

目录

目录	1
安装CUDA驱动指南	2
安装依赖包	2
下载驱动	2
安装cuDNN和NCCL指南	3
安装cuDNN	3
安装NCCL	3
安装CAFFE指南	4
前期工作	4
安装CAFFE	4
配置pycaffe	5
安装TensorFlow指南	5
安装TensorFlow	6
网络安装	6
源码安装	6
验证环境	6
产品常见问题	6
购买常见问题	6
购买渠道	6
购买注意事项	6

安装CUDA驱动指南

CUDA (Compute Unified Device Architecture) 是显卡厂商 NVIDIA 推出的运算平台。CUDA™ 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。开发人员现在可以使用 C 语言, C++, FORTRAN 来为 CUDA™ 架构编写程序, 所编写出的程序可以在支持 CUDA™ 的处理器的超高性能运行。

GPU 裸金属采用 NVIDIA 显卡, 需要安装 CUDA 开发运行环境。以Ubuntu 16.0.4为例, 可参照以下步骤进行安装。

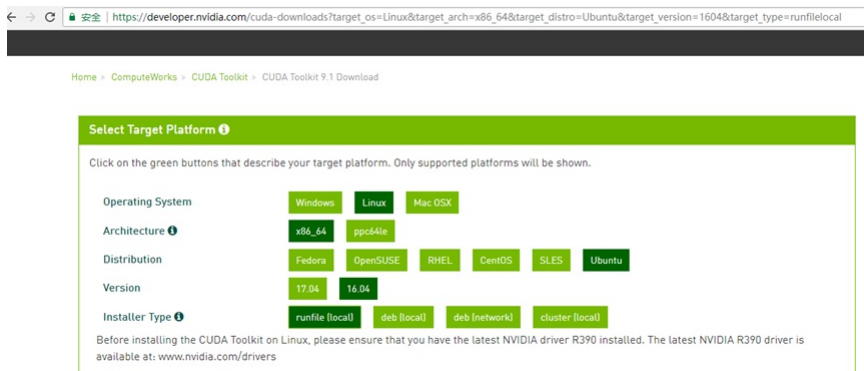
安装依赖包

登录 GPU 实例, 下载安装需要的依赖库, 运行如下指令:

```
sudo apt-get install libGLU* libXi* libXmu* -y
```

下载驱动

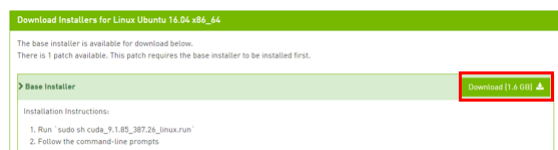
1. 登录GPU实例, 进行CUDA驱动下载或复制链接 <https://developer.nvidia.com/cuda-downloads>
2. 选择与自己的操作系统相匹配的安装包。以Ubuntu 16.0.4 64 位为例, 可按如下方式进行选择:



注意:

- Installer Type 这里推荐选择 runfile (local)。
- network: 网络安装包, 安装包较小, 需要在主机内联网下载实际的安装包。
- local: 本地安装包。安装包较大, 包含每一个下载安装组件的安装包。

3. 点击【Download】, 选择下载存放地址:



4. 切换到CUDA安装包所在的目录, 执行以下命令:

```
sudo sh cuda_9.1.85_387.26_linux.run
```

根据提示选择accept -yes -enter。

注意:

- 若执行后出现如下结果:

```
Driver: installed require reboot
Toolkit: install skip
Samples: install skip
```

说明这个CUDA安装包包含了Driver, Toolkit和Samples三部分, 但此次安装时只把驱动装上了。

此时需重新安装, 再次执行以下命令:

```
sudo sh cuda_9.1.85_387.26_linux.run
```

- CUDA安装成功结果如下:

```
Driver: Installed
Toolkit: Installed in /usr/local/cuda
Samples: Installed in /home/XX
```

