Ubuntu 16.04×86_64 + NVIDIA GeForce GTX 1080Ti + CUDA 9.0 + cuDNN 7.0 配置 TensorFlow-GPU 1.6.0

万 震
wanzhen@cqu.edu.cn

2018 年 3 月 18 日

TensorFlow 有 CPU 和 GPU 两个版本，GPU 版本需要 NVIDIA 的 CUDA 和 cuDNN 支持，CPU 版本不需要[1]。CUDA(Compute Unified Device Architecture) 是显卡厂商 NVIDIA 推出的运算平台。$CUDA^{TM}$ 是一种由 NVIDIA 推出的通用并行计算架构，该架构使 GPU 能够解决复杂的计算问题。它包含了 CUDA 指令集架构 (ISA) 以及 GPU 内部的并行计算引擎。NVIDIA CUDA® Deep Neural Network library (cuDNN) 是 NVIDIA 专门针对深度神经网络 (Deep Neural Networks) 中的基础操作而设计基于 GPU 的加速库，其被广泛用于各种深度学习框架，例如 TensorFlow, Caffe, Theano, Torch, CNTK 等。cuDNN 为深度神经网络中的标准流程 (forward and backward convolution, pooling, normalization, and activation layers) 提供了高度优化的实现方法[2]。

本教程配置环境为：

- Ubuntu 16.04×86_64 LTS、NVIDIA GeForce GTX 1080Ti
- CUDA 9.0 、cuDNN 7.0、TensorFlow-GPU 1.6.0

文中首先介绍了配置之前的相关准备工作，包括查看 NVIDIA 显卡型号和对应驱动是否安装、验证 NVIDIA 显卡是否支持 CUDA、修改 ubuntu 默认 python 版本、安装 pip 并升级、完全卸载旧版本 CUDA；其次介绍了 CUDA 9.0 、cuDNN 7.0 以及 libcupti-dev 库的安装；再次介绍了环境变量的配置、

[1] https://github.com/tensorflow/tensorflow/releases

[2] https://developer.nvidia.com/cuda-toolkit

测试 CUDA 9.0 和 cuDNN 7.0 的安装情况；随后介绍了配置 TensorFlow-GPU 环境并测试其安装情况；最后介绍了导入 tensorflow 出现的常见问题及其解决办法。

本文所述方法适用于以上环境但所体现的思想不限于此种组合，读者可根据本教程配置类似组合。

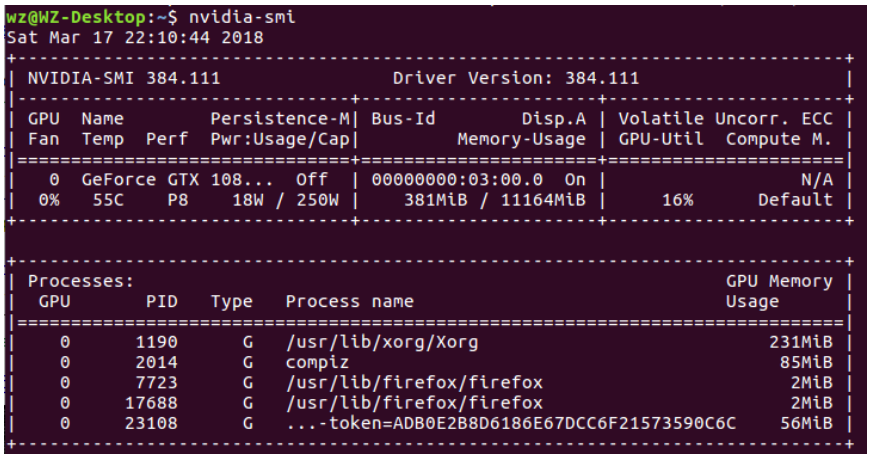
1 准备工作

1.1 查看 NVIDIA 显卡型号和相应驱动是否安装

快捷键 Ctrl+Alt+T 调出终端，输入如下命令：

$ nvidia-smi

查看 NVIDIA 显卡属性，若本机驱动已安装，则显示类似下图信息。



从上图第一行可以看出本机安装的 NVIDIA 显卡驱动版本为 384.111；第二行中可以看出本机 NVIDIA 显卡名字为：GeForce GTX 1080，GPU 编号为 0。

