链滴

如何在 Ubuntu 下编译 CUDA 10.0 Tensor Flow-GPU r.1.13 版本

作者：lai-bluejay

原文链接：https://ld246.com/article/1553577574298

来源网站：链滴

如何在Ubuntu下编译CUDA 10.0 TensorFlow-GPU r.1 13版本

注意:

1. CUDA 10.1 版本在编译时会报错，提示版本无法适配kernel。可以使用 mazar的方法解决部分题。但是使用CUDA 10.0 + cuDNN 7.5.0会容易很多

The driver installation is unable to locate the kernel source. Please make sure that the kernel ource packages are installed and set up correctly.

2. bazel的版本建议是0.15~0.17，最高不要超过0.21，会导致无法编译。

3. 该重启的时候需要重启。

4. 编译会花较长时间(3~4H)，建议再空闲时间编译。

5. 请严格按照此教程的顺序进行安装和编译~~~

这是一个如何编译适配 CUDA 10.0 版本TensorFlow GPU r1.13的教程。(截至2019年03月26日12:00由于最新的官方发布中，TensorFlow 的版本为1.12，使用cuda 9.0 + cuDNN 7。最新编译同样如。但从TensorFlow的项目issues来看，最新r.1.13是适配CUDA 10.0的，而CUDA每次大版本的更新有很大的性能上的提升。由于TensorFlow 的github release只有源码，因此我们考虑从源码编译最版的TensorFlow。

最新版本的TensorFlow提供了最新的特性和优化方法，最新的CUDA Toolkit能提供速度上的提升，能使用最新cuDNN减少深度学习训练时间。

Step 1. 更新系统

```
sudo apt-get update
sudo apt-get upgrade
```

Step 2. 确定linux的版本，安装最新的依赖

```
uname -m && cat /etc/*release
sudo apt-get install build-essential
sudo apt-get install cmake git unzip zip
# 这边更建议下载最新的conda进行安装
```

同时使用Conda安装Python3。可以参考Conda加速下载。可以下载最新安装包之后conda install python=3.6，也可以创建虚拟环境安装conda create -n env_name python=3.6

Step 3. 安装linux kernel header

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

安装好之后，可以查看其它内核版本是否已经安装在boot目录下。

如果出现如下类似问题，那可能是boot目录下空间不够

```
gzip: stdout: No space left on device
update-initramfs: failed for /boot/initrd.img-4.4-143-generic with 1
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

采用如下方案解决:

```
sudo dpkg --configure -a
dpkg -l | grep linux-
# 把和uname -r不一致的旧内核都可以删了
sudo dpkg --purge linux-headers-x.x.x-xx-blabla
sudo apt autoremove
```

比如我的uname -r是4.4.0-143-generic, 把列表中其他header和image删除就好了。

Step 4. 确定GPU适配的CUDA版本

查看官网的cuda-gpus ，由于安装新版的CUDA，建议移除旧版CUDA

```
lspci | grep -i nvidia
sudo apt-get purge nvidia*
sudo apt-get autoremove
sudo apt-get autoclean
sudo rm -rf /usr/local/cuda*
```

接下来安装cuda和驱动:

```
sudo apt-get update
sudo apt-get -o Dpkg::Options::="--force-overwrite" install cuda-10-0 cuda-drivers
```

也可以使用deb进行安装。

Step 5.安装后按提示重启。

Step 6.更新 ~/.bashrc

首先更新~/.bashrc中的环境变量

```
echo 'export CUDA_HOME=/usr/local/cuda-10.0'
echo 'export PATH=$CUDA_HOME/bin${PATH:+:${PATH}}' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$CUDA_HOME/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_
ATH}}' >> ~/.bashrc
## 或者直接编辑~/.bashrc

source ~/.bashrc
sudo ldconfig
```

之后查看nvidia-smi，确认cuda版本和驱动版本。比如我的是NVIDIA-SMI 418.39 Driver Version: 4
8.39 CUDA Version: 10.0。之后到nvidia-samples验证是否安装成功。

```
# sample 可以指定路径安装
cuda-install-samples-10.0.sh ~
cd ~/NVIDIA_CUDA-10.0_Samples/5_Simulations/nbody

# 也可能在安装cuda的过程中安装，路径位置 /usr/local/cuda-10.0/samples 再复制到指定位置即

# cp -r /usr/local/cuda-10.0/samples ~/NVIDIA_CUDA-10.0_Samples

```

```
make ./nbody
```

不报错就成功了。

Step 7. 安装cuDNN。

打开cuDNN网站进行下载，选择cuDNN Library for Linux。下载之后是.tgz压缩包。

