Manufacturing_Scenario

July 11, 2020

1 Smart Queue Monitoring System - Manufacturing Scenario

1.1 Overview

Now that you have your Python script and job submission script, you're ready to request an edge node and run inference on the different hardware types (CPU, GPU, VPU, FPGA).

After the inference is completed, the output video and stats files need to be retrieved and stored in the workspace, which can then be viewed within the Jupyter Notebook.

1.2 Objectives

- Submit inference jobs to Intel's DevCloud using the `qsub` command.
- Retrieve and review the results.
- After testing, go back to the proposal doc and update your original proposed hardware device.

1.3 Step 0: Set Up

IMPORTANT: Set up paths so we can run Dev Cloud utilities You *must* run this every time you enter a Workspace session. (Tip: select the cell and use **Shift+Enter** to run the cell.)

[13]:
```
%env PATH=/opt/conda/bin:/opt/spark-2.4.3-bin-hadoop2.7/bin:/opt/conda/bin:/usr/
 ↪local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/
 ↪intel_devcloud_support
import os
import sys
sys.path.insert(0, os.path.abspath('/opt/intel_devcloud_support'))
sys.path.insert(0, os.path.abspath('/opt/intel'))
```

```
env: PATH=/opt/conda/bin:/opt/spark-2.4.3-bin-hadoop2.7/bin:/opt/conda/bin:/usr/
local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/intel_devcloud_supp
ort
```

1.3.1 Step 0.1: (Optional-step): Original Video

If you are curious to see the input video, run the following cell to view the original video stream we'll be using for inference.

[ ]:
```
import videoHtml
videoHtml.videoHTML('Manufacturing', ['original_videos/Manufacturing.mp4'])
```

1.4 Step 1 : Inference on a Video

In the next few cells, You'll submit your job using the `qsub` command and retrieve the results for each job. Each of the cells below should submit a job to different edge compute nodes.

The output of the cell is the `JobID` of your job, which you can use to track progress of a job with `liveQStat`.

You will need to submit a job for each of the following hardware types: * **CPU** * **GPU** * **VPU** * **FPGA**

Note: You will have to submit each job one at a time and retrieve their results.

After submission, they will go into a queue and run as soon as the requested compute resources become available. (Tip: **shift+enter** will run the cell and automatically move you to the next cell.)

If your job successfully runs and completes, once you retrieve your results, it should output a video and a stats text file in the `results/manufacturing/<DEVICE>` directory.

For example, your **CPU** job should output its files in this directory: > **results/manufacturing/cpu**

Note: To get the queue labels for the different hardware devices, you can go to this link.

The following arguments should be passed to the job submission script after the `-F` flag: * Model path - `/data/models/intel/person-detection-retail-0013/<MODEL PRECISION>/`. You will need to adjust this path based on the model precision being using on the hardware. * Device - `CPU`, `GPU`, `MYRIAD`, `HETERO:FPGA,CPU`. * Manufacturing video path - `/data/resources/manufacturing.mp4`. * Manufacturing queue_param file path - `/data/queue_param/manufacturing.npy`. * Output path - `/output/results/manufacturing/<DEVICE>` This should be adjusted based on the device used in the job. * Max num of people - This is the max number of people in queue before the system would redirect them to another queue.

1.5 Step 1.1: Submit to an Edge Compute Node with an Intel CPU

In the cell below, write a script to submit a job to an edge node with an Intel Core™ i5-6500TE processor. The inference workload should run on the CPU.

[15]:
```
#Submit job to the queue
MODEL_PATH = "/data/models/intel/person-detection-retail-0013/FP32/
↪person-detection-retail-0013"
DEVICE = "CPU"
VIDEO = "/data/resources/manufacturing.mp4"
QUEUE_COORDS = "/data/queue_param/manufacturing.npy"
OUTPUT_PATH = "/output/results/manufacturing/cpu"
MAX_NUM_PEOPLE = "2"


CMD = " ".
↪join([MODEL_PATH,DEVICE,VIDEO,QUEUE_COORDS,OUTPUT_PATH,MAX_NUM_PEOPLE])
```

```
cpu_job_id = !qsub queue_job.sh -d . -l nodes=1:tank-870:i5-6500te -F "$CMD" -N␣
 ↪store_core
print(cpu_job_id[0])
```

```
HZCFe7TSmihgMAfffsqigLWOn5patWgC
```

Check Job Status To check on the job that was submitted, use `liveQStat` to check the status of the job.

Column `S` shows the state of your running jobs.

For example: - If `JOB ID`is in Q state, it is in the queue waiting for available resources. - If `JOB ID` is in R state, it is running.

[16]:
```
import liveQStat
liveQStat.liveQStat()
```

Get Results Run the next cell to retrieve your job's results.

[17]:
```
import get_results
get_results.getResults(cpu_job_id[0], filename='output.tgz', blocking=True)
```

```
getResults() is blocking until results of the job
(id:HZCFe7TSmihgMAfffsqigLWOn5patWgC) are ready.
Please wait…Success!
output.tgz was downloaded in the same folder as this notebook.
```

Unpack your output files and view stdout.log

[18]:
```
!tar zxf output.tgz
```

[ ]:
```
!cat stdout.log
```

View stderr.log This can be used for debugging.

[20]:
```
!cat stderr.log
```

View Output Video Run the cell below to view the output video. If inference was successfully run, you should see a video with bounding boxes drawn around each person detected.

[ ]:
```
import videoHtml

videoHtml.videoHTML('Manufacturing CPU', ['results/manufacturing/cpu/
 ↪output_video.mp4'])
```

1.6 Step 1.2: Submit to an Edge Compute Node with CPU and IGPU

In the cell below, write a script to submit a job to an IEI Tank* 870-Q170 edge node with an Intel® Core i5-6500TE. The inference workload should run on the **Intel® HD Graphics 530** integrated GPU.

```
[22]: #Submit job to the queue
MODEL_PATH = "/data/models/intel/person-detection-retail-0013/FP32/
 ↪person-detection-retail-0013"
DEVICE = "GPU"
VIDEO = "/data/resources/manufacturing.mp4"
QUEUE_COORDS = "/data/queue_param/manufacturing.npy"
OUTPUT_PATH = "/output/results/manufacturing/gpu"
MAX_NUM_PEOPLE = "2"


CMD = " ".
 ↪join([MODEL_PATH,DEVICE,VIDEO,QUEUE_COORDS,OUTPUT_PATH,MAX_NUM_PEOPLE])

gpu_job_id = !qsub queue_job.sh -d . -l nodes=1:tank-870:i5-6500te:intel-hd-530␣
 ↪-F "$CMD" -N store_core
print(gpu_job_id[0])
```

```
6Porr9PlPpR92hdXWlUztVJnOCEUHU85
```

1.6.1 Check Job Status

To check on the job that was submitted, use `liveQStat` to check the status of the job.

Column `S` shows the state of your running jobs.

For example: - If `JOB ID`is in Q state, it is in the queue waiting for available resources. - If `JOB ID` is in R state, it is running.

```
[23]: import liveQStat
liveQStat.liveQStat()
```

Get Results Run the next cell to retrieve your job's results.

```
[24]: import get_results
get_results.getResults(gpu_job_id[0], filename='output.tgz', blocking=True)
```

```
getResults() is blocking until results of the job
(id:6Porr9PlPpR92hdXWlUztVJnOCEUHU85) are ready.
Please wait…Success!
output.tgz was downloaded in the same folder as this notebook.
```

Unpack your output files and view stdout.log

```
[25]: !tar zxf output.tgz
```

```
[ ]: !cat stdout.log
```

View stderr.log This can be used for debugging.

```
[27]: !cat stderr.log
```

View Output Video Run the cell below to view the output video. If inference was successfully run, you should see a video with bounding boxes drawn around each person detected.

```
[ ]: import videoHtml

     videoHtml.videoHTML('Manufacturing GPU', ['results/manufacturing/gpu/
      ↪output_video.mp4'])
```