若 NVIDIA 显卡驱动未安装，则上述命令无法查询以上信息，可以使用如下命令进行驱动安装：

$ sudo ubuntu-drivers autoinstall

然后再次使用命令 $ nvidia-smi 查看 NVIDIA 显卡驱动是否已经安装；或者进入系统设置，点击底部的详细信息，若概况栏目正确显示出本机 CPU、GPU 信息则表明驱动已经正确安装，若上述操作均已顺利执行，但概况栏目仍未正确显示出本机 CPU、GPU 信息，则重启 Ubuntu 系统即可。

1.2 验证 NVIDIA 显卡是否支持 CUDA

点击 https://developer.nvidia.com/cuda-gpus 进入 NVIDIA GPUs 页面，根据本机 NVIDIA 显卡型号进入相应列表查看其是否在列，同时查看其是否满足 Compute Capability $\geq$ 3.0。若在列且满足 Compute Capability $\geq$ 3.0，那么恭喜你，请继续阅读下文；否则你懂的。

1.3 安装 Python 3.6/3.5 ×64

由于 Ubuntu 16.04 LTS 版本自带 Python 2.7 和 Python 3.5，为后续顺利搭建 TensorFlow-GPU 环境，请务必安装 Python 3.5 及其以上版本，当然系统自带的 Python 3.5 版本足以满足要求，本着“少折腾”原则，可以“就地取材”使用系统自带 Python 3.5。

但是，Ubuntu 16.04 LTS 默认使用 Python 2.7（可在终端输入：python -V 查看本机默认采用的 python 版本，然后进入/usr/local/lib 查看当前系统中已安装的 python 版本；若有需要，可使用命令 sudo apt-get install python3.6 安装 Python 3.6 版本），为此，需要修改系统默认的版本（并不是删除不需要的版本，因为系统的许多底层是依赖 python2 的，删除后可能会导致系统某些功能无法正常运行，谨慎操作），方法是：

1. 删除/usr/bin 目录下的 python link 文件，在终端输入如下命令：

$$\$ \text{ cd /usr/bin}$$
$$\$ \text{ sudo rm -rf python}$$

2. 删除后再建立新的 python3 链接关系：

$$\$ \text{ sudo ln -s /usr/bin/python3 /usr/bin/python}$$

1.4 安装 pip 并升级到 9.0 及以上

Ubuntu 16.04 LTS 自带的 Python 3.5 使用的 pip 版本为 8.1 的，可用 pip -V 查看当前 pip 版本（若提示未安装，可使用 sudo apt-get install python3-pip 进行安装）。后续操作需要 pip 版本为 9.0 及以上，可使用如下命令对当前 pip 进行升级：

$$\$ \text{ pip install –upgrade pip}$$

升级完毕之后，此时用 pip -V 查看当前 pip 版本。

1.5 卸载并清理原始 CUDA 版本

若不是在 ubuntu16.04 上首次安装 CUDA 及对应的 cuDNN，或者要升级 CUDA 及对应的 cuDNN 以便配置更高版本的 TensorFlow-GPU，请务必先将其卸载并清理干净，为后续高版本的配置提供清朗的安装环境。

以卸载 CUDA 8.0 和 cuDNN 6.0 为例，具体方法如下：

1. 先使用如下命令卸载 CUDA8.0 安装包：

   $ sudo apt-get –purge remove cuda-repo-ubuntu1604-8-0-local-ga2

2. 使用如下命令查找残留文件（带对应版本号的均需要清理）

   $ sudo apt-cache search cuda*

   经过上述查找，结果如下（仅列出部分）：
   cuda-cudart-8-0 - CUDA Runtime native Libraries
   cuda-driver-dev-8-0 - CUDA Driver native dev stub library
   cuda-demo-suite-8-0 - Demo suite for CUDA
   cuda-documentation-8-0 - CUDA documentation
   cuda-cusolver-8-0 - CUDA solver native runtime libraries
   .....

3. 再次使用第一步中的命令清理第二步带对应版本号的残留文件（不带对应版本号的无需清理）

   sudo apt-get –purge remove cuda-cudart-8-0 cuda-driver-dev-8-0 .....