```
tar -xf cudnn-10.0-linux-x64-v7.5.0.56.tgz

sudo cp -R cuda/include/* /usr/local/cuda-10.0/include

sudo cp -R cuda/lib64/* /usr/local/cuda-10.0/lib64
```

查看CUDA和cuDNN版本验证一下

```
cat /usr/local/cuda/version.txt
cat /usr/local/cuda/include/cudnn.h | grep CUDNN_MAJOR -A 2
```

Step 8. 安装新版NCCL

这里需要下载对应系统版本的nccl，例如ubuntu 16.04下载Local installer for Ubuntu 16.04

参考nccl

```
# 或者
sudo apt install libnccl2=2.4.2-1+cuda10.0 libnccl-dev=2.4.2-1+cuda10.0
sudo ldconfig
```

Step 9. 更新python 依赖

建议在虚拟环境中

```
pip install -U pip six numpy wheel mock
pip install -U keras_applications==1.0.5 --no-deps
pip install -U keras_preprocessing==1.0.3 --no-deps
```

Step 10. 配置编译

下载安装bazel：

```
wget -c https://github.com/bazelbuild/bazel/releases/download/0.21.0/bazel-0.21.0-installer-
inux-x86_64.sh
chmod +x bazel-0.21.0-installer-linux-x86_64.sh
./bazel-0.21.0-installer-linux-x86_64.sh --user
# which bazel; whereis bazel 查看一下路径
# 看是否需要加到环境配置中。
echo 'export PATH="$PATH:$HOME/bin"' >> ~/.bashrc

# 重新加载环境变量
source ~/.bashrc
sudo ldconfig
```

下载TensorFlow 源码:

```
cd ~
git clone https://github.com/tensorflow/tensorflow.git
cd tensorflow
git checkout r1.13
./configure
```

配置编译config，有默认项的可以直接使用默认项，除非像cuda和python path之类的指定版本。

```
# 最好指定python3.6的路径，比如我的是虚拟环境/home/dxm/data/anaconda2/envs/py36torch

Please specify the location of python. [Default is /usr/bin/python]: /home/dxm/data/anacon
a2/envs/py36torch/bin/python3

# 配置其他选项
Do you wish to build TensorFlow with Apache Ignite support? [Y/n]: Y
Do you wish to build TensorFlow with XLA JIT support? [Y/n]: Y
Do you wish to build TensorFlow with OpenCL SYCL support? [y/N]: N
Do you wish to build TensorFlow with ROCm support? [y/N]: N
Do you wish to build TensorFlow with CUDA support? [y/N]: Y
Please specify the CUDA SDK version you want to use. [Leave empty to default to CUDA 9.0]:
10.0
Please specify the location where CUDA 10.0 toolkit is installed. Refer to README.md for mo
e details. [Default is /usr/local/cuda]: /usr/local/cuda-10.0
Please specify the cuDNN version you want to use. [Leave empty to default to cuDNN 7]: 7.5.


Please specify the location where cuDNN 7 library is installed. Refer to README.md for more
details. [Default is /usr/local/cuda-10.0]: /usr/local/cuda-10.0
Do you wish to build TensorFlow with TensorRT support? [y/N]: N
Please specify the NCCL version you want to use. If NCCL 2.2 is not installed, then you can us
version 1.3 that can be fetched automatically but it may have worse performance with multip
e GPUs. [Default is 2.2]: 2.4.2

Do you want to use clang as CUDA compiler? [y/N]: N

Please specify which gcc should be used by nvcc as the host compiler. [Default is /usr/bin/gc
]: /usr/bin/gcc

Do you wish to build TensorFlow with MPI support? [y/N]: N

Please specify optimization flags to use during compilation when bazel option "--config=opt"
is specified [Default is -march=native]: -march=native

Would you like to interactively configure ./WORKSPACE for Android builds? [y/N]:N
```

Step 11 使用bazel编译TensorFlow

```
bazel build --config=opt --config=cuda //tensorflow/tools/pip_package:build_pip_package
```

接下来，等着吧。等几个小时，睡觉或者打游戏去吧。

编译好了之后，构建wheel文件

```
bazel-bin/tensorflow/tools/pip_package/build_pip_package tensorflow_pkg
cd tensorflow_pkg
tensorflow-1.13.1-cp36-cp36m-linux_x86_64.whl
```

使用tensorflow-1.13.1-cp36-cp36m-linux_x86_64.whl安装在需要的环境就OK啦。

接下来验证一下是否安装成功。

```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
# b'Hello, TensorFlow!'
print(tf.__version__)
# 1.13.1
```