Towards Serverless Sky Computing

An Investigation of Global Workload Distribution to Mitigate Carbon Intensity, Network Latency, and Cost

Robert Cordingly, Jasleen Kaur, Divyansh Dwivedi, Wes Lloyd
rcording@uw.edu

School of Engineering and Technology
University of Washington Tacoma
11th IEEE International Conference on Cloud Engineering
IC2E 2023

Outline

- Background and Motivation
- Research Questions
- Serverless Proxy System
- Methodology and Results
- Conclusions

Why Serverless?

DigitalOcean

IBM Cloud Functions

Apache OpenWhisk

Serverless function-as-a-service (FaaS) platforms offer many desirable features:

- Rapid elastic scaling
- Scale to zero
- No infrastructure management
- Fine grained billing
- Fault tolerance
- No up front cost to deploy an application

What is Sky Computing?

Azure

Google Cloud

aws

- The Sky sits above the clouds.
- Consists of compatibility layers allowing interoperability between multiple cloud providers.

Goals for Sky Computing:

- Reduce vendor lock-in
- Allow applications to take advantage of resources of multiple cloud providers.

Serverless Sky Computing

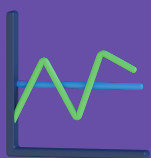
Sky Computing has potential to enhance Serverless Computing by enabling:

- Reduce carbon intensity
- Improve performance
- Reduce latency
- Reduce hosting costs
- Improve fault tolerance

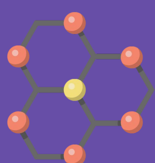
Outline

- Background and Motivation
- Research Questions
- Serverless Proxy System
- Methodology and Results
- Conclusions

Research Questions



- RQ-1 (**Performance Variation**): How does function network latency and runtime of a serverless platform vary over time by region?



- RQ-2 (**Carbon Intensity**): How is the carbon intensity of a serverless application impacted by different cloud aggregations? How does the carbon intensity of cloud regions change over time?

Research Questions

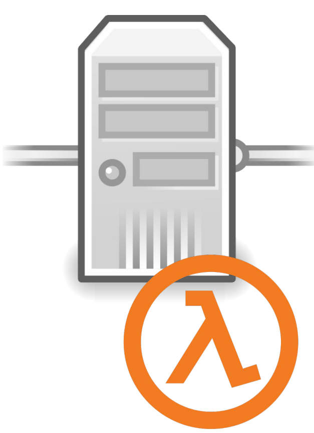
- RQ-3 (**Sustainability Costs**): What are the latency and performance implications of minimizing the carbon footprint of a serverless application through carbon-aware load distribution?

- RQ-4 (**Multi-configuration Aggregation**): How can serverless resource aggregation be leverages to reduce application hosting costs by utilizing function deployments with many different configurations?

Outline

- Background and Motivation
- Research Questions
- Serverless Proxy System
- Methodology and Results
- Conclusions

Serverless Proxy System



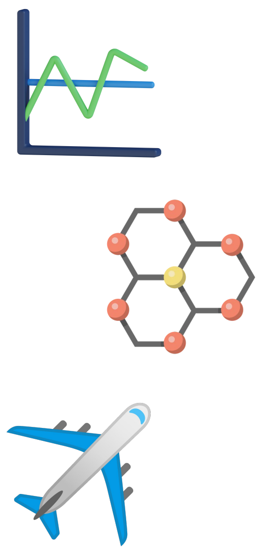
Analyzer

- Collects carbon intensity metrics for every AWS region
- Stores data in S3 for future use
- Informs Smart Proxies to make routing decisions
- Deployed to US-West-1 and is called every 15 minutes
- Can be expanded in the future to collect latency/runtime metrics

Smart Proxies

- Deployed in every AWS region
- Designed to minimize overhead cost :
  - Performs optimally at the lowest memory setting
  - Has minimal runtime
- Invoked synchronously and handles routing request
- Makes routing decisions using 5 load distribution techniques

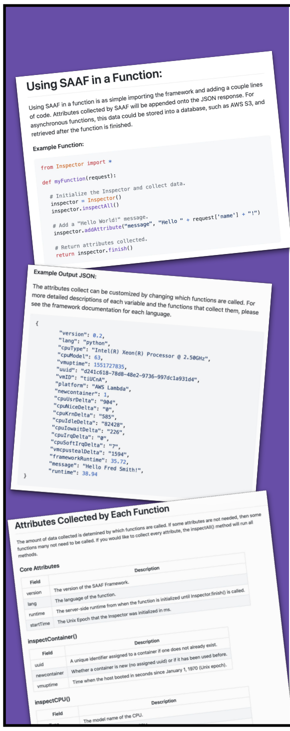
Smart Proxy Load Distribution Techniques



1. **Ohio**: All requests, route to a single region: Ohio
2. **Minimize Carbon**: Requests route to the region with the lowest carbon footprint (nearest if there is a tie)
3. **Minimize Distance**: Requests route to the nearest region
4. **Balanced**: Weighs increases in distance and carbon equally to make routing decisions
5. **Weighted on Distance**: 3X weight is applied to distance over carbon, prioritizing low latency but also considering carbon footprint

Workloads

Function	TLP	Description
MST*	1	Generates a graph and calculates the min spanning tree.
BFS*	1	Generates a graph and processes a breadth first search.
Page Rank*	1.2	Generates a graph and processes page rank of each node.
DNA*	0.9	Pulls DNA sequence from S3 and creates visualization data.
Compress*	1	Generates files and compresses them into a zip file.
Resize*	1	Pulls an image from S3, resizes it and saves it back to S3.
Stress	n	Tool used to generate CPU stress.
Writer	1	Generates text and repeatedly writes it to disk and deletes.
CSV Processor	1	Generates a large CSV file and performs calculates on columns.
Calcs	n	Executes random math operations.
Matrix Calcs	n	Generates random large matrices and performs matrix operations.
HTTP Request	1	Makes a HTTP request with a defined payload to a URL.



