https://infohub.delltechnologies.com/p/running-the-mlperf-tm-inference-v1-0-benchmark-on-dell-emc-systems/

https://github.com/mlperf

git clone <https://github.com/mlperf/inference_results_v0.7.git>

git clone <https://github.com/mlperf/training_results_v0.6.git>

git clone <https://github.com/mlperf/inference_results_v1.1.git>

**2 Prerequisites**

Prerequisites for running the MLPerf inference v1.0 tests include:

An x86\_64 system

Docker installed with the NVIDIA runtime hook

Ampere-based NVIDIA GPUs (Turing GPUs include legacy support, but are no longer maintained for optimizations)

NVIDIA Driver Version 455.xx or later

ECC set to ON

To set ECC to ON, run the following command:

sudo nvidia-smi --ecc-config=1

$/home/training\_results\_v0.6$ ls

Alibaba CONTRIBUTING.md Fujitsu Google Intel LICENSE NVIDIA README.md

$/home/training\_results\_v0.6$ cd NVIDIA

$/home/training\_results\_v0.6/NVIDIA$ ls

benchmarks LICENSE.md README.md results systems

$/home/training\_results\_v0.6/NVIDIA$ cd benchmarks; ls

gnmt maskrcnn minigo resnet ssd transformer

$/home/training\_results\_v0.6/NVIDIA$ cd gnmt/implementations; ls

download\_dataset.sh pytorch verify\_dataset.sh

$/home/training\_results\_v0.6/NVIDIA/benchmarks/gnmt/implementations$ bash download\_dataset.sh

$/home/training\_results\_v0.6/NVIDIA/benchmarks/gnmt/implementations$ bash verify\_dataset.sh

OK: correct data/train.tok.clean.bpe.32000.en

OK: correct data/train.tok.clean.bpe.32000.de

OK: correct data/newstest\_dev.tok.clean.bpe.32000.en

OK: correct data/newstest\_dev.tok.clean.bpe.32000.de

OK: correct data/newstest2014.tok.bpe.32000.en

OK: correct data/newstest2014.tok.bpe.32000.de

OK: correct data/newstest2014.de

$/home/training\_results\_v0.6/NVIDIA/benchmarks/gnmt/implementations$ ls

data download\_dataset.sh logs pytorch verify\_dataset.sh

$/home/training\_results\_v0.6/NVIDIA/benchmarks/gnmt/implementations$cd pytorch; ls -l

bind\_launch.py

config\_DGX1\_multi.sh

config\_DGX1.sh

config\_DGX2\_multi\_16x16x32.sh

config\_DGX2\_multi.sh

config\_DGX2.sh

Dockerfile

LICENSE

mlperf\_log\_utils.py

preprocess\_data.py

README.md

requirements.txt

run\_and\_time.sh

run.sub

scripts

seq2seq

setup.py

train.py

translate.py

$training\_results\_v0.6/NVIDIA/benchmarks/gnmt/implementations/pytorch$cp config\_DGX1.sh config\_SUT.sh

Edit config\_SUT.sh to reflect your system config

## System config params

DGXNGPU=4

DGXSOCKETCORES=20

DGXHT=2 # HT is on is 2, HT off is 1

DGXIBDEVICES=''

DGXNSOCKET=2

BIND\_LAUNCH=1

docker build -t <docker/registry>/mlperf-nvidia:rnn\_translator .

docker push <docker/registry>/mlperf-nvidia:rnn\_translator

nvidia-docker run -it --rm mlperf-nvidia:rnn\_translator

root@4e944d91164e:/workspace/rnn\_translator# ls -l \*.sh

config\_DGX1.sh

config\_DGX1\_multi.sh

config\_DGX2.sh

config\_DGX2\_multi.sh

config\_DGX2\_multi\_16x16x32.sh

config\_SUT.sh

run\_and\_time.sh

git clone <https://github.com/mlcommons/inference_results_v1.0>

**>>DOCKER:-**

**1>Install using the apt repository**

**sudo apt-get update**

**sudo apt-get install ca-certificates curl gnupg curl python3-pip apt-transport-https ca-certificates gnupg-agent software-properties-common -y**

