Installation de machine avec RT Kernel et accélération graphique

Déploiement avec FAI-Project

https://fai-project.org/FAIme/

- Sélectionner Ubuntu
- Basculer en mode avancé en cliquant sur Toggle
- Si le mdp est laissé vide, ce sera le code du projet FAI
- Choisir un utilisateur, par ex. `etudiant`
- Attention si vous choisissez US, le clavier sera QWERTZ

Your job BYDEGIKS is currently being processed. https://fai-project.org/myimages/BYDEGIKS

Your web config:

```
type="install"
partition="ONE"
desktop="XORG"
suite="ubuntu"
keyboard="fr"
addpkgs="software-properties-common terminator nano htop wget curl gpg apt-transport-https pytho
email="robotics@hentz.eu"
rootpw=''
username="etudiant"
userpw=''
gittype=""
gituser=""
repo="deb [trusted=yes] http://download.docker.com/linux/ubuntu noble stable"
postinst="install_pc_aica.sh"
rclocal="1"
```

```
datapart=""
cmdline=""
options="STANDARD REBOOT RECOMMENDS"
```

```
curl "https://fai-project.org/cgi/faime.cgi?type=install;username=etudiant;partition=ONE;repo=http%3A%2F%2Fdownload.docker.com%2Flinux%2Fubuntu%20noble%20stable;keyboard=fr;suite=ubuntu;desktop=XORG;cl6=STANDARD;addpkgs=software-properties-common%20terminator%20nano%20htop%20wget%20curl%20gpg%20apt-transport-https%20python3-rosdep2%20ubuntu-realtime%20docker-ce%20docker-ce-cli%20containerd.io%20docker-buildx-plugin%20docker-compose-plugin;cl9=RECOMMENDS;email=robotics%40hentz.eu;postinst=install_pc_aica.sh;rclocal=1;cl8=REBOOT;sbm=2"
```

```
sudo apt install ubuntu-realtime
sudo nano /etc/default/grub
sudo update-grub
uname -v | cut -d" " -f1-4
sudo apt install cpufrequtils
sudo systemctl disable ondemand
sudo systemctl enable cpufrequtils
sudo sh -c 'echo "GOVERNOR=performance" > /etc/default/cpufrequtils'
sudo systemctl daemon-reload && sudo systemctl restart cpufrequtils
sudo systemctl status docker
cpufreq-info

wget <https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/x86_64/cuda-keyring_1.1-1_all.deb>
sudo dpkg -i cuda-keyring_1.1-1_all.deb
sudo apt update
sudo apt install nvidia-driver-pinning-570
sudo IGNORE_PREEMPT_RT_PRESENCE=1 apt install cuda-drivers

sudo apt-get update && sudo apt-get install -y --no-install-recommends    ca-certificates curl    gnupg2
sudo apt-get remove docker docker-engine docker.io containerd runc
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
# Add the repository to Apt sources:
sudo tee /etc/apt/sources.list.d/docker.sources <<EOF
Types: deb
```

```
URIs: <https://download.docker.com/linux/ubuntu>
Suites: $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")
Components: stable
Signed-By: /etc/apt/keyrings/docker.asc
EOF
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
sudo docker run hello-world
sudo usermod -aG docker $USER

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/stable/deb/nvidia-container-toolkit.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 update
sudo apt install nvidia-container-toolkit
sudo nvidia-ctk runtime configure --runtime=docker
sudo reboot
sudo docker run --rm --runtime=nvidia --gpus all ubuntu nvidia-smi

git clone <https://github.com/gautz/unistra-aica-practical> aica
cd ~/aica/package-template
docker build -f aica-package.toml -t object-detection-utils .
```

RT Kernel

RT Kernel ou Low Latency ?

https://unix.stackexchange.com/questions/553980/why-would-anyone-choose-not-to-use-the-lowlatency-kernel

Soit on build à la main (https://innovation.iha.unistra.fr/books/robotique-open-source/page/commander-un-robot-ur-avec-le-driver-ros2#bkmrk-installation-d%27ubunt), soit on prend un Kernel qui est dispo dans les dépôts

`sudo apt list *realtime* linux*rt`

```
ubuntu-realtime
linux-realtime
linux-image-6.8.1-1015-realtime
```

