Triton inference server on KISTI NEURON

Contents	
Created	@July 19, 2022 9:46 AM
Tags	Daily

1. Prepare a TensorRT model

- Tutorial by NVIDIA Jinho Lee https://github.com/leejinho610/TRT_Triton_HandsOn/blob/main/1. Model preperation (starting Point).ipynb
- **trtexec** command (ONNX → tensorRT)

```
!trtexec \
  --onnx=model.onnx \
  --explicitBatch \
  --optShapes=actual_input_1:16x3x224x224 \
  --maxShapes=actual_input_1:32x3x224x224 \
  --minShapes=actual_input_1:1x3x224x224 \
  --best \
  --saveEngine=model.plan
```

- you may use TensorRT in the singularity image that I built on the KISTI server(/scratch/kedu05/leecy/tensorRT)
- Run the singularity image with “singularity run --nv tensorRT” and use the trtexec command inside the container

```
<model-repository-path>/
    <model-name>/
        [config.pbtxt]
        [<output-labels-file> ...]
        <version>/
            <model-definition-file>
        <version>/
            <model-definition-file>
        ...
    <model-name>/
        [config.pbtxt]
        [<output-labels-file> ...]
        <version>/
            <model-definition-file>
        <version>/
            <model-definition-file>
        ...
    ...
```

model directory should be in this format

2. Run tritonserver with the TensorRT model

- you may use the singulairty iamge that I built on the KISTI server (/scratch/kedu05/leecy/project1/triton.sif) or build your own singulairty image with the corrsesponding docker image)
- Run the singularity image with “singularity run --nv triton.sif”

tritonserver --model-repository={**/path/to/your/model**} --allow-http=true --http-port=50712 --allow-grpc=true --grpc-port=50722 --allow-metrics=true --metric-port=50723 **(SPECIFY YOUR PORT NUMBER)**

- Port in singularity container will be automatically mapped to serverside port without specifying port numbers with --net flag

3. Client side request to the tritonserver

- You may run client side code with tritonserver_client both in the local environment as long as you connected your local port to the KISTI server by portforwarding. You can also run it on the KISTI server
- portforward both grpc port and http port (ex: ssh -L localhost:8000:gpu22:50712 kedu05@neuron.ksc.re.kr)
- I used docker environment in my local computer.
- docker pull nvcr.io/nvidia/tritonserver:22.01-py3-sdk
- docker run -it --rm --net=host -v $(pwd):/hello nvcr.io/nvidia/tritonserver:22.01-py3-sdk (mounted local volume on the container to access to the inference dataset)
- run client.py (your code) in the docker container

```python
import tritonclient.http as tritonhttpclient
import tritonclient.grpc as tritongrpcclient
import numpy as np
# from PIL import Image
# from torchvision import transforms
import json

###CONFIGURATION########
VERBOSE = False
input_name = 'modelInput'
input_shape = (1, 1, 128, 128, 128)
input_dtype = 'FP32'
output_name = 'modelOutput'
model_name = 'amyProg'
http_url = 'localhost:8000'
grpc_url = 'localhost:8001'
model_version = '1'
########################




image = np.load('/hello/target_nor.npy')
image = np.expand_dims(image, axis=0)
image = np.expand_dims(image, axis=0)
image = image.astype('float32')

# triton_client = tritonhttpclient.InferenceServerClient(url=http_url, verbose=VERBOSE)
triton_client = tritongrpcclient.InferenceServerClient(url=grpc_url, verbose=VERBOSE)
# model_metadata = triton_client.get_model_metadata(model_name=model_name, model_version=model_version) #You can remove this line
model_config = triton_client.get_model_config(model_name=model_name, model_version=model_version)

#input0 = tritonhttpclient.InferInput(input_name, input_shape, input_dtype)
input0 = tritongrpcclient.InferInput(input_name, input_shape, input_dtype)
#input0.set_data_from_numpy(image_numpy, binary_data=False)
input0.set_data_from_numpy(image)

#output = tritonhttpclient.InferRequestedOutput(output_name, binary_data=False)
output = tritongrpcclient.InferRequestedOutput(output_name)
response = triton_client.infer(model_name, model_version=model_version,
                               inputs=[input0], outputs=[output])
logits = response.as_numpy(output_name)
logits = np.asarray(logits, dtype=np.float32)

print(logits)
```

client.py