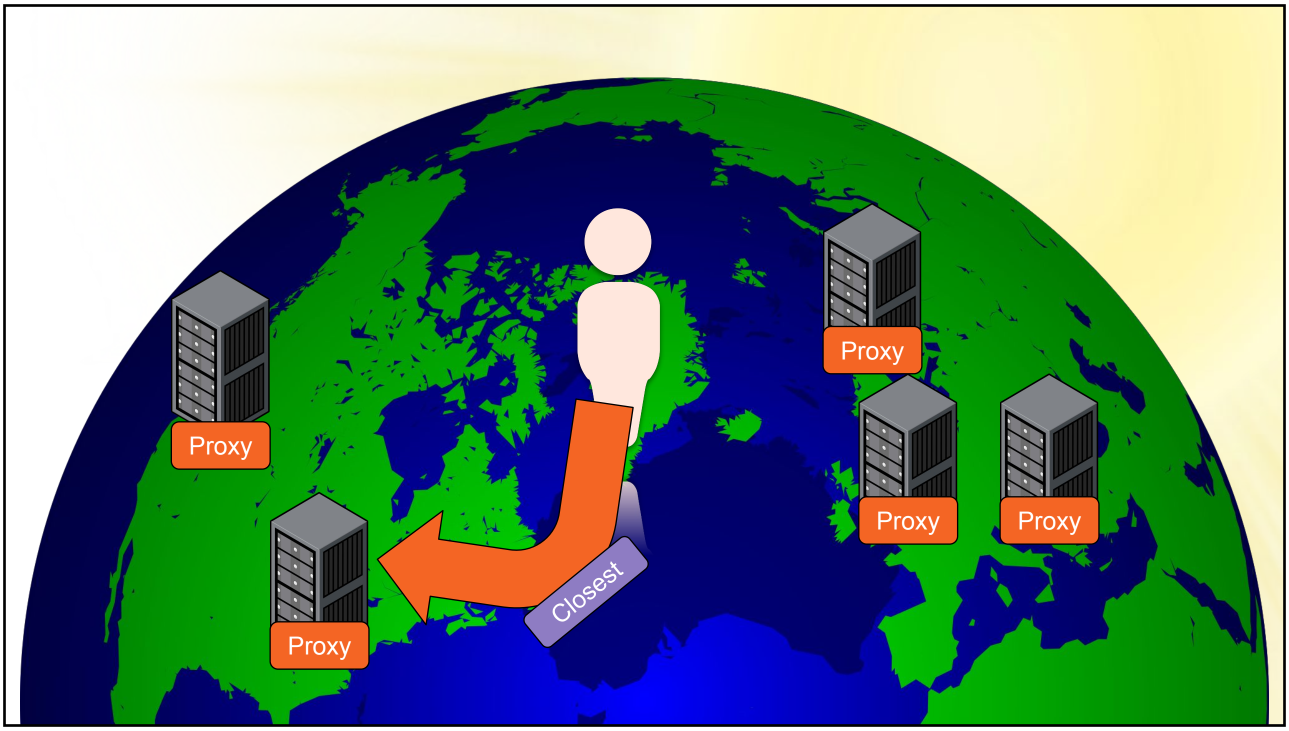
Supporting Tools - SAAF

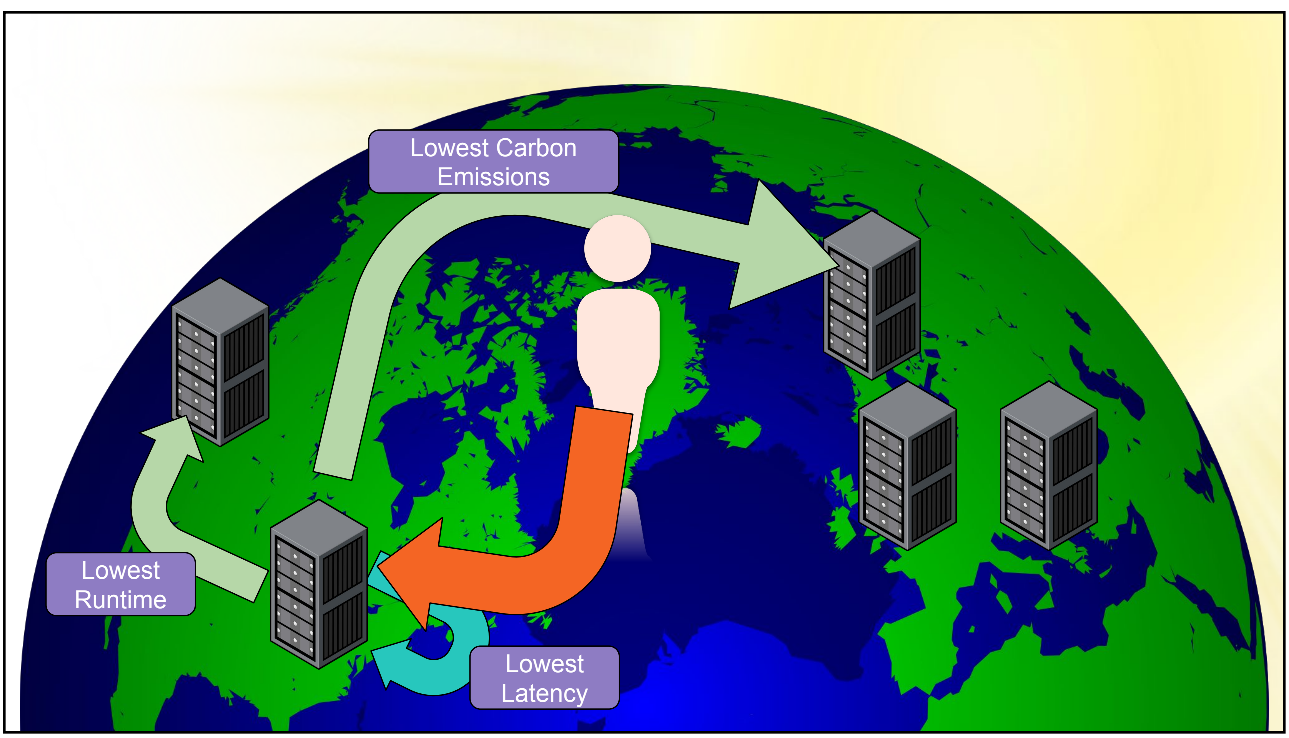
We utilize the Serverless Application Analytics Framework to collect metrics from serverless functions.

Metrics such as CPU timing accounting, runtime, latency, and more can collected by the Analyzer function and used to make routing decisions by the Proxies.

SAAF and our other tools are is available here: https://github.com/wlloyduw/SAAF

Routing Demonstration


Proxy
Proxy
Proxy
Proxy
Proxy
Closest


Lowest Carbon Emissions
Lowest Runtime
Lowest Latency

Outline

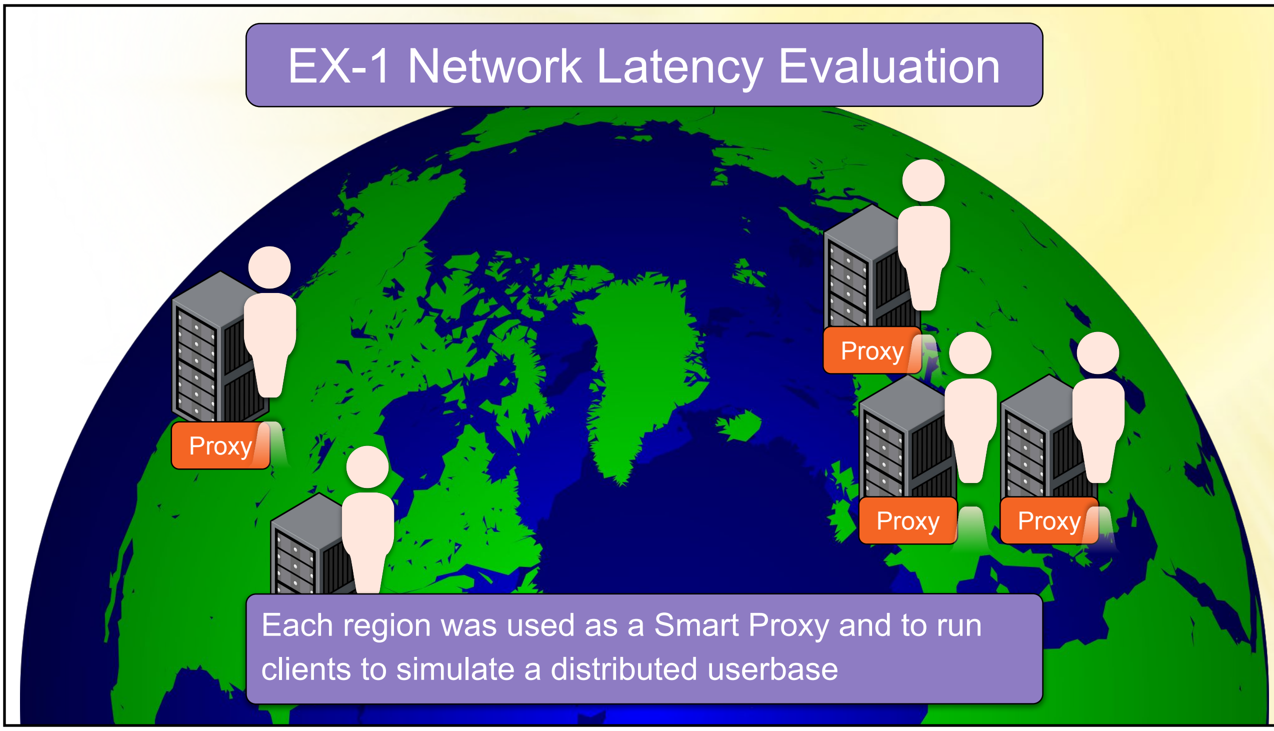
- Background and Motivation
- Research Questions
- Serverless Proxy System
- ➤ Methodology and Results
- Conclusions