- On install `sudo apt install ubuntu-realtime`
- Permettre le choix du noyau au démarrage (grub)
- `sudo nano /etc/default/grub`

```
GRUB_SAVEDEFAULT=true
GRUB_DEFAULT="saved" # Le dernier Kernel choisi devient le Kernel par défaut
#GRUB_DEFAULT="Advanced options for Ubuntu>Ubuntu, with Linux 6.8.1-1015-realtime"
#GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=3 # 3 secondes pour choisir le Kernel à démarrer
```

- `sudo update-grub`

https://unix.stackexchange.com/questions/198003/set-the-default-kernel-in-grub

https://www.gnu.org/software/grub/manual/grub/html_node/Simple-configuration.html

Accélération GPU

Que ce soit pour LeRobot, pour AICA, ou en général pour utiliser des modèles d'IA, l'installation de Linux avec les bons drivers graphiques n'est pas toujours aisé.

On veut accéder dans Docker aux capacités Compute d'un GPU, essentiellement fournie par NVidia Cuda. On différentiera l'installation d'un serveur qui n'a pas besoin de l'accélération graphique liée à l'affichage, ni de capacités temps-réel, contrairement à un PC de commande de robot.

NVidia Driver et RT Kernel

https://interfacinglinux.com/2024/01/16/nvidia-cuda-on-debian-real-time-kernel/

https://gist.github.com/pantor/9786c41c03a97bca7a52aa0a72fa9387

Prérequis

- Les paquets Nvidia pour les Kernels installés ne sont pas nécessairement dispos, mais le patch est automatique
- Par contre le patch échoue si un RT Kernel est détecté
- Il faut forcer le patch à s'appliquer sur le RT Kernel
- `export IGNORE_PREEMPT_RT_PRESENCE=1`

- On peut l'ajouter au `~.bashrc` pour ne pas oublier de le faire lors d'un upgrade de Kernel)
- On installe les headers pour le Kernel en cours
- `sudo apt install linux-headers-$(uname -r)`
- S'il y a d'autres Kernel, installer les header
- `sudo apt install linux-headers-`...

> `sudo apt install nvidia-driver` installe la dernière version du driver dispo, mais cuda-drivers n'est pas forcément dispo pour cette version. Donc on va chercher la bonne version.

> Pour supprimer tous les drivers nvidia précédemment installés. **A utiliser avec précautions** : `sudo apt autoremove --purge *nvidia*`

Installation du `nvidia-driver` et de cuda

- On installe le dépôt CUDA
- ```
  wget https://developer.download.nvidia.com/compute/cuda/repos/$distro/x86_64/cuda-keyring_1.1-1_all.deb
  sudo dpkg -i cuda-keyring_1.1-1_all.deb
  sudo apt update
  ```
- On regarde les version de `nvidia-driver` qui peuvent être pin :
- `sudo apt install nvidia-driver-pinning-*`
- On regarde les version des `cuda-drivers` qui peuvent être installées :
- `sudo apt list cuda-drivers-*`
- On regarde quelle version de `nvidia-driver` serait installé pour une version voulue de cuda :
- Exemple : `sudo apt install cuda-13-1` OU `sudo apt install cuda-12-8`
- On choisi de pin la version de `nvidia-driver` qui a un `cuda-drivers` et qui permet d'installer la bonne version de CUDA :
- Exemple : `sudo apt install nvidia-driver-pinning-550` installe CUDA 12.5
- Exemple : `sudo apt install nvidia-driver-pinning-560` installe CUDA 12.6
- Exemple : `sudo apt install nvidia-driver-pinning-570` installe CUDA 12.8
- Exemple : `sudo apt install nvidia-driver-pinning-580` installe CUDA 13.0
- On install les drivers propriétaires (Calcul et Affichage GPU) :
- `sudo IGNORE_PREEMPT_RT_PRESENCE=1 apt install cuda-drivers` (pour patcher même les RT Kernel)
- Installer nvidia-container-toolkit pour Docker : `sudo apt install nvidia-container-toolkit`

https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html#configuring-docker