1.7 Step 1.3: Submit to an Edge Compute Node with a Neural Compute Stick 2

In the cell below, write a script to submit a job to an IEI Tank 870-Q170 edge node with an Intel Core i5-6500te CPU. The inference workload should run on an Intel Neural Compute Stick 2 installed in this node.

```
[29]: #Submit job to the queue
      MODEL_PATH = "/data/models/intel/person-detection-retail-0013/FP16/
       ↪person-detection-retail-0013"
      DEVICE = "MYRIAD"
      VIDEO = "/data/resources/manufacturing.mp4"
      QUEUE_COORDS = "/data/queue_param/manufacturing.npy"
      OUTPUT_PATH = "/output/results/manufacturing/vpu"
      MAX_NUM_PEOPLE = "2"


      CMD = " ".
       ↪join([MODEL_PATH,DEVICE,VIDEO,QUEUE_COORDS,OUTPUT_PATH,MAX_NUM_PEOPLE])

      vpu_job_id = !qsub queue_job.sh -d . -l nodes=tank-870:i5-6500te:intel-ncs2 -F␣
       ↪"$CMD" -N store_core
      print(vpu_job_id[0])
```

```
mbocTqi5lhwCWOCemcLtxOtT4iO36C5e
```

1.7.1 Check Job Status

To check on the job that was submitted, use `liveQStat` to check the status of the job.

Column `S` shows the state of your running jobs.

For example: - If `JOB ID`is in Q state, it is in the queue waiting for available resources. - If `JOB ID` is in R state, it is running.

```
[30]: import liveQStat
      liveQStat.liveQStat()
```

Get Results Run the next cell to retrieve your job's results.

```
[31]: import get_results
      get_results.getResults(vpu_job_id[0], filename='output.tgz', blocking=True)
```

```
getResults() is blocking until results of the job
(id:mbocTqi5lhwCWOCemcLtxOtT4iO36C5e) are ready.
Please wait…Success!
output.tgz was downloaded in the same folder as this notebook.
```

Unpack your output files and view stdout.log

```
[32]: !tar zxf output.tgz
```

```
[ ]: !cat stdout.log
```

View stderr.log This can be used for debugging.

```
[34]: !cat stderr.log
```

View Output Video Run the cell below to view the output video. If inference was successfully run, you should see a video with bounding boxes drawn around each person detected.

```
[ ]: import videoHtml

     videoHtml.videoHTML('Manufacturing VPU', ['results/manufacturing/vpu/
      ↪output_video.mp4'])
```

1.8 Step 1.4: Submit to an Edge Compute Node with IEI Mustang-F100-A10

In the cell below, write a script to submit a job to an IEI Tank 870-Q170 edge node with an Intel Core™ i5-6500te CPU . The inference workload will run on the IEI Mustang-F100-A10 FPGA card installed in this node.

```
[36]: #Submit job to the queue
      MODEL_PATH = "/data/models/intel/person-detection-retail-0013/FP16/
       ↪person-detection-retail-0013"
      DEVICE = "HETERO:FPGA,CPU"
      VIDEO = "/data/resources/manufacturing.mp4"
      QUEUE_COORDS = "/data/queue_param/manufacturing.npy"
      OUTPUT_PATH = "/output/results/manufacturing/fpga"
      MAX_NUM_PEOPLE = "2"
```

```
CMD = " ".
↪join([MODEL_PATH,DEVICE,VIDEO,QUEUE_COORDS,OUTPUT_PATH,MAX_NUM_PEOPLE])

fpga_job_id = !qsub queue_job.sh -d . -l nodes=1:tank-870:i5-6500te:
↪iei-mustang-f100-a10 -F "$CMD" -N store_core
print(fpga_job_id[0])
```

```
Co4BNHd3ujzq1Q9Nayy6E12ocowALhQu
```

1.8.1 Check Job Status

To check on the job that was submitted, use `liveQStat` to check the status of the job.

Column `S` shows the state of your running jobs.

For example: - If `JOB ID`is in Q state, it is in the queue waiting for available resources. - If `JOB ID` is in R state, it is running.

[37]:
```
import liveQStat
liveQStat.liveQStat()
```

Get Results Run the next cell to retrieve your job's results.

[38]:
```
import get_results
get_results.getResults(fpga_job_id[0], filename='output.tgz', blocking=True)
```

```
getResults() is blocking until results of the job
(id:Co4BNHd3ujzq1Q9Nayy6E12ocowALhQu) are ready.
Please wait…Success!
output.tgz was downloaded in the same folder as this notebook.
```

Unpack your output files and view stdout.log

[39]:
```
!tar zxf output.tgz
```

[ ]:
```
!cat stdout.log
```

View stderr.log This can be used for debugging.