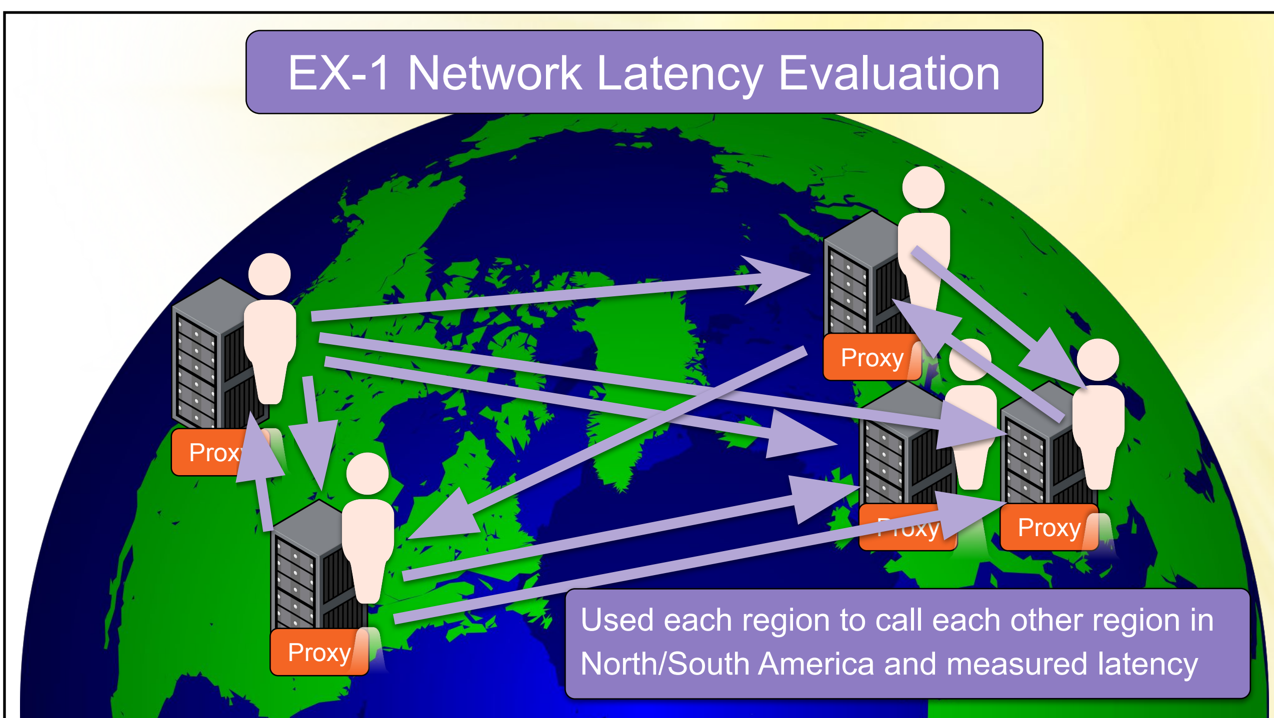
Research Question 1

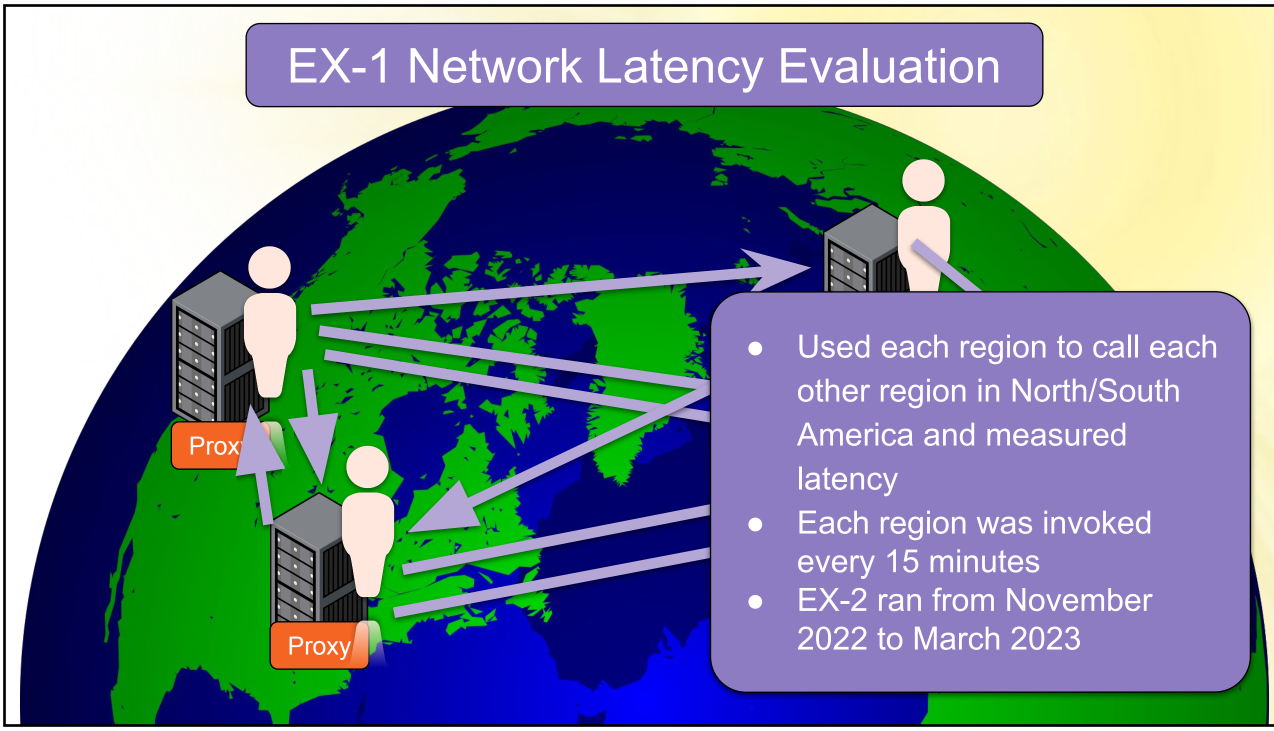
How does function network latency and runtime of a serverless application vary over time by region?

