Automobile Traffic Detection on the Edge: A Performance Comparison

Team 8: Tootiya Giyahchi, Ritwik Nandakumar, and Beomjun Aaron Bae

Motivation - Edge Computing

- For many IoT scenarios, a framework for data processing and data computation with as little latency and bandwidth utilization as possible, is required.
- To this end, edge computing offers a possible solution. Edge Computing reduces latency by offloading computation, to be performed on the edge itself.
- In our scenario, we require an edge device (Raspberry Pi) to run a machine learning object recognition model to detect traffic and perform the classifications on the device itself.
- This enables the system to reduce response time and improve resource utilization, which are both critical in real-time IoT environments.

Goals

- Our goal is to test and benchmark two edge computing platforms - ***Microsoft Azure IoT*** and ***Amazon AWS Greengrass***, using the same configuration and resources.
- Both these platforms function by enabling computation to be offloaded to the edge device, rather than sending it to the cloud to perform the computation, just to send it back again to the device.
- We aim to test and evaluate the performance of these processing frameworks (Azure IoT Edge, Greengrass IoT) and aim to extrapolate the performance in classifying the image dataset (In our case CIFAR10 dataset) for each instance.

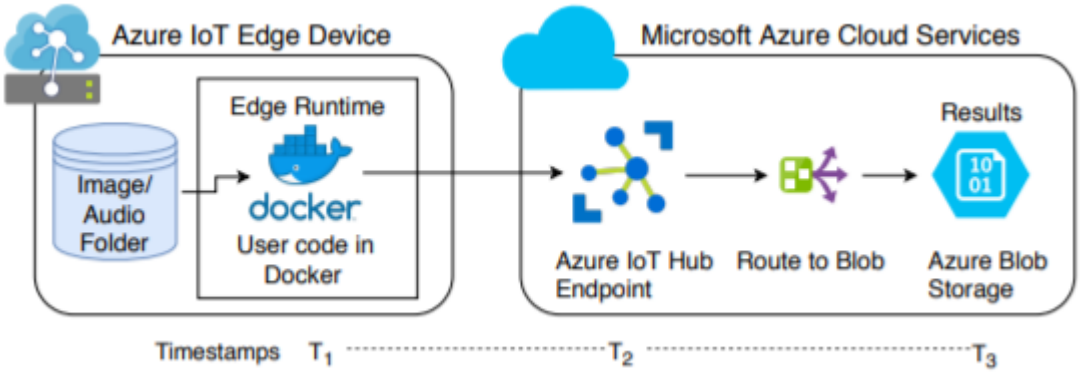
Related Work

- The paper entitled “Towards a Methodology for Benchmarking Edge Processing Frameworks” offers a preliminary benchmark methodology for any given edge processing platform. This aids us in creating evaluation criterion, benchmark objectives and an edge workflow.

- “EdgeBench: Benchmarking Edge Computing Platforms” offers an open-source benchmark tool to compute and compare performance parameters for Azure and Greengrass. We will utilize parts of this system tool to evaluate our own setup but with a different application scenario in mind, as well as our own custom hardware resources and image dataset.

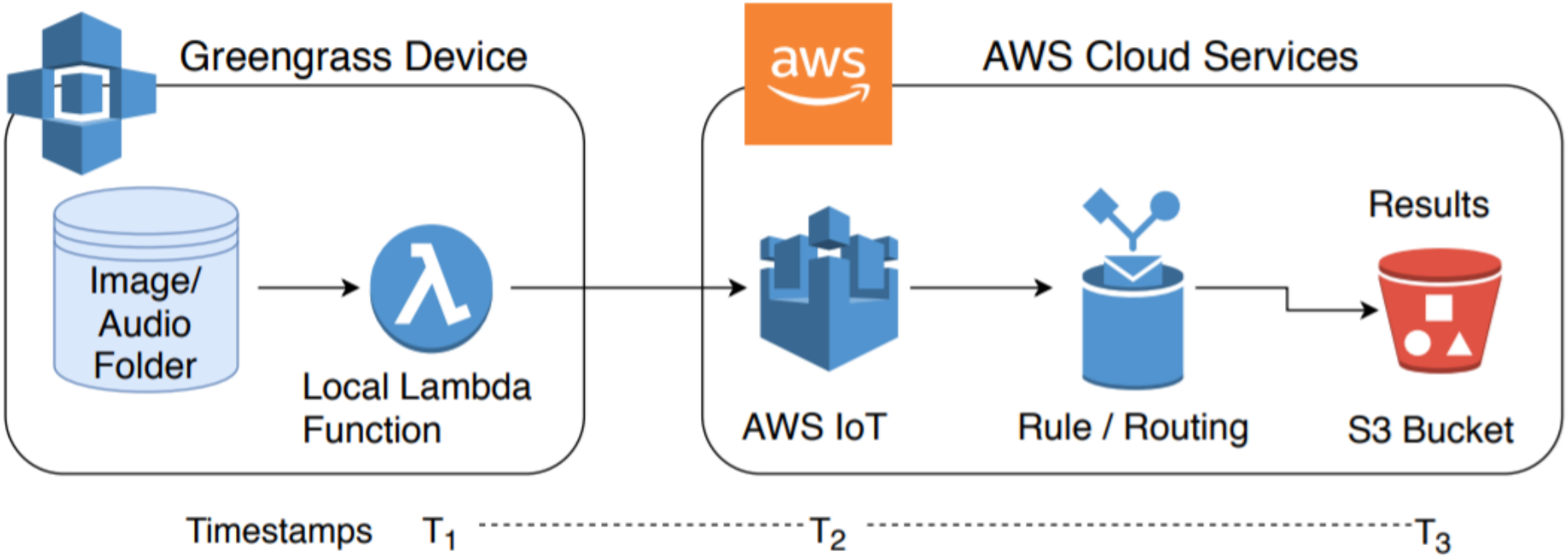
System Design Specifics (Azure)

