NEUCHIPS

MLPerf™ Inference v3.0 NEUCHIPS Implementation for DLRM Benchmark

Quick Start Guide

ML30QSG-v01 | April 28, 2023

Document History

ML30QSG-v01

Version	Date	Authors	Description of Change
01	28th April, 2023	Chance	First release.

Table of Contents

This is a repository of NEUCHIPS implementations for MLPerf™ Inference Benchmark v3.0. This README is a quick start guide introducing how to execute RecAccel™ N3000 for DLRM benchmark.

NEUCHIPS Submission

NEUCHIPS submits RecAccel™ N3000 inference results to 'closed' and 'closed-power' division for DLRM benchmarks. The implementations are in `closed/Neuchips/code`.

System Processing

System Dependencies

The system for running the inference is based on following configurations:

Item	Description
Server Machine	Gigabyte G482-Z54
Host Processor	AMD EPYC 7452
Storage Space	more than 4TB
Memory Size	more than 256GB
Operating System	Ubuntu 22.04.1
Linux Kernel	v5.18.17
RecAccel™ N3000 PCIe Version	v1.0
RecAccel™ Device Driver	v1.0
RecAccel™ N3000 Firmware	v1.0

NUMA Configuration

The AMD EPYC 7452 processor supports Non-Uniformed Memory Access (NUMA). In this submission, the system processor is partitioned into four NUMA domains (NPS4) to get the best use of 8 RecAccel™ N3000 PCIe cards. For further information, please refer to the description from AMD EPYC HPC Tuning Guide.

Setting up Storage Spaces

Once you have obtained an available space (more than 4TB), set the `MLPERF_SCRATCH_PATH` environment variable. This is the space that code tracks where the data is stored. The default variable of MLPERF_SCRATCH_PATH is `/home/mlperf_inference_data`, if this environment variable is not set.

```
$ export MLPERF_SCRATCH_PATH=/path/to/myspace
```

Then create empty directories in your storage to store the data:

```
$ mkdir $MLPERF_SCRATCH_PATH/data $MLPERF_SCRATCH_PATH/models
$MLPERF_SCRATCH_PATH/preprocessed_data
```

After you have done so, you will need to download the models and datasets, and run the preprocessing scripts on the datasets.

```
$ echo $MLPERF_SCRATCH_PATH  # Make sure that the
MLPERF_SCRATCH_PATH has set correctly
$ ls -al $MLPERF_SCRATCH_PATH  # Make sure that the scratch space is
mounted correctly
$ make clean  # Make sure that the build/ directory isn't dirty
$ make link_dirs  # Link the build/ directory to the scratch space
$ ls -al build/  # You should see output like the following:
total 8
drwxrwxr-x  2  neuchips neuchips 4096  Mar 2 09:23 .
drwxrwxr-x 15 neuchips neuchips 4096  Mar 2 09:23 ..
lrwxrwxrwx  1  neuchips neuchips   35   Mar 2 09:23 data ->
$MLPERF_SCRATCH_PATH/data
lrwxrwxrwx  1  neuchips neuchips   37   Mar 2 09:23 models ->
$MLPERF_SCRATCH_PATH/models
lrwxrwxrwx  1  neuchips neuchips   48   Mar 2 09:23 preprocessed_data ->
$MLPERF_SCRATCH_PATH/preprocessed_data
```

Once you have verified that the `build/data, build/models/,` and `build/preprocessed_data` point to the correct directories in your scratch space, you can continue.

Download Datasets

For processing the inference, you need to download the Criteo Terabyte dataset and unzip the files to `$MLPERF_SCRATCH_PATH/data/criteo/`. After the files unzip, you can use following instruction to setup the benchmarks to dlrm.
`$make download_data BENCHMARKS="dlrm"`

Note that if the dataset already exists, the script will print out a message confirming that the directory structure is as expected.

Download Model Files for RecAccel™ N3000

NEUCHIPS uses MLCommons DLRM model for closed-division inferencing. You can manually download the files from the MLCommons Github to check. In this repo, we compile MLCommons DLRM model to RecAccel™ N3000 setup files in the `./download_aie/` folder. You can follow the instructions as below for download and setup of model to the RecAccel™ N3000 PCIe card(s).
(You can also check readme.md in `./download_aie/` directory.)

1. `$cd ./download_aie/`

2. Download DLRM model data from remote storage
   ```
   wget
   https://zenodo.org/record/7672349/files/n3000.dlrm.tar.gz?download=1
   ```
3. Create ./model_data folder `$mkdir model_data`
4. Extract the .gz file to ./model_data
   `tar xzf n3000.dlrm.tar.gz -C ./model_data/`

5. Download to RecAccel™ PCIe cards For single card, please use following instruction to set the card: $./n3000_dl 0
   For 8 cards, please use following instructions to set those cards:
   $./n3000_dl 0
   $./n3000_dl 1
   $./n3000_dl 2
   $./n3000_dl 3
   $./n3000_dl 4
   $./n3000_dl 5
   $./n3000_dl 6
   $./n3000_dl 7

Preprocessing Datasets for Inference

A preprocessing of original MLPerf™ dataset for evaluating is needed to continue the inferencing work. In this section, we will do following works:

- Converting the data format to INT8
- Converting the raw data to NumPy arrays

```
$ make preprocess_data BENCHMARKS="dlrm"
```

NEUCHIPS Submission

Running NEUCHIPS DLRM Benchmark

Enter `closed/Neuchips.` From now on, all of the commands detailed in this guide should be executed from this directory. This directory contains our submission code for the MLPerf Inference Closed Division.

