本教程适用于CUDA9.0+CUDNN7.1+OPENCV(3.4.0)

一、依赖包安装

在Ubuntu的Terminal中输入：

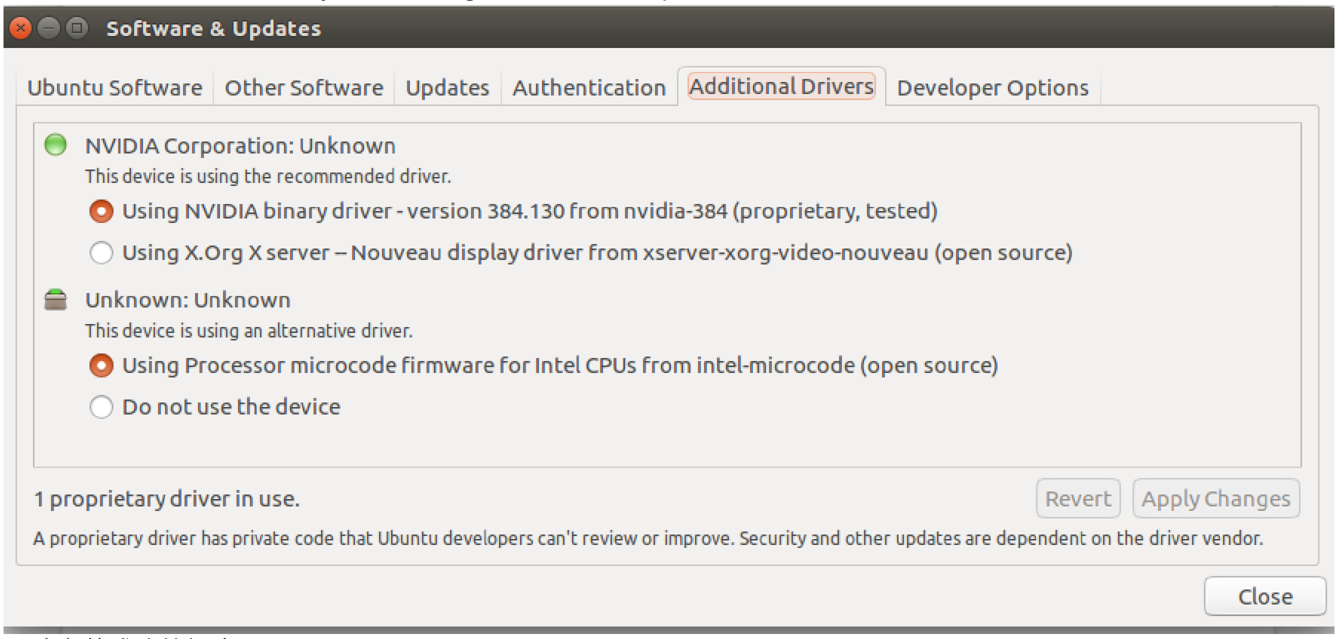
```
sudo apt-get install libprotobuf-dev libleveldb-dev libsnappy-dev libopencv-dev libhdf5-serial-dev protobuf-compiler
sudo apt-get install --no-install-recommