EdgeBench: Benchmarking Edge Computing Platforms

*Anirban Das**, Stacy Patterson*, Mike P. Wittie^
*Department of Computer Science, RPI
Networked Systems Lab
^Gianforte School of Computing, Montana State University

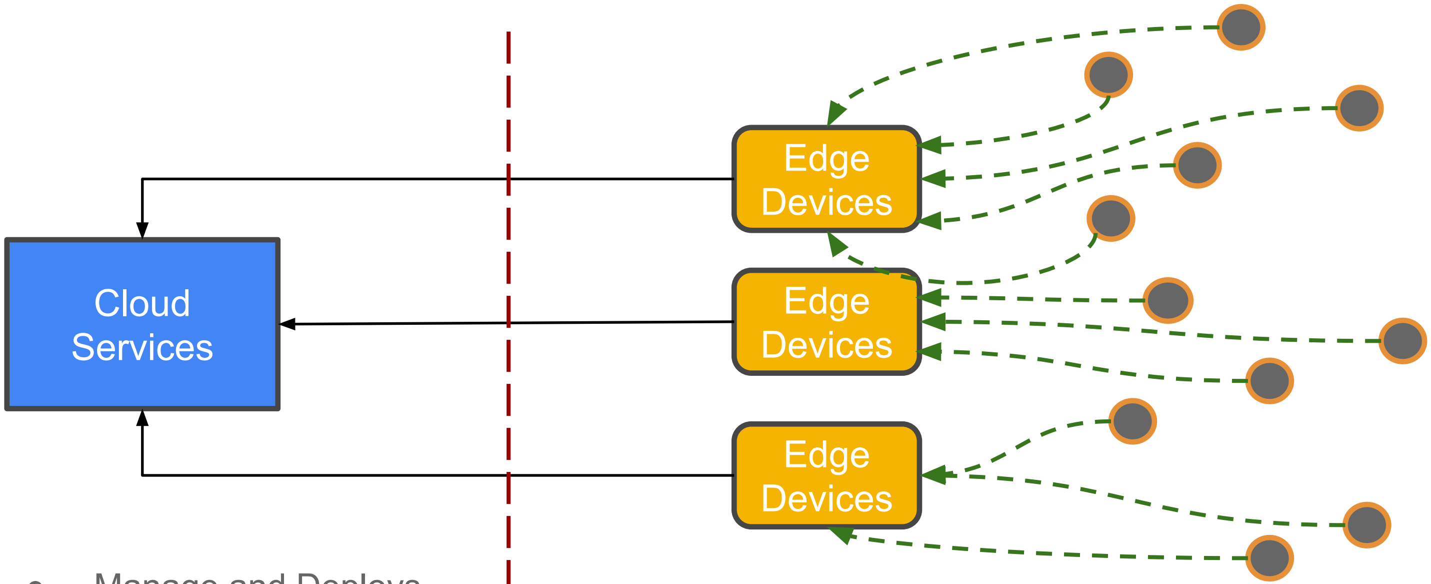
Rensselaer
why not change the world?®

12/20/2018

Roadmap

- Motivation
- Related Work
- System Architectures
- Pipelines and Applications/ Workloads
- Experimental Results
- Conclusion

What is edge computing?



