Hardware-accelerated Function-as-a-Service Using AWS Greengrass on Clear Linux

Table of Contents:

1. Introduction:

Hardware accelerated Function-as-a-Service (FaaS) enables cloud developers to deploy inference functionalities [1] on Intel IoT edge devices with accelerators (Integrated GPU, FPGA, and Movidius). These functions provide a great developer experience and seamless migration of visual analytics from cloud to edge in a secure manner using containerized environment. Hardware-accelerated FaaS provides the best-in-class performance by accessing optimized deep learning libraries on Intel IoT edge devices with accelerators.

This document describes implementation of FaaS inference samples (based on Python 2.7) using AWS Greengrass [1] and lambdas [2]. These lambdas can be created, modified, or updated in the cloud and can be deployed from cloud to edge using AWS Greengrass. This document covers description of samples, pre-requisites for Intel edge device, configuring a Greengrass group, creating and packaging lambda functions, deployment of lambdas and various options to consume the inference output.

2. Supported Platforms

- Operating System: Clear Linux Build 25930+
- Hardware:
    - Intel core platforms (This release supports inference on CPU only)

3. Description of Samples

The Greengrass samples are located at https://github.com/intel/Edge-Analytics-FaaS/tree/master/AWS%20Greengrass.

We provide the following Greengrass samples:

- greengrass_classification_sample.py
  This Greengrass sample classifies a video stream using classification networks such as AlexNet and GoogLeNet and publishes top-10 results on AWS IoT Cloud every second.
- greengrass_object_detection_sample_ssd.py
  This Greengrass sample detects objects in a video stream and classifies them using single-shot multi-box detection (SSD) networks such as SSD Squeezenet, SSD

Mobilenet, and SSD300. This sample publishes detection outputs such as class label, class confidence, and bounding box coordinates on AWS IoT Cloud every second.

4. Deep Learning Models

Edge optimized deep learning models are distributed on https://github.com/intel/Edge-optimized-models. The models need to be converted to intermediate representation (IR) using Model Optimizer for inference. We also provide intermediate representation for these edge-optimized models in FP32 directory for each model. For example, the FP32 IR for Squeezenet SSD detection model can be found at https://github.com/intel/Edge-optimized-models/tree/master/SqueezeNet%205-Class%20detection/FP32.

In this offering, model optimizer is not integrated into Clear Linux. User's custom models can be converted to intermediate representation on any other device using the steps listed here: https://software.intel.com/en-us/articles/OpenVINO-ModelOptimizer.

Note: For Mobilenet SSD models, the image pixels from all channels must be mean-subtracted and scaled before input using the below formula:

new_pixel_value = (original_pixel_value – 127.5) * 0.007843

5. Installing Clear Linux on the edge device

Before installing the Greengrass core software, set up your Clear Linux system by following the "bare metal" installation guide and choosing the "automatic" install:

- https://clearlinux.org/documentation/clear-linux/get-started/bare-metal-install#bare-metal-install

After the core OS is installed, create a user as shown here:

- https://clearlinux.org/documentation/clear-linux/guides/maintenance/enable-user-space

And then use the "bundle-add" option of the swupd software updater to add the following bundles to your system:

- $sudo swupd bundle-add os-clr-on-clr desktop-autostart computer-vision-basic

In particular, the "computer-vision-basic" bundle will provide the OpenVINO environment required to support the Hardware Acceleration described in this document.

6. Configuring a Greengrass group

For each Intel edge platform, we need to create a new Greengrass group and install Greengrass core software to establish the connection between cloud and edge.

- To create a Greengrass group, follow the instructions in the AWS Greengrass developer guide at: https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-config.html
- To install and configure Greengrass core on edge platform, follow the instructions at https://docs.aws.amazon.com/greengrass/latest/developerguide/gg-device-start.html

7. Creating and Packaging Lambda Functions

- To download the AWS Greengrass Core SDK for python 2.7, follow the steps 1-4 at: https://docs.aws.amazon.com/greengrass/latest/developerguide/create-lambda.html
- Replace greengrassHelloWorld.py with Greengrass sample (greengrass_classification_sample.py/greengrass_object_detection_sample_ssd.py) and zip it with extracted Greengrass SDK folders from the previous step into greengrass_sample_python_lambda.zip. The zip should contain:
    - greengrasssdk
    - greengrass sample(greengrass_classification_sample.py or greengrass_object_detection_sample_ssd.py)

  For example,
  ```
  zip -r greengrass_lambda.zip greengrasssdk greengrass_object_detection_sample_ssd.py
  ```