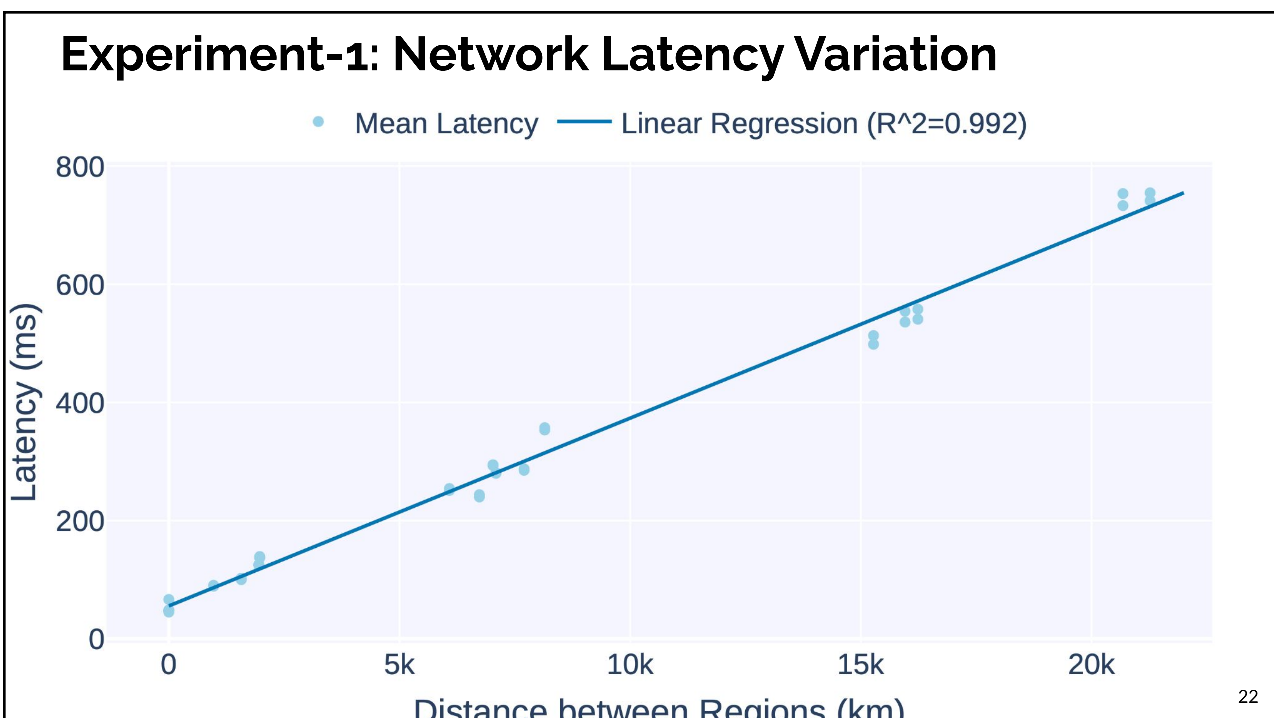
Experiment 1

- Performance and Network Latency Evaluation

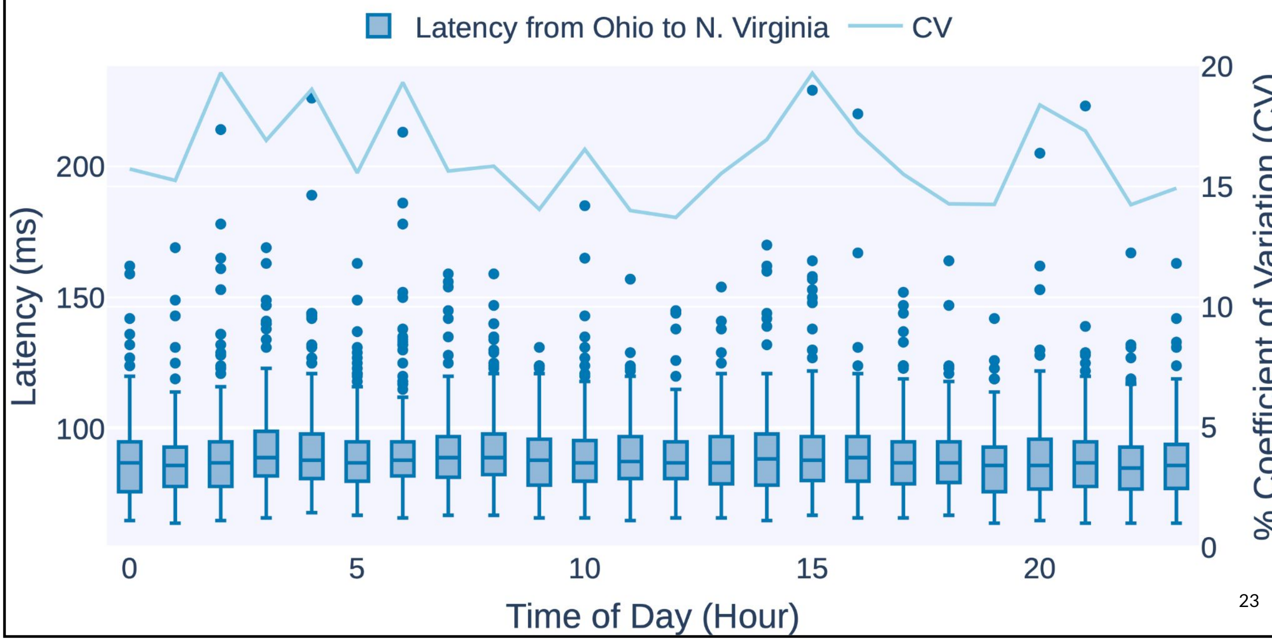

EX-1 Network Latency Evaluation
Proxy
Proxy
Proxy
Proxy
Each region was used as a Smart Proxy and to run clients to simulate a distributed userbase


EX-1 Network Latency Evaluation
Proxy
Proxy
Proxy
Used each region to call each other region in North/South America and measured latency


EX-1 Network Latency Evaluation
Used each region to call each other region in North/South America and measured latency
Each region was invoked every 15 minutes
EX-2 ran from November 2022 to March 2023
Proxy
Prox


Experiment-1: Network Latency Variation
Mean Latency
Linear Regression (R^2=0.992)
Latency (ms)
800
600
400
200
0
0
5k
10k
15k
20k
Distance between Regions (km)
22

Experiment-1: Network Latency Variation

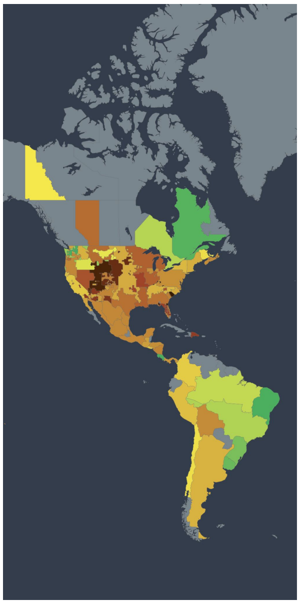


Research Question 2

How is the carbon intensity of a serverless application impacted by different cloud aggregations? How does the carbon intensity of cloud regions change over time?

Experiment 2

- Carbon Data Collection



Electricity Maps API

Electricity Maps is a leading resource for up-to-date electricity and carbon emissions data and is utilized by major corporations such as Google, Microsoft, and Cisco.

We estimated Carbon Intensity of a Serverless workload using Fossil Fuel Gigabyte Seconds:

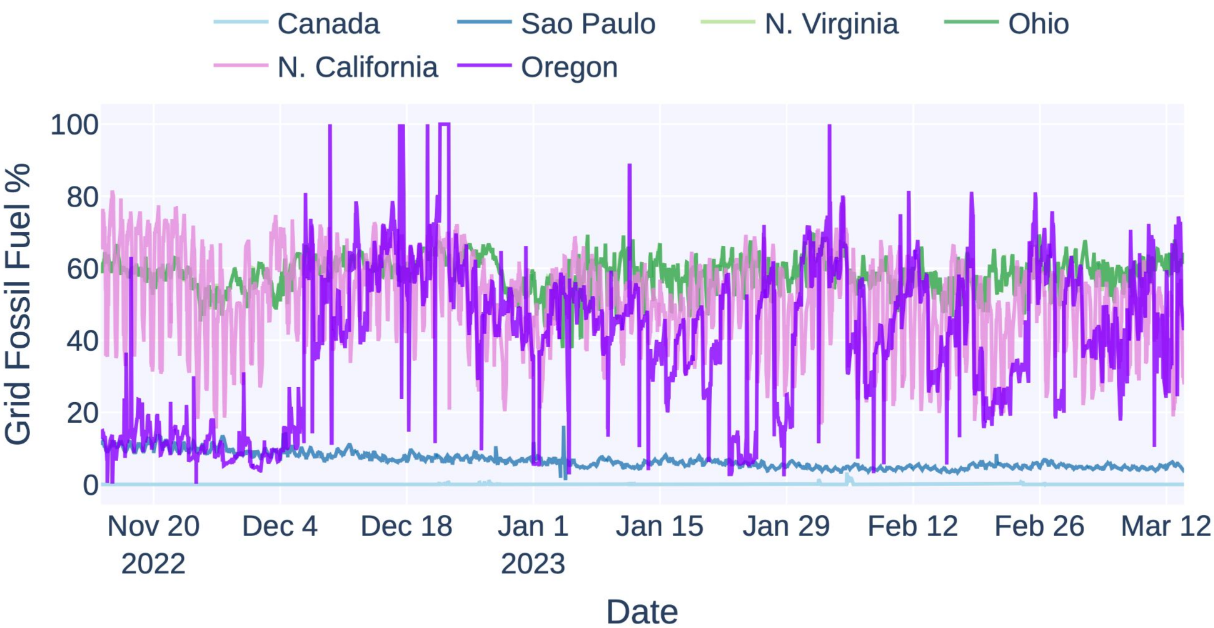
$$FFGBS = Runtime_{sec} * Memory_{GB} * FossilFuel_{\%}$$

1. Hong Kong
2. Tokyo
3. Seoul
4. Osaka
5. Mumbai
6. Singapore
7. Sydney
8. Frankfurt
9. Stockholm
10. Milan
11. Ireland
12. London
13. Paris
14. Canada
15. Sao Paulo
16. N. Virginia
17. Ohio
18. N. California
19. Oregon