经过上述步骤，绝大部分残留文件均已清除，但少部分文件由于权限原因，还需进一步删除，具体表现在，第三步进行后会有如下提示：
dpkg:警告:卸载 cuda-nvrtc-dev-8-0 时,目录 /usr/local/cuda-8.0/targets/x86_64-linux/include 非空，因而不会删除该目录
dpkg:警告:卸载 cuda-nvrtc-8-0 时,目录 /usr/local/cuda-8.0/targets/x86_64-linux/lib 非空，因而不会删除该目录
dpkg：警告：卸载 cuda-license-8-0 时，目录 /usr/local/cuda-8.0 非空，因而不会删除该目录

此时使用如下命令强制删除根目录下的 CUDA 安装目录，即可彻底清理：

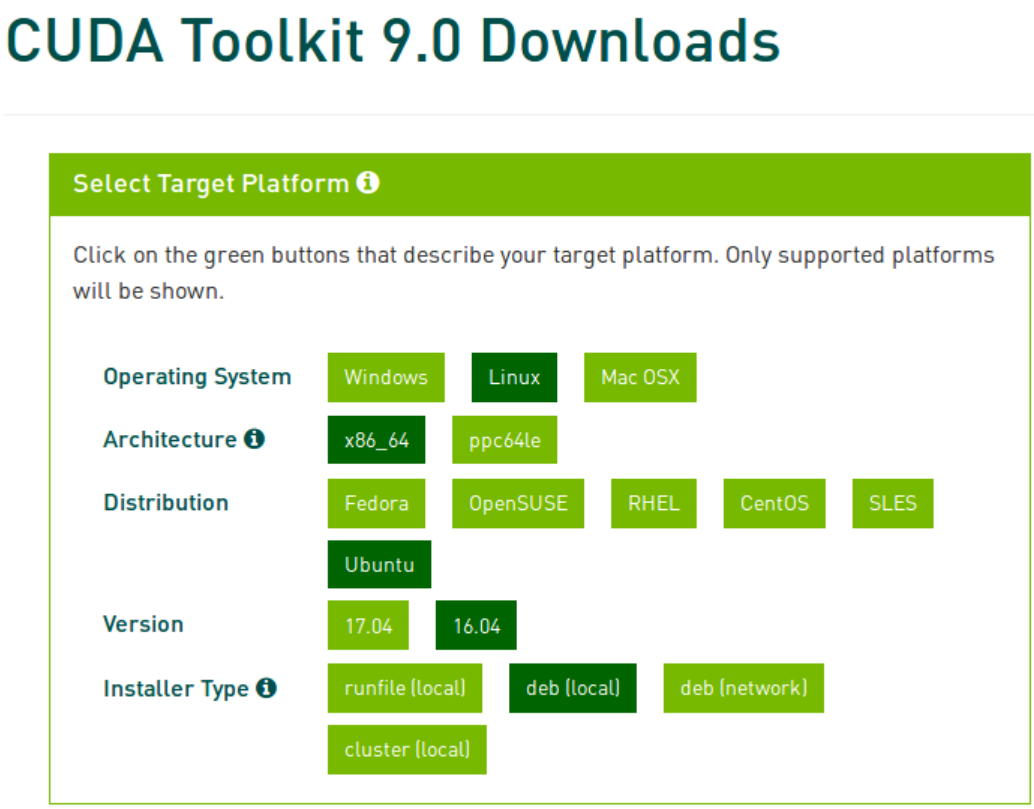
$ sudo rm -rf /usr/local/cuda-8.0

至此，准备工作已经结束，好戏才刚刚开始！

2 CUDA 9.0 和 cuDNN 7.0 的下载与安装

2.1 CUDA 9.0 的下载与安装

进入 https://developer.nvidia.com/cuda-toolkit-archive 下载 CUDA 9.0 版本。为保证后续安装和配置顺利进行，CUDA 9.0 下载页中的 Operating System、Architecture、Distribution、Version、Installer Type 这几项请按照下图所示进行选择。



共三个 deb 文件：1 为 CUDA 9.0 基础安装包，2-3 为其更新包。

1. cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64.deb（1.2G）
2. cuda-repo-ubuntu1604-9-0-local-cublas-performance-update_1.0-1_amd64.deb（100.2M）
3. cuda-repo-ubuntu1604-9-0-local-cublas-performance-update-2_1.0-1_amd64.deb（100.0M）

按照 1-2-3 的顺序进行安装，先基础安装包，首先在终端切换路径到 deb 安装包所在路径（cd ~/<*debDirectory*>），然后依次输入如下四行命令：