- Manage and Deploys Edge Modules
- Provide connectors to other services

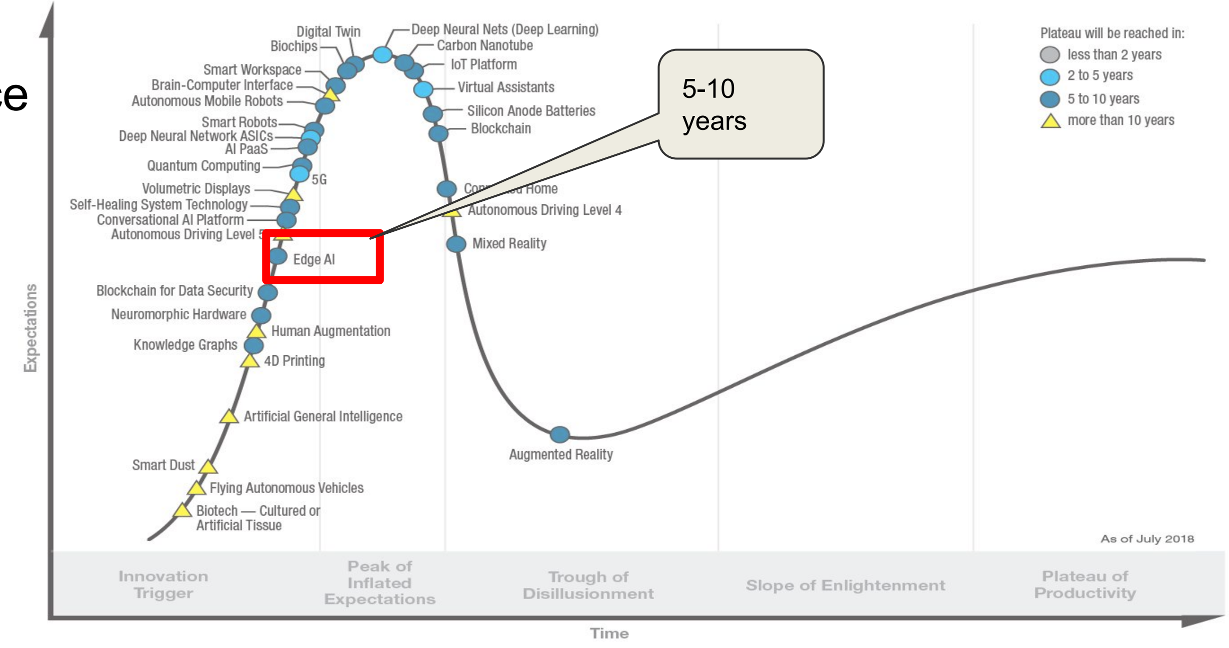
- Partial or Full computation

Sensors / Data sources
Sensor data Data flow

Motivations

- Ubiquitous intelligence
- Speech detection
- Image recognition
- Sensor data stream
- Autonomous cars
- Augmented Reality

Hype Cycle for Emerging Technologies, 2018



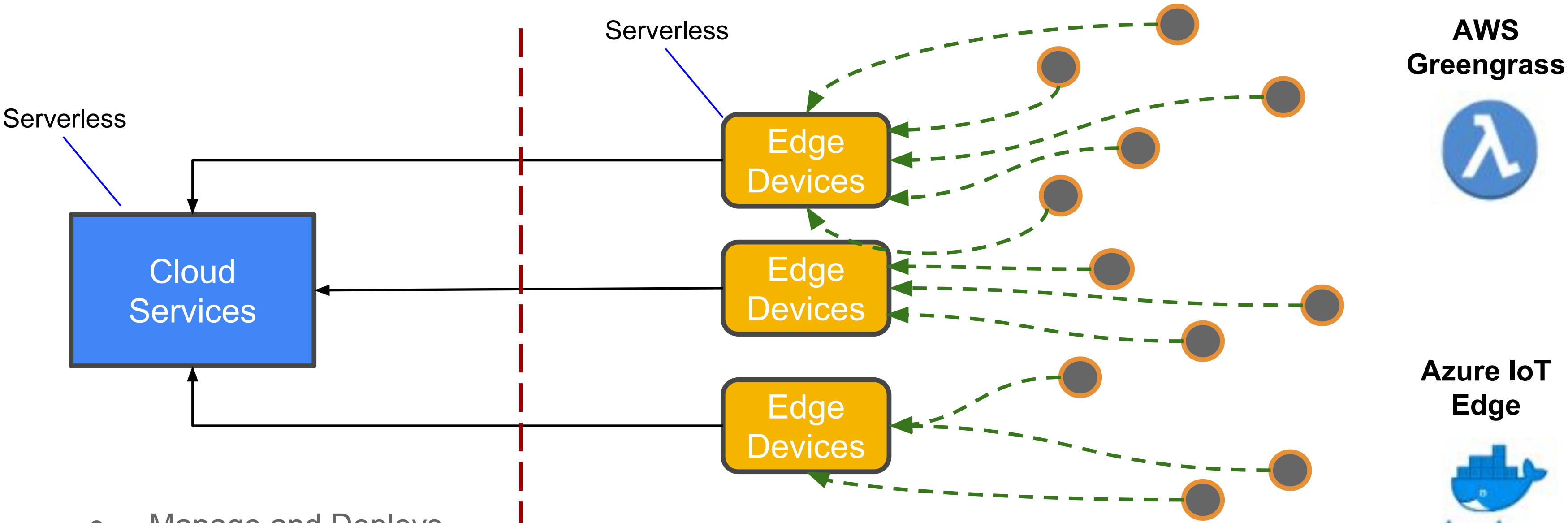
gartner.com/SmarterWithGartner

Source: Gartner (August 2018)

Gartner

Ref: https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/

How serverless fits in the picture?