Building Binaries

```
$ make build_dlrm_n3000
```

- This command will execute following subjects:

1. Sets up symbolic links to the models, datasets, and preprocessed datasets in the MLPerf Inference scratch space in `build/`
2. Pulls the specified hashes for the subrepositories in our repo:
    i. MLCommons Inference Repo (Official repository for MLPerf Inference tools, libraries, and references)
3. Builds all necessary binaries for the system.

Running DLRM Benchmark

NEUCHIPS provides 2 types of DLRM benchmark, **closed** and **closed-power**. The closed-power would need additional setup for power meter and its ssh server to get detail power information during the run. Please refer to related section for guiding you how to make the run.

Closed

The usage of run_harness command is as below:

```
$ make run_harness RUN_ARGS= [--benchmarks=dlrm]
                          [--scenarios=offline | server]
                          [--test_mode=PerformanceMode | AccuracyMode]
                  NEU_CARDS=["1" | "8"]
```

You can run the harness in `offline` using the `--scenarios` flag. Example for run offline and performance mode:

```
$ make run_harness RUN_ARGS="--benchmarks=dlrm --scenarios=offline --
test_mode=PerformanceMode" NEU_CARDS="8"
```

You can run the harness in `AccuracyMode` using the `--test_mode` flag. Example for run server and accuracy mode:

```
$ make run_harnes RUN_ARGS="--benchmarks=dlrm --scenarios=server --
test_mode=AccuracyMode" NEU_CARDS="8"
```

Closed – Power

Closed-power division requires power measurement by compliant power meter. You can check the power measurement setup in MLCommons power-dev repo. Anyone for checking the repo must follow the MLCommons instruction to sign PTD EULA license agreement and join the Power WG.

In NEUCHIPS submission, we clone the 'power-dev repo' to `~/power-dev` for this result submission.

System Setup for Power Measurement

For getting the detail power information during the inference, we have to install PTDaemon in the inference machine. The PTDaemon executable file is located in a private repo: https://github.com/mlcommons/power and submitters must join the Power WG and sign the PTD EULA license agreement to get it.
The MLCommons power-dev repo is cloned in `~/power-dev` and is on the correct branch for the MLPerf Inference version (i.e. r3.0 for MLPerf Inference v3.0)
There exists an administrator user lab with password labuser. If your administrator account has different login information, `set POWER_SERVER_USERNAME and POWER_SERVER_PASSWORD in closed/Neuchips/Makefile` to the correct credentials.
In NEUCHIPS's submission, we use the Yokogawa WT332 meter in single channel mode.
To get the detail power data of the meter, you need to set another server and to connect it with Yokogawa WT332 meter by USB cable. In our case, the OpenSSH server is installed and enabled on the machine, listening on port 22.
Set the power meter configuration in `power/server-neuchips-linux.cfg.`

Running Close-Power Benchmark

Instead of `make run_harness,` please use `make run_harness_power` to get the power-close result. The usage of close-power instruction is as below. And each instruction can only run a certain combination of offline scenario or server scenario :

```
$ make run_harness_power RUN_ARGS= [--benchmarks=dlrm]
                                  [--scenarios=offline | server]
                                  [--test_mode=PerformanceMode]
                        NEU_CARDS="8"
```

With this make target, LoadGen logs will be located in `build/power_logs instead of build/logs.`
You can run different scenarios by using the `--scenarios` flag. Here comes the example for running offline scenario and performance mode:

```
$ make run_harness_power RUN_ARGS="--benchmarks=dlrm --scenarios=offline
--test_mode=PerformanceMode" NEU_CARDS="8"
```

When `make run_harness_power` is called, the script runs the harness twice:

1. First run is called the `ranging run`, which is used to gather the maximum voltage and current that the system consumes so as to configure the power meter correctly.
2. Second run is called the `testing run`, which actually collects the power readings.

After second run is finished. The logs in `build/power_logs` will be automatically copied to `results/` directory if the run is valid.

How Do I Know the Run is Finished?

The latest section for the run is checking the accuracy. Once `Accuracy test PASSED` is shown, it means the run is finished.

How Do I View the Logs of My Previous Runs?

The logs will be saved to `build/logs/[timestamp]/N3000_CPU_2S_Neuchips/...` every time. You can follow the folder naming rule to check your previous runs.

How to Run INT8 Calibration?

You can check the `calibration_process.adoc` file we shared to know our calibration method.

Appendix

The `neuchips_ai_ep.ko` is the RecAccel™ N3000 Linux driver for this submission, its hash value of SHA256 is `238c67c34a3c55cd5721570b22737fc56249bc295e3e279e227ecc58778b6402.`

NEUCHIPS Inc. | 220 State Street, Los Altos, CA 94022

https://www.neuchips.ai/