Author: Charles Garrett Eason

Date: 8/31/2019

Contact: eason@gwu.edu

Setting up CUDA 10.1 Update 2, cuDNN 7.6.3, and Python on a personal Ubuntu 18.04 system for remote computing

Preliminary Steps

1) Install Ubuntu 18.04: https://ubuntu.com/download/desktop
2) Tether or get internet connected somehow and update Ubuntu, install git, and install bc:
    a. In the terminal enter:
    sudo apt update
    sudo apt upgrade -y
    sudo apt install git -y
    sudo apt install bc -y
3) Setup your wireless drivers, ethernet cable, or however you will be connecting to the internet.
4) Make sure you are using the most recent NVIDIA driver metapackage:
    a. In the terminal enter:
    software-properties-gtk --open-tab=4
    b. In the "Additional Drivers" tab make sure that you are using the latest driver; make sure the largest XXX in "Using NVIDIA driver metapackage from nvidia-driver-XXX (open source)" is selected.
5) Make sure your system has a CUDA-capable GPU:
    a. In the terminal enter:
    lspci | grep -i nvidia
    b. Then go and make sure your GPU is listed on: https://developer.nvidia.com/cuda-gpus
6) Verify you are using Ubuntu 18.04 64-bit:
    a. In the terminal enter:
    uname -m && cat /etc/*release
    b. It should return that you are on an "x86_64" system and that it is Ubuntu 18.04
7) Make sure that gcc is installed with 18.04:
    a. In the terminal enter:
    gcc --version
    b. If you don't get an error you are good.
    c. If you receive an error you need to install the Ubuntu 18.04 development tools:
    sudo apt-get install build-essential -y
8) Make sure your kernel version and kernel headers / development packages matchup:
    a. In the terminal enter:

sudo apt-get install linux-headers-$(uname -r)

9) Reboot:
    a. In the terminal enter:
    sudo reboot

Installing CUDA 10.1 Update 2

1) Installing CUDA:
    a. Note, this step will change for newer updates; go to NVIDIA for the most recent CUDA version: https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1804&target_type=deblocal
    b. In the terminal enter:
    wget
    https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin
    sudo mv cuda-ubuntu1804.pin /etc/apt/preferences.d/cuda-repository-pin-600
    wget
    http://developer.download.nvidia.com/compute/cuda/10.1/Prod/local_installers/cuda-repo-ubuntu1804-10-1-local-10.1.243-418.87.00_1.0-1_amd64.deb
    sudo dpkg -i cuda-repo-ubuntu1804-10-1-local-10.1.243-418.87.00_1.0-1_amd64.deb
    sudo apt-key add /var/cuda-repo-10-1-local-10.1.243-418.87.00/7fa2af80.pub
    sudo apt-get update
    sudo apt-get -y install cuda
2) Adding CUDA path to the PATH variable:
    a. In the terminal enter:
    echo $PATH
    echo 'export PATH=/usr/local/cuda-10.1/bin:/usr/local/cuda-10.1/NsightCompute-2019.1${PATH:+:${PATH}}' >> ~/.bashrc
    echo 'export LD_LIBRARY_PATH=/usr/local/cuda-10.1/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}' >> ~/.bashrc
3) Reboot:
    a. In the terminal enter:
    sudo reboot

Testing if CUDA 10.1 is successfully installed

1) Checking the versions / removing .deb file:
    a. In the terminal enter:
    cat /proc/driver/nvidia/version
    nvidia-smi
    nvcc --version
    rm -r ~/cuda-repo-ubuntu1804-10-1-local-10.1.243-418.87.00_1.0-1_amd64.deb
2) Building the CUDA samples:
    a. In the terminal enter:
    cd /usr/local/cuda-10.1/samples

sudo make

3) Running built sample 1:
    a. In the terminal enter:
    ./1_Utilities/deviceQuery/deviceQuery
4) Running built sample 2:
    a. In the terminal enter:
    ./1_Utilities/bandwidthTest/bandwidthTest
5) If both tests result in PASS then everything is working and you can remove the built samples. Otherwise something went wrong and needs to be fixed before moving on.