1. $ sudo dpkg -i cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64.deb
2. $ sudo apt-key add /var/cuda-repo-9-0-local/7fa2af80.pub
3. $sudo apt-get update
4. $ sudo apt-get install cuda

注意，第二行命令官网给的是 sudo apt-key add /var/cuda-repo-<version>/7fa2af80.pub，其中的<version>是需要我们输入安装的 CUDA 9.0 版本号，为便于确定版本号的正确输入形式，大家可以输入这行命令的前一半命令：sudo apt-key add /var/cuda-repo-，然后按Tab键自动补齐版本号。这里已经给大家确定了其正确的版本号输入形式，直接 copy 运行即可顺利安装。

接着安装更新包（默认三个 deb 均在同一路径下），在终端依次输入如下二行命令：

1. $ sudo dpkg -i cuda-repo-ubuntu1604-9-0-local-cublas-performance-update_1.0-1_amd64.deb
2. $ sudo dpkg -i cuda-repo-ubuntu1604-9-0-local-cublas-performance-update-2_1.0-1_amd64.deb

至此，CUDA 9.0 安装完毕。下面进行 cuDNN 7.0 的下载与安装。

2.2 cuDNN 7.0 的下载与安装

进入 https://developer.nvidia.com/rdp/cudnn-archive 下载与 CUDA 9.0 版本相匹配的 cuDNN 版本，这里需要注册账号、登录并进行问卷调查（问卷不多于 5 个问题）才能下载。本教程下载 Download cuDNN v7.0.5 [Dec 5, 2017], for CUDA 9.0 -cuDNN v7.0.5 Library for Linux 进行配置，下载文件为 cudnn-9.0-linux-x64-v7.tgz（333M）或是 cudnn-9.0-linux-x64-v7.solitairetheme8（348.8M）。

若是下载的 cudnn-9.0-linux-x64-v7.tgz 压缩文件，先使用如下命令进行解压：

$ tar xvf cudnn-9.0-linux-x64-v7.tgz

也可直接右键“提取到此处”，简单粗暴。

若是下载的 cudnn-9.0-linux-x64-v7.solitairetheme8 文件，先使用如下命令将其转换为 tgz 文件：

```
$ cp cudnn-9.0-linux-x64-v7.solitairetheme8 cudnn-9.0-linux-x64-v7.tgz
```

然后采用上述方法解压，或者直接右键“提取到此处”。当然也可直接将 cudnn-9.0-linux-x64-v7.solitairetheme8的后缀名改为.tgz，然后右“提取到此处”，简单粗暴、疗效快。

在终端依次输入如下三行命令：

1. $ sudo cp cuda/include/cudnn.h /usr/local/cuda/include
2. $ sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
3. $ sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*

至此，cuDNN 7.0 安装完毕。

3 libcupti-dev 库的安装

根据TensorFlow 安装教程说明：

The libcupti-dev library, which is the NVIDIA CUDA Profile Tools Interface. This library provides advanced profiling support. To install this library, issue the following command for CDDA Toolkit ≥ 8.0:

```
$ sudo apt-get install cuda-command-line-tools
```

但是，当你输入上述命令后，你会得到如下的 **错误提示：**

E: Unable to locate package cuda-command-line-tools

这是因为**没有指定** cuda-command-line-tools **的版本**，那么如何解决呢？经过检索发现，这是 TensorFlow Linux 安装教程的一个 bug，在 TensorFlow 的 github 主页下，有人提交了该问题的 Issues：Unable to locate package cuda-command-line-tools。其中二楼 cyrilzh 给出了一个解决办法：

```
$ sudo apt-cache search cuda-command-line-tool
```

使用上述命令在源软件列表中查找相应的软件包，此时会返回包含“cuda-command-line-tool”字段的所有 cuda-command-line-tools 版本。例如搜索结果如下：

cuda-command-line-tools-8-0 - CUDA command-line tools

cuda-command-line-tools-9-0 - CUDA command-line tools

然后选择相应的版本安装即可。例如本教程安装的是 CUDA 9.0，因此选择与之匹配的“cuda-command-line-tools-9-0”进行安装，命令如下：

