Guide d'utilisation des GPUs

Version 2.0

Jérôme Pansanel

5 décembre 2023

Contents

CHAPTER 1

Introduction

1.1 Utilisation des GPUs

Cette documentation détaille l'installation des outils NVIDIA pour utiliser les GPUs au sein d'une machine virtuelle déployée sur le service de Cloud Computing de la plateforme SCIGNE, ainsi que l'installation d'outils pour réaliser des analyses basées sur l'intelligence artificielle.

Elle se base sur l'utilisation d'une image Ubuntu 22.04 à jour.

1.2 Gabarits GPU

Il existe trois gabarits disposants d'un 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)

Tous les gabarits ont 8 cœurs, 64 Go de RAM et 40 Go de disque, et se différencient par les GPUs proposés :

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

Pour les besoins importants de disque, il est conseillé de créer un volume avec Cinder et de l'attacher au serveur. Ce point est détaillé dans le guide d'utilisation d'OpenStack.

CHAPTER 2

Installation des outils NVIDIA

Afin d'exploiter efficacement le GPU disponible dans la machine virtuelle, il est nécessaire d'installer le pilote de la carte GPU et les outils CUDA.

2.1 Configuration du dépôt NVIDIA

Bien que les pilotes et outils NVIDIA soient disponibles dans les dépôts Ubuntu, nous recommandons d'utiliser le dépôt NVIDIA afin de bénéficier des dernières 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
```

Il est nécessaire d'installer les logiciels nécessaires à la compilation des drivers et dépendances NVIDIA :

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

2.2 Driver NVIDIA

Dans un premier temps, la liste des pilotes disponibles est affichée avec :

```
$ 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
```

(suite sur la page suivante)

(suite de la page précédente)

```
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
```

Cette liste varie en fonction des mises à jour et ne correspondra probablement pas avec le résultat ci-dessus lorsque vous exécuterez cette commande. Le pilote dont la ligne se termine par *recommended* est installé avec la commande suivante :

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

Cette version sera probablement mise à jour lors de l'installation de CUDA. Pour prendre en compte les mises à jour et l'installation du nouveau pilote, le serveur est redémarré avec :

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

Après le redémarrage du serveur, l'installation du pilote est vérifiée avec :

```
$ 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

L'installation de CUDA est relativement simple, mais nécessite d'indiquer la version de CUDA que nous souhaitons utiliser. En effet, les versions actuelles de PyTorch et TensorFlow ne fonctionneint avec la toute dernière version de CUDA :

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

L'installation de CUDA est vérifiée avec :

```
$ /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 Vérification

Il est fortement recommandé de vérifier le fonctionnement de CUDA. Pour cela, les outils fournis par CUDA sont utilisés à travers les commandes suivantes :

```
$ /usr/local/cuda-11.8/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)
```

(suite sur la page suivante)

(suite de la page précédente)

```
  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.
```

Dans les deux cas, le résultat (*result*) doit être *PASS*.

CHAPTER 3

Outil pour l'intelligence artificielle

Le gabarit intégrant un GPU peut être utilisé avec des outils logiciels tel que TensorFlow ou PyTorch pour accélerer les codes d'apprentissage en intelligence artificielle.

Cette section détaille l'installation de TensorFlow, PyTorch, ainsi que les dépendances nécessaires à une utilisation optimale des gabarits proposant des GPUs NVIDIA.

3.1 Outils Python

TensorFlow et PyTorch sont disponibles via des bibliothèques Python. Afin d'utiliser des environnements indépendants, et simplifier l'installation des outils, l'usage d'un outil pour gérer les environnements virtuels, tel que **virtualenv** ou **conda**, est recommandé. Dans cette documentation, **virtualenv** est utilisé, car il est bien plus léger que conda. L'outil **pip** est également installé, puisqu'il permet d'installer les paquets TensorFlow et PyTorch.

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

3.2 cuDNN

Afin de bénéficier des accélérations sur les GPUs pour l'apprentissage automatique, il est nécessaire d'installer la bibliothèque cudaNN. vous pouvez l'installer avec les commandes suivantes :

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

3.3 Tensorflow

L'installation de TensorFlow et de dépendances NVIDIA se déroule assez simplement, en utilisant un environnement virtuel et 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
```

Le résultat est vérifié avec :

```
(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

L'installation de PyTorch est réalisée avec la commande ci-dessous. La version des différents outils à installer est obtenue sur le site de PyTorch :

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

Elle est vérifiée avec :

```
(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 4

Documentation complémentaire

Les sites suivants peuvent être consultés pour obtenir plus d'informations concernant l'installation des outils présentés dans cette documentation :

- Guide d'installation de TensorFlow
- Guide d'installation de PyTorch