jsapede Update README.md f288947 · 2 months ago 11 Commits

README.md Update README.md 2 months ago

README

frigate-proxmox-docker-openvino

Complete setting for OpenVINO hardware acceleration in frigate, instead of CORAL

tutorial is adapted for Docker version of frigate installed in a proxmox LXC and deals mainly with GPU passthroughs

Prerequisites

- Intel iX > GEN6 architecture (i.e. compatible with openvino acceleration)
- A proxmox working installation

check in your PVE Shell that `/dev/dri/renderD128` is available :

```
ce /dev/dri
ls
```

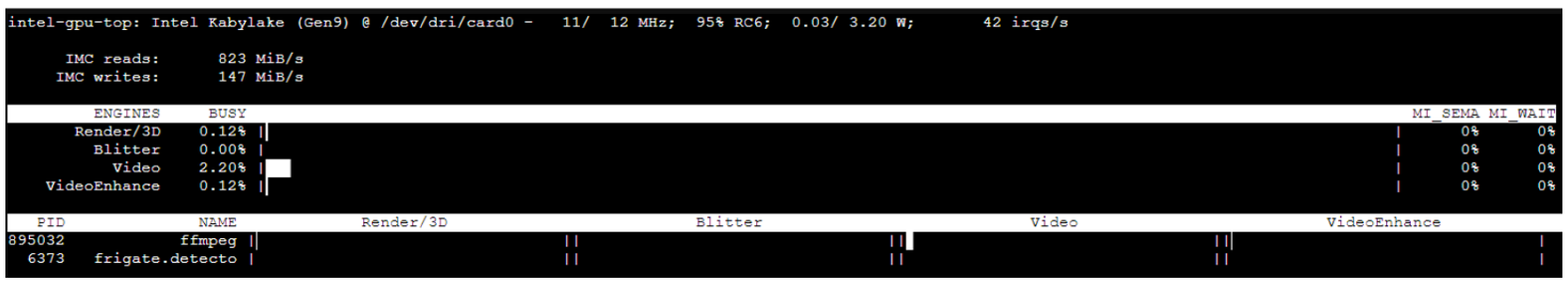
optionnally install intel GPU tools :

```
apt install intel-gpu-tools
```

now you can check GPU access / usage :

```
intel_gpu_top
```

should lead to something like this :



create Docker LXC :

The easiest way is to use Tteck's scripts

first in the PVE console launch the tteck's script to install a new docker LXC :

```
bash -c "$(wget -qLO - https://github.com/tteck/Proxmox/raw/main/ct/docker.sh)"
```

during installation :

- switch to "advanced mode"
- select debian 12
- make the LXC **PRIVILEGED**
- you'd better choose 8Go ram and 2 or 4 cores
- add portainer if needed
- add docker compose

Once the LXC is created you have to can also install intel-gpu-tools **inside** the LXC

```
apt install intel-gpu-tools
```

Next you have to add GPU passthrough to the LXC to allow frigate access the OpenVINO acceleratons. On your LXC "Ressources" add "Device Passtrough" :

About

Complete setting for OpenVINO hardware acceleration in friggate

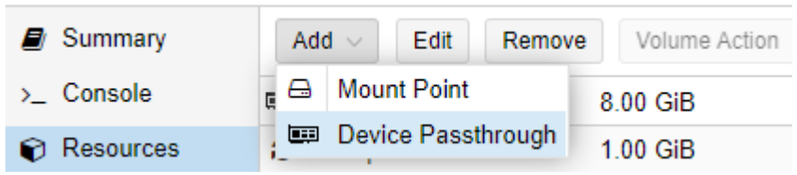
Readme
Activity
3 stars
1 watching
2 forks
Report repository

Releases

No releases published

Packages

No packages published



and specify the path you want to add : `/dev/dri/renderD128`

Reboot

now your LXC has access to the GPU.

Frigate Docker

create folders

On the LXC shell, create folders to organize your frigate storage for videos / captures / models and configs.

