GPU User Guide

Release 2.0

Jérôme Pansanel

5 décembre 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

1.1 Using the GPU

This documentation details the installation of NVIDIA tools to use GPU efficiently within a virtual machine deployed on the SCIGNE plateform. It covers also the installation of tools for artificial intelligence-based analyses.

It is based on the use of an up-to-date Ubuntu 22.04 image.

1.2 GPU Flavors

There are three flavors with a GPU:

- g1.xlarge-4xmem (73730dea-ca08-47fb-ac0b-2ebd6dbe1465)
- g2.xlarge-4xmem (242a578e-b314-4f33-83ea-b05f50f08960)
- g4.xlarge-4xmem (00b54b02-63d0-4d15-927d-6e2f5a4d4920)

All flavors have 8 cores, 64 GB of RAM and 40 GB of disks. They differ in the type of GPU offered:

Name	GPU
g1.xlarge-4xmem	NVIDIA Tesla P100
g2.xlarge-4xmem	NVIDIA GeForce RTX 2080 Ti
g4.xlarge-4xmem	NVIDIA Tesla T4

If you need a large disk space, it is recommended to create a volume with Cinder and attache it to the server. This point is covered in the OpenStack User Guide.

CHAPTER

TWO

NVIDIA TOOLS INSTALLATION

In order to use the GPU available in the virtual machine effectively, you need to install the GPU driver and the CUDA GPU driver and CUDA tools.

2.1 NVIDIA Repository Configuration

Although NVIDIA drivers and tools are available in the Ubuntu repositories, we recommend that you use the NVIDIA repository to benefit from the latest versions:

```
$ sudo wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64/
↪cuda-keyring_1.1-1_all.deb
$ sudo dpkg -i cuda-keyring_1.1-1_all.deb
$ sudo apt update
```

It is necessary to install of all required software for the compilation of NVIDIA drivers and dependencies:

```
$ sudo apt install -y gcc g++ ubuntu-drivers-common libglfw3-dev libfreeimage-dev \
  alsa-utils
```

2.2 NVIDIA Driver

First, the list of available drivers is displayed with:

```
$ sudo ubuntu-drivers devices
== /sys/devices/pci0000:00/0000:00:05.0 ==
modalias : pci:v000010DEd00001EB8sv000010DEsd000012A2bc03sc02i00
vendor   : NVIDIA Corporation
model    : TU104GL [Tesla T4]
driver   : nvidia-driver-470 - distro non-free recommended
driver   : nvidia-driver-450-server - distro non-free
driver   : nvidia-driver-418-server - distro non-free
driver   : nvidia-driver-535-server - distro non-free
driver   : nvidia-driver-525-server - distro non-free
driver   : nvidia-driver-470-server - distro non-free
driver   : xserver-xorg-video-nouveau - distro free builtin
```

This list may vary depending on Ubuntu system updates and will probably differs from the above result when you run the command. The driver whose line ends with the *recommended* tag is installed with the following command:

```
$ sudo apt install -y nvidia-driver-470
```

This version will probably be updated when you install CUDA. To take into account the updates and the installation of the new driver, the server is restarted with:

```
$ sudo shutdown -r now
```

Once the server is restarted, the driver installation is verified with:

```
$ cat /proc/driver/nvidia/version
NVRM version: NVIDIA UNIX x86_64 Kernel Module  470.223.02  Sat Oct  7 15:39:11 UTC 2023
GCC version:  gcc version 11.4.0 (Ubuntu 11.4.0-1ubuntu1~22.04)

$ sudo nvidia-smi
Wed Dec  6 07:13:55 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.223.02   Driver Version: 470.223.02   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:05.0 Off |                    0 |
| N/A   53C    P0    27W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

2.3 CUDA

Installing CUDA is relatively straightforward, but requires you to specify the version of CUDA you want to use. PyTorch and TensorFlow software do not work with the latest version of CUDA:

```
$ sudo apt install -y cuda=11.8.0-1
$ sudo apt-mark hold cuda
```

The installation of CUDA is verified with:

```
$ /usr/local/cuda/bin/nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
```

2.4 Verification

We strongly recommend you to check that CUDA works. To do this, the tools provided by CUDA are used through the following commands:

```
$ /usr/local/cuda/extras/demo_suite/deviceQuery
/usr/local/cuda/extras/demo_suite/deviceQuery Starting...

 CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Tesla T4"
  CUDA Driver Version / Runtime Version          12.3 / 11.8
  CUDA Capability Major/Minor version number:    7.5
  Total amount of global memory:                 15110 MBytes (15843721216 bytes)
  (40) Multiprocessors, ( 64) CUDA Cores/MP:     2560 CUDA Cores
  GPU Max Clock rate:                            1590 MHz (1.59 GHz)
  Memory Clock rate:                             5001 Mhz
  Memory Bus Width:                              256-bit
  L2 Cache Size:                                 4194304 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536),␣
↪3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1024
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 3 copy engine(s)
  Run time limit on kernels:                     No
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Enabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 0 / 5
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device␣
↪simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.3, CUDA Runtime Version = 11.
↪8, NumDevs = 1, Device0 = Tesla T4
Result = PASS
```

```
$ /usr/local/cuda/extras/demo_suite/bandwidthTest

[CUDA Bandwidth Test] - Starting...
Running on...

 Device 0: Tesla T4
 Quick Mode

 Host to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     12748.0

 Device to Host Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     12851.4

 Device to Device Bandwidth, 1 Device(s)
 PINNED Memory Transfers
   Transfer Size (Bytes)        Bandwidth(MB/s)
   33554432                     239369.8

Result = PASS

NOTE: The CUDA Samples are not meant for performance measurements. Results may vary when␣
↪GPU Boost is enabled.
```

In both cases, the result should be *PASS*.

CHAPTER

THREE

ARTIFICIAL INTELLIGENCE TOOLS

The flavor providing GPU can be used with softaware like TensorFlow or PyTorch to accelerate the execution of artificial intelligence-related codes.

This section describes the installation of TensorFlow, PyTorch, as well as all required dependencies.

3.1 Python Tools

TensorFlow and PyTorch are made available as Python libraries. In order to use independent environments, and to simplify the installation of this tools, we recommend to use virtual environments, like **virtualenv** or **conda**. In this documentation, we will cover the use of **virtualenv**. The **pip** tool is also used, as it permits to install all the packages.

```
$ sudo apt install -y python3-pip python3-testresources python3-virtualenv
```

3.2 cuDNN

To take advantage of GPU acceleration for machine learning, you need to install the cudaNN library. It can be installed with the following commands:

```
$ sudo apt install -y libcudnn8
$ echo 'export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH' >> ~/.bashrc
$ source ~/.bashrc
```

3.3 Tensorflow

Installing TensorFlow and its NVIDIA dependencies is fairly straightforward by using a virtual environment and **pip**:

```
$ virtualenv tf
$ source tf/bin/activate
(tf) $ pip install --extra-index-url https://pypi.nvidia.com tensorrt-bindings==8.6.1␣
↪tensorrt-libs==8.6.1
(tf) $ pip install tensorflow==2.9.3
```

The installation is verified with:

```
(tf) $ python3 -c "from tensorflow.python.client import device_lib; device_lib.list_
↪local_devices()"
2023-12-06 08:57:53.061432: I tensorflow/core/platform/cpu_feature_guard.cc:193] This␣
↪TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use␣
↪the following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler␣
↪flags.
2023-12-06 08:57:53.611112: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2023-12-06 08:57:53.633516: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2023-12-06 08:57:53.633861: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2023-12-06 08:57:53.693124: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2023-12-06 08:57:53.693424: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2023-12-06 08:57:53.693676: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:975]␣
↪successful NUMA node read from SysFS had negative value (-1), but there must be at␣
↪least one NUMA node, so returning NUMA node zero
2023-12-06 08:57:53.693906: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1532]␣
↪Created device /device:GPU:0 with 13956 MB memory:  -> device: 0, name: Tesla T4, pci␣
↪bus id: 0000:00:05.0, compute capability: 7.5
```

3.4 PyTorch

PyTorch is installed using the commands below. The version of the various tools to be installed can be found at on the PyTorch site:

```
$ virtualenv torch
$ source torch/bin/activate
(torch) $ pip3 install torch torchvision torchaudio \
          --index-url https://download.pytorch.org/whl/cu118
```

The installation is verified with:

```
(torch) $ python3 -c "import torch; print(torch.rand(5, 3))"
tensor([[0.7061, 0.2096, 0.7064],
        [0.6548, 0.5531, 0.8631],
        [0.2996, 0.8940, 0.7171],
        [0.8309, 0.1354, 0.1753],
        [0.1878, 0.3823, 0.6253]])

(torch) $ python3 -c "import torch; print(torch.cuda.is_available())"
True
```

CHAPTER

FOUR

COMPLEMENTARY DOCUMENTATION

The following websites can be consulted for further information about the installation of the tools presented in this documentation:

- TensorFlow Installation Guide
- PyTorch Installation Guide