- Both architectures are quite similar. We have an edge device locally performing computations. It can then send the data results to the cloud.
- The cloud can then send the results to the device or any other storage unit.



(T_1, T_2, T_3 - timestamps at each stage used for performance correlation / evaluation)

System Design Specifics (Greengrass)



(T_1 , T_2, T_3 are the timestamps at each stage used for performance correlation / evaluation)

Predictive Model - MXNet

The image recognition / classification task will be done by employing Apache MXNet. This library is useful for running many ML and Deep Learning tasks. In our case, we will generate class labels for the objects in each of the sampled image(s).

The MXNet Classification Flow on the Raspberry Pi is as follows:

Step 1: Feed each of the 32x32 CIFAR10 image set as input.

Step 2: Recognition of the image object by MXNet Machine Learning Model.

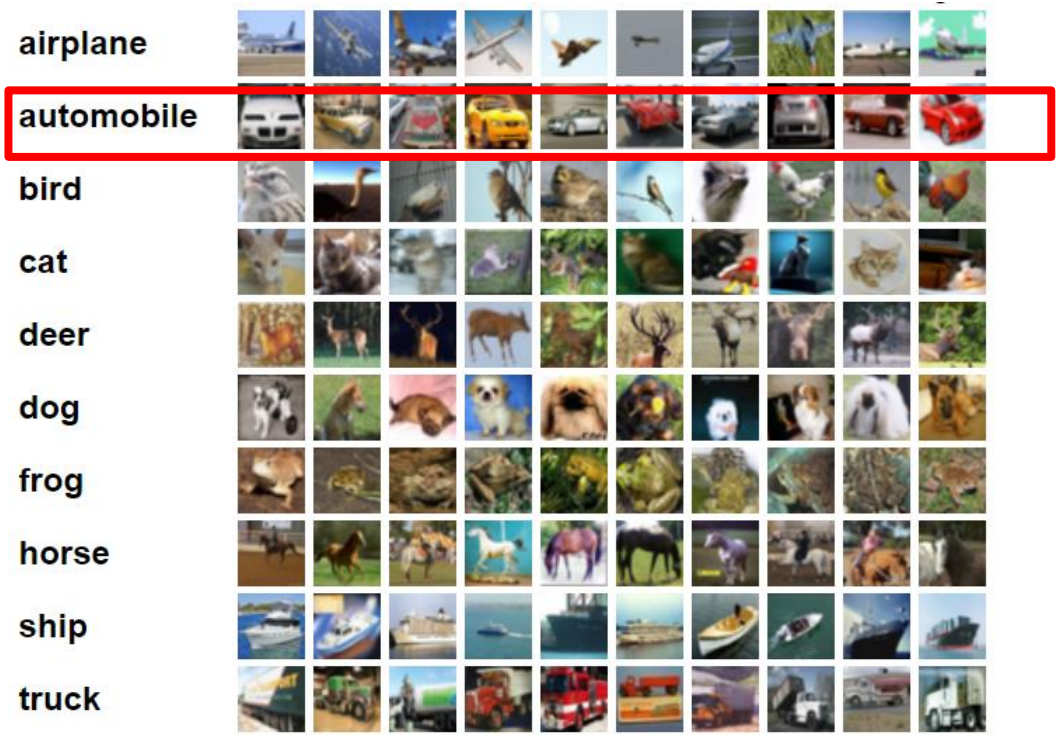
Step 3: Generation of a class label for each image (Note CIFAR 10 uses mutually exclusive classes so we should not run into intersecting class labels.)

Step 4: Classify the images and send it to the cloud for analysis and eventual knowledge gain.

Dataset - CIFAR10

Due to our requirements for the module to be lightweight and images to be size conscious, we will be using the CIFAR 10 image dataset. CIFAR10 is a publicly available dataset that contains 10 classes.

Size of Each Image : 32 x 32
Total Number of Images : 60000
Number of Classes : 10
Images Per Class : 6000
Training Batch Size : 50000
Test Batch Size : 10000



Evaluation / Metrics Plan

1. Total Compute Time (C_{edge})

We plan to use a number of metrics during evaluation. The first is the total compute time. It is the total processing time for an image in the Raspberry Pi.

2. Propagation Time (Dprop)

TIme taken to reach the hub after it is sent from the device.

3. Response time($T_{resp.}$)

The time taken after processing the request and outputting a response.

4. End-to-End Latency (T_{lat})

This is the total time taken, starting from the point of registering an input image (CIFAR10) to the time when the results are finally available in the storage unit.

.

5. Resource Utilization

We also plan to calculate average memory and CPU utilization for our setup on each platform.

Thank You