Here are my usually settings :

```
mkdir /opt/frigate
mkdir /opt/frigate/media
mkdir /opt/frigate/config
```

create the forlders according to your needs

next we will build the docker container.

create a docker-compose.yml at the root folder

```
cd /opt/frigate
nano docker-compose.yml
```

or create a stack in portainer :

and add :

```
version: "3.9"
services:
  frigate:
    container_name: frigate
    privileged: true
    restart: unless-stopped
    image: ghcr.io/blakeblackshear/frigate:0.14.1
    cap_add:
      - CAP_PERFMON
    shm_size: "256mb"
    devices:
      - /dev/dri/renderD128:/dev/dri/renderD128
      - /dev/dri/card0:/dev/dri/card0
    volumes:
      - /etc/localtime:/etc/localtime:ro
      - /opt/frigate/config:/config
      - /opt/media:/media/frigate
      - type: tmpfs
        target: /tmp/cache
        tmpfs:
          size: 1G
    ports:
      - "5000:5000"
      - "8971:8971"
      - "1984:1984"
      - "8554:8554" # RTSP feeds
      - "8555:8555/tcp" # WebRTC over tcp
      - "8555:8555/udp" # WebRTC over udp
    environment:
      FRIGATE_RTSP_PASSWORD: ****
      PLUS_API_KEY: ****
```

as you can see :

- container is **privileged**
- /dev/dri/renderD128 is passtrhough from the LXC to the container
- created folders are bind to frigate usual folders
- shm_size has to be set according to documentation
- tmpfs has to be adjusted to your configuration, see documentation
- ports for UI, RTSP and webRTC are forwarded
- define some FRIGATE_RTSP_PASSWORD and PLUS_API_KEY if needed

From now the docker container is ready, and have access to the GPU.

Do not start it right now as you have to provide frigate configuraton !

Setup Frigate for OpenVINO acceleration

add your frigate configutration :

```
cd /opt/frigate/config
nano config.yml
```

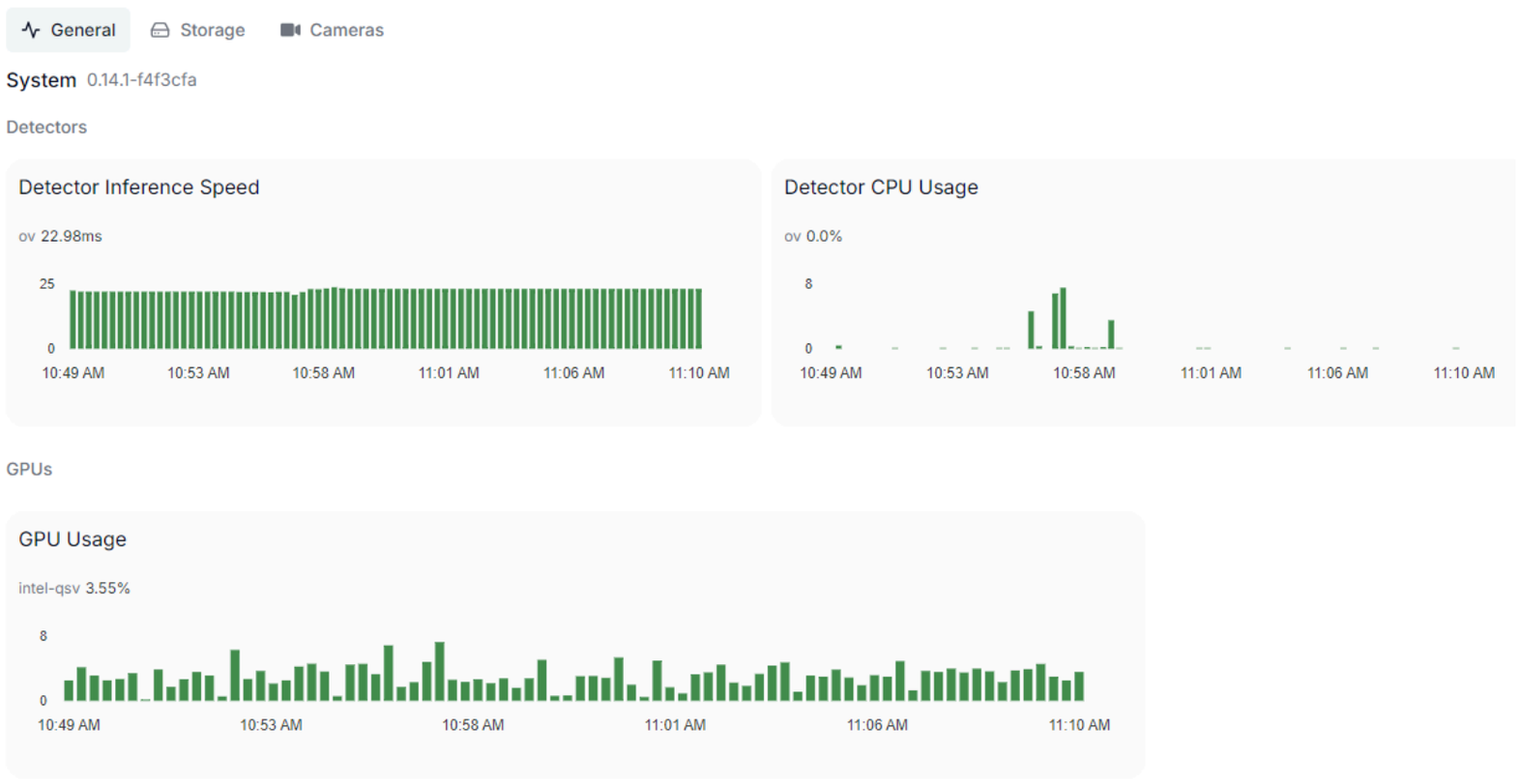
edit it accroding to your setup and now you must add the following lines to your frigate config :

```
detectors:
  ov:
    type: openvino
    device: GPU

model:
  width: 300
  height: 300
  input_tensor: nhwc
  input_pixel_format: bgr
  path: /openvino-model/ssdlite_mobilenet_v2.xml
  labelmap_path: /openvino-model/coco_91cl_bkgr.txt
```

