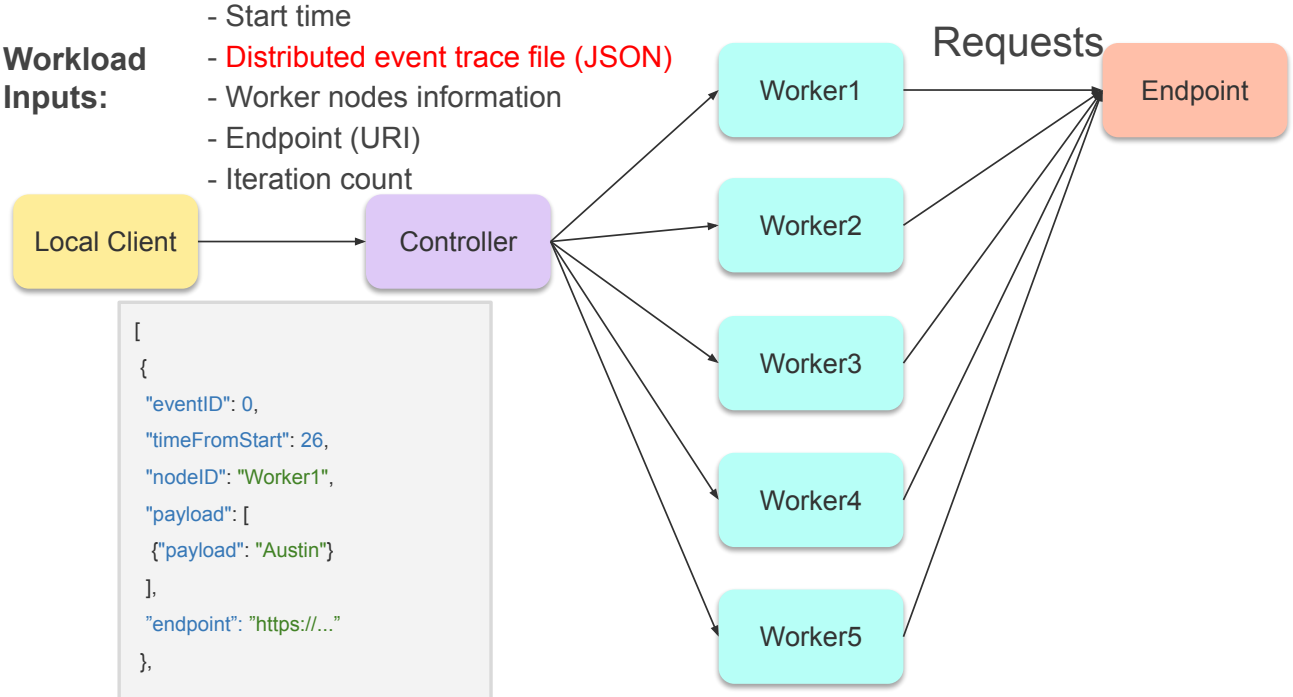
Design of Distributed FaaSRunner



Distributed FaaSRunner: Event Trace Mode

Workload Inputs:

- Start time
- Distributed event trace file (JSON)
- Worker nodes information
- Endpoint (URI)
- Iteration count