5. 在 /usr/local/cuda/samples/1_Utilities/deviceQuery 目录下, 执行 make 命令, 可以编译出 deviceQuery 程序。执行 deviceQuery 正常显示如下设备信息, 此刻认为 CUDA 安装正确。

```
./deviceQuery Starting...
CUDA Device Query (runtime API) version (CUDA8T static linking)
Detected 1 CUDA Capable device(s)
Device 0: "Tesla M40 24GB"
  CUDA Driver Version / Runtime Version      8.0 / 7.5
  CUDA Capability Major/Minor version number: 5.2
  Total amount of global memory:             24505 MBytes (2569592736 bytes)
  (24) Multiprocessors, (120) CUDA Cores/MP: 3072 CUDA Cores
  GPU Max Clock rate:                        1112 MHz (1.11 GHz)
  Memory Clock rate:                         3004 MHz
  Memory Bus Width:                          304-bit
  L2 Cache Size:                             3145728 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
  Maximum Layered 1D Texture Size, (num) layers 1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(16384, 16384), 2048 layers
  Total amount of constant memory:          65536 bytes
  Total amount of shared memory per block:  49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                 32
  Maximum number of threads per multiprocessor: 2048
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                     512 bytes
  Texture alignment:                         Yes with 2 copy engine(s)
  Concurrent copy and kernel execution:      Yes with 2 copy engine(s)
  Run time limit on kernels:                 No
  Integrated GPU sharing Host Memory:        No
  Support host page-locked memory mapping:   Yes
  Alignment requirement for Surfaces:        Yes
  Device has ECC support:                    Disabled
  Device supports Unified Addressing (UVA):  Yes
  Device PCI Bus ID / Bus ID / Location ID:  0 / 0 / 7
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 8.0, CUDA Runtime Version = 7.5, NumDevs = 1, Device0 = Tesla M40
Result = PASS
```

安装cuDNN和NCCL指南

安装cuDNN

cuDNN的全称为NVIDIA CUDA® Deep Neural Network library，是NVIDIA专门针对深度学习（Deep Neural Networks）中的基础操作而设计基于GPU的加速库。cuDNN为深度学习中的标准流程提供了高度优化的实现方式，例如convolution、pooling、normalization以及activation layers的前向以及后向过程。

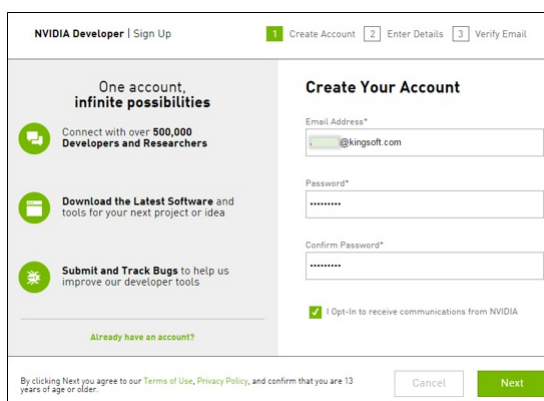
下面以Ubuntu 16.0.4为例说明如何配置cuDNN进行神经网络的加速。

1. 登录GPU实例，进行[cuDNN下载](https://developer.nvidia.com/cudnn)或复制下载地址 <https://developer.nvidia.com/cudnn>，点击【下载cuDNN】进行下载。

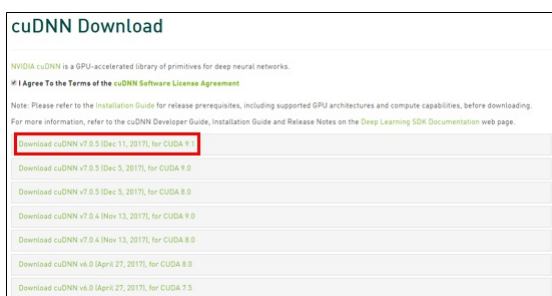
NVIDIA cuDNN

NVIDIA CUDA® 深度学习神经网络库 (cuDNN) 是经 GPU 加速的深度学习神经网络基元库。cuDNN 可大幅优化标准例程（例如用于前向传播和反向传播的卷积层、池化层、归一化层和激活层）的实施。