Once your `config.yml` is ready build your container by either `docker compose up` or "deploy Stack" if you're using portainer

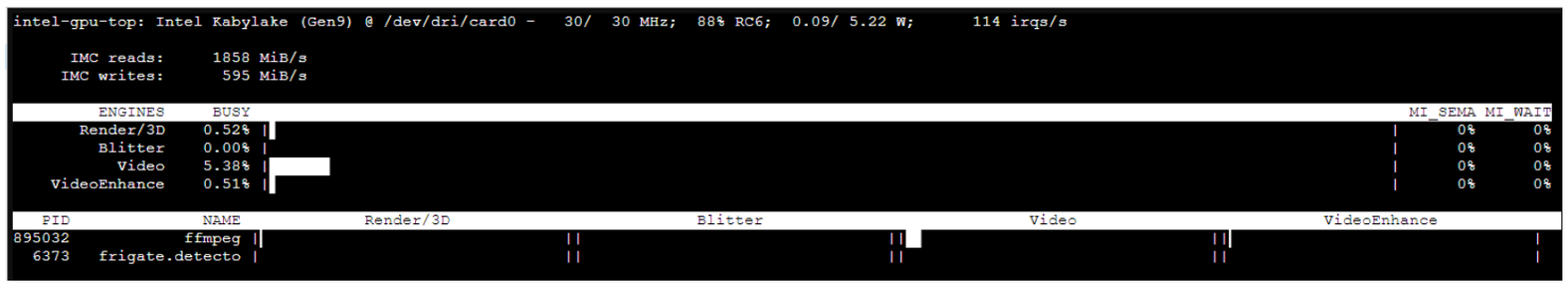
reboot all, and go to frigate UI to check everything is working :



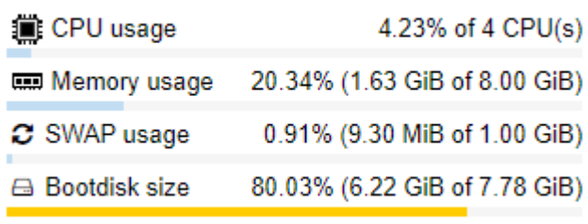
you should see :

- low inference time : ~20 ms
- low CPU usage
- GPU usage

you can also check with `intel_gpu_top` inside the LXC console and see that Render/3D has some loads according to frigate detections



and on your PROXMOX, you can see that CPU load of the LXC is drastically lowered :



Extra settings

CPU load

i experimentally found that running those 2 Tteck's scripts int the PVE console greatly reduces the CPU consumption in "idle mode" (i.e. when frigate only "observes" and has no detection running) :

- Filesystem Trim
- CPU Scaling Governor : set governor to **powersave**

experiment on your own !

YOLO NAS models

Except default SSDLite model, YOLO NAS model is also available for OpenVINO acceleration.

To use it you have to build the model to make it compatible with frigate. this can be easily done with the dedicated google collab

the only thing to do is to define the dimensions of the input image shape. 320x320 leads to higher inference time, i'd use 256x256.

```
            input_image_shape=(256,256),
```

and select the base precision of the model. **S** version is good enough, **M** induces much higer inference time :