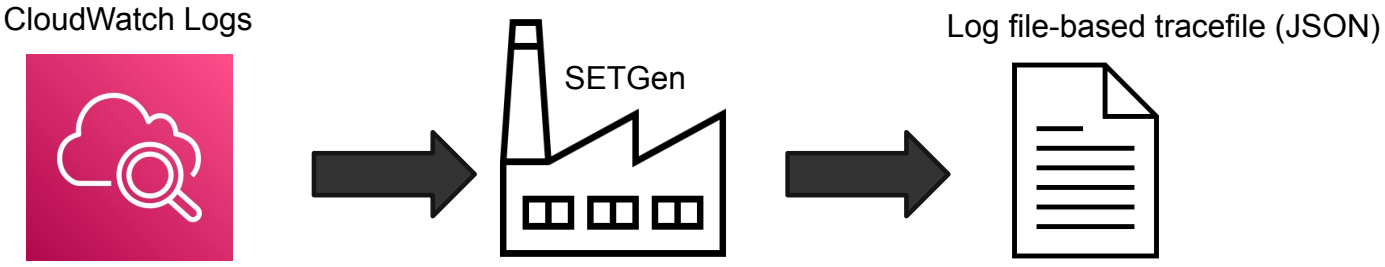


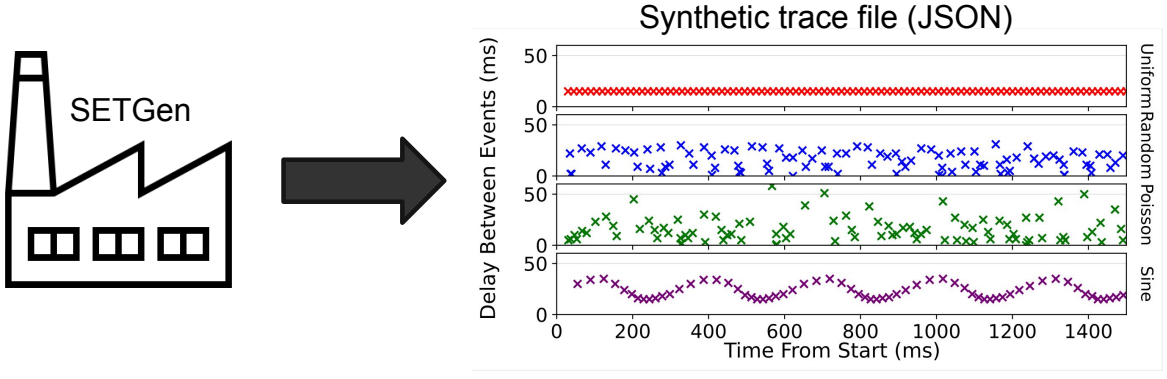

```
[
 {
  "eventID": 0,
  "timeFromStart": 26,
  "nodeID": "Worker1",
  "payload": [
   {"payload": "Austin"}
  ],
  "endpoint": "https://..."
 },
 ...
```

Outline

- Background and Motivation
- Research Questions
- Methodology
  - Design of Distributed FaaSRunner
  - SETGen
  - SAAF
  - Experimental Design
- Results
- Conclusions

SETGen (Serverless Event Trace Generator)





Outline

- Background and Motivation
- Research Questions
- Methodology
  - Design of Distributed FaaSRunner
  - SETGen
  - SAAF
  - Experimental Design
- Results
- Conclusions

SAAF (Serverless Application Analytics Framework)

SAAF supports profiling Function-as-a-Service (FaaS) workload performance, resource utilization, and infrastructure enabling accurate performance and cost characterizations.

Cloud:

…etc

Language:

…etc

Inspection:

- cpuType
- vmID
- runtime
- newcontainer
- startTime
- endTime

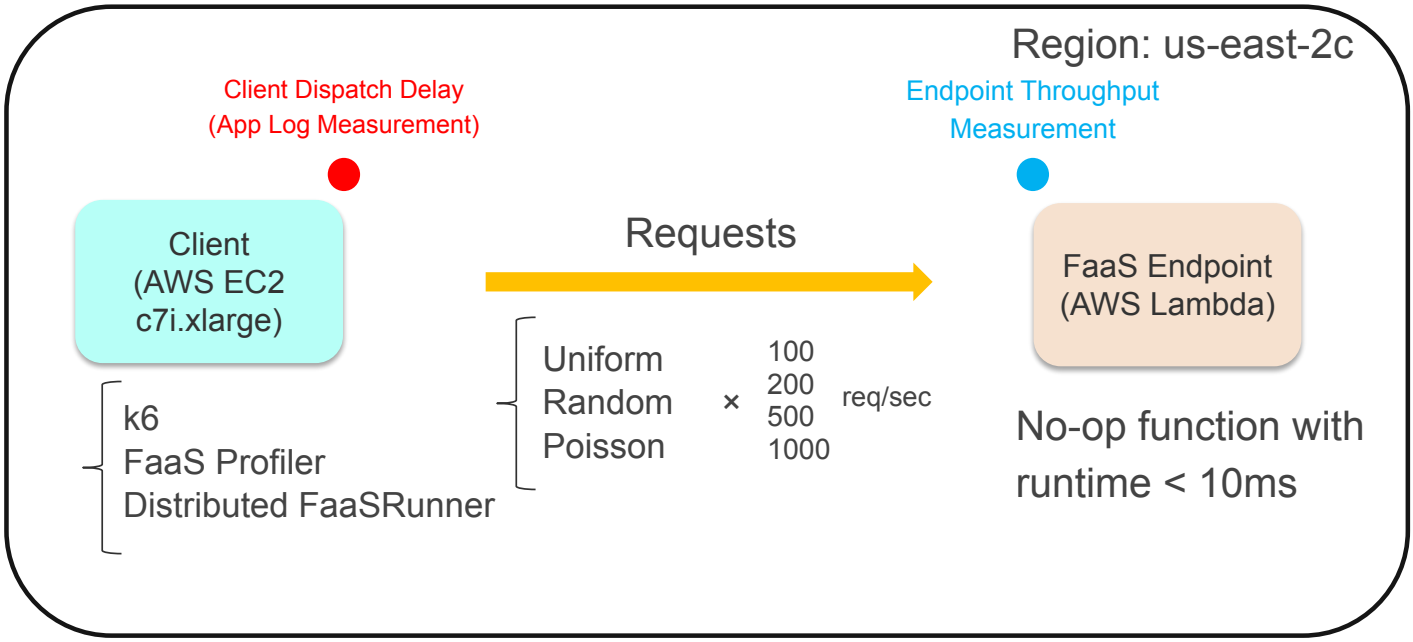
…etc

Outline

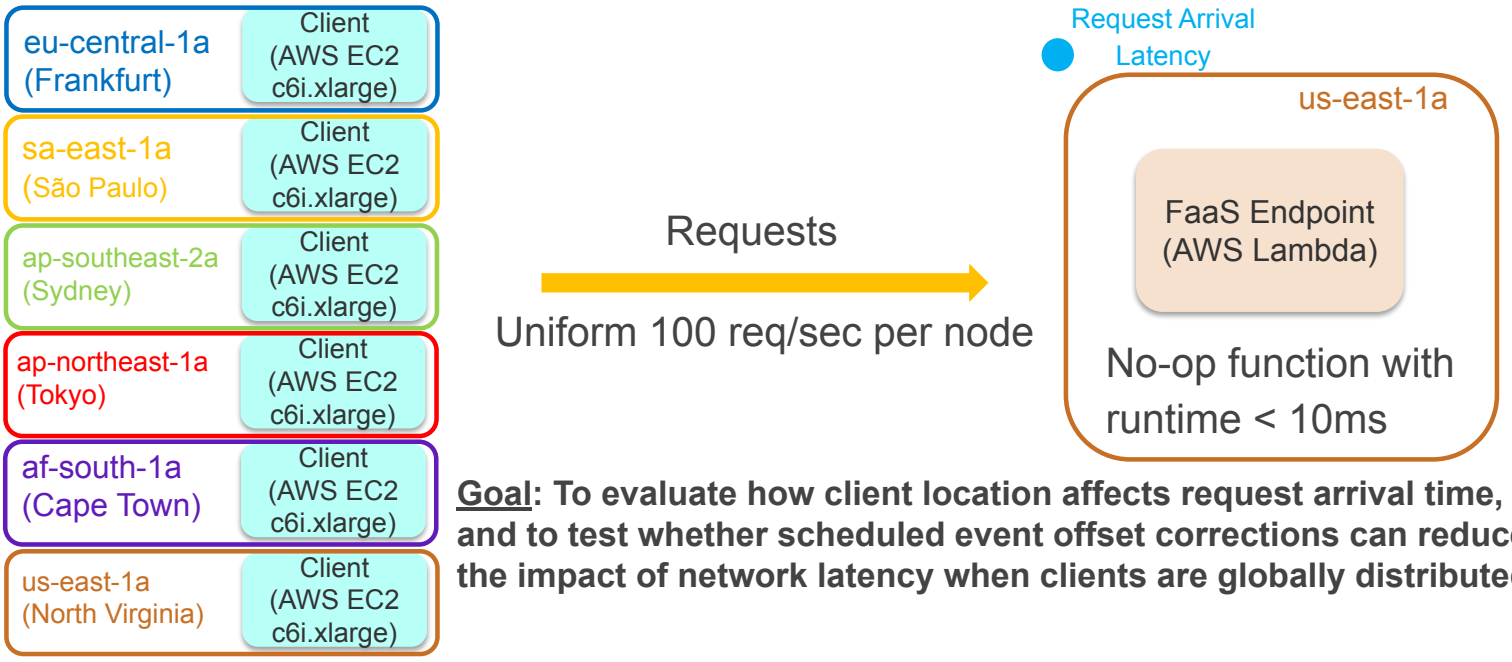
- Background and Motivation
- Research Questions
- Methodology
  - Design of Distributed FaaSRunner
  - SETGen
  - SAAF
- Experimental Design
- Results
- Conclusions

