WHITE PAPER–OCTOBER 2019

Bitfusion Guide to CUDA Installation

Bitfusion Guides

Table of Contents

Purpose

Bitfusion FlexDirect provides a GPU virtualization solution. It allows you to use the GPUs, or even partial GPUs (e.g., half of a GPU, or a quarter, or one-third of a GPU), on different servers as if they were attached to your local machine, a machine on which you can now run a CUDA application even though it has no GPUs of its own.

Since Bitfusion FlexDirect software and ML/AI (machine learning/artificial intelligence) applications require other CUDA libraries and drivers, this document describes how to install the prerequisite software for both Bitfusion FlexDirect and for various ML/AI applications.

If you already have your CUDA drivers and libraries installed, you can skip ahead to the chapter on installing Bitfusion FlexDirect.

This document gives instruction and examples for Ubuntu, Red Hat, and CentOS systems.

Introduction

Bitfusion's software tool is called FlexDirect. FlexDirect easily installs with a single command, but installing the third-party prerequisite drivers and libraries, plus any application and its prerequisite software, can seem a Gordian Knot to users new to CUDA code and ML/AI applications. We will begin untangling the knot with a table of the components involved. Starting with GPU hardware, then CUDA driver software, we will work our way all the way up the stack to ML/AI applications following the order in which you'd install the packages. There is some installation information in the following table, followed by more details in the body of the document.

HIGH-LEVEL DESCRIPTION	PACKAGES, FILES AND COMMENTS
GPU hardware	attached to system via PCIe
NVIDIA driver Needed on GPU servers, but not on the clients. *https://www.nvidia.com/Download/index.aspx*	nvidia-smi (application) /usr/lib/x86_64-linux-gnu/libnvidia-ml.so (NVIDIA Management Library, NVML) (used by nvidia-smi app e.g.) libcuda.so (low-level CUDA library & API, or CUDA-driver API: e.g., `cu_funcName`) (pretty good backwards compatibility) NVIDIA driver (.ko) And other libraries
CUDA (or SDK) Needed where the application is run. Applications may be sensitive to backward or forward version differences. *https://developer.nvidia.com/cuda-toolkit-archive*	cublas.so cusparse.so cufft.so etc. (NVIDIA helper libraries for various math and problem domains) libcudart.so (high-level or user-level CUDA library & API: e.g., `cuda_funcName`) nvcc (CUDA compiler) Sample applications

NVIDIA CUDA Deep Neural Network library *https://developer.nvidia.com/rdp/cudnn-archive* (You need use or create an NVIDIA developer's account to download from here.)	cudnn.so (A NVIDIA helper library for deep neural networks, kept separate from other libraries in the Toolkit.) Note: This must be selected carefully to match the versions of the **CUDA Toolkit** and of the **NVIDIA driver.**

FlexDirect
GPU virtualization software from Bitfusion.
Requires license.
	Install script run in bash:
	`$ wget -O installfd getflex.bitfusion.io ; sudo bash installfd`
	Example: Run an application using GPU from remote server:
	`$ flexdirect run -n 1 app_needing_gpu`

TensorFlow A Google ML project.	The build can be rather involved. We recommend getting a prebuilt image, at least to start: `$ sudo pip[3] install tensorflow-gpu==1.9.0 # or other version`

TensorFlow benchmarks
A representative CUDA application, uses GPU(s) through the CUDA library.
Commonly used for Bitfusion evaluations.
	GitHub repository — no build needed
	`$ git clone https://github.com/tensorflow/benchmarks.git`
	`$ cd benchmarks`
	`$ git checkout cnn_tf_v1.9_compatible`

Some Sources of Confusion

- When someone talks to you about "the CUDA library," they may mean a specific library (such as the CUDA runtime library used directly by applications or the lower-level libcuda.so, etc.) or a collection of libraries and possibly the NVIDIA driver, too. If you get confused, keep asking questions.
- In this doc, we will use the CUDA toolkit package which will let us install everything (driver, too), except for the cuDNN library (for deep neural network functionality).
- The NVIDIA driver package is more than just a driver. It includes the low-level CUDA library and the application **nvidia-smi**, too. At some point in the future, you can easily forget this and spend a bit of time trying to figure which toolkit or other NVIDIA download or package that **nvidia-smi** comes in, while never stopping to consider that it might be in the NVIDIA driver package.