- Manage and Deploys Edge Modules
- Provide connectors to other services

- Partial or Full computation

Sensors / Data sources

Sensor data Data flow

Related Work (Cloud-Only Serverless Benchmarks)

Big 4: AWS Lambda, Azure Functions, GCF, IBM Openwhisk

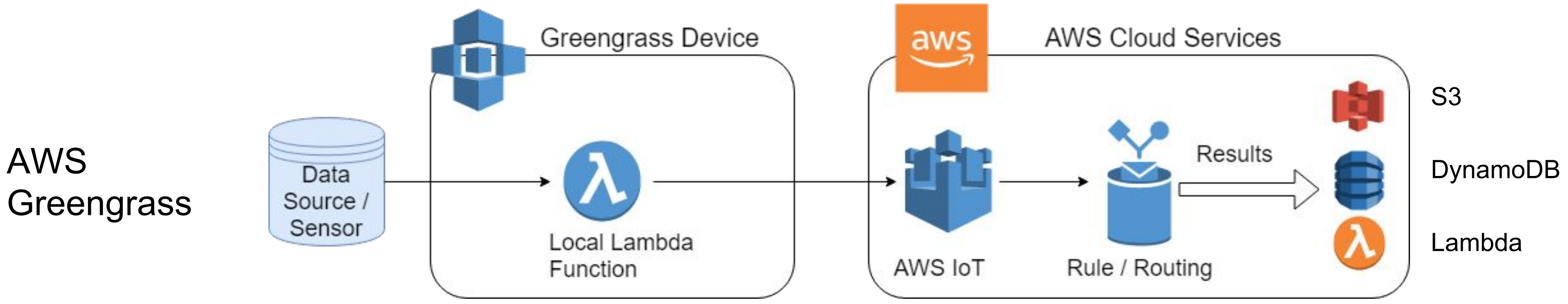
- CPU intensive benchmarks using Serverless and Hyperflow
  - Malawski et. al., 2017
- Azure based prototype for performance oriented serverless and measures performance using custom made tool
  - McGrath and Brenner, 2017
- Propose a micro benchmark for cost and performance modeling
  - Back and Andrikopoulos, 2018
- Provides a real world example of running k-Means clustering on AWS Lambda
  - Deese, 2018

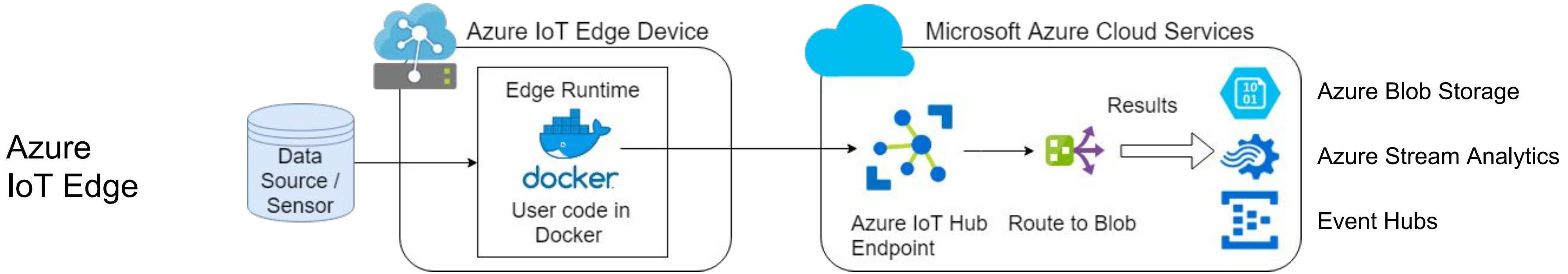
Research Questions and Contributions

- Need to compare vendors in Edge Computing
- Need to compare edge architectures with cloud only architectures
- Feasibility of edge architectures

- Contributions:
  - Developed benchmark EdgeBench
  - Developed benchmarking methodologies and metrics of interest
  - Developed applications based on real world use cases
  - Studied two platforms / industry vendors:
    - **AWS Greengrass**
    - **Microsoft Azure IoT Edge**