Ex. 1. Single Node Performance Comparison

Goal: Establish single-node throughput (req/sec) and Client Dispatch Delay for each tool

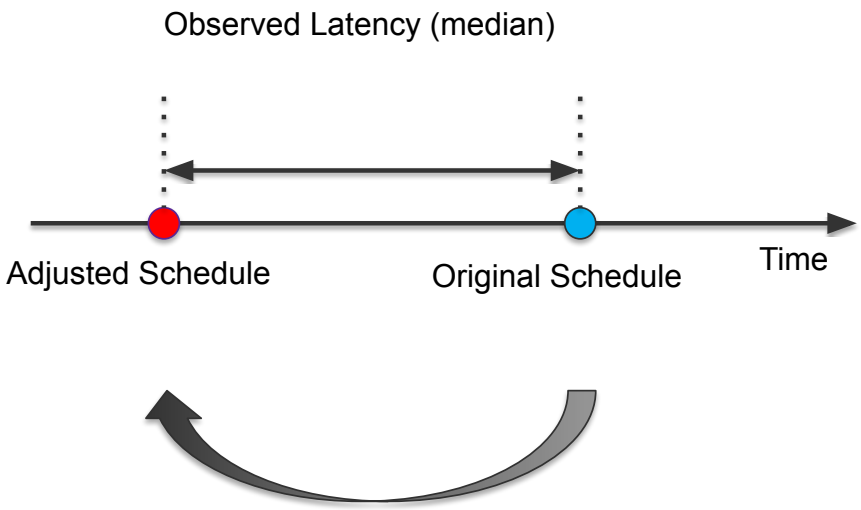


Ex. 4. Impact of Test Cluster Configuration and Offset Correction on Request Arrival Time