- To complete creating lambdas, follow steps 6-11 at: https://docs.aws.amazon.com/greengrass/latest/developerguide/create-lambda.html
- In step 9(a), while uploading the zip file, make sure to name the handler as below depending on the Greengrass sample you are using:
  `greengrass_object_detection_sample_ssd.function_handler` (or)
  `greengrass_classification_sample.function_handler`

8. Deployment of Lambdas

7.1. Configuring the Lambda function

- After creating the Greengrass group and the lambda function, start configuring the lambda function for AWS Greengrass by following the steps 1-8 in AWS Greengrass developer guide at: https://docs.aws.amazon.com/greengrass/latest/developerguide/config-lambda.html

- In addition to the details mentioned in step 8 of the AWS Greengrass developer guide, change the Memory limit to 2048MB to accommodate large input video streams.
- Add the following environment variables as key-value pair when editing the lambda configuration and click on update:

Key	Value
PARAM_MODEL_XML	<MODEL_DIR>/<IR.xml>, where <MODEL_DIR> is user specified and contains IR.xml, the Intermediate Representation file from Intel Model Optimizer
PARAM_INPUT_SOURCE	<DATA_DIR>/input.webm to be specified by user. Holds both input and output data. For webcam, set PARAM_INPUT_SOURCE to '/dev/video0'
PARAM_DEVICE	For CPU, specify "CPU"
PARAM_CPU_EXTENSION_PATH	/usr/lib64/libcpu_extension.so
PARAM_OUTPUT_DIRECTORY	<DATA_DIR> to be specified by user. Holds both input and output data
PARAM_NUM_TOP_RESULTS	User specified for classification sample.(e.g. 1 for top-1 result, 5 for top-5 results)

- Add subscription to subscribe or publish messages from Greengrass lambda function by following the steps 10-14 in AWS Greengrass developer guide at: https://docs.aws.amazon.com/greengrass/latest/developerguide/config-lambda.html. The "Optional topic filter" field should be the topic mentioned inside the lambda function.
  For example, `openvino/ssd or openvino/classification`

7.2. Local Resources

- Add local resources and access privileges by following the instructions https://docs.aws.amazon.com/greengrass/latest/developerguide/access-local-resources.html. Following are the local resources needed for CPU:

Name	Resource Type	Local path	Access
ModelDir	Volume	<MODEL_DIR> to be specified by user	Read-Only
Webcam	Device	/dev/video0	Read-Only
DataDir	Volume	<DATA_DIR> to be specified by user. Holds both input and output data.	Read and Write

7.3. Deploy

- To deploy the lambda function to AWS Greengrass core device, select "Deployments" on group page and follow the instructions at: https://docs.aws.amazon.com/greengrass/latest/developerguide/configs-core.html

9. Output Consumption

There are four options available for output consumption. These options are used to report/stream/upload/store inference output at an interval defined by the variable `'reporting_interval'` in the Greengrass samples.

a. IoT Cloud Output:

This option is enabled by default in the Greengrass samples using a variable `'enable_iot_cloud_output'`. We can use it to verify the lambda running on the edge device. It enables publishing messages to IoT cloud using the subscription topic specified in the lambda (For example, '`openvino/classification`' for classification and '`openvino/ssd`' for object detection samples). For classification, top-1 result with class label are published to IoT cloud. For SSD object detection, detection results such as bounding box co-ordinates of objects, class label, and class confidence are published. To view the output on IoT cloud, follow the instructions at https://docs.aws.amazon.com/greengrass/latest/developerguide/lambda-check.html

b. Kinesis Streaming:

This option enables inference output to be streamed from the edge device to cloud using Kinesis [3] streams when 'enable_kinesis_output' is set to True. The edge devices act as data producers and continually push processed data to the cloud. The users need to set up and specify Kinesis stream name, Kinesis shard, and AWS region in the Greengrass samples.

c. Cloud Storage using AWS S3 Bucket:

This option enables uploading and storing processed frames (in JPEG format) in an AWS S3 bucket when '`enable_s3_jpeg_output`' variable is set to True. The users need to set up and specify the S3 bucket name in the Greengrass samples to store the JPEG images. The images are named using the timestamp and uploaded to S3.

d. Local Storage:

This option enables storing processed frames (in JPEG format) on the edge device when '`enable_s3_jpeg_output`' variable is set to True. The images are named using the timestamp and stored in a directory specified by `'PARAM_OUTPUT_DIRECTORY'`.

10. References:

[1] AWS Greengrass: https://aws.amazon.com/greengrass/
[2] AWS Lambda: https://aws.amazon.com/lambda/
[3] AWS Kinesis: https://aws.amazon.com/kinesis/