System architectures of AWS Greengrass and Azure IoT Edge



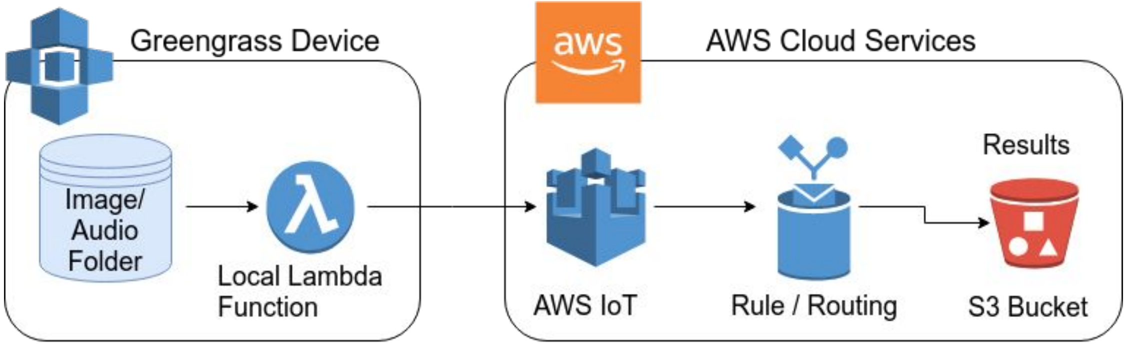


Benchmark Applications

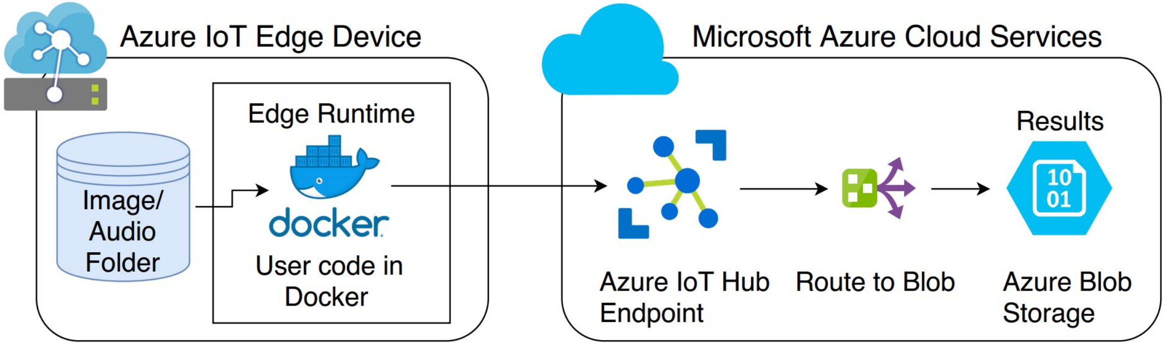
- Canonical applications from real-world use cases
  - Scalar Sensor Emulator:
    - Extremely light-weight workload - A scalar sensor value generator
  - Image Classification:
    - A representative workload from the image processing/ classification domains like autonomous cars, AR
  - Speech to Text Decoding/Translation:
    - An edge use-case of speech to text decoding inspired from the popularity of Amazon Echo and Google Home

Edge Pipelines for Benchmark Applications

AWS Greengrass



Azure IoT Edge



Edge Pipelines for Benchmark Applications

Image Classification/ Object Recognition

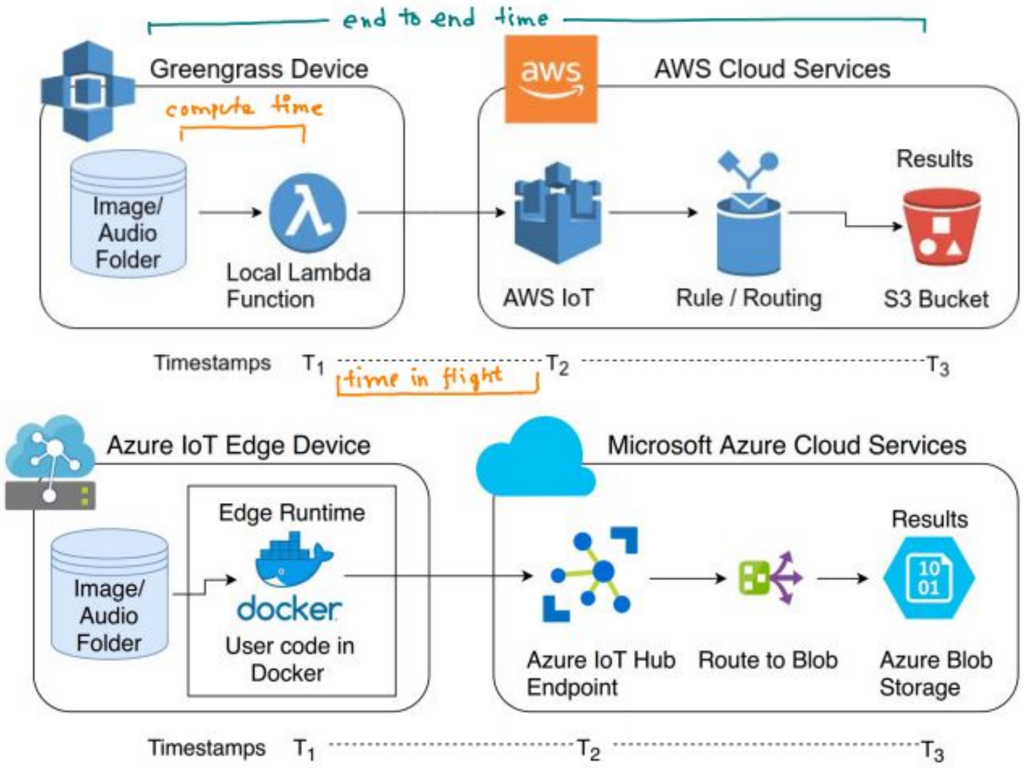
- Python
- MXNet framework (Squeezenet)
- Workload: Imagenet 2012 dataset

Speech to Text

- Python
- PocketSphinx: Python Port: (https://github.com/bambocher/pocketsphinx-python)
- Workload: Samples from Tatoeba Database from Mozilla Common Voice platform

