TMVA with GPU Installation

This installation is for:

Fedora 33
cuda toolkit 11.4
cuDNN 8
root 24.00

Pre-installation

- Verify the system has a CUDA-capable GPU.
- Verify the system is running a supported version of Linux.
- Verify the system has gcc installed.
- Verify the system has the correct kernel headers and development packages installed.
- Download the NVIDIA CUDA Toolkit.
- Handle conflicting installation methods.

cuda toolkit:

https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html

```
First:
dnf install gcc-c++ mesa-libGLU-devel libX11-devel libXi-devel libXmu-devel
dnf install freeglut freeglut-devel

Get cuda toolkit:
wget https://developer.download.nvidia.com/compute/cuda/11.4.0/local_installers/cuda-repo-fedora33-11-4-local-
11.4.0_470.42.01-1.x86_64.rpm
sudo rpm -i cuda-repo-fedora33-11-4-local-11.4.0_470.42.01-1.x86_64.rpm
sudo dnf clean all
sudo dnf -y module install nvidia-driver:latest-dkms
sudo dnf -y install cuda
```

After installation

```
cat << EOF > /etc/profile.d/cuda.sh
pathmunge /usr/local/cuda-11.4/bin before

if [ -z "${LD_LIBRARY_PATH}" ]; then
    LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64
else
    LD_LIBRARY_PATH=/usr/local/cuda-11.4/lib64:$LD_LIBRARY_PATH
fi

export PATH LD_LIBRARY_PATH
EOF
```

After installation (check all is well):

cd /(insallation location)/NVIDIA_CUDA-11.4_Samples/5_Simulations/nbody

make

./nbody

if you see the simulation all is working with cuda toolkit

Installation of cuDNN libraries:

https://docs.nvidia.com/deeplearning/cudnn/install-guide/index.html

https://developer.nvidia.com/cudnn
You can download by registering or get them from our drop box under cuDNN
Download cuDNN v8.2.2 (July 6th, 2021), for CUDA 11.4

These work with Fedora 33 as well:

cuDNN Library for Linux (x86_64)

cuDNN Runtime Library for RedHat/Centos 8 PPC (RPM)

cuDNN Developer Library for RedHat/Centos 8 PPC (RPM)

cuDNN Code Samples and User Guide for RedHat/Centos 8 PPC (RPM)

```
tar -xzvf cudnn-x.x-linux-x64-v8.x.x.x.tgz

sudo cp cuda/include/cudnn*.h /usr/local/cuda/include
sudo cp -P cuda/lib64/libcudnn* /usr/local/cuda/lib64
sudo chmod a+r /usr/local/cuda/include/cudnn*.h /usr/local/cuda/lib64/libcudnn*
```

Install root:

https://root.cern/install/build_from_source/

dependencies:

```
sudo yum install git make cmake gcc-c++ gcc binutils \
libX11-devel libXpm-devel libXft-devel libXext-devel python openssl-devel

sudo yum install redhat-lsb-core gcc-gfortran pcre-devel \
mesa-libGL-devel mesa-libGLU-devel glew-devel ftgl-devel mysql-devel \
fftw-devel cfitsio-devel graphviz-devel libuuid-devel \
avahi-compat-libdns_sd-devel openldap-devel python3-numpy \
libxml2-devel gsl-devel readline-devel R-devel R-Rcpp-devel R-RInside-devel
```

sudo dnf install python-devel python3-yaml `python-pygments`

git clone --branch v6-24-00 https://github.com/root-project/root.git root_src

mkdir root_build root_install

cd root_build/

cmake -DCMAKE_INSTALL_PREFIX=/home/ptgroup/root_install/ -DCMAKE_CXX_STANDARD=14 -Dminuit2=ON -Dvmc=ON -Dcuda=ON -Dcudnn=ON -Dtmva-gpu=ON -Dtmva-cpu=ON /home/ptgroup/root_src/

cmake --build . --target install

https://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html

https://www.if-not-true-then-false.com/2018/install-nvidia-cuda-toolkit-on-fedora/

This installation is for:

Ubuntu 22.04

Install the following before installing CERN ROOT TMVA

```
You can try to install (PyTorch, TensorFlow, CUDA, cuDNN, and NVIDIA Drivers) using the Lambda Stack https://lambdalabs.com/lambda-stack-deep-learning-software
wget -nv -O- https://lambdalabs.com/install-lambda-stack.sh | sh -
sudo reboot

If you use pip install directly for each of these, I think you will have to move your .local
mv ~/.local ~/.local.bak

For virtual environment installs (This is recommended for both TensorFlow and ROOT)
```

Miniconda installation

Check your:
   pip -v list | grep -v "/usr/lib/python3/dist-packages"
    * This will show you any package installed that is not part of Lambda Stack or Ubuntu
      (Ideally these other packages should only be in a virtual environment Python venv or Anaconda/Miniconda

1. Install Miniconda:
   From: https://docs.conda.io/en/latest/miniconda.html#linux-installers
   Current version for Python 3.10:

    $ wget https://repo.anaconda.com/miniconda/Miniconda3-py310_22.11.1-1-Linux-x86_64.sh
    $ bash Miniconda3-py310_22.11.1-1-Linux-x86_64.sh

2. Activate the environment for miniconda to be active:
   $. $HOME/.bashrc

   If you want Miniconda installed but not always active, I would not mix Miniconda with python venv,
   since Miniconda blocks the default install.