Installing and testing cuDNN 7.6.3

1) Make an account and download cuDNN 7.6.3 for CUDA 10.1 (Ubuntu x86_64architecture) from: https://developer.nvidia.com/cudnn
    a. Specifically download these options:
    cuDNN Runtime Library for Ubuntu18.04 (Deb)
    cuDNN Developer Library for Ubuntu18.04 (Deb)
    cuDNN Code Samples and User Guide for Ubuntu18.04 (Deb)
    b. Make sure that the .deb files are located in: ~/Downloads
2) Install cuDNN 7.6.3 and build the MNIST example:
    a. In the terminal enter:
    cd ~/Downloads
    sudo dpkg -i libcudnn7_7.6.3.30-1+cuda10.1_amd64.deb
    sudo dpkg -i libcudnn7-dev_7.6.3.30-1+cuda10.1_amd64.deb
    sudo dpkg -i libcudnn7-doc_7.6.3.30-1+cuda10.1_amd64.deb
    cp -r /usr/src/cudnn_samples_v7/ ~
    cd ~/cudnn_samples_v7/mnistCUDNN
    sudo make clean && sudo make
3) Test cuDNN:
    a. In the terminal enter:
    ./mnistCUDNN
    b. If the tests are passed then everything is working and you can move on, otherwise something went wrong and needs to be fixed before moving on.
4) Remove unnecessary files and directories:
    a. In the terminal enter:
    cd
    rm -r ./cudnn_samples_v7
    rm -r ~/Downloads/libcudnn7_7.6.3.30-1+cuda10.1_amd64.deb
    rm -r ~/Downloads/libcudnn7-dev_7.6.3.30-1+cuda10.1_amd64.deb
    rm -r ~/Downloads/libcudnn7-doc_7.6.3.30-1+cuda10.1_amd64.deb

Setting up Python 3.6

1) Installing necessary Deep Learning packages:
    a. In the terminal enter:
    sudo apt install python3-pip -y

```
sudo apt install build-essential libssl-dev libffi-dev python3-dev -y
sudo apt-get install tcl-dev tk-dev python-tk python3-tk -y
sudo pip3 install --upgrade pip
sudo -H pip3 install matplotlib
sudo -H pip3 install pandas
sudo -H pip3 install h5py
sudo -H pip3 install leveldb
sudo -H pip3 install seaborn
sudo -H pip3 install --upgrade tensorflow-gpu
#sudo -H pip3 install tensorflow-gpu==2.0.0-rc0 #for when tensorflow 2.0 comes out
sudo -H pip3 install Theano
sudo -H pip3 install keras
sudo -H pip3 install sklearn
sudo -H pip3 install cython
sudo -H pip3 install torch
sudo -H pip3 install torchvision
sudo -H pip3 install opencv-python
sudo -H pip3 install lmdb
sudo -H pip3 install sympy
sudo -H pip3 install pydotplus
sudo apt-get install -y p7zip-full
sudo apt install unzip
sudo -H pip3 install gpustat
sudo apt-get install -y stress htop iotop lm-sensors
sudo apt-get install -y python3-skimage
```

2) Testing that the python Deep Learning frameworks are installed:
    a. In the terminal enter:
    Python3
    b. Then in Python3 enter:
    import matplotlib, numpy, torch, tensorflow, keras, Theano, pandas, scipy, cv2, lmdb, sympy
    c. If a package doesn't load then you need to figure out why it won't import (perhaps some dependencies are missing?).

Getting Amir's Repos from Git

1) Installing Amir's Repos:
    a. In the terminal enter:
    cd
    git clone https://github.com/amir-jafari/Cloud-Computing.git
    git clone https://github.com/amir-jafari/Machine-Learning.git
    git clone https://github.com/amir-jafari/Deep-Learning.git
2) Setting permissions:
    a. In the terminal enter:
    sudo chmod -R 777 Cloud-Computing

sudo chmod -R 777 Machine-Learning
sudo chmod -R 777 Deep-Learning

3) Setting Display Size:
    a. In the terminal enter:
    export DISPLAY=localhost:10.0

Setting up an SSH server

1) Setting up an Open SSH server:
    a. In the terminal enter:
    sudo apt update
    sudo apt install openssh-server
2) Getting ssh.socket to listen on startup:
    a. In the terminal enter:
    sudo update-rc.d ssh defaults
    sudo systemctl enable ssh.socket
3) Testing the SSH server:
    a. In the terminal enter:
    sudo service ssh status
    b. Note: “q” exits the status
4) Now you just need to SSH in with your laptop and setup a key pair. After that you are good to go!

Notes

1) Be aware the some of the above code is a single line on multiple lines because of spacing issues in word. E.g. this code from above,

    wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu1804/x86_64/cuda-ubuntu1804.pin

    is a single line in the terminal.
2) This was installed on a system using an RTX 2080, there may be differences that need to be accounted for depending on the hardware used.
3) The information here could be wrapped into a single shell script for ease of use; however, a shell script has not been provided specifically because the rebooting requirements make for natural test opportunities. This has the advantage of providing new users a break point at which they can test to be sure the setup has not gone awry, and if it has, the user knows in what step things went wrong. Finally, writing a shell script that continues after a reboot unnecessarily complicates the steps above for new users.
4) Given point 3) above, note that rebooting is a necessary step for this setup to complete.

Resources

1) Amirs Github page: https://github.com/amir-jafari/

2) The NVIDIA CUDA installation guide: https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html
3) The NVIDA cuDNN installation guide: https://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html