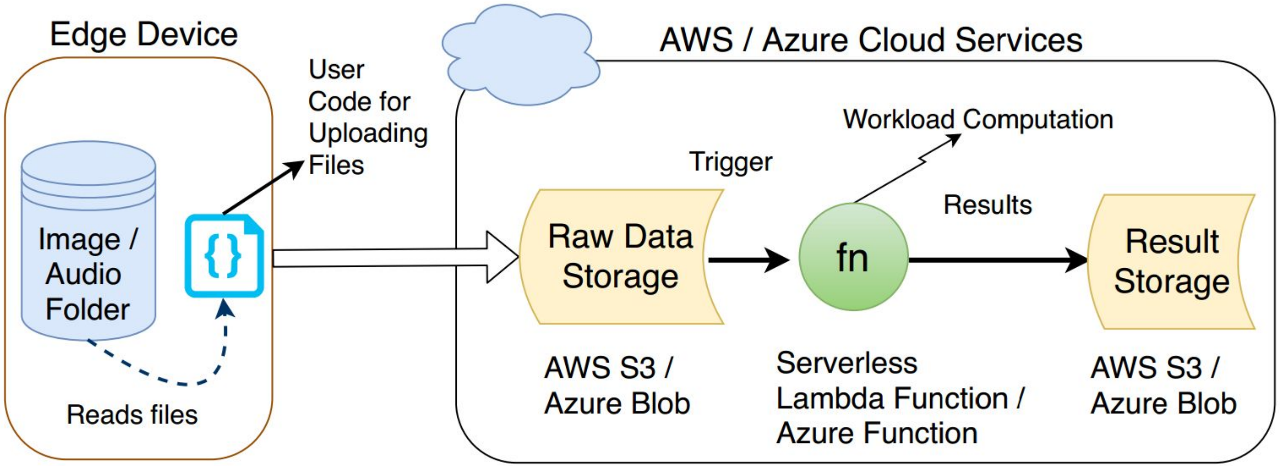
Metrics for Edge

- 3 UTC timestamps:
  - T_1 at the edge
  - T_2 at IoT Hub
  - T_3 at S3/Blob
- Feasibility of edge device
  - Compute time
  - Memory and CPU utilization
- Feasibility of applications
  - Time in Flight / Flight time
  - End to End Latency
- Bandwidth Savings
  - Payload Size



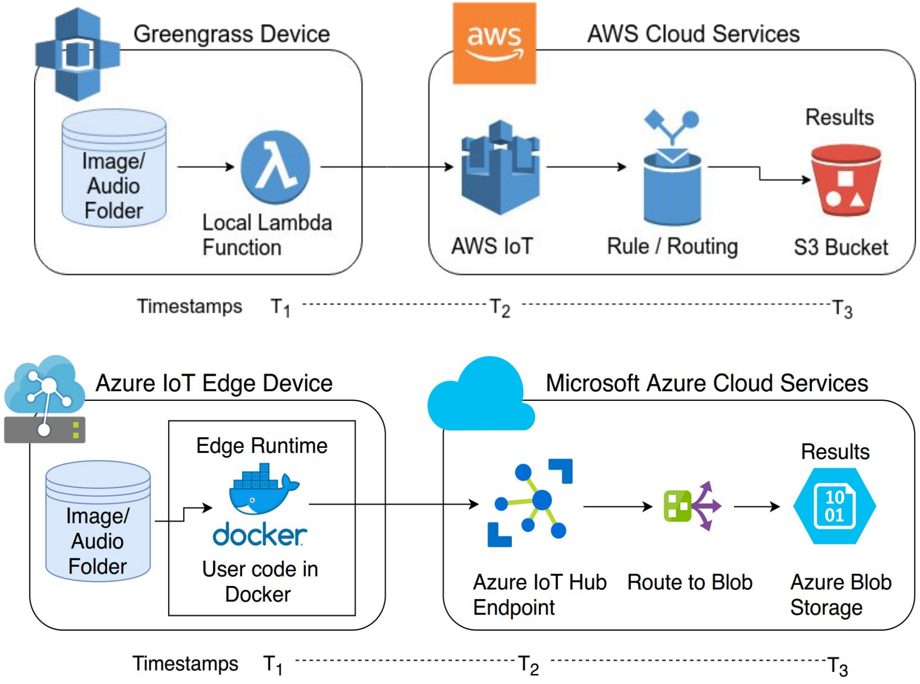
Cloud Pipelines for Benchmark Applications

- Files send sequentially (10-15 s delay)
- Lambda memory at 3008 MB and Azure Consumption Host Plan
- Metric: End to End Latency