世界各地的深度学习研究人员和框架开发者都依赖 cuDNN 实现高性能 GPU 加速。借助 cuDNN，研究人员和开发者可以专注于训练神经网络及开发软件应用，而不必花时间进行低层级的 GPU 性能调整。cuDNN 可加速广泛应用的深度学习框架，包括 Caffe2、Chainer、Keras、MATLAB、MxNet、PyTorch 和 TensorFlow。如需获取经 NVIDIA 优化且已在框架中集成 cuDNN 的深度学习框架容器，请访问 NVIDIA GPU CLOUD 了解详情并开始使用。

[下载 cuDNN >](#)
[GTC2020 >](#)
[开发者指南 >](#)
[论坛 >](#)


2. 需要登录/注册，请按步骤提示，完成输入邮箱和密码，用户等信息，确认邮件等步骤。



3. 选择与已安装的CUDA对应的版本，这里我们选择CUDA 9.1。
4. 选择与操作系统对应的版本，这里我们选择【cuDNN v7.0.5 Library for Linux】，下载得到压缩文件cudnn-9.1-linux-x64-v7.tgz。



5. 切换到cuDNN压缩包所在目录，在命令行输入以下指令进行解压。

```
sudo tar -xvf cudnn-9.1-linux-x64-v7.tgz -C /usr/local ---- 完成安装
```

安装NCCL

NCCL是Nvidia Collective multi-GPU Communication Library的简称，它是一个实现多GPU的collective communication通信库，Nvidia做了很多优化，以在PCIe、Nvlink、InfiniBand上实现较高的通信速度。

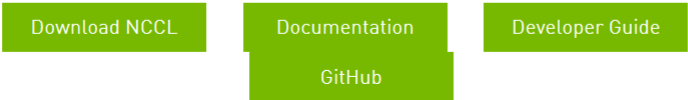
1. 登录GPU实例，进行[NCCL下载](https://developer.nvidia.com/nccl)或复制下载地址 <https://developer.nvidia.com/nccl>，点击【Download NCCL】进行下载。

NVIDIA NCCL

The NVIDIA Collective Communication Library (NCCL) implements multi-GPU and multi-node communication primitives optimized for NVIDIA GPUs and Networking. NCCL provides routines such as all-gather, all-reduce, broadcast, reduce, reduce-scatter as well as point-to-point send and receive that are optimized to achieve high bandwidth and low latency over PCIe and NVLink high-speed interconnects within a node and over NVIDIA Mellanox Network across nodes.

Leading deep learning frameworks such as Caffe2, Chainer, MxNet, PyTorch and TensorFlow have integrated NCCL to accelerate deep learning training on multi-GPU multi-node systems.

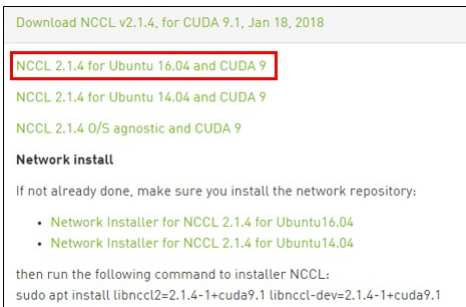
NCCL is available for download as part of the NVIDIA HPC SDK and as a separate package for Ubuntu and Red Hat.



- 需要登录/注册，请按步骤提示耐心操作。登录后，选择对应版本的 NCCL，这里我们选择【Download NCCL v2.1.4, for CUDA 9.1, Jan 18, 2018】。



- 选择与操作系统对应的版本，以Ubuntu 16.0.4为例，这里我们选择【NCCL 2.1.4 for Ubuntu 16.04 and CUDA 9】，下载得到nccl-repo-ubuntu1604-2.1.4-ga-cuda9.1_1-1_amd64.deb。



- 切换到NCCL文件所在目录，运行以下命令：

```
sudo dpkg -i nccl-repo-ubuntu1604-2.1.4-ga-cuda9.1_1-1_amd64.deb
```

