**Hardware-accelerated Function-as-a-Service Using AWS Greengrass**

**(Beta Release)**

**Table of Contents:**

1. Introduction
2. Description of Samples
3. Pre-requisites for Intel Edge Device
4. Configuring a Greengrass Group
5. Creating and Packaging Lambda Functions
6. Deployment of Lambdas

6.1. Configuring the Lambda function

6.2. Local Resources

6.3. Deployment

1. Output Consumption
2. References
3. **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.

1. **Description of Samples**

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.

1. **Pre-requisites for Intel Edge Device**

* Operating System: Ubuntu 16.04
* Hardware:
  + Aaeon Up2 kit with integrated GPU (<https://software.intel.com/en-us/blogs/2018/05/16/kits-to-accelerate-your-computer-vision-deployments>)
  + IEI 870 tank with integrated GPU (<https://software.intel.com/en-us/blogs/2018/05/16/kits-to-accelerate-your-computer-vision-deployments>)
  + Any Greengrass certified Intel gateway with Atom Apollo Lake processor, Core Skylake and Xeon Skylake in <https://aws.amazon.com/greengrass/faqs/>. These platforms come with an integrated GPU that can be used for inference.
  + Accelerators: Arria10 1150 FPGA (<https://www.buyaltera.com/Search/?keywords=arria+kit>)
* Download and install OpenVINO Toolkit from <https://software.intel.com/en-us/openvino-toolkit>
* Python 2.7 with opencv-python, numpy, boto3 (use sudo pip install to install the packages in locations accessible by Greengrass.
* Download Intel Edge optimized models available at: <https://github.com/intel/Edge-optimized-models>. Any custom pre-trained classification or SSD models can be used.
* Convert the above models to Intermediate Representation (IR) using Model Optimizer. Follow the instructions at: <https://software.intel.com/en-us/articles/OpenVINO-ModelOptimizer>. For CPU, models should be converted with data type FP32 and for GPU/FPGA, it should be with data type FP16 for the best performance.

1. **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>

1. **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:
* greengrass\_common,
* greengrass\_ipc\_python\_sdk
* greengrasssdk
* greengrass sample(greengrass\_classification\_sample.py or greengrass\_object\_detection\_sample\_ssd.py)

For example,

zip -r greengrass\_sample\_python\_lambda.zip greengrass\_common greengrass\_ipc\_python\_sdk 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>

1. **Deployment of Lambdas**

**6.1. 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** |
| LD\_LIBRARY\_PATH | <INSTALL\_DIR>/opencv/share/OpenCV/3rdparty/lib:<INSTALL\_DIR>/opencv/lib:/opt/intel/opencl:<INSTALL\_DIR>/deployment\_tools/inference\_engine/external/cldnn/lib:<INSTALL\_DIR>/deployment\_tools/inference\_engine/external/mkltiny\_lnx/lib:<INSTALL\_DIR>/deployment\_tools/inference\_engine/lib/ubuntu\_16.04/intel64:<INSTALL\_DIR>/deployment\_tools/model\_optimizer/model\_optimizer\_caffe/bin:<INSTALL\_DIR>/openvx/lib |
| PYTHONPATH | <INSTALL\_DIR>/deployment\_tools/inference\_engine/python\_api/Ubuntu\_1604/python2 |
| 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.mp4 to be specified by user. Holds both input and output data. For webcam, set PARAM\_INPUT\_SOURCE to ‘0’ |
| PARAM\_DEVICE | For CPU, specify "CPU"  For GPU, specify "GPU"  For FPGA, specify “HETERO:FPGA,CPU” |
| PARAM\_CPU\_EXTENSION\_PATH | <INSTALL\_DIR>/deployment\_tools/inference\_engine/lib/Ubuntu\_16.04/intel64/<CPU\_EXTENSION\_LIB>, where CPU\_EXTENSION\_LIB is libcpu\_extension\_sse4.so for Intel Atom processors and libcpu\_extension\_avx2.so for Intel Core and Xeon processors |
| 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) |

Use below LD\_LIBRARY\_PATH and additional environment variables for Arria10 FPGA:

|  |  |
| --- | --- |
| **Key** | **Value** |
| LD\_LIBRARY\_PATH | /opt/altera/aocl-pro-rte/aclrte-linux64/board/a10\_ref/linux64/lib:/opt/altera/aocl-pro-rte/aclrte-linux64/host/linux64/lib:<INSTALL\_DIR>/opencv/share/OpenCV/3rdparty/lib:<INSTALL\_DIR>/opencv/lib:/opt/intel/opencl:<INSTALL\_DIR>/deployment\_tools/inference\_engine/external/cldnn/lib:<INSTALL\_DIR>/deployment\_tools/inference\_engine/external/mkltiny\_lnx/lib:<INSTALL\_DIR>/deployment\_tools/inference\_engine/lib/ubuntu\_16.04/intel64:<INSTALL\_DIR>/deployment\_tools/model\_optimizer/model\_optimizer\_caffe/bin:<INSTALL\_DIR>/openvx/lib |
| DLA\_AOCX | <INSTALL\_DIR>/a10\_devkit\_bitstreams/0-8-1\_a10dk\_fp16\_8x48\_arch06.aocx |
| CL\_CONTEXT\_COMPILER\_MODE\_INTELFPGA | 3 |

* + 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

**6.2. 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 various hardware (CPU,GPU and FPGA) options:

**General (for all hardware options):**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| OpenVINOPath | Volume | <INSTALL\_DIR> where INSTALL\_DIR is the OpenVINO installation directory | Read-Only |

**GPU:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Resource**  **Type** | **Local path** | **Access** |
| GPU | Device | /dev/dri/renderD128 | Read and Write |

**FPGA:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Resource**  **Type** | **Local path** | **Access** |
| FPGA | Device | /dev/acla10\_ref0 | Read and Write |
| FPGA\_DIR1 | Volume | /opt/Intel/OpenCL/Boards | Read and Write |
| FPGA\_DIR2 | Volume | /etc/OpenCL/vendors | Read and Write |

**Movidius:**

Movidius hasn’t been validated with Greengrass yet. This section will be updated in future releases.

**6.3. 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>

1. **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’.

1. **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/>