So Many Prerequisites

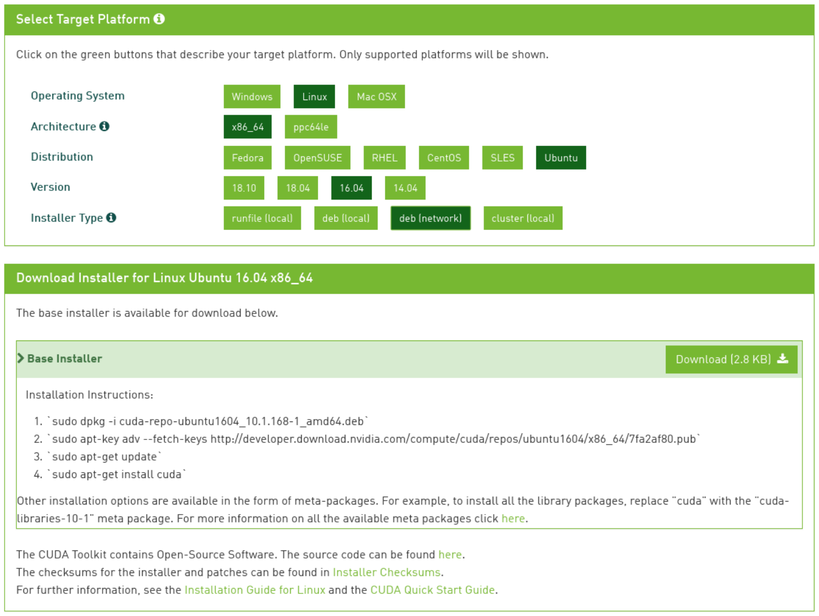
Let's walk through the details from the bottom of the stack up. However, we will assume that you have already installed a GPU that supports CUDA or will seek help from NVIDIA if you run into problems.

Installing the NVIDIA Repository

To get the NVIDIA driver installed, first install the NVIDIA repository for your operating system. You can find the links for installing the appropriate deb or rpm package on the *CUDA Toolkit Archive* page.

The version of the NVIDIA driver you install limits which versions of CUDA you can install. The version of CUDA that you install limits which versions of cuDNN you can install. If you want to target a specific version of CUDA or cuDNN, pick the version you want and work backwards to find the latest supported versions of the other packages.

Pick the CUDA version you want to install, and you'll be taken to a page where you select your operating system.



Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows [✓ Linux] Mac OSX
Architecture	[✓ x86_64] ppc64le
Distribution	Fedora OpenSUSE RHEL CentOS SLES [✓ Ubuntu]
Version	18.10 18.04 [✓ 16.04] 14.04
Installer Type	runfile (local) deb (local) [✓ deb (network)] cluster (local)

Download Installer for Linux Ubuntu 16.04 x86_64

The base installer is available for download below.

Base Installer Download (2.8 KB)

Installation Instructions:

1. `sudo dpkg -i cuda-repo-ubuntu1604_10.1.168-1_amd64.deb`
2. `sudo apt-key adv --fetch-keys http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub`
3. `sudo apt-get update`
4. `sudo apt-get install cuda`

Other installation options are available in the form of meta-packages. For example, to install all the library packages, replace "cuda" with the "cuda-libraries-10-1" meta package. For more information on all the available meta packages click here.

The CUDA Toolkit contains Open-Source Software. The source code can be found here.
The checksums for the installer and patches can be found in Installer Checksums.
For further information, see the Installation Guide for Linux and the CUDA Quick Start Guide.

Select the Linux operating system, architecture, distribution, version and installer type. For installer type, we recommend the "network" installation.

Once you click the Installer Type, a download button appears along with some instructions. The download button will download a deb or rpm package onto your computer; the instructions will tell you how to install it.

The package you downloaded will install the NVIDIA package repository on your computer, allowing you to download and install additional software packages directly from NVIDIA, using your operating system's package manager. The steps in the instructions:

- Install the repository package.
- Install NVIDIA's public keys used to sign their software packages.
- Update the repository by downloading a list of the latest available packages.

CAUTION

The last step says to install the "cuda" package. This will install the latest NVIDIA driver and the latest version of CUDA. If you want an older driver or an older of version of CUDA, do not do this step! Keep reading and we'll tell you how to install a specific version of the NVIDIA driver and the CUDA package.

Installing the NVIDIA Driver

Now that the repository is installed, to get the NVIDIA driver installed just use your operating system's package manager.

NVIDIA Driver Install on Ubuntu

```
# See all available driver versions.
sudo apt-cache show cuda-drivers | grep Version

# Install the 418.67-1 driver.
sudo apt-get install cuda-drivers=418.67-1 --no-install-recommends
```

And for Red Hat and Centos:

NVIDIA Driver Install on Red Hat or CentOS

```
# See all available driver versions.
sudo repoquery --show-duplicates cuda-drivers

# Install the 418.67-1 driver.
sudo yum install cuda-drivers=418.67-1
```

Installing NVIDIA CUDA

At the time of this writing, TensorFlow packages were meant to use CUDA 10.1. Here is an installation example using CUDA 10.1 for Ubuntu 16.04:

CUDA Toolkit Install on Ubuntu

```
# Install CUDA 10.1 using the meta package name.
sudo apt-get install cuda-10-1

# Install the latest version of CUDA available from the meta package.
sudo apt-get install cuda
```

And here is an example for RHEL 7 and Centos 7:

CUDA Toolkit Install on Red Hat or CentOS

```
# Install CUDA 10.1 using the meta package name.
sudo yum install cuda-10-1

# Install the latest version of CUDA available from the meta package.
sudo yum install cuda
```

✓ REBOOT!