完成解压安装，将NCCL的 include 和 lib 文件夹下文件放到对应 /usr/local/include/usr/local/lib 目录下。

安装CAFFE指南

前期工作

- 已安装成功CUDA，若未安装请查看教程 [安装CUDA驱动指南](#)。
- 已安装cuDNN和NCCL优化库，若未安装请查看教程 [安装cuDNN和NCCL指南](#)。
- 安装必要的依赖库，输入以下指令：

```
sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler libgflags-dev libgoogle-glog-dev liblmbd-dev libatlas-base-dev git -y
sudo apt-get install --no-install-recommends libboost-all-dev -y
```

安装CAFFE

- 从 Github 下载CAFFE的对应发布版本，下载地址如下：<https://github.com/NVIDIA/caffe/releases>，选择【Source code (tar.gz)】。或者复制链接地址，利用指令进行下载：

```
wget https://github.com/NVIDIA/caffe/archive/v0.16.6.tar.gz
```



2. 进行Openblas下载与安装，输入如下指令：

```
wget https://github.com/xianyi/OpenBLAS/archive/v0.2.20.tar.gz
tar -xvf v0.2.20.tar.gz
cd OpenBLAS-0.2.20
make PREFIX=/opt/OpenBLAS
make install
```

3. 配置lib路径，使lib库生效，输入以下指令：

```
echo "/opt/OpenBLAS/lib/
/usr/local/cuda/lib64 " >> /etc/ld.so.conf.d/cuda.conf
ldconfig
ldconfig -v
```

4. 解压CAFFE压缩包，请把文件放到/home 目录下，输入以下指令：

```
tar -xvf v0.16.6.tar.gz
cd caffe-0.16.6
cp Makefile.config.example Makefile.config
```

5. 对Makefile.config进行编辑，输入如下指令：

```
vi Makefile.config
```

```
## Refer to http://caffe.berkeleyvision.org/installation.html
# Contributions simplifying and improving our build system are welcome!

# cuDNN acceleration switch (uncomment to build with cuDNN).
# cuDNN version 6 or higher is required.
# USE_CUDNN := 1

# NCCL acceleration switch (uncomment to build with NCCL).
# See https://github.com/NVIDIA/nccl
# USE_NCCL := 1

# Builds tests with 16 bit float support in addition to 32 and 64 bit.
# TEST_FP16 := 1

# uncomment to disable IO dependencies and corresponding data layers
# USE_OPENCV := 0
# USE_LEVELDB := 0
# USE_LMDB := 0
```

去掉USE_NCCL := 1 , USE_CUDNN := 1 之前的注释#，使其生效。

注：

- 取消对行 USE_CUDNN := 1 的注释可以启用 cuDNN 加速。
- 取消对行 USE_NCCL := 1 的注释可以启用在多个 GPU 上运行 Caffe 所需的 NCCL。

6. 保存并关闭文件，对CAFFE进行编译，输入指令：

```
make -all -j4
```

等待编译打开编辑器，添加bin 路径，输入指令：

```
vi /etc/profile
export PATH=$PATH:/home/caffe-0.16.6/build/tools
```

保存并退出，输入指令，使文件生效。

```
source /etc/profile
```

配置pycaffe

1. 根据caffe根目录python文件夹下的requirements.txt的清单文件，安装上面列出的需要的依赖库。输入如下指令查看requirements.txt：

```
cd caffe-0.16.6/python
cat requirements.txt
```

显示如下信息：

```
cython>=0.19.2
numpy>=1.7.1,<=1.11.0
scipy>=0.13.2
scikit-image>=0.9.3
matplotlib>=1.3.1
ipython>=3.0.0,<=5.4.1
h5py>=2.2.0
leveldb>=0.191
networkx>=1.8.1
nose>=1.3.0
pandas>=0.12.0
protobuf>=2.5.0
pydot2
python-dateutil>=1.4,<2
python-gflags>=2.0
pypam1>=3.10
Pillow>=2.3.0
six>=1.1.0
```