   I do the following so conda is not always activated, and I can switch between Miniconda and Python venv:
    (base) $ conda deactivate
    $ conda config --set auto_activate_base false

3. Create a environment:
sudo apt install nvidia-driver-525

(You can use 'sudo ubuntu-drivers devices' to figure out what the recommended driver is for you GPU)

conda create --name tf_gpu -c nvidia -c conda-forge tensorflow=2.11 cudatoolkit=11.8 cudnn
a. Create from the command line: $ conda create --name tensorrt_quick tensorflow -c nvidia b. Activate the
Miniconda environment: $ conda activate tensorrt_quick c. Install other dependencies: $tensorrt_quick) $ pip
install pandas scikit-learn matplotlib numpy 4. Run the code: (tensorrt_quick) $ python quick.py * I did not need
to set LD_LIBRARY_PATH or anything

Intial setup:
   Check your:
      $ pip -v list | grep -v "/usr/lib/python3/dist-packages"
       * This will show you any package installed that is not part of Lambda Stack or Ubuntu
         (Ideally these other packages should only be in a virtual environment Python venv or Anaconda/Miniconda

   Make sure you are not inside of Anaconda/Miniconda
      $ deactivate # Deactivates python venv environments
      $ conda deactivate # Deactivates a layer of Anaconda/Minicond keep doing that until there is no (base) or
other.

1. Install python3-venv
    $ sudo apt update
    $ sudo apt install python3-venv
2. Create a new environment
    $ python -m venv --system-site-packages myvenv
    (--system-site-packages allows it to used the default installed packages, and new packages
      override in the environment and keep them versioned - not affecting the default or versioned environment)
3. Activate the environment:
    $ ./myvenv/bin/activate
4. Run a simple test with the base install:
   TF_CPP_MIN_LOG_LEVEL=3 python -c 'import tensorflow as tf; print("\nTensorflow version: ",tf.__version__,
"\nTensorflow file: ",tf.__file__) ; print("Num GPUs Available: ", len(tf.config.experimental.
list_physical_devices("GPU")))'
5. Install packages (like tensorrt).
   $ pip install nvidia-tensorrt

   On a older python3.8 release (still the latest 525.60.* driver, CUDA 11.8, tensorflow 11.2), I needed to
install
   $ pip install nvidia-pyindex
   $ pip install nvidia-tensorrt

  Other dependencies, just to make sure they are installed (default current version works fine):
   $ pip install pandas scikit-learn matplotlib numpy

6. The good news is nvidia-tensorrt does install the libraries in the correct place so they are picked up.
   (By default Anaconda/Miniconda does not for cuDNN, and I need to setup up the LD_LIBRARY_PATH)

7. Run the code:
   Quietly (without the verbose tensorflow messages):
     $ TF_CPP_MIN_LOG_LEVEL=3 python quick.py
   Or by default if you want to see the specifics:
     $ python quick.py


```
Example:
joe@box:~/lambda/documentation/lambdastack-python-venv-tensorflow$ python -m venv --system-site-packages myvenv

joe@box:~/lambda/documentation/lambdastack-python-venv-tensorflow$ . myvenv/bin/activate
(myvenv) joe@box:~/lambda/documentation/lambdastack-python-venv-tensorflow$
(myvenv) joe@box:~/lambda/documentation/lambdastack-python-venv-tensorflow$ TF_CPP_MIN_LOG_LEVEL=3 python -c
'import tensorflow as tf; print("\nTensorflow version: ",tf.__version__, "\nTensorflow file: ",tf.__file__) ;
print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices("GPU")))'

   Tensorflow version:  2.10.1
   Tensorflow file:  /usr/lib/python3/dist-packages/tensorflow/__init__.py
   Num GPUs Available:  2

      * Note if it is not seeing your GPUs, you have a basic issue
          1. Check: nvidia-smi

Then install other packages that you need customized.
  (myvenv) joe@tensorbook-server:~$ pip install nvidia-pyindex
  (myvenv) joe@tensorbook-server:~$ pip install nvidia-tensorrt

  (myvenv) joe@tensorbook-server:~$ TF_CPP_MIN_LOG_LEVEL=3 python quick.py
    * quiet run output
  (myvenv) joe@tensorbook-server:~$ python quick.py
    * verbose output



Installation of ROOT:
Build from source using - https://root.cern/install/build_from_source/

prerequisits

sudo apt-get install gfortran libpcre3-dev xlibmesa-glu-dev libglew1.5-dev libftgl-dev libmysqlclient-dev libfftw3-dev libcfitsio-dev graphviz-dev libavahi-
compat-libdnssd-dev libldap2-dev python-dev libxml2-dev libkrb5-dev libgsl0-dev qtwebengine5-dev

sudo apt-get install libxft-dev libxext-dev python libssl-dev



$git clone --branch latest-stable --depth=1 https://github.com/root-project/root.git root_src
$mkdir <builddir> <installdir>
$cd <builddir>



cmake -DCMAKE_INSTALL_PREFIX=/home/dir/root_inst -Dcuda=ON -Dcudnn=ON -Dtmva-gpu=ON -Dtmva-cpu=ON /home/dustin/root_src/

cmake --build . --target install -j5

source /home/dustin/root_inst/bin/thisroot.sh
```