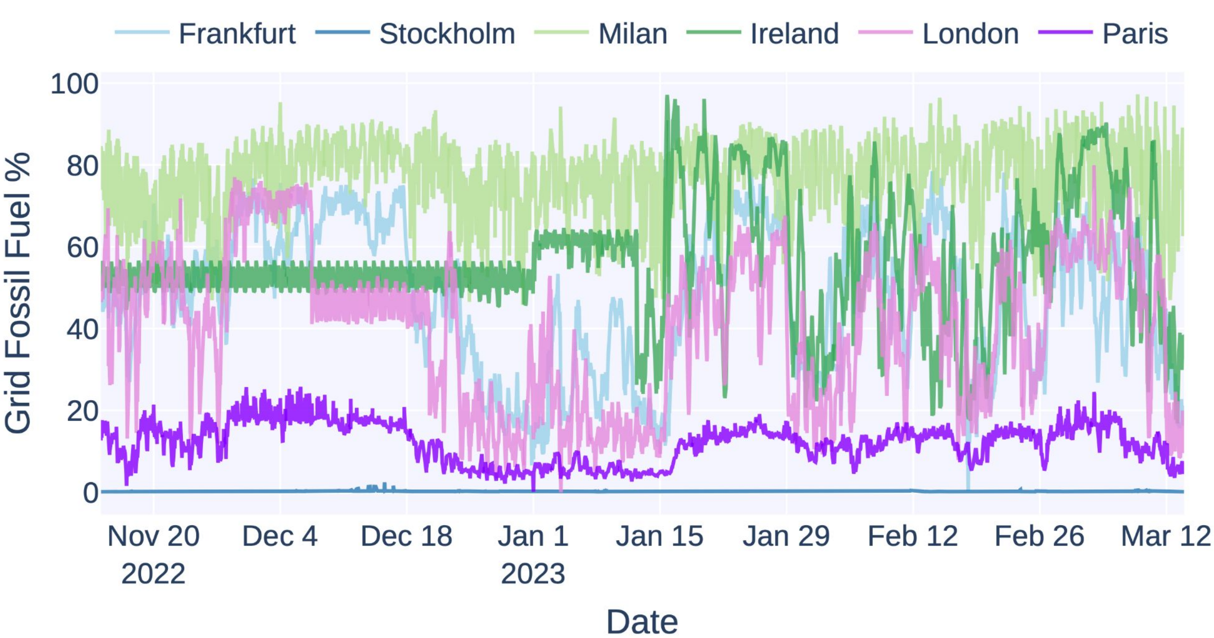
Experiment 2 - Carbon Data

1. From November 2022 to March 2023 the Analyzer function collected carbon information from Electricity Maps
2. Data was collected from 19 regions on AWS
    a. This represents every location on AWS that Electricity Maps had data for

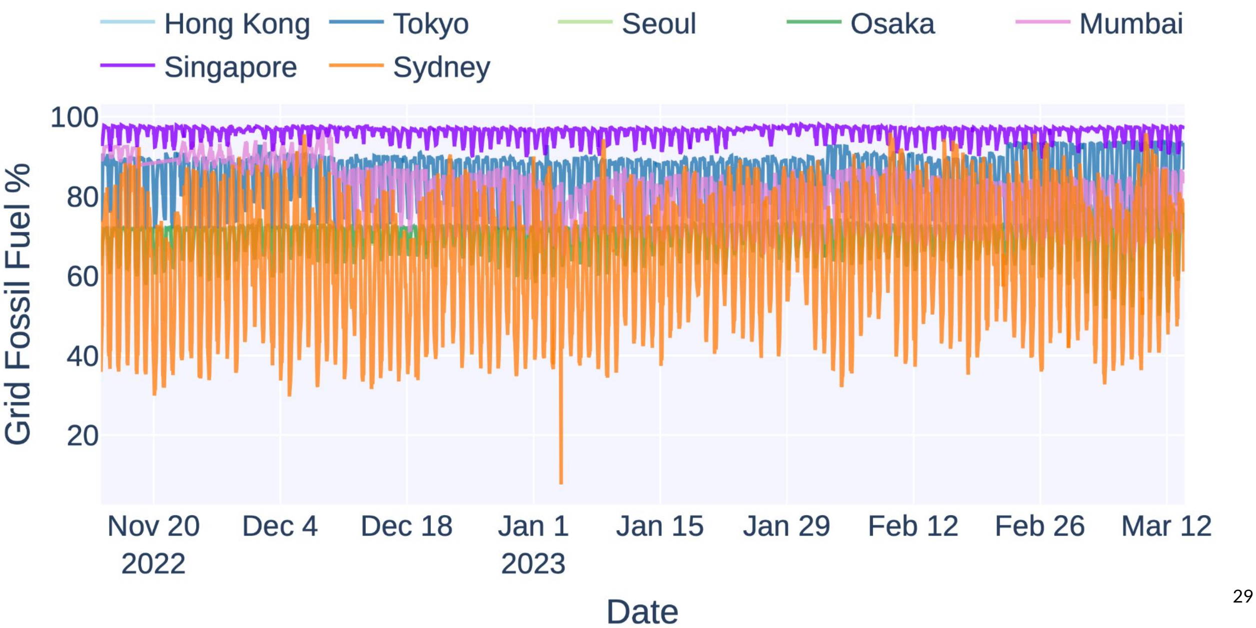
Experiment-2: Carbon Variation (Americas)



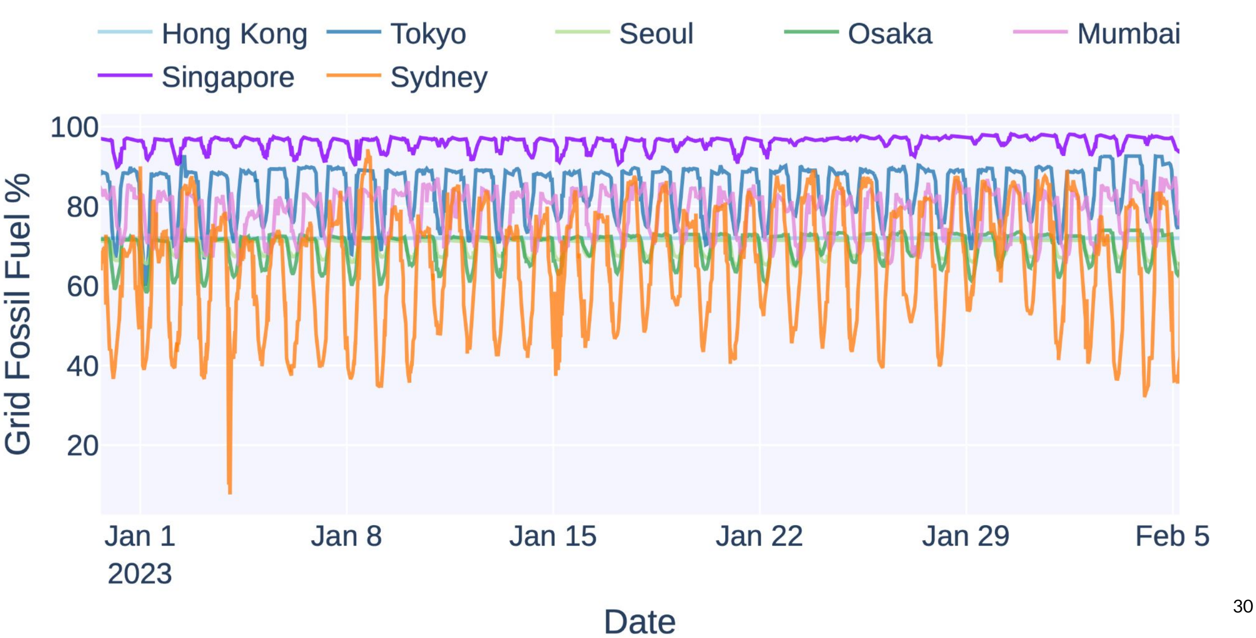
Experiment-2: Carbon Variation (Europe)



Experiment-2: Carbon Variation (Asia/Oceania)



Experiment-2: Carbon Variation (Asia - 1 Month)



Research Question 3

What are the latency and performance implications of minimizing the carbon footprint of a serverless application through carbon-aware load distributions?