**sudo install -m 0755 -d /etc/apt/keyrings**

**curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg**

**sudo chmod a+r /etc/apt/keyrings/docker.gpg**

**echo \**

**"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \**

**"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \**

**sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**

**sudo apt-get update**

**apt list -a docker-ce**

**sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**

**apt install docker-ce=<VERSION> docker-ce-cli=<VERSION> containerd==1.3.3-0ubuntu2**

**apt-get install containerd****=1.3.3-0ubuntu2**

**apt-get install docker.io**

**docker run hello-world**

**docker rmi -f 9c7a54a9a43c // to remove the image**

**2>Uninstall Docker Engine**

1. sudo apt-get purge docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-ce-rootless-extras
2. sudo rm -rf /var/lib/docker
3. sudo rm -rf /var/lib/containerd

**3> Install NVIDIA Container Toolkit**

**distribution=$(. /etc/os-release;echo $ID$VERSION\_ID) \**

**&& curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \**

**&& curl -s -L https://nvidia.github.io/libnvidia-container/$distribution/libnvidia-container.list | \**

**sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \**

**sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list**

**sudo apt-get update \**

**&& sudo apt-get install -y nvidia-container-toolkit-base**

**nvidia-ctk --version**

**sudo nvidia-ctk cdi generate --output=/etc/cdi/nvidia.yaml**

**grep " name:" /etc/cdi/nvidia.yaml**

**sudo apt-get install -y nvidia-container-toolkit nvidia-docker2**

**docker --version**

**systemctl status docker**

**systemctl start docker**

**nvidia-ctk runtime configure --runtime=docker**

**apt-mark hold docker-ce**

**sudo systemctl restart docker**

**docker run -it 3286289241d7**

**nvidia-docker run -it 3286289241d7 bash**

**https://hub.docker.com/r/nvidia/cuda/tags?page=2&name=ubuntu20.04**

**docker pull nvidia/cuda:11.8.0-devel-ubuntu20.04 ##{copy}**

**https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/containerd.io\_1.2.13-2\_amd64.deb**

**MLPERF :-**

git clone <https://github.com/mlperf/inference_results_v0.7.git>

cd inference\_results\_v1.0/closed/NVIDIA

vim Makefile

132 **BASE\_IMAGE := nvidia/cuda:11.8.0-devel-ubuntu20.04**

mkdir scratch

docker images ###empty

make prebuild

export MLPERF\_SCRATCH\_PATH=/home/user/inference\_results\_v1.0/closed/NVIDIA/scratch

**>>Set up the configuration file**

cd inference\_results\_v1.0/closed/NVIDIA/configs/resnet50/Server/

Notes-

1. R7525 server with a one-A100 GPU writes as R7525\_A100-PCIe-40GBx1
2. QPS parameter (server\_target\_qps) to match the number of GPUs QPS = number of GPUs x QPS per GPU.
3. server\_target\_qps set to 26000 in accordance with one GPU performance
4. https://en.wikipedia.org/wiki/List\_of\_Nvidia\_graphics\_processing\_units

vim config.json

"R7525\_A100-PCIe-40GBx2": {

         "config\_ver": {

         },

         "deque\_timeout\_us": 2000,

         "gpu\_batch\_size": 64,

         "gpu\_copy\_streams": 4,

         "gpu\_inference\_streams": 3,

         "server\_target\_qps": 52000,

         "use\_cuda\_thread\_per\_device": true,

         "use\_graphs": true

     },

This example shows the modified configuration for one GPU:

"R7525\_A100-PCIe-40GBx1": {

         "config\_ver": {

         },

         "deque\_timeout\_us": 2000,

         "gpu\_batch\_size": 64,

         "gpu\_copy\_streams": 4,

         "gpu\_inference\_streams": 3,

         "server\_target\_qps": 26000,

         "use\_cuda\_thread\_per\_device": true,

         "use\_graphs": true

     },

**>>Add the new system to the list of available systems**

cd inference\_results\_v1.0/closed/NVIDIA/code/common/

Where:

For <system ID>, enter the system ID with which you want to identify this system.