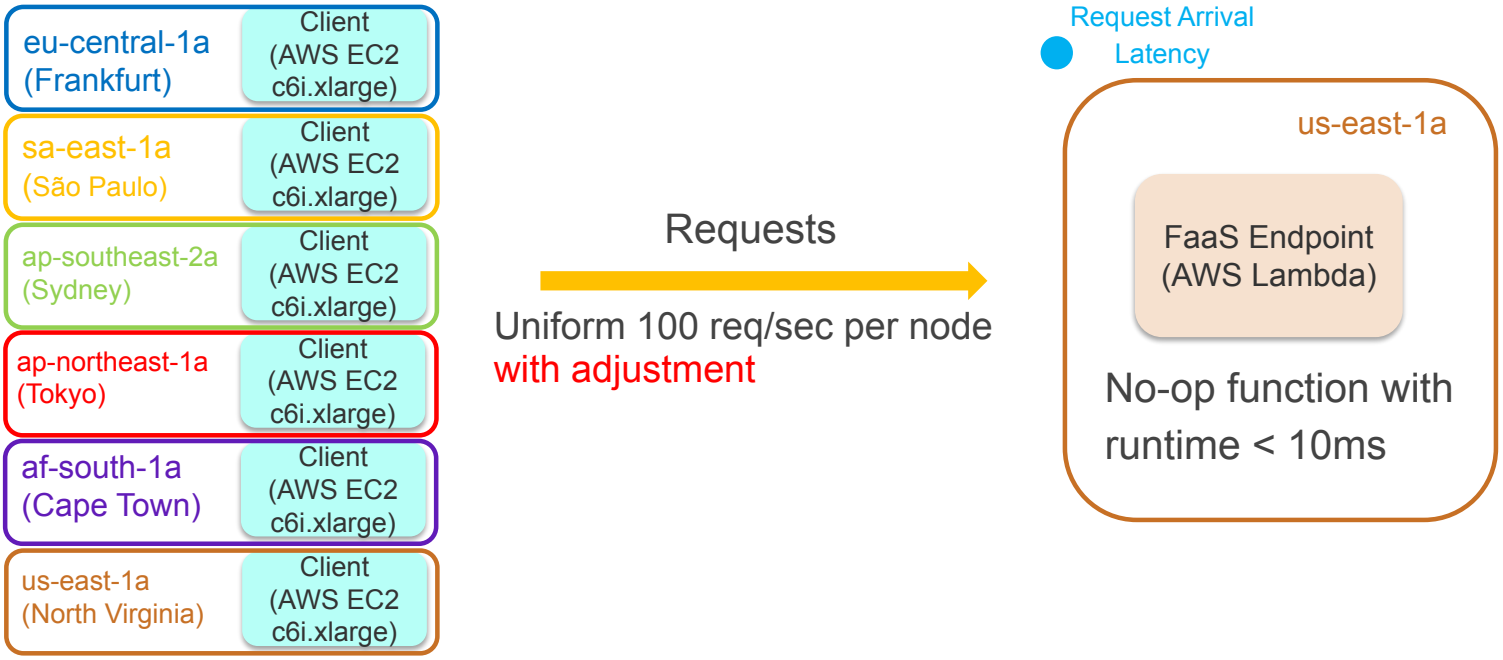


Goal: To evaluate how client location affects request arrival time, and to test whether scheduled event offset corrections can reduce the impact of network latency when clients are globally distributed.

Ex. 4. Impact of Test Cluster Configuration and Offset Correction on Request Arrival Time



Ex. 4. Impact of Test Cluster Configuration and Offset Correction on Request Arrival Time



Outline

- Background and Motivation
- Research Questions
- Methodology
- Results
- Conclusions

Results: Ex. 1 (Single Node Performance Comparison)

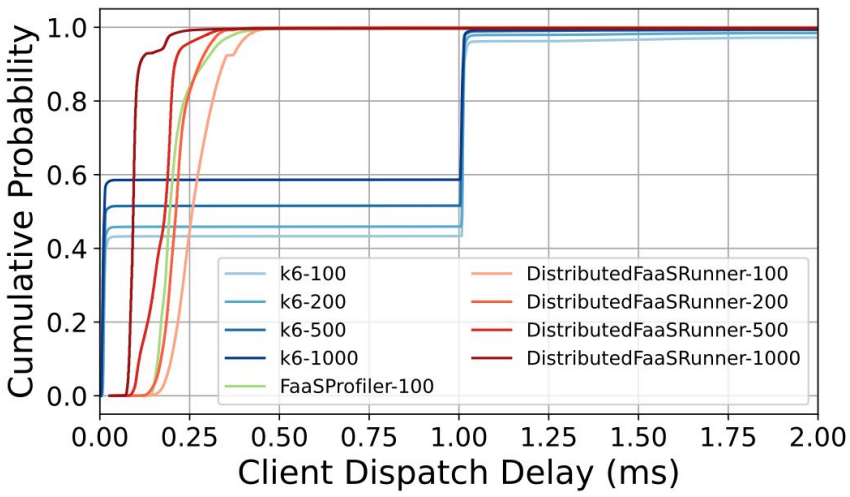
Endpoint Throughput

Tool	Target rate	Mean (rps)	Std
k6	100	100.036	0.007
k6	200	200.062	0.028
k6	500	500.080	0.040
k6	1000	1000.055	0.113
FaaSProfiler	100	100.047	0.012
FaaSProfiler	200	186.591	2.389
FaaSProfiler	500	188.807	1.783
FaaSProfiler	1000	188.020	1.336
Distributed FaaSRunner	100	100.031	0.006
Distributed FaaSRunner	200	200.049	0.033
Distributed FaaSRunner	500	500.093	0.030
Distributed FaaSRunner	1000	1000.153	0.151