[41]:
```
!cat stderr.log
```

View Output Video Run the cell below to view the output video. If inference was successfully run, you should see a video with bounding boxes drawn around each person detected.

[ ]:
```
import videoHtml

videoHtml.videoHTML('Manufacturing FPGA', ['results/manufacturing/fpga/
↪output_video.mp4'])
```

Wait!

Please wait for all the inference jobs and video rendering to complete before proceeding to the next step.

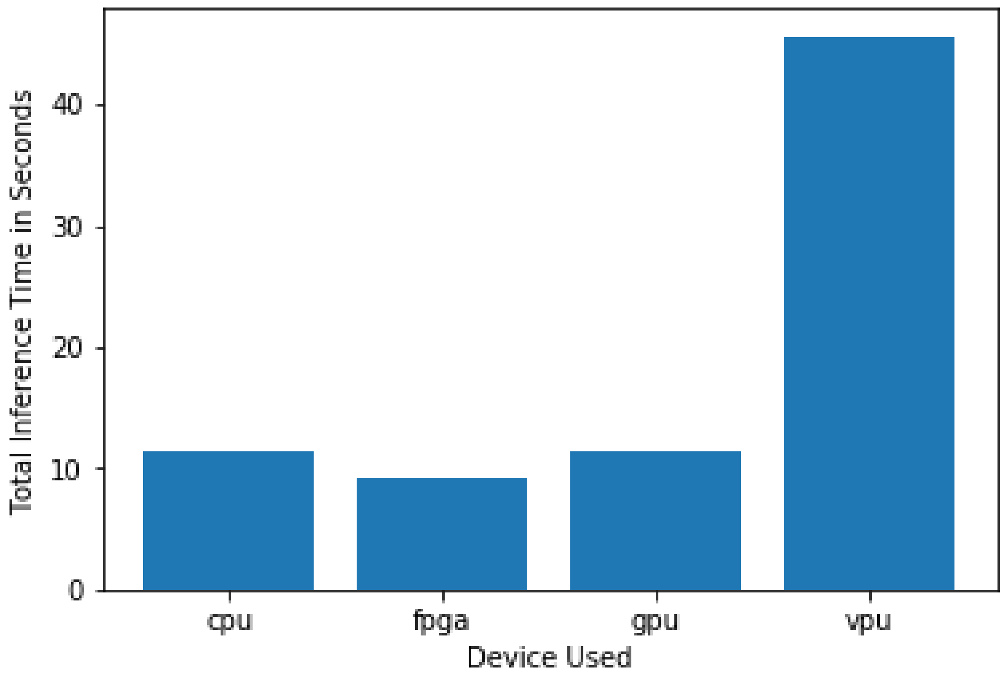
1.9 Step 2: Assess Performance

Run the cells below to compare the performance across all 4 devices. The following timings for the model are being compared across all 4 devices:

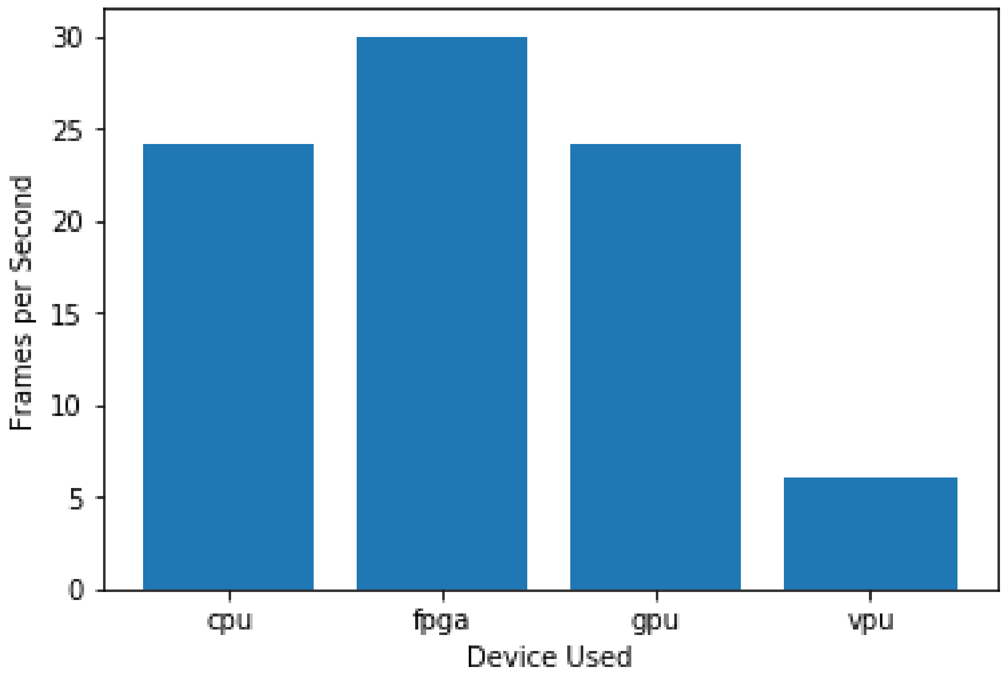
- Model Loading Time
- Average Inference Time
- FPS