For <list of names reported by nvidia-smi>, run the nvidia-smi -L command and use the name that is returned.

For <known PCI IDs of this system>, run the following command:

$ CUDA\_VISIBLE\_ORDER=PCI\_BUS\_ID nvidia-smi --query-gpu=gpu\_name,pci.device\_id --format=csv

vim system\_list.py

system\_list = ([

( "X12DAI-N6\_A4000-PCIe-16GB", "NVIDIA RTX A4000","24B0","Architecture.Ampere",1)

])

system\_list = ([

("X11SPM-F\_T400-PCIe-2GBx1", "NVIDIA T400","1FB2",0),

])

Apt update

vim Makefile

128 ifndef BASE\_IMAGE

129 ifeq ($(shell bash $(PROJECT\_ROOT)/scripts/check\_intranet.sh),0)

130 BASE\_IMAGE := gitlab-master.nvidia.com/compute/mlperf-inference:$(DOCKER\_IMAGE\_NAME)

131 else

132 BASE\_IMAGE := nvidia/cuda:11.5-cudnn8-devel-ubuntu20.04

nvidia/cuda 11.5.0-runtime-ubuntu20.04

docker run --rm --gpus all nvidia/cuda:11.7.0-cudnn8-runtime-ubuntu20.04 nvidia-smi

root@user:/opt/inference\_results\_v1.0/closed/NVIDIA/docker# cp -r Dockerfile /opt/inference\_results\_v0.7/closed/NVIDIA/docker/

vim docker/Dockerfile

python-dev libffi-dev python-numpy python-dev python-setuptools libboost- libboost-thread-dev \

37 # remove python-pip

52 RUN python2 -m pip install --upgrade setuptools \

53 && python2 -m pip install absl-py==0.7.1

56 && python2 -m pip install wheel==0.35.1 \

58 && python3 -m pip install pillow==6.0.0

90 RUN python3 -m pip install tensorflow==1.13.1

93 RUN python3 -m pip install torch==1.4.0+cpu torchvision==0.5.0+cpu

94 && python3 -m pip install onnx==1.6.0

95 && python3 -m pip install pycuda==2019.1.2

&& python3 -m pip install Cython==0.29.10

52 RUN python3 -m pip install --upgrade setuptools \

53 && python3 -m pip install absl-py==0.7.1

56 && python3 -m pip install wheel==0.35.1 \

58 && python3 -m pip install pillow==7.0.0

90 RUN python3 -m pip install tensorflow==2.2.0

93 RUN python3 -m pip install torch==1.4.0 torchvision==0.5.0

94 && python3 -m pip install onnx==1.8.0

&& python3 -m pip install numpy

94 && python3 -m pip install cuvarbase \

95 && python3 -m pip install pycuda==2022.2.2 \ error install pyproject.toml-based projects

pip install Cython --install-option="--no-cython-compile"

go this side <http://mirror.cs.uchicago.edu/nvidia/cuda/ubuntu/focal/>

157 ARG CUDNN\_DEB\_URL=https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86\_64/

158 RUN cd /tmp \

159 && install\_deb\_pkg() { wget $CUDNN\_DEB\_URL/$1 -O $1 && dpkg -i $1 && rm $1; } \