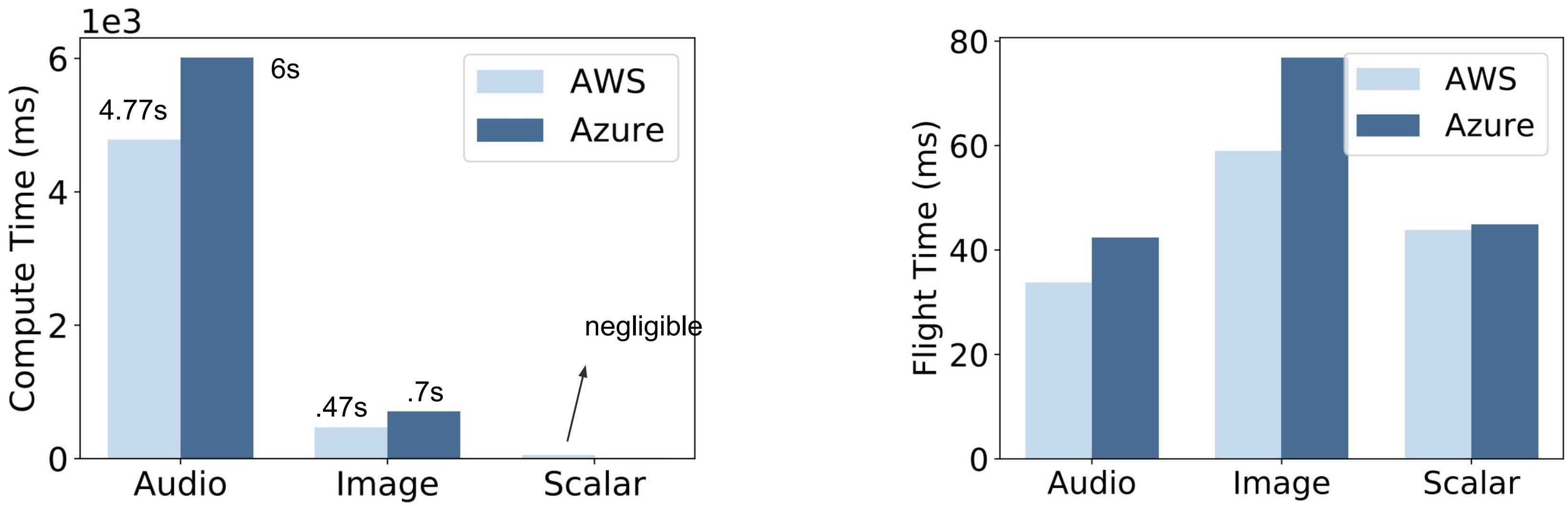
Experimental Setup

- Raspberry Pi 3B
- TM 2000A Stratum 1 for time sync
- Azure and AWS locations US East North Virginia
- Local Lambda Long running
- GGC Core 1.5.0, Azure IoT Hub device client 1.4.0.