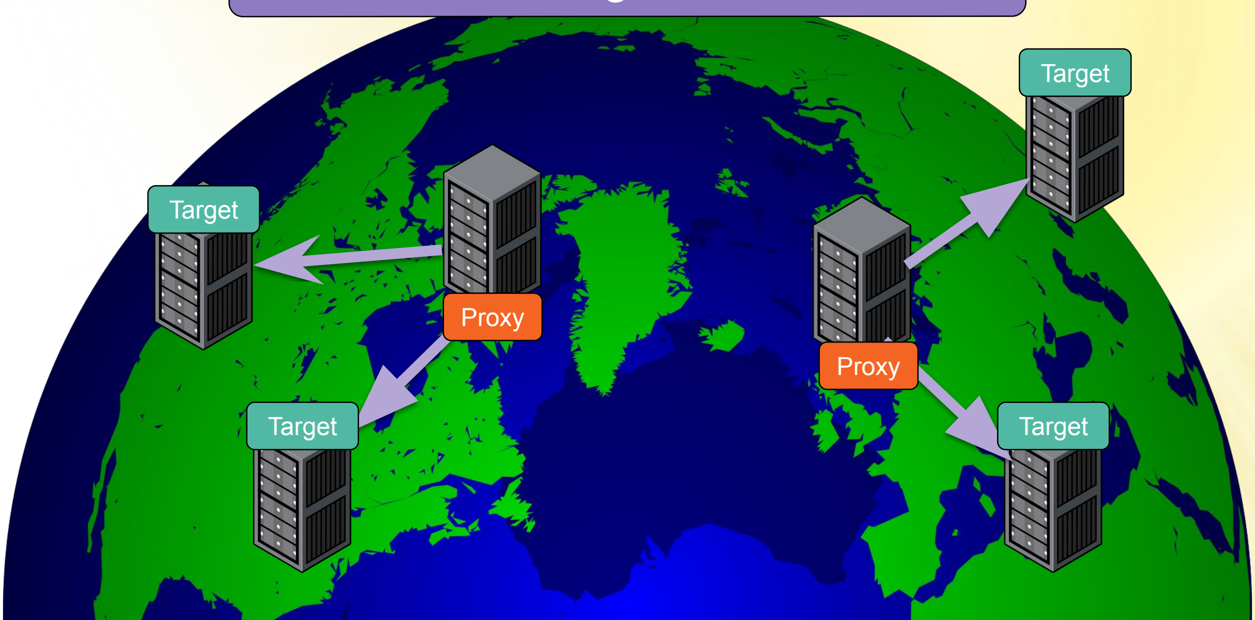
Experiment 3

- Dual-region Load Distribution

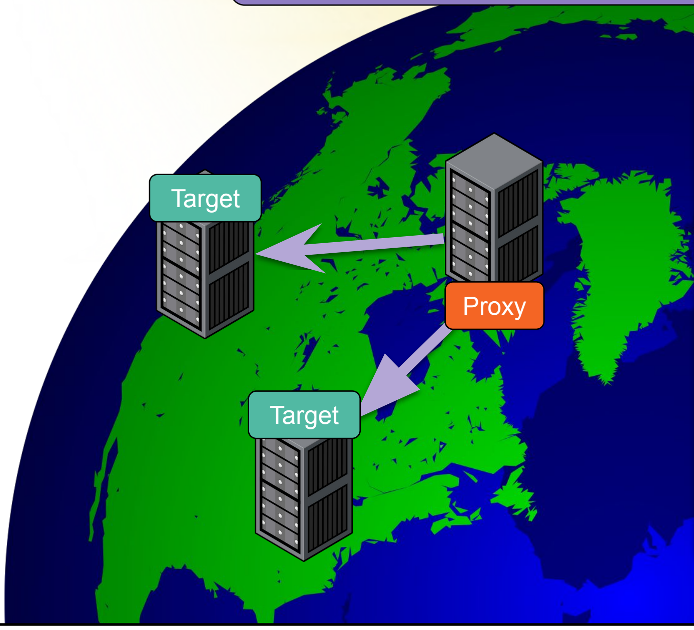
Experiment 4

- Global Load Distribution

EX-3 Dual-Region Distribution

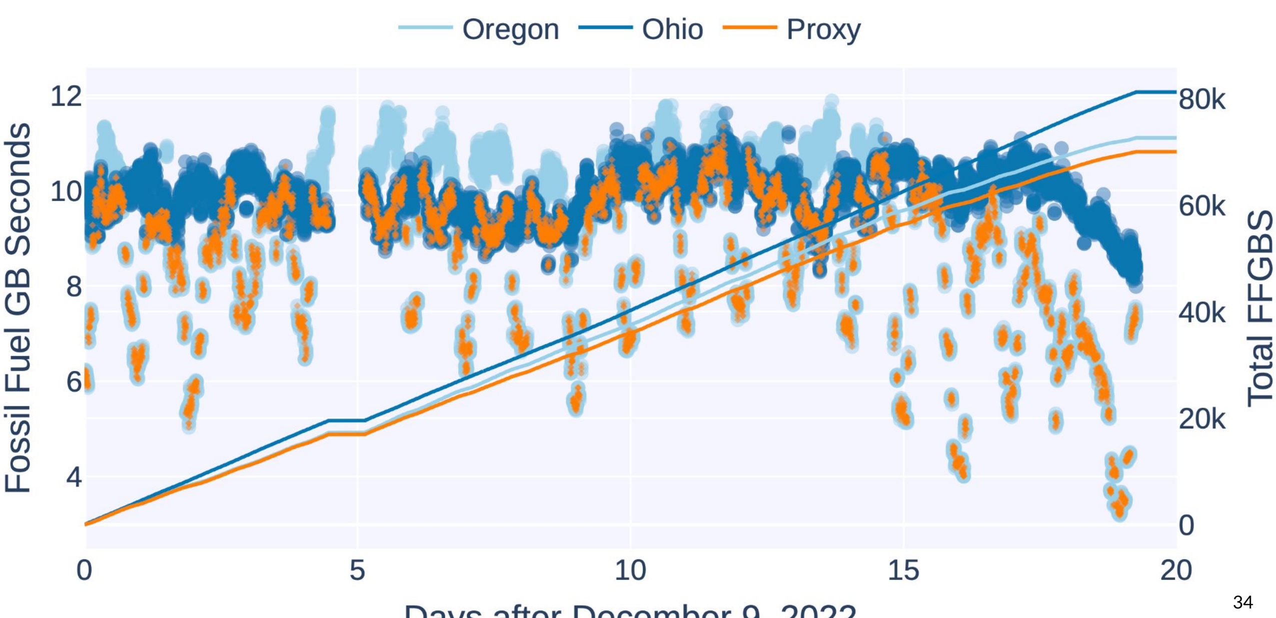


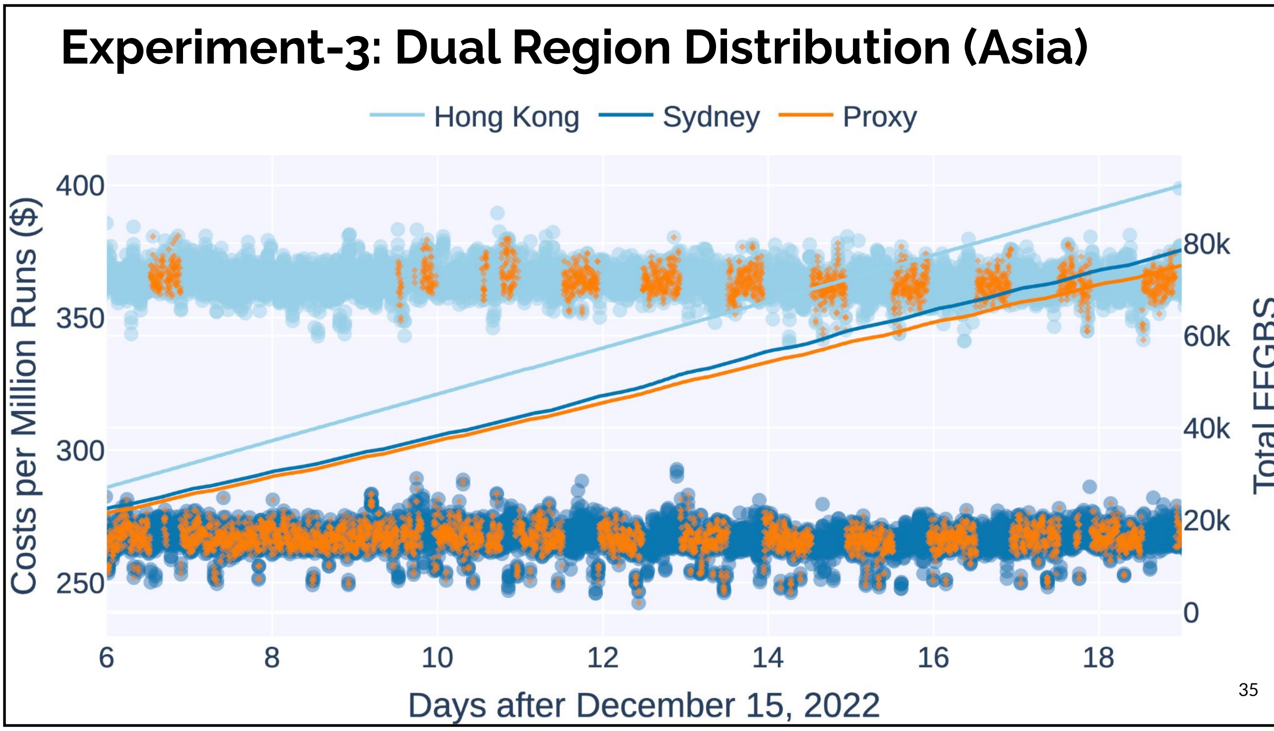
EX-3 Dual-Region Distribution

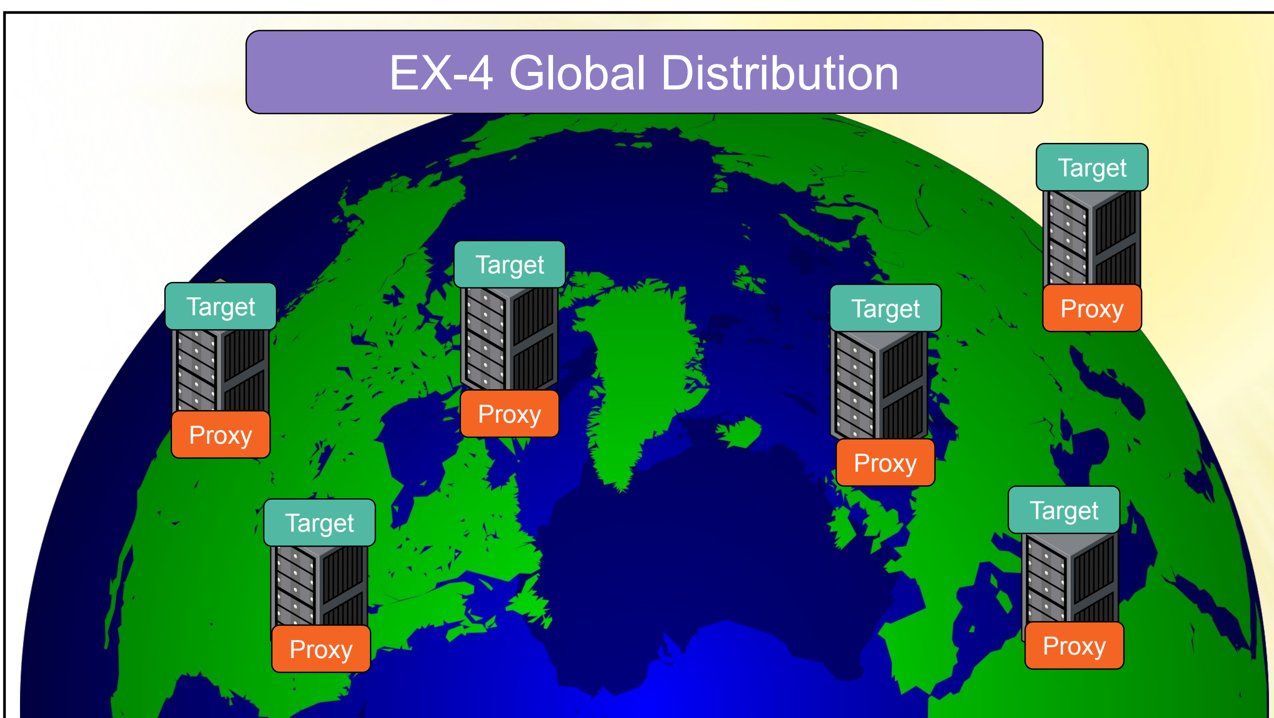


- Divided all AWS regions into American, European, and Asian aggregations.
- We picked two regions in each aggregation to act as the target region.
- Each other region in the aggregation was used as client/proxies to route requests to the target regions
- We measured the impact on carbon intensity…

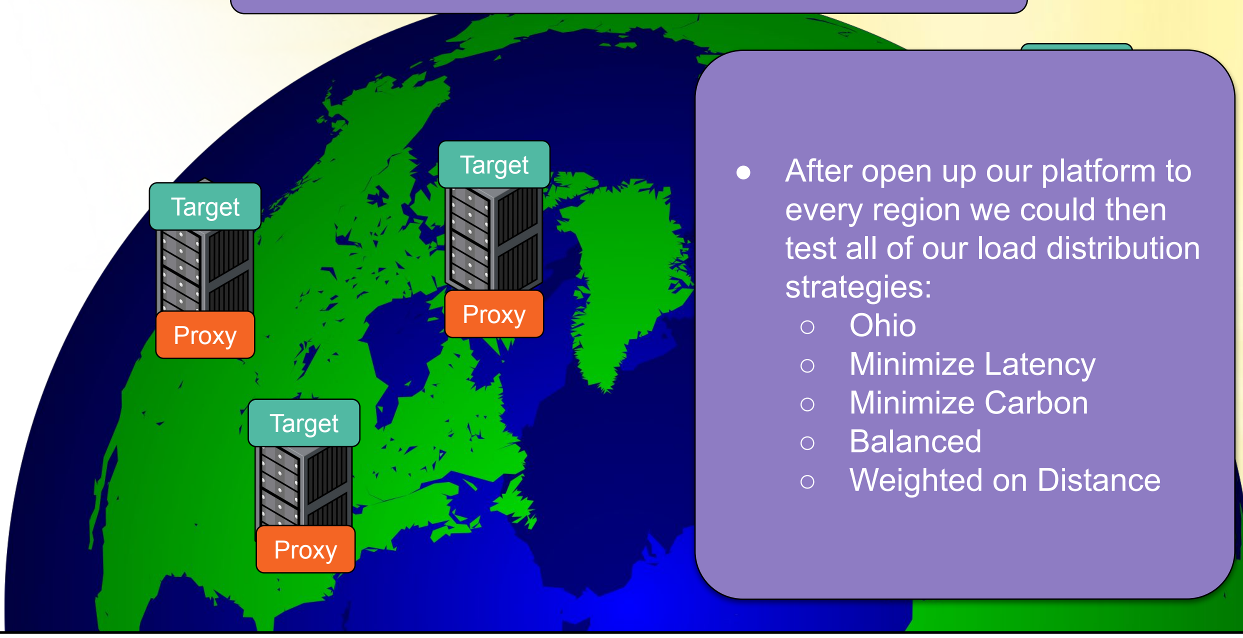
Experiment-3: Dual Region Distribution (Americas)




Experiment-3: Dual Region Distribution (Asia)
Hong Kong
Sydney
Proxy
Costs per Million Runs ($)
400
350
300
250
Total FFGBS
80k
60k
40k
20k
0
6
8
10
12
14
16
18
Days after December 15, 2022
35


EX-4 Global Distribution
Target
Proxy
Target
Proxy
Target
Proxy
Target
Proxy
Target
Proxy
Target
Proxy

EX-4 Global Distribution