$ sudo apt-get install cuda-command-line-tools-9-0

当然，上述办法也可使用 Tab 键的自动补齐功能来辅助查找相应版本，两者有异曲同工之妙。

4 环境变量配置

在终端依次输入如下命令：

$ sudo gedit /.bash_profile

打开个人配置文件，然后在文件末尾添加下列三行内容（以 export 开头）：

export LD_LIBRARY_PATH=”$LD_LIBRARY_PATH:/usr/local/cuda/lib64:/usr/local/cuda/extras/CUPTI/lib64”

export CUDA_HOME=/usr/local/cuda

export LD_LIBRARY_PATH=/usr/local/cuda/lib64/

保存并退出，然后在终端输入

$ source /.bash_profile

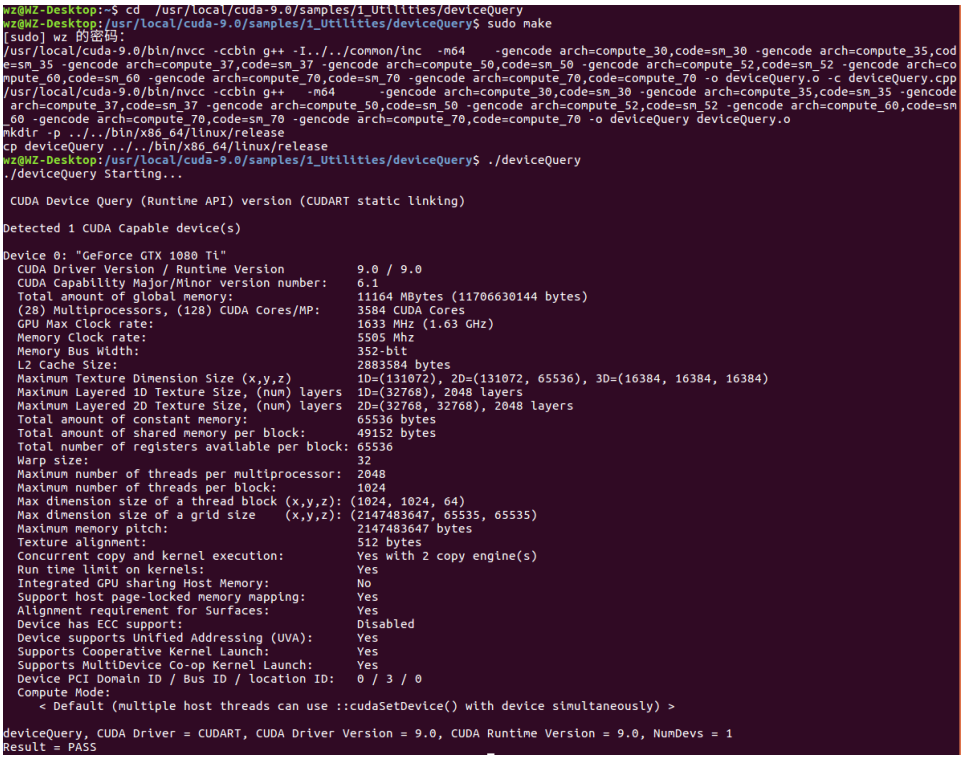
更新当前变量环境。

5 测试 CUDA 9.0 和 cuDNN 7.0 的安装情况

在终端输入如下命令：

1. $ cd /usr/local/cuda-9.0/samples/1_Utilities/deviceQuery
2. $ sudo make
3. $ ./deviceQuery

若显示本机 GPU 属性信息，如下图所示，则表明上述安装成功。



若出现错误：

CUDA Device Query (Runtime API) version (CUDART static linking)

cudaGetDeviceCount returned 35

-> CUDA driver version is insufficient for CUDA runtime version

Result = FAIL

此种情况多是由于笔记本电脑是双显卡（含 Intel 集成显卡与 NVIDIA 显卡）配置，但其 NVIDIA 显卡没启用所致。解决办法如下，终端输入：

$ nvidia-settings

打开 NVIDIA 控制面板，在左侧 PRIME Profiles 项中切换到 NVIDIA 显卡，然后重启 Ubuntu 系统即可，再次测试便能成功。

也可在终端再次输入如下命令进行二次测试：

1. $ cd /usr/local/cuda-9.0/samples/1_Utilities/bandwidthTest
2. $ sudo make
3. $ ./bandwidthTest