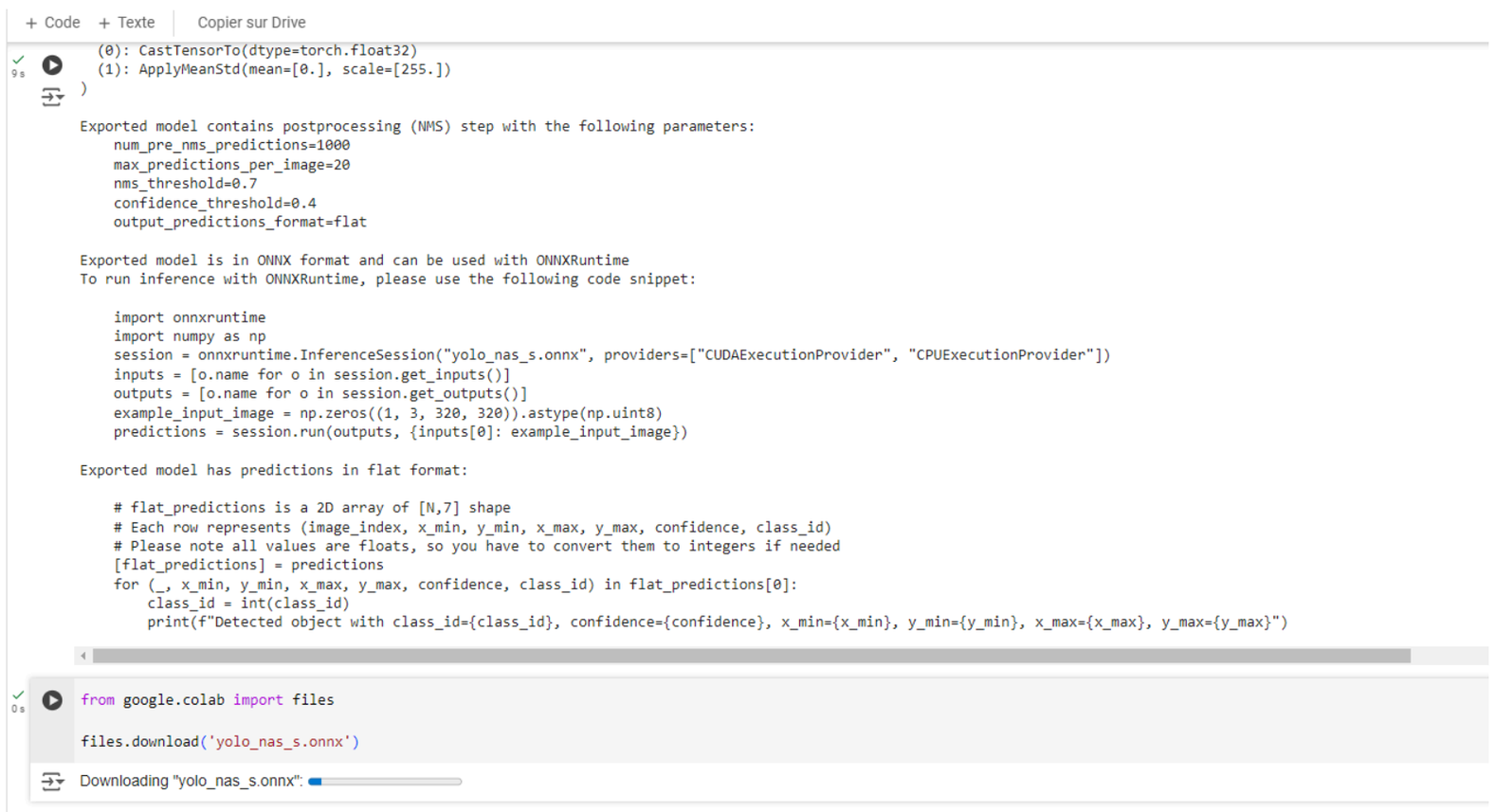
```
model = models.get(Models.YOLO_NAS_S, pretrained_weights="coco")
```

NOTE: you can make some tests and find the good combination for your hardware. try to limit inference time around 20 ms

and specify the name of the model file you will generate :

```
files.download('yolo_nas_s.onnx')
```

now simply launch all the steps of the collab, 1 by 1, and it will download the model file :



Copy the model file you generated to your frigate config folder `/opt/frigate/config`

and now change your detector and adapt it accordingly to your settings :

```
detectors:
  ov:
    type: openvino
    device: GPU

model:
  model_type: yolonas
  width: 256 # <--- should match whatever was set in notebook
  height: 256 # <--- should match whatever was set in notebook
  input_tensor: nchw # <--- take care, it changes from the setting fot the SSDLite model !
  input_pixel_format: bgr
  path: /config/yolo_nas_s_256.onnx # <--- should match the path and the name of your model
file
  labelmap_path: /labelmap/coco-80.txt # <--- should match the name and the location of the
COCO80 labelmap file
```

NOTE : YOLO NAS uses the COCO80 labelmap instead of COCO91