- After open up our platform to every region we could then test all of our load distribution strategies:
  - Ohio
  - Minimize Latency
  - Minimize Carbon
  - Balanced
  - Weighted on Distance

Experiment-4: Global Load Distribution

Name	Regions Used	Average Latency	Latency CV	Average FF-GBS	Cost Per 1m
Ohio	1	474	50	568,000	$65.25
Minimize Carbon	2	600	49	128	$64.64
Minimize Distance	12	166	72	560,000	$67.01
Weighted Evenly	2	516	70	134	$64.05
Weighted Distance	6	489	71	440	$64.64

Experiment-4: Global Load Distribution

Name	Regions Used	Average Latency	Latency CV	Average FF-GBS	Cost Per 1m
Ohio	1	474	50	568,000	$65.25
Minimize Carbon	2	600	49	128	$64.64
Minimize Distance	12	166	72	560,000	$67.01
Weighted Evenly	2	516	70	134	$64.05
Weighted Distance	6	489	71	440	$64.64

Experiment-4: Global Load Distribution

Name	Regions Used	Average Latency	Latency CV	Average FF-GBS	Cost Per 1m
Ohio	1	474	50	568,000	$65.25
Minimize Carbon	2	600	49	128	$64.64
Minimize Distance	12	166	72	560,000	$67.01
Weighted Evenly	2	516	70	134	$64.05
Weighted Distance	6	489	71	440	$64.64