Results - Compute Time and Flight Time

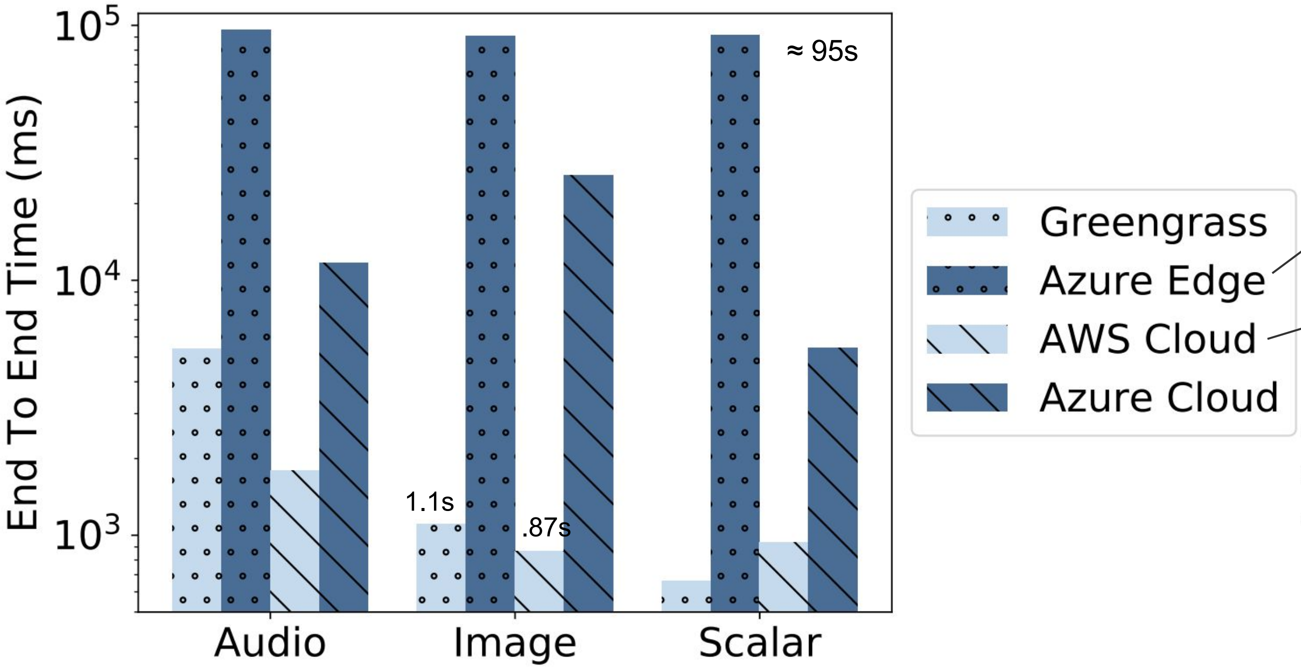
Edge Only Pipelines



- Image Recognition Sub second, Audio slow

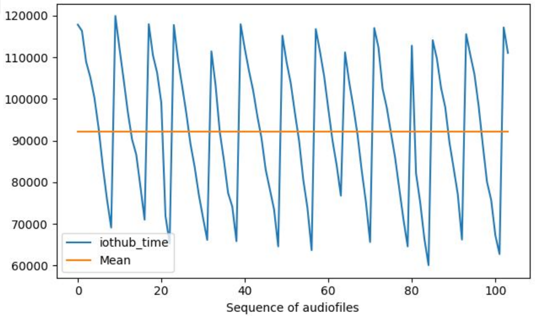
Results - End to End Time

Both Edge and Cloud



Slowest, due to batching

Fastest



Results - Bandwidth Usage

		Total Input Size (Mbytes)	Total raw Payload Size (Mbytes)	Total MBytes Transmitted in Network	
				AWS	Azure
Audio Trials = 104	Edge	8.83	0.02	0.25	0.26
	Cloud		8.83	9.06	9.09
Image Trials = 500	Edge	71.69	0.38	0.9	0.96
	Cloud		71.69	73.10	73.49
Scalar Trials = 200	Edge	0.05	0.05	0.33	0.26
	Cloud			0.47	0.38

- Massive reduction in BW usage in cloud vs edge pipelines:
  - AWS: 36x in audio and 81x in image
  - Azure: 36x in audio and 77x in image

- Average *single* payload size for edge apps:

Image: 752 bytes	Audio: 162 bytes	Scalar: 234 bytes

Rough Infrastructure Cost Estimate (August 2018)

- Image Pipeline: 1 traffic camera, image every 10 second for 1 month
- Input data size : 259,200 x 143 KB
- Cloud Config: 3008 MB Lambda

- Cost:
  - Greengrass : ≈ 1.56 USD / month
  - AWS Cloud Solution : ≈ 8.027 USD / month

- Cloud solution 5.3x more expensive at least.

- Data Transfer:
  - Greengrass : 253 MB
  - AWS Cloud Solution : 35.4 GB

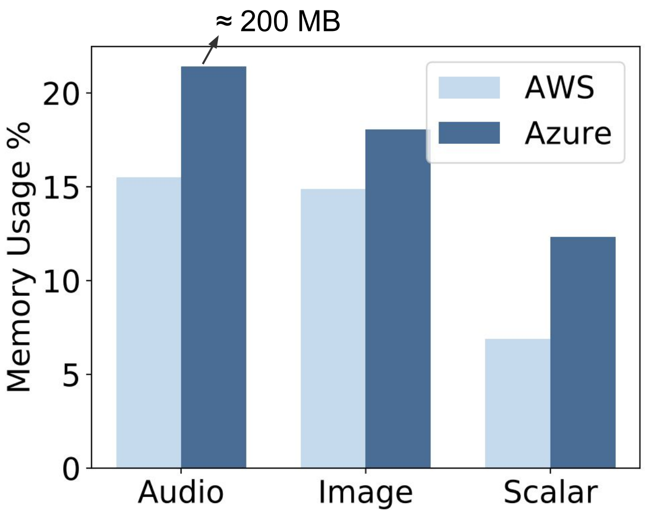
Conclusion

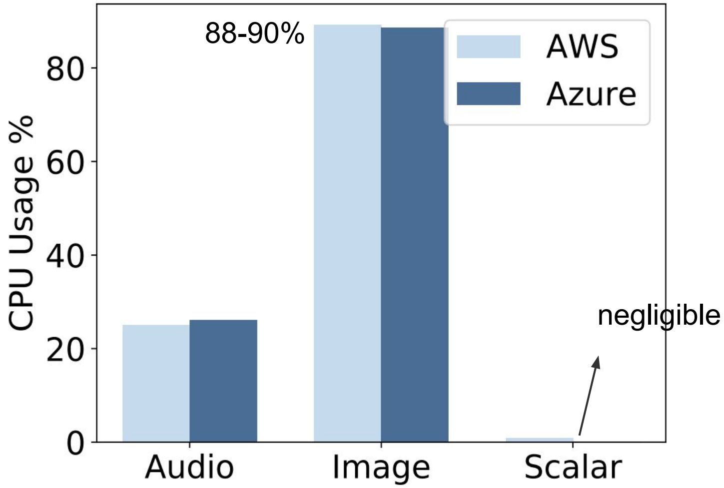
- Presented EdgeBench (https://github.com/akaanirban/edgebench)
  - Methodologies, Applications, Performance on Greengrass and Azure IoT Edge
- Our results show:
  - Performance on the edge comparable for both platforms
  - Cloud is faster than edge
  - Bandwidth saving is massive using edge architectures
- Is one platform better than the other?
  - Depends on use case for e.g., batching vs event based
- Future work:
  - Expanding into Google and IBM's products
  - Expand study with different model sizes and applications
  - Standardize deployment procedure (open problem)
    - Need for frameworks like Serverless for homogeneity
    - Greengo for Greengrass (https://github.com/dzimine/greengo)