Optionnel
  ```

- Si on est sur un serveur, on peut aussi n'installer que les composants de Calcul GPU :

`sudo apt -V install libnvidia-compute nvidia-dkms`

- Si on est sur un PC, on peut aussi n'installer que les composants d'Affichage GPU :

`sudo apt -V install libnvidia-gl nvidia-dkms`

- Pour installer les composants Libres (Open Kernel Modules), voir :

https://docs.nvidia.com/datacenter/tesla/driver-installation-guide/ubuntu.html

- Installer le SDK CUDA `sudo apt install cuda-toolkit`
- puis les paquets GDS `sudo apt install nvidia-gds` (qui contient `nvidia-fs` kernel module)
- pour `arm64-jetson` : `sudo apt install cuda-compat`

https://docs.nvidia.com/cuda/cuda-installation-guide-linux/#network-repo-installation-for-ubuntu

Vérification de la compatibilité GPU - Application IA

Procédure :

- Lister les applications d'IA qu'on va vouloir faire tourner
- Regarder les **prérequis de la version actuelle de l'application** IA
- Regarder les prérequis de ses dépendances
- Par exemple YOLO requiert onnxruntime-gpu
- Regarder les **prérequis en termes GPU** (Compute Capability, etc.)
- Regarder les **prérequis en termes d'environnement d'exécution**
  - Version d'Ubuntu et drivers dispos
  - Version de CUDA, cuDNN, Pytorch
- Acheter le GPU nécessaire et installer la version d'Ubuntu nécessaire
- Si on est motivé et qu'on veut essayer avec notre GPU et notre version d'Ubuntu actuelle, regarder les prérequis des versions précédentes et voir s'il y en a une qui passe

On a une application d'IA donnée

Par exemple YOLO :

- utilise onnxruntime https://onnxruntime.ai/docs/execution-providers/CUDA-ExecutionProvider.html
- onnxruntime-gpu >=1.18.1 : CUDA 12.x ; cuDNN 9.x ; PyTorch >= 2.4.0
- onnxruntime-gpu < 1.18.0 : CUDA 12.x ; cuDNN 8.x ; PyTorch < 2.4.0
- onnxruntime-gpu 1.18-1.20 : CUDA 11.8 ; cuDNN 8.x ; PyTorch <= 2.3.1

ONNX Runtime (ORT) qui sert à l'inférence des modèles Yolo sur GPU nécessitent des Compute Capability >= 6.0

- **onnxruntime-gpu >= 1.16 supports CC 6.0 or above**
- **onnxruntime-gpu < 1.18 requiert CUDA 11.x**
- **L'environnement YOLO de AICA utilise onnxruntime-gpu=1.22.2**
- **onnxruntime-gpu >=1.24 requiert CC 6.0 pour CUDA 12 et CC 7.5 pour CUDA 13**

Différentes versions de l'environnement d'exécution