经过以上两次测试，即可验证前述工作初见成效。下面正式开始配置 TensorFlow-GPU 环境！

6 配置 TensorFlow-GPU 环境

TensorFlow 官方安装教程 提供了五种方式来安装 TensorFlow：Virtualenv、“native” pip、Docker、Anaconda 以及从源码编译安装，官方推荐使用 Virtualenv 来进行安装：

We recommend the Virtualenv installation. *Virtualenv is a virtual Python environment isolated from other Python development, incapable of interfering with or being affected by other Python programs on the same machine. During the Virtualenv installation process, you will install not only TensorFlow but also all the packages that TensorFlow requires. (This is actually pretty easy.) To start working with TensorFlow, you simply need to "activate" the virtual environment. All in all, Virtualenv provides a safe and reliable mechanism for installing and running TensorFlow.*

Native pip installs TensorFlow directly on your system without going through any container system. ***We recommend the native pip install for system administrators aiming to make TensorFlow available to everyone on a multi-user system***. *Since a native pip installation is not walled-off in a separate container, the pip installation might interfere with other Python-based installations on your system. However, if you understand pip and your Python environment, a native pip installation often entails only a single command.*

Docker completely isolates the TensorFlow installation from pre-existing packages on your machine. The Docker container contains TensorFlow and all its dependencies. Note that the Docker image can be quite large (hundreds of MBs). You might choose the Docker installation if you are incorporating TensorFlow into a larger application architecture that already uses Docker.

In Anaconda, you may use conda to create a virtual environment. However, within Anaconda, we recommend installing TensorFlow with the pip install *command, not with the* conda install *command.*

***NOTE**: The conda package is community supported, not officially supported. That is, the TensorFlow team neither tests nor maintains the conda package. Use*

that package at your own risk.

下面逐步实现：

1. 安装 pip （前面已经完成）和 Virtualenv：
   $ sudo apt-get install python3-pip python3-dev python-virtualenv

2. 创建 Virtualenv 环境：
   $ virtualenv –system-site-packages -p python3 /home/wz/App/tensorflow
   注意：路径 */home/wz/App/tensorflow* 为自己定义，本文设定安装在此路径下，读者根据实际情况做相应修改（下同）。

3. 激活 Virtualenv 环境：
   $ source /home/wz/App/tensorflow/bin/activate
   此时，终端的源命令提示符前部分应标识了“(tensorflow)”字段：
   (tensorflow) wz@WZ-Desktop:~$

4. 在激活的 Virtualenv 环境中安装 TensorFlow-GPU（当然也可以安装 CPU 版本的 TensorFlow）
   (tensorflow) wz@WZ-Desktop:~$ pip3 install tensorflow-gpu==1.6.0
   注意：
   1. tensorflow-gpu-1.6.0 版本大约 *210M*，这里配置的是 *1.6.0* 版本的 *tensorflow-gpu*，当然 *CUDA 9.0* 和 *cuDNN 7.0* 对 *tensorflow-gpu* 版本是向下兼容的，也就是说这里也可以安装 *1.5.0*、*1.4.0* 或 *1.3.0* 等低版本的 *tensorflow-gpu*。*tensorflow-gpu* 版本之间的切换并无特定要求，具备“回滚”功能，也就是说安装 *tensorflow-gpu-1.X.0* 版本的同时会自动卸载 *tensorflow-gpu-1.Y.0* 版本（*X* 与 *Y* 没有大小、先后之分），可使用上述命令指定安装版本来切换不同的 *tensorflow-gpu* 版本，没有后顾之忧。
   2. 至于最新版的 *tensorflow-gpu-1.7.0* ，笔者未曾测试，不敢妄下断言，还请读者自证。

此时终端提示信息如下：

Successfully installed absl-py-0.1.11 astor-0.6.2 bleach-1.5.0 gast-0.2.0 grpcio-1.10.0 html5lib-0.9999999 markdown-2.6.11 numpy-1.14.2 protobuf-3.5.2.post1 tensorboard-1.6.0 tensorflow-gpu-1.6.0 termcolor-1.1.0 werkzeug-0.14.1

表示 tensorflow-gpu-1.6.0 初步成功安装（也同时安装 tensorboard）。

7 测试 TensorFlow-GPU 安装情况

需要注意的是，每次使用 TensorFlow 的时候，必须先激活 Virtualenv 环境。

测试流程如下：

(tensorflow) wz@WZ-Desktop:~$ python

Python 3.5.2 (default, Nov 23 2017, 16:37:01)