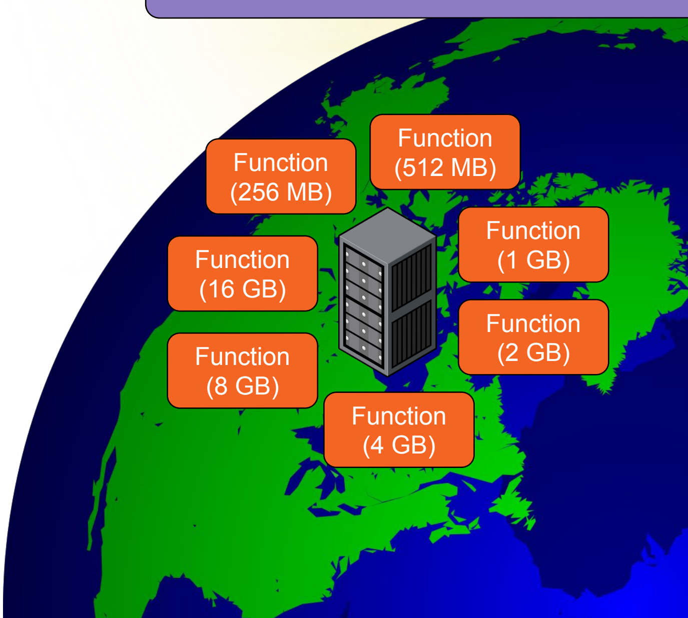
Research Question 4

How can serverless resource aggregation be leverages to reduce application hosting costs by utilizing function deployments with many different configurations?

Experiment 5

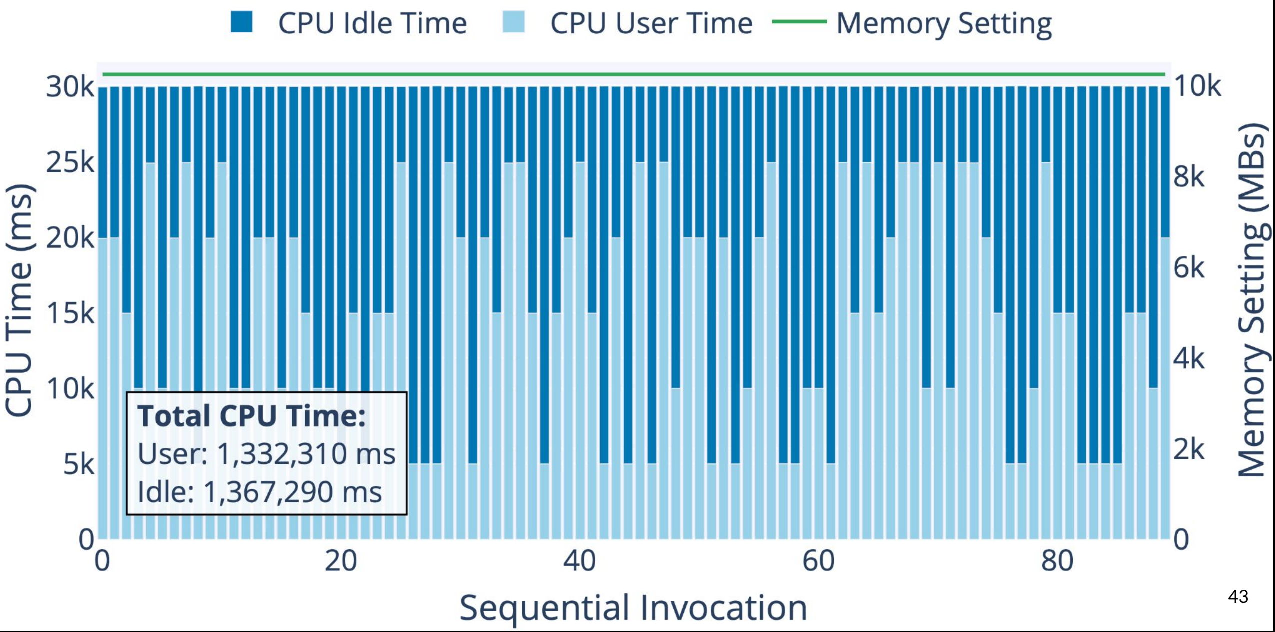
- Performance-based Load Distribution

EX-5 Performance Based Load Distribution

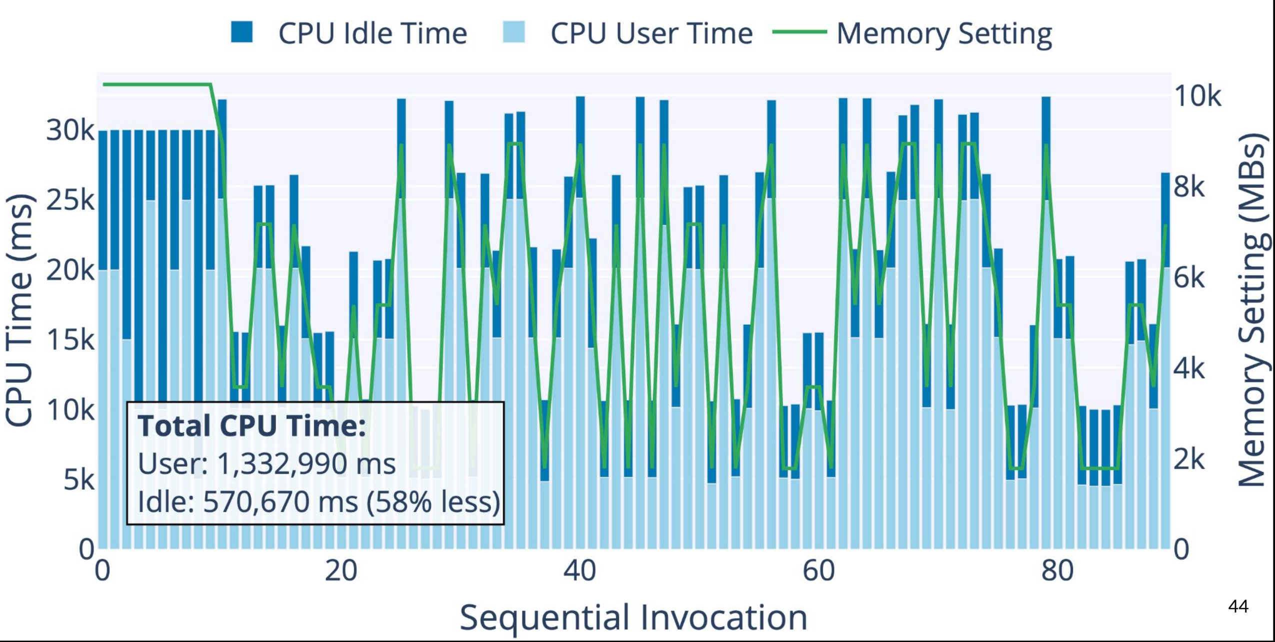


- Instead of using a Proxy to distribute between regions, we can distribute between multiple deployments of the same function with different configurations
- We can then optimize function runtime and cost using the CPU-TAMS model
- The proxy function predicts optimal memory setting based off function request parameters

Experiment-5: Performance Based Distribution

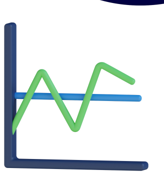


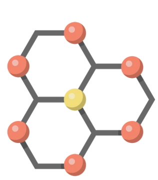
Experiment-5: Performance Based Distribution



Outline

- Background and Motivation
- Research Questions
- Serverless Proxy System
- Methodology and Results
- Conclusions





Conclusion Summary

- We executed large experiments using 19 regions on AWS over than ran continuously for 6 months.
- RQ-1: We observed latency variation of 2-29% CV during the day, averaging +/-10ms. Distance was a strong predictor of latency with R^2 of 0.992
- RQ-2: Each region had varying carbon intensity with Canada and Stockholm regions having almost no fossil fuel usage.

Conclusion Summary

- RQ-3: The serverless proxy on a global distribution was able to reduce carbon intensity by up to 99.8% while also reducing latency by 65% compared to a single region deployment.

- RQ-4: By utilizing multiple configurations of the same function we were able to reduce runtime and hosting costs by 58%.

Thank You!

This research has been supported by AWS Cloud Credits for Research.