- Grâce à la compatibilité des versions mineures de Nvidia CUDA, une application d'IA compilée avec CUDA 11.8 peut être exécutée avec n'importe laquelle des version CUDA 11.x . Idem pour CUDA 12.x .
- Par contre une application cuDNN 8.x n'est pas compatible avec cuDNN 9.x .
- PyTorch 2.3 et inférieur utilisent cuDNN 8.x
- PyTorch 2.4 et supérieur utilisent cuDNN 9.x
- Il faut donc choisir la version de Pytorch en fonction des versions de CUDA et cuDNN qui sont compatibles avec l'environnement d'exécution (carte graphique et driver)

https://docs.nvidia.com/deeplearning/cudnn/backend/latest/reference/support-matrix.html

cuDNN>=v9.12.0 supporte :

- CUDA 12.x (nvidia-driver>=525.60.13)
- CUDA 13.x (nvidia-driver>=580.65.06)
- CUDA CC>=7.5

cuDNN<=v9.11.1

- CUDA 12.x (nvidia-driver>=525.60.13)
- CUDA CC>=5.0-9.0 (10.0-12.0 avec CUDA>=12.8 et nvidia-driver>=570.26)

cuDNN<=v9.10.2 :

- CUDA 12.x (nvidia-driver>=525.60.13) (12.8-9 pour >=570.26 et CC 10-12)
- CUDA 11.x (nvidia-driver>=450.80.02)
- CUDA CC>=5.0-9.0 (10.0-12.0 avec CUDA>=12.8 et nvidia-driver>=570.26)

Compatibilité de version d'Ubuntu

Sur Ubuntu 24.04 :

- les `sudo apt list nvidia-driver*` dispos sont 460-470,515-590
- les `sudo apt list cuda-drivers*` dispos sont 550,555,560,565,570,575,580
- les `sudo apt list nvidia-driver-pinning*` dispo sont 570,580,590

- les `sudo apt list cuda-toolkit*` seuls CUDA 12.5-13.1 sont dispos

On a une carte graphique donnée :

- Vérification du **Compute Capability (CC)** https://developer.nvidia.com/cuda/gpus
- Si on est dans cette liste, donc **CC >= 7.5 c'est TOP, on est compatible avec onnx et CUDA 13**
- A partir de 6.0 on est compatible avec onnx et CUDA 12 (mais pas CUDA 13)
- On conseille de **laisser tomber l'IA avec des CC de 5.0** ou inférieur
- Si on a **5.2 >= CC < 6.0** , on a une carte Legacy, voir ci-dessous, **c'est compliqué**
- On vérifie la version de CUDA Max supportée en fonction du CC :

  https://stackoverflow.com/questions/28932864/which-compute-capability-is-supported-by-which-cuda-versions
- On va donc pouvoir chercher une compatibilité d'exécution avec CUDA 12.x et CUDA 11.x , mais pas CUDA 13.x

Problèmes avec les GPU NVidia anciens : https://github.com/ultralytics/ultralytics/issues/5305

GPU NVidia anciens / Legacy

On lance Yolo et on a l'erreur suivante :