[GCC 5.4.0 20160609] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import tensorflow as tf

>>> a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')

>>> b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')

>>> c = tf.matmul(a, b)

>>> sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

显示如下信息：

2018-03-18 17:46:08.206276: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1212] Found device 0 with properties:

name: GeForce GTX 1080 Ti major: 6 minor: 1 memoryClockRate(GHz): 1.6325

pciBusID: 0000:03:00.0

totalMemory: 10.90GiB freeMemory: 10.44GiB

2018-03-18 17:46:08.206317: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1312] Adding visible gpu devices: 0

2018-03-18 17:46:08.416506: I tensorflow/core/common_runtime/gpu/gpu_device.cc:993] Creating TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 10104 MB memory)

-> physical GPU (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:03:00.0, compute capability: 6.1)

Device mapping:

/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:03:00.0, compute capability: 6.1

2018-03-18 17:46:08.526242: I tensorflow/core/common_runtime/direct_session.cc:297] Device mapping:

>>> print(sess.run(c)) MatMul: (MatMul): /job:localhost/replica:0/task:0/device:GPU:0
2018-03-18 17:46:31.856736: I tensorflow/core/common_runtime/placer.cc:875]
MatMul: (MatMul)/job:localhost/replica:0/task:0/device:GPU:0
b: (Const): /job:localhost/replica:0/task:0/device:GPU:0
2018-03-18 17:46:31.856797: I tensorflow/core/common_runtime/placer.cc:875]
b: (Const)/job:localhost/replica:0/task:0/device:GPU:0
a: (Const): /job:localhost/replica:0/task:0/device:GPU:0
2018-03-18 17:46:31.856822: I tensorflow/core/common_runtime/placer.cc:875]
a: (Const)/job:localhost/replica:0/task:0/device:GPU:0

$$\begin{bmatrix} 22. & 28. \\ 49. & 64. \end{bmatrix}$$

到此，即验证了 TensorFlow-GPU 完全安装成功！

若退出 TensorFlow 环境，可使用如下命令：

(tensorflow) wz@WZ-Desktop:~$ deactivate

若卸载 TensorFlow，只要移除创建的 tensorflow 根目录即可，使用如下命令：

(tensorflow) wz@WZ-Desktop:~$ rm -r /home/wz/App/tensorflow

8 常见问题与解决办法

一般来说，按照上述流程顺利走下来，TensorFlow-GPU 便可以完全配置成功。但有时候也会出现某种“意外”，尤其是当 ubuntn 安装某些依赖失败导致当前环境紊乱时。

笔者就遇到过这样一个问题：按照上述流程配置 CUDA 8.0 + cuDNN6.0 + TensorFlow-GPU 1.4.0，在导入 tensorflow 时候，提示：

ImportError: libcublas.so.6.0: cannot open shared object file: No such file or directory

初步分析原因是本地/usr/local/lib 缺少 cuDNN6.0 的动态连接库文件 libcublas.so.6.0（因为 cuDNN6.0 最高可支持 TensorFlow-GPU 1.4.0 并向下兼容，cuDNN7.0 可支持 TensorFlow-GPU 1.6.0 并向下兼容，所以排除版本之间不兼容原因；同时相关路径已经添加到系统环境）。

解决办法是将与 libcublas.so.6.0（在/usr/local/cuda-8.0/lib64/下）相关的三个文件复制到/usr/local/lib 文件夹下（注意与安装的 cuda 版本号）：

$ sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so /usr/local/lib/libcudnn.so && sudo ldconfig

$ sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so.6 /usr/local/lib/libcudnn.so.6 && sudo ldconfig

$ sudo cp /usr/local/cuda-8.0/lib64/libcudnn.so.6.0.21 /usr/local/lib/libcudnn.so.6.0.21 && sudo ldconfig

若读者在按照本教程配置 CUDA 9.0 + cuDNN8.0 + TensorFlow-GPU 1.6.0 或其他组合时出现类似问题：

ImportError: libcublas.so.x.0: cannot open shared object file: No such file or directory

可参照上述方法复制相关文件到/usr/local/lib 文件夹下，即可解决问题。

参考文献

[1] https://tensorflow.google.cn/install/install_linux

[2] http://blog.csdn.net/u014696921/article/details/60140264

[3] https://blog.csdn.net/10km/article/details/61665578