160 && install\_deb\_pkg libcudnn8\_8.9.2.26-1+cuda11.8\_amd64.deb \

161 ## libcudnn8\_8.1.1.33-1+cuda11.2\_amd64.deb \

162 && install\_deb\_pkg libcudnn8-dev\_8.9.2.26-1+cuda11.8\_amd64.deb \

163 ## libcudnn8-dev\_8.1.1.33-1+cuda11.2\_amd64.deb\

164 && unset -f install\_deb\_pkg

166 # Install TRT 7.2

167 ARG TRT\_DEB\_URL=http://mirror.cs.uchicago.edu/nvidia/cuda/ubuntu/focal/

168 ##https://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1804/x86\_64

169 RUN cd /tmp \

170 && install\_deb\_pkg() { wget $TRT\_DEB\_URL/$1 -O $1 && dpkg -i $1 && rm $1; } \

171 && install\_deb\_pkg libnvinfer8\_8.6.1.6-1+cuda11.8\_amd64.deb \

172 ## libnvinfer7\_7.2.3-1+cuda11.2\_amd64.deb \

173 && install\_deb\_pkg libnvinfer-plugin8\_8.6.1.6-1+cuda11.8\_amd64.deb \

174 ## libnvinfer-plugin7\_7.2.3-1+cuda11.2\_amd64.deb \

175 && install\_deb\_pkg libnvparsers8\_8.6.1.6-1+cuda11.8\_amd64.deb \

176 ## libnvparsers7\_7.2.3-1+cuda11.2\_amd64.deb \

177 && install\_deb\_pkg libnvonnxparsers8\_8.6.1.6-1+cuda11.8\_amd64.deb \

178 ## libnvonnxparsers7\_7.2.3-1+cuda11.2\_amd64.deb \

179 && install\_deb\_pkg libnvinfer-dev\_8.6.1.6-1+cuda11.8\_amd64.deb \

180 ## libnvinfer-dev\_7.2.3-1+cuda11.2\_amd64.deb \

181 && install\_deb\_pkg libnvinfer-plugin-dev\_8.6.1.6-1+cuda11.8\_amd64.deb \

182 ## libnvinfer-plugin-dev\_7.2.3-1+cuda11.2\_amd64.deb \

183 && install\_deb\_pkg libnvparsers-dev\_8.6.1.6-1+cuda11.8\_amd64.deb \

184 ## libnvparsers-dev\_7.2.3-1+cuda11.2\_amd64.deb \

185 && install\_deb\_pkg libnvonnxparsers-dev\_8.6.1.6-1+cuda11.8\_amd64.deb \

186 ## libnvonnxparsers-dev\_7.2.3-1+cuda11.2\_amd64.deb \

187 && install\_deb\_pkg python3-libnvinfer\_8.6.1.6-1+cuda11.8\_amd64.deb \

188 ## python3-libnvinfer\_7.2.3-1+cuda11.2\_amd64.deb \

189 && install\_deb\_pkg python3-libnvinfer-dev\_8.6.1.6-1+cuda11.8\_amd64.deb \

190 ## python3-libnvinfer-dev\_7.2.3-1+cuda11.2\_amd64.deb \

191 && unset -f install\_deb\_pkg

/

nvidia-docker run --rm -it -w /work \

-v /opt/inference\_results\_v1.1/closed/NVIDIA:/work -v /root:/mnt//root \

--cap-add SYS\_ADMIN --cap-add SYS\_TIME \

-e NVIDIA\_VISIBLE\_DEVICES=all \

--shm-size=32gb \

-v /etc/timezone:/etc/timezone:ro -v /etc/localtime:/etc/localtime:ro \

--security-opt apparmor=unconfined --security-opt seccomp=unconfined \

--name mlperf-inference-root-x86\_64 -h mlperf-inference-root-x86\_64 --add-host mlperf-inference-root-x86\_64:127.0.0.1 \

--user 0:0 --net host --device /dev/fuse \

\

-e MLPERF\_SCRATCH\_PATH= \

-e HOST\_HOSTNAME=user \

\

mlperf-inference:root-x86\_64