[43]:
```
import matplotlib.pyplot as plt

device_list=['cpu', 'gpu', 'fpga', 'vpu']
inference_time=[]
fps=[]
model_load_time=[]

for device in device_list:
    with open('results/manufacturing/'+device+'/stats.txt', 'r') as f:
        inference_time.append(float(f.readline().split("\n")[0]))
        fps.append(float(f.readline().split("\n")[0]))
        model_load_time.append(float(f.readline().split("\n")[0]))
```

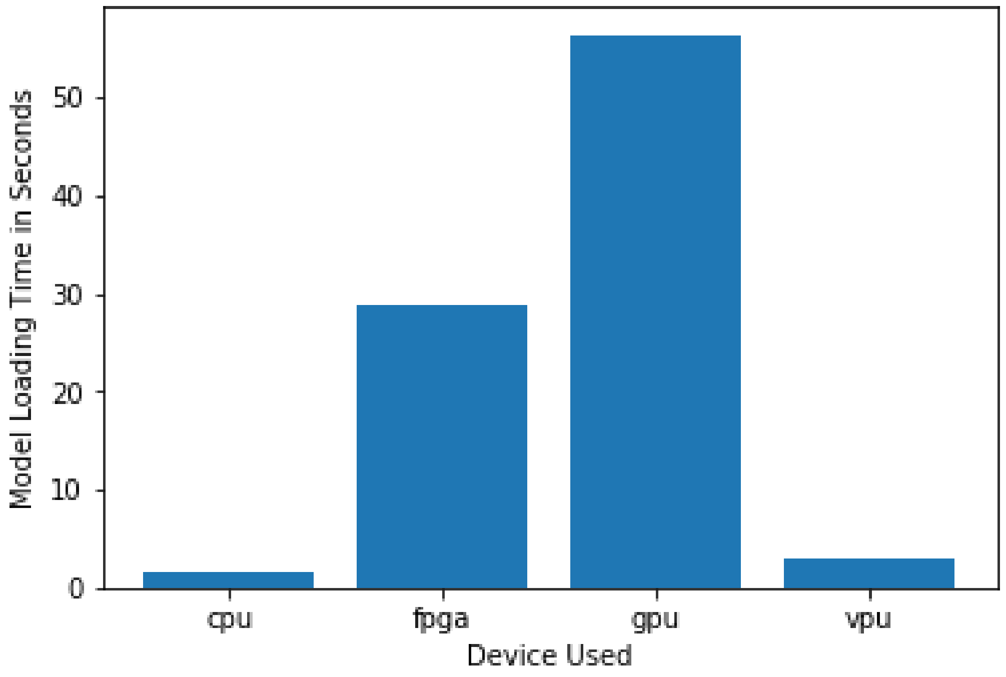
[44]:
```
plt.bar(device_list, inference_time)
plt.xlabel("Device Used")
plt.ylabel("Total Inference Time in Seconds")
plt.show()
```



[45]:
```
plt.bar(device_list, fps)
plt.xlabel("Device Used")
plt.ylabel("Frames per Second")
plt.show()
```



[46]:
```
plt.bar(device_list, model_load_time)
plt.xlabel("Device Used")
plt.ylabel("Model Loading Time in Seconds")
plt.show()
```



2 Step 3: Update Proposal Document

Now that you've completed your hardware testing, you should go back to the proposal document and validate or update your originally proposed hardware. Once you've updated your proposal, you can move on to the next scenario.