```
10:17:40.111 [yolo_executor] error: ONNX Runtime error during model query: Non-zero status code returned while running QuickGelu node. Name:'node_silu/QuickGeluFusion/' Status Message: CUDA error cudaErrorNoKernelImageForDevice:no kernel image is available for execution on the device
```

ONNX Runtime (ORT) qui sert à l'inférence des modèles Yolo sur GPU nécessitent des Compute Capability > 5.2

- onnxruntime-gpu 1.14 supports CC 3.7 or above
- **onnxruntime-gpu 1.15 supports CC 5.2 or above**
- onnxruntime-gpu >=1.16 supports CC 6.0 or above
- **onnxruntime-gpu <1.18 requiert CUDA 11.x**
- **>1.24 requiert CC 6.0 pour CUDA 12 et CC 7.5 pour CUDA 13**
- **L'environnement YOLO de AICA utilise onnxruntime-gpu=1.22.2**
- FP16 models ne tournent qu'à partir de CC 5.3
- C'est pour ça que l'erreur indique qu'il n'y a pas de kernel FP16 compilé pour notre GPU
- Problème : CUDA 11.x n'est pas dispo sur Ubuntu 24.04
- Mais peut être installé : https://askubuntu.com/questions/1536271/can-i-install-cuda-11-4-in-ubuntu-24-04-if-so-how

Solution (voir analyse ci-dessous) :

- Installer le driver 470
- Installer CUDA 11.8? en suivant https://askubuntu.com/questions/1536271/can-i-install-cuda-11-4-in-ubuntu-24-04-if-so-how
- Exporter le modèle onnx en kernel FP32
- Déployer le modèle YOLO dans un environnement Legacy avec onnxruntime-gpu 1.15

https://github.com/microsoft/onnxruntime/issues/17861#issuecomment-1758098712

On a une carte graphique donnée | par exemple Quadro M620 (GM107GLM) :

- Vérification du Compute Capability https://developer.nvidia.com/cuda/gpus/legacy | CC5.2
- On conseille de **laisser tomber l'IA avec des Compute Capability de 5.0** ou inférieur |
- Si on a une Compute Capability entre 5.2 et 7.2 , on peut envisager des choses mais c'est compliqué
- On vérifie la version de CUDA Max supportée : https://stackoverflow.com/questions/28932864/which-compute-capability-is-supported-by-which-cuda-versions | pour CC5.2 : CUDA 12.x
- On va donc pouvoir chercher une compatibilité d'exécution avec CUDA 12.x et CUDA 11.x , mais pas CUDA 13.x
- Les `nvidia-driver` dispos sont 460-470,515-590 (535 est testé/recommandé)
- mais les `cuda-drivers` dispos sont 550,555,560,565,570,575,580
- et les `nvidia-driver-pinning` et dispo de 570,580,590
- Exemple : `sudo apt install nvidia-driver-470` mais CUDA 11.x pas dispo sur Ubuntu 24.04
- Exemple : `sudo apt install nvidia-driver-pinning-550` installe CUDA 12.5
- Exemple : `sudo apt install nvidia-driver-pinning-560` installe CUDA 12.6
- Exemple : `sudo apt install nvidia-driver-pinning-570` installe CUDA 12.8
- Exemple : `sudo apt install nvidia-driver-pinning-580` installe CUDA 13.0
- Avec CUDA 12.0-1 et le driver 535 on peut avoir v8.9.2<=cuDNN<=v9.11.1
- Avec CUDA 12.2 v8.9.7<=cuDNN<=v9.11.1
- Avec CUDA >=12.3 v9<=cuDNN<=v9.11.1
- Donc il faudrait CUDA 12.0-2
- mais `sudo apt list cuda-toolkit*` seuls CUDA 12.5-13.1 sont dispos

Option 1 : Driver Version: 580 CUDA Version: 13.0

- On n'a pas cuDNN v8
- on a `cuda-drivers-580`

Option 2 : Driver Version: 570 CUDA Version: 12.8

-
- on a `cuda-drivers-570`

Option 3 : Driver Version: 535.288.01 CUDA Version: 12.2

- On a v8.9.7<=cuDNN<=v9.11.1
- Mais on n'a pas `cuda-drivers`

Option 4 : Driver Version: 470 CUDA Version: 11.x

- Installer CUDA 11.8? en suivant https://askubuntu.com/questions/1536271/can-i-install-cuda-11-4-in-ubuntu-24-04-if-so-how
- On n'a pas `cuda-drivers` -> Problème ?

https://pytorch.org/get-started/previous-versions/

- Avec pytorch 2.1.1 et 2.2.0 , c'est cuddn 8.9.2 qui est installé par défaut
- pytorch <2.2.0 n'est plus supporté dans pip

https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-892/support-matrix/index.html

cuDNN=v8.9.2 :

- CUDA 12.0-1 (nvidia-driver>=525.60.13)
- CUDA 11.x (nvidia-driver>=450.80.02)
- CUDA CC>=5.0 <=8.6 (8.9-9.0 avec CUDA>=11.8)
- Avec pytorch ... c'est cuddn 8.9.7 qui est installé par défaut

https://docs.nvidia.com/deeplearning/cudnn/archives/cudnn-897/support-matrix/index.html

cuDNN=v8.9.7 :

- CUDA 12.0-2 (nvidia-driver>=525.60.13)
- CUDA 11.x (nvidia-driver>=450.80.02)
- CUDA CC>=5.0 <=8.6 (8.9-9.0 avec CUDA>=11.8)

Voir aussi https://docs.nvidia.com/deeplearning/cudnn/onnxruntime-gpuarchives/index.html

Avec :