- Both k6 and Distributed FaaSRunner effectively met the target throughput of up to 1000 req/sec.
- FaaSProfiler satisfied the requirement at 100 req/sec but was unable to maintain performance beyond 200 req/sec. Its maximum achievable throughput is approximately 188 req/sec.

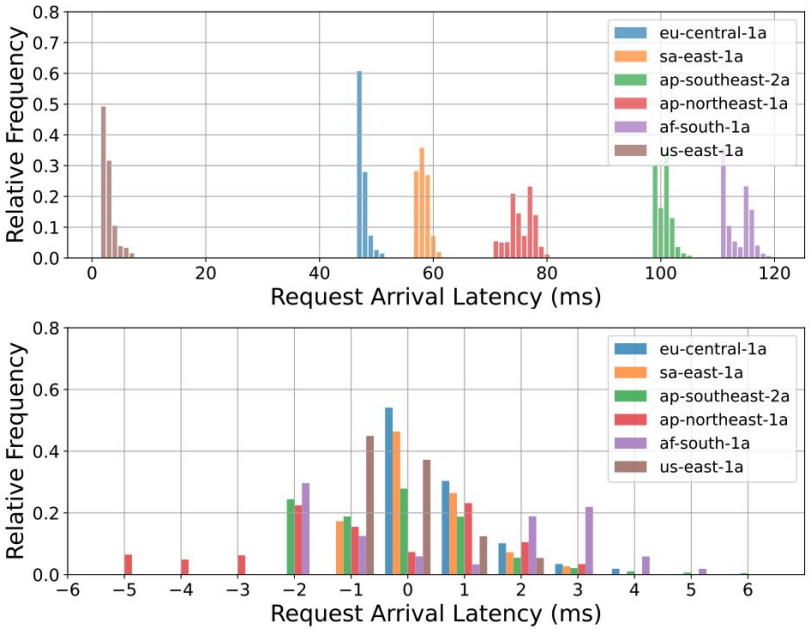
Results: Ex. 1 (Single Node Performance Comparison)

Client Dispatch Delay



- k6 exhibited high precision for about half of the requests, while the other requests displayed delays near 1 ms, indicating a bimodal dispatch pattern.
- The Distributed FaaSRunner showed strong consistency, with almost all requests dispatched in under 0.5 ms.

Results: Ex. 4 (Impact of Test Cluster Configuration and Offset Correction on Request Arrival Time)



- Request arrival latency generally increased as geographic distance grew, often surpassing 50 to 100 ms.
- By applying a static time offset correction derived from the observed median latency per node, the median arrival time across all regions was reduced to within a few milliseconds of zero.

Outline

- Background and Motivation
- Research Questions
- Methodology
- Results
- Conclusions

Conclusion: RQ-1

RQ-1 (Workload Event Latency Characterization): What is our ability to reproduce distributed workload traces using distributed test clusters consisting of nodes dispersed across multiple cloud regions at continental vs. global levels?

We observe that the median delay between scheduled and actual request arrival time often exceeds 50–100 ms, especially between geographically distant regions. Although Distributed FaaSRunner can issue requests with sub-millisecond accuracy, such end-to-end delays constrain the workload traces reproducibility at the server.

Conclusion: RQ-2

RQ-2 (Adjustment Capability): By adjusting request dispatch times of scheduled events based on expected network latency, to what extent can we reduce request arrival latency when reproducing distributed event traces?

Applying static adjustments derived from the Measurement-based Approach consistently reduces the median request arrival latency to within a few milliseconds (e.g. 0-2 ms) across all tested regions.

Thank You!

Questions?