Thank You

Extra Slides

Results - Resource usage on Pi





Edge Only Pipelines

Feature Comparison

	AWS Greengrass	Azure IoT Edge
Runtime	Python 2.7, Node.JS 6.10, Java 8, C, C++	C#, C, Node.JS ver > 0.4.x.x, Python (both 2.7 and 3.6), and Java 7+
Deployment Method	Lambda Functions ● Greengrass Containerized Non Containerized (as of ggc core 1.7) ● Install heavier libraries directly on Raspberry Pi	Docker Containers ● Orchestrated using Moby ● Can package anything in Containers
Triggers Routes available	15 (e.g. S3, Dynamo DB, Lambda, Cloudwatch logs, SNS, Step Functions etc.)	4 (e.g.Blog Storage, Event Hub, Service Bus Queue, Service bus topic) (Can directly deploy Azure ML models and ASA jobs into IoT Edge)
Parallel Execution	Parallel Lambdas can be triggered to run locally	N/A
Deployment	boto3, aws-cli	azure-cli, VSCode

(December 2018)

Azure / **azure-iot-sdk-python**

Watch 81 | Unstar 128 | Fork 138

Code | Issues 26 | Pull requests 1 | Projects 0 | Wiki | Insights

Releases | Tags

Latest release

release_2018_10...

568d7f9

Azure IoT SDKs for Devices v1.4.4

pierreca released this on Oct 31 · 4 commits to master since this release

Assets 2

Source code (zip)

Source code (tar.gz)

We are snapping to the latest LTS (Long-Term Support) version of the C SDK and therefore declaring our 1.4.x series of update the LTS branch.

We will be publishing new features under the 1.5.x denomination in the future.

This is also a good time to let everybody know that we've seen and heard the feedback loud and clear about the many pains caused by having to wrap the C SDK using Boost (and the ensuing platform incompatibilities). At that point we've started a complete re-write of the SDK in pure, cross platform python.

As soon as we have a partial-feature SDK preview ready we will communicate this in the readme and start redirecting new users to the preview of the v2 SDK.

Change	Description	Date
Amazon SageMaker Neo Deep Learning Runtime	The Amazon SageMaker Neo deep learning runtime supports machine learning models that have been optimized by the Amazon SageMaker Neo deep learning compiler.	November 28, 2018
Run AWS IoT Greengrass in a Docker container	You can run AWS IoT Greengrass in a Docker container by configuring your Greengrass group to run with no containerization.	November 26, 2018
AWS IoT Greengrass Version 1.7.0 Released	New features: Greengrass connectors, local secrets manager, isolation and permission settings for Lambda functions, hardware root of trust security, connection using ALPN or network proxy, and Raspbian Stretch support.	November 26, 2018
AWS IoT Greengrass Software Downloads	The AWS IoT Greengrass Core Software, AWS IoT Greengrass Core SDK, and AWS IoT Greengrass Machine Learning SDK packages are available for dowload through Amazon CloudFront.	November 26, 2018
AWS IoT Device Tester for AWS IoT Greengrass	Use AWS IoT Device Tester for AWS IoT Greengrass to verify that your CPU architecture, kernel configuration, and drivers work with AWS IoT Greengrass.	November 26, 2018
AWS CloudTrail Logging for AWS IoT Greengrass API Calls	AWS IoT Greengrass is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS IoT Greengrass.	October 29, 2018
Support for TensorFlow v1.10.1 on NVIDIA Jetson TX2	The TensorFlow precompiled library for NVIDIA Jetson TX2 that AWS IoT Greengrass provides now uses TensorFlow v1.10.1. This supports Jetpack 3.3 and CUDA Toolkit 9.0.	October 18, 2018
Support for MXNet v1.2.1 Machine Learning Resources	AWS IoT Greengrass supports machine learning models that are trained using MXNet v1.2.1.	August 29, 2018
AWS IoT Greengrass Version 1.6.0 Released	New features: Lambda executables, configurable message queue, configurable reconnect retry interval, volume resources under /proc, and configurable write directory.	July 26, 2018