```
#syntax=ghcr.io/aica-technology/app-builder:v2
# launch with bash:
[core]
"image" = "v5.1.0"

[packages]
# add components
```

```
#"@aica/components/rl-policy-components" = "v3.0.0"
"@aica/components/advanced-perception" = "v1.0.0" # contains YoloExecutor
"@aica/components/core-vision" = "v1.1.2" # contains CameraStreamer
"@aica/foss/toolkits/cuda" = "v1.0.0-cuda24.12" # prerequisite for NVidia GPU acceleration
"@aica/foss/toolkits/ml" = "v1.0.0-gpu24.12" # prerequisite for ML model inference

# other extensions
"object-detection-utils" # self-built component to feed goal frame extracted from yolo to
robot velocity controller

# add hardware collections
"@aica/collections/intel-realsense-collection" = "v2.0.1" # D4XX and L515 cameras models,
stream RGBD
"@aica/collections/ur-collection" = "v4.3.0"
```

On a :

```
ros2@iha-portrob-1:~/examples$ python3 -m torch.utils.collect_env
<frozen runpy>:128: RuntimeWarning: 'torch.utils.collect_env' found in sys.modules after
import of package 'torch.utils', but prior to execution of 'torch.utils.collect_env'; this may
result in unpredictable behaviour
Collecting environment information...
PyTorch version: 2.6.0+cu126
Is debug build: False
CUDA used to build PyTorch: 12.6
ROCM used to build PyTorch: N/A

OS: Ubuntu 24.04.2 LTS (x86_64)
GCC version: (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Clang version: Could not collect
CMake version: version 3.28.3
Libc version: glibc-2.39

Python version: 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (64-bit runtime)
Python platform: Linux-6.17.0-14-generic-x86_64-with-glibc2.39
Is CUDA available: True
CUDA runtime version: 12.6.85
CUDA_MODULE_LOADING set to: LAZY
GPU models and configuration: GPU 0: Quadro M620
Nvidia driver version: 570.211.01
```

```
cuDNN version: Probably one of the following:
/usr/lib/libcudnn.so.9.6.0
/usr/lib/libcudnn_adv.so.9.6.0
/usr/lib/libcudnn_cnn.so.9.6.0
/usr/lib/libcudnn_engines_precompiled.so.9.6.0
/usr/lib/libcudnn_engines_runtime_compiled.so.9.6.0
/usr/lib/libcudnn_graph.so.9.6.0
/usr/lib/libcudnn_heuristic.so.9.6.0
/usr/lib/libcudnn_ops.so.9.6.0
HIP runtime version: N/A
MIOpen runtime version: N/A
Is XNNPACK available: True

CPU:
Architecture:                         x86_64
CPU op-mode(s):                       32-bit, 64-bit
Address sizes:                        39 bits physical, 48 bits virtual
Byte Order:                           Little Endian
CPU(s):                               4
On-line CPU(s) list:                  0-3
Vendor ID:                            GenuineIntel
Model name:                           Intel(R) Core(TM) i5-7440HQ CPU @ 2.80GHz
CPU family:                           6
Model:                                158
Thread(s) per core:                   1
Core(s) per socket:                   4
Socket(s):                            1
Stepping:                             9
CPU(s) scaling MHz:                   84%
CPU max MHz:                          3800.0000
CPU min MHz:                          800.0000
BogoMIPS:                             5599.85
Flags:                                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp
lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpriority ept vpid ept_ad
fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt
xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
```

```
vnmi md_clear flush_l1d arch_capabilities
Virtualization:                         VT-x
L1d cache:                              128 KiB (4 instances)
L1i cache:                              128 KiB (4 instances)
L2 cache:                               1 MiB (4 instances)
L3 cache:                               6 MiB (1 instance)
NUMA node(s):                           1
NUMA node0 CPU(s):                      0-3
Vulnerability Gather data sampling:     Vulnerable
Vulnerability Ghostwrite:               Not affected
Vulnerability Indirect target selection: Not affected
Vulnerability Itlb multihit:            KVM: Mitigation: Split huge pages
Vulnerability L1tf:                     Mitigation; PTE Inversion; VMX conditional cache
flushes, SMT disabled
Vulnerability Mds:                      Mitigation; Clear CPU buffers; SMT disabled
Vulnerability Meltdown:                 Mitigation; PTI
Vulnerability Mmio stale data:          Mitigation; Clear CPU buffers; SMT disabled
Vulnerability Old microcode:            Not affected
Vulnerability Reg file data sampling:   Not affected
Vulnerability Retbleed:                 Mitigation; IBRS
Vulnerability Spec rstack overflow:     Not affected
Vulnerability Spec store bypass:        Mitigation; Speculative Store Bypass disabled via
prctl
Vulnerability Spectre v1:               Mitigation; usercopy/swapgs barriers and __user
pointer sanitization
Vulnerability Spectre v2:               Mitigation; IBRS; IBPB conditional; STIBP disabled;
RSB filling; PBRSB-eIBRS Not affected; BHI Not affected
Vulnerability Srbds:                    Mitigation; Microcode
Vulnerability Tsa:                      Not affected
Vulnerability Tsx async abort:          Mitigation; TSX disabled
Vulnerability Vmscape:                  Mitigation; IBPB before exit to userspace