You will want to reboot at this point to make sure all your changes have been picked up.

Installing cuDNN

You will need a NVIDIA Developer's account to download the cuDNN library and its installation guide.

cuDNN is built against particular versions of CUDA. It is critical to pick a cuDNN version that is built for the CUDA Toolkit version you installed above (which you chose to be compatible with the version of TensorFlow you want to use). You will also pick a cuDNN built for your OS. Go to the *cuDNN Archive* and follow the links. For instance, to get the cuDNN 7.6.0 package for CUDA 10.1 and Ubuntu 16.04, click the "Download cuDNN v7.6.0 (May 20, 2019), for CUDA 10.1," then select "cuDNN Runtime Library for Ubuntu16.04 (Deb)." This will download the cuDNN package file to your computer.

Install the package on Ubuntu:

Install cuDNN 7.6.0 for CUDA 10.1 on Ubuntu 16.04

```
# Install cuDNN 7.6.0 for CUDA 10.1 on Ubuntu 16.04.
sudo dpkg -i libcudnn7_7.6.0.64-1+cuda10.1_amd64.deb
sudo apt-get install -f
```

And with the .rpm file on Red Hat 7.x and Centos 7.x systems:

Install cuDNN 7.6.0 for CUDA 10.1 on RHEL 7.x

```
# Install cuDNN 7.6.0 for CUDA 10.1 on RHEL 7.x.
sudo rpm -ivh libcudnn7-7.6.0.64-1.cuda10.1.x86_64.rpm
```

REBOOT!

You will want to reboot at this point to make sure all your changes have been picked up.

DID IT WORK?

Type `nvidia-smi` to get the current status of your GPU. It should also tell you what version of the NVIDIA driver is installed and what version of CUDA is running.

`/sbin/ldconfig -N -v $(echo $LD_LIBRARY_PATH | sed "s/:/ /g") 2> /dev/null | grep libcudnn` will show you the versions of currently loaded cuDNN libraries.

IS A PARTICULAR CUDA LIBRARY PRESENT?

`ldconfig -p | grep <libname, e.g. cudnn>` should tell you if the system can find any particular library.

PROBLEMS?

If you have any problems, consult *NVIDIA's Installation Guide* for the latest information or for other operating systems.

Speed It Up!

FlexDirect takes advantage of GPU features like stream memops and peer mapping override to make the most efficient use of each GPU. These settings are highly recommended for best results.

To enable these settings, create the file `nvidia-bf.conf` containing the following:

nvidia-bf.conf (template)

```
options nvidia NVreg_EnableStreamMemOPs=1 NVreg_RegistryDwords=”PeerMappingOverride=1;”

options nvidia_${nvidia_driver_major_version} NVreg_EnableStreamMemOPs=1 NVreg_RegistryD
words=”PeerMappingOverride=1;”
```

Only where the file says `${nvidia_driver_major_version}`, replace that with the NVIDIA driver major version. If you installed NVIDIA driver 418.67-1, the major version number is “418,” and the file would look like:

nvidia-bf.conf

```
options nvidia NVreg_EnableStreamMemOPs=1 NVreg_RegistryDwords=”PeerMappingOverride=1;”

options nvidia_418 NVreg_EnableStreamMemOPs=1 NVreg_RegistryDwords=”PeerMappingOverr
ide=1;”
```

WHAT DRIVER VERSION IS INSTALLED?

If you’re not sure what the driver version is, just type `nvidia-smi` and it will tell you what version is installed.

Once you have the file you need to stick it in the right place.

- For Ubuntu install the file as `/etc/modprobe.d/nvidia-bf.conf`
- For Red Hat and Centos, install the file as `/lib/modprobe.d/nvidia-bf.conf`

Next, update initramfs. On Ubuntu 16.04:

Update initramfs on Ubuntu

```
# Update initramfs on Ubuntu.
sudo update-initramfs -k all -u
```

On Red Hat 7.x and Centos 7.x:

Update initramfs on Red Hat and Centos

```
# Update initramfs on RedHat and Centos.
sudo dracut -f
```

Reboot.

When your computer is back up, you can verify that the settings activated by checking the `/proc` file system. The two `grep` commands below should return something if the settings are active:

Verify the Driver Settings

```
# Verify that NVIDIA parameter ‘EnableStreamMemOPs: 1’ is set.
grep ‘EnableStreamMemOPs: 1’ /proc/driver/nvidia/params

# Verify that NVIDIA parameter ‘RegistryDwords: “PeerMappingOverride=1;”’ is set.
grep “RegistryDwords: \”PeerMappingOverride=1;\”” /proc/driver/nvidia/params
```

Upgrading CUDA

If you want to upgrade any of the packages listed above, just follow the same steps above with newer packages. The newer packages will replace the older packages.