2. 执行安装代码：

```
for seq in $(cat requirements.txt )
do
pip install $seq
done
```

3. 安装完成以后，再次回到caffe根目录，可以执行如下指令：

```
cd .. ---退出软件python 目录，回到caffe根目录
sudo pip install -r python/requirements.txt
```

安装成功的，都会显示Requirement already satisfied; 没有安装成功的，会继续安装。

4. 编译python接口，输入如下指令：

```
make pycaffe
```

配置路径，添加到 /etc/profile，输入以下指令：

```
vi /etc/profile
export PYTHONPATH=/home/caffe-0.16.6/python:$PYTHONPATH
```

保存并退出，输入指令，使文件生效：

```
source /etc/profile
```

5. 输入python，开启pyhton测试pycaffe，若运行如下三行不出错 说明配置完成。

```
import caffe
from caffe import layers as L
from caffe import params as P
```

安装TensorFlow指南

安装TensorFlow

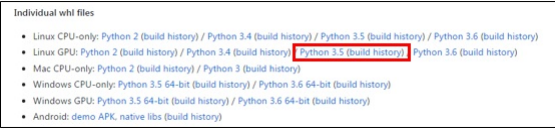
网络安装

网络良好时，可直接指令安装：

```
pip3 install --upgrade tensorflow-gpu
```

源码安装

1. 进行[tensorflow 1.5.0](https://github.com/tensorflow/tensorflow)下载或复制下载地址 <https://github.com/tensorflow/tensorflow>
2. 下载对应 GPU版本，这里我们选择python 3.5 版本，得到文件【tf_nightly_gpu-1.head-cp35-cp35m-linux_x86_64.whl】。



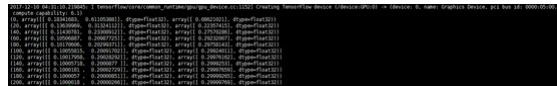
3. 输入以下指令，进行安装。

```
pip3 install tf_nightly_gpu-1.head-cp35-cp35m-linux_x86_64.whl
```

验证环境

TensorFlow安装完成后，执行python脚本，根据输出结果，确认是否有异常。Python代码如下：

```
import tensorflow as tf
import numpy as np
x_data = np.float32(np.random.rand(2, 100))
y_data = np.dot([0.100, 0.200], x_data) + 0.300
b = tf.Variable(tf.zeros([1]))
W = tf.Variable(tf.random_uniform([1, 2], -1.0, 1.0))
y = tf.matmul(W, x_data) + b
loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)
init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)
for step in range(0, 201):
    sess.run(train)
    if step % 20 == 0:
        print (step, sess.run(W), sess.run(b))*
```



使用以上代码写入python 文件，Python 指向文件运行即可，如果运行成功则结果如下：

更加详细的安装流程，可以参考[tensorflow社区文档](#)。

产品常见问题

Q: GPU相对于CPU有哪些优势？

A: GPU比CPU拥有更多的逻辑运算单元(ALU) 支持并行计算，可以多线程大规模并行计算。

Q: GPU裸金属服务器（GPU Elastic Physical Compute，简称GEPC）如何计费？

A: 目前GEPC支持包年包月和按量付费（按日月结）的计费模式，用户可选择提前按单月或数月支付GEPC的费用，或者使用后按单日或数日支付GEPC的费用。

Q: GEPC是否可以支持配置升级和降配？

A: GEPC使用固定配置，不支持对实例进行升级配置或者降配。

Q: GEPC是否可以访问云服务器（Kingsoft Cloud Elastic Compute，简称KEC）和裸金属服务器（Elastic Physical Compute，简称EPC）？

A: 支持，GEPC默认加入VPC，可以与用户VPC内的KEC和EPC互联互通。

购买常见问题

购买渠道

如果您还未取得购买权限GPU裸金属服务器（GPU Elastic Physical Compute，简称GEPC），请发邮件到ksc_gpu@kingsoft.com申请。

如果您已有购买权限，您可以在金山云控制台购买GEPC。

购买注意事项

在购买金山云GEPC前，请确保您了解金山云GEPC的价格以及所提供的服务，并且根据您的实际需求购买，一旦购买成功，则不能退款。