Versions of relevant libraries:
[pip3] ament-flake8==0.17.2
[pip3] flake8==7.0.0
[pip3] flake8-builtins==2.1.0
[pip3] flake8-comprehensions==3.14.0
[pip3] flake8-docstrings==1.6.0
[pip3] flake8-import-order==0.18.2
[pip3] flake8-quotes==3.4.0
```

```
[pip3] numpy==1.26.4
[pip3] nvidia-cublas-cu12==12.6.4.1
[pip3] nvidia-cuda-cupti-cu12==12.6.80
[pip3] nvidia-cuda-nvrtc-cu12==12.6.77
[pip3] nvidia-cuda-runtime-cu12==12.6.77
[pip3] nvidia-cudnn-cu12==9.5.1.17
[pip3] nvidia-cufft-cu12==11.3.0.4
[pip3] nvidia-curand-cu12==10.3.7.77
[pip3] nvidia-cusolver-cu12==11.7.1.2
[pip3] nvidia-cusparse-cu12==12.5.4.2
[pip3] nvidia-cusparselt-cu12==0.6.3
[pip3] nvidia-nccl-cu12==2.21.5
[pip3] nvidia-nvjitlink-cu12==12.6.85
[pip3] nvidia-nvtx-cu12==12.6.77
[pip3] onnxruntime-gpu==1.22.2
[pip3] pytorch3d==0.7.8
[pip3] torch==2.6.0+cu126
[pip3] torchaudio==2.6.0+cu126
[pip3] torchvision==0.21.0+cu126
[pip3] triton==3.2.0
[conda] Could not collect
```

Avec :

```
"@aica/foss/toolkits/cuda" = "v1.0.0-cuda24.12" # prerequisite for NVidia GPU acceleration
"@aica/foss/toolkits/ml" = "v1.0.0-gpu24.12" # prerequisite for ML model inference
```

On a :

```
ros2@iha-portrob-1:~/examples$ python3 -m torch.utils.collect_env
<frozen runpy>:128: RuntimeWarning: 'torch.utils.collect_env' found in sys.modules after
import of package 'torch.utils', but prior to execution of 'torch.utils.collect_env'; this may
result in unpredictable behaviour
Collecting environment information...
PyTorch version: 2.6.0+cu126
Is debug build: False
CUDA used to build PyTorch: 12.6
ROCM used to build PyTorch: N/A

OS: Ubuntu 24.04.2 LTS (x86_64)
```

```
GCC version: (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Clang version: Could not collect
CMake version: version 3.28.3
Libc version: glibc-2.39

Python version: 3.12.3 (main, Feb  4 2025, 14:48:35) [GCC 13.3.0] (64-bit runtime)
Python platform: Linux-6.17.0-14-generic-x86_64-with-glibc2.39
Is CUDA available: True
CUDA runtime version: 12.6.85
CUDA_MODULE_LOADING set to: LAZY
GPU models and configuration: GPU 0: Quadro M620
Nvidia driver version: 570.211.01
cuDNN version: Probably one of the following:
/usr/lib/libcudnn.so.9.6.0
/usr/lib/libcudnn_adv.so.9.6.0
/usr/lib/libcudnn_cnn.so.9.6.0
/usr/lib/libcudnn_engines_precompiled.so.9.6.0
/usr/lib/libcudnn_engines_runtime_compiled.so.9.6.0
/usr/lib/libcudnn_graph.so.9.6.0
/usr/lib/libcudnn_heuristic.so.9.6.0
/usr/lib/libcudnn_ops.so.9.6.0
HIP runtime version: N/A
MIOpen runtime version: N/A
Is XNNPACK available: True

CPU:
Architecture:                         x86_64
CPU op-mode(s):                       32-bit, 64-bit
Address sizes:                        39 bits physical, 48 bits virtual
Byte Order:                           Little Endian
CPU(s):                               4
On-line CPU(s) list:                  0-3
Vendor ID:                            GenuineIntel
Model name:                           Intel(R) Core(TM) i5-7440HQ CPU @ 2.80GHz
CPU family:                           6
Model:                                158
Thread(s) per core:                   1
Core(s) per socket:                   4
Socket(s):                            1
Stepping:                             9
```

```
CPU(s) scaling MHz:                      81%
CPU max MHz:                             3800.0000
CPU min MHz:                             800.0000
BogoMIPS:                                5599.85
Flags:                                   fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge
mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp
lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpriority ept vpid ept_ad
fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt
xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
vnmi md_clear flush_l1d arch_capabilities
Virtualization:                          VT-x
L1d cache:                               128 KiB (4 instances)
L1i cache:                               128 KiB (4 instances)
L2 cache:                                1 MiB (4 instances)
L3 cache:                                6 MiB (1 instance)
NUMA node(s):                            1
NUMA node0 CPU(s):                       0-3
Vulnerability Gather data sampling:      Vulnerable
Vulnerability Ghostwrite:                Not affected
Vulnerability Indirect target selection: Not affected
Vulnerability Itlb multihit:             KVM: Mitigation: Split huge pages
Vulnerability L1tf:                      Mitigation; PTE Inversion; VMX conditional cache
flushes, SMT disabled
Vulnerability Mds:                       Mitigation; Clear CPU buffers; SMT disabled
Vulnerability Meltdown:                  Mitigation; PTI
Vulnerability Mmio stale data:           Mitigation; Clear CPU buffers; SMT disabled
Vulnerability Old microcode:             Not affected
Vulnerability Reg file data sampling:    Not affected
Vulnerability Retbleed:                  Mitigation; IBRS
Vulnerability Spec rstack overflow:      Not affected
Vulnerability Spec store bypass:         Mitigation; Speculative Store Bypass disabled via
prctl
Vulnerability Spectre v1:                Mitigation; usercopy/swapgs barriers and __user
pointer sanitization
Vulnerability Spectre v2:                Mitigation; IBRS; IBPB conditional; STIBP disabled;
RSB filling; PBRSB-eIBRS Not affected; BHI Not affected
Vulnerability Srbds:                     Mitigation; Microcode
```

```
Vulnerability Tsa:                      Not affected
Vulnerability Tsx async abort:          Mitigation; TSX disabled
Vulnerability Vmscape:                  Mitigation; IBPB before exit to userspace

Versions of relevant libraries:
[pip3] ament-flake8==0.17.2
[pip3] flake8==7.0.0
[pip3] flake8-builtins==2.1.0
[pip3] flake8-comprehensions==3.14.0
[pip3] flake8-docstrings==1.6.0
[pip3] flake8-import-order==0.18.2
[pip3] flake8-quotes==3.4.0
[pip3] numpy==1.26.4
[pip3] nvidia-cublas-cu12==12.6.4.1
[pip3] nvidia-cuda-cupti-cu12==12.6.80
[pip3] nvidia-cuda-nvrtc-cu12==12.6.77
[pip3] nvidia-cuda-runtime-cu12==12.6.77
[pip3] nvidia-cudnn-cu12==9.5.1.17
[pip3] nvidia-cufft-cu12==11.3.0.4
[pip3] nvidia-curand-cu12==10.3.7.77
[pip3] nvidia-cusolver-cu12==11.7.1.2
[pip3] nvidia-cusparse-cu12==12.5.4.2
[pip3] nvidia-cusparselt-cu12==0.6.3
[pip3] nvidia-nccl-cu12==2.21.5
[pip3] nvidia-nvjitlink-cu12==12.6.85
[pip3] nvidia-nvtx-cu12==12.6.77
[pip3] onnxruntime-gpu==1.22.2
[pip3] pytorch3d==0.7.8
[pip3] torch==2.6.0+cu126
[pip3] torchaudio==2.6.0+cu126
[pip3] torchvision==0.21.0+cu126
[pip3] triton==3.2.0
[conda] Could not collect
```

Sources

https://stackoverflow.com/questions/30820513/what-is-the-correct-version-of-cuda-for-my-nvidia-driver/30820690#30820690

https://interfacinglinux.com/2024/01/16/nvidia-cuda-on-debian-real-time-kernel/

https://forum.zorin.com/t/nvidia-drivers-cannot-be-installed-because-of-real-time-kernel/46419/22

Revision #31
Created 20 February 2026 16:36:48 by admin_idf
Updated 13 April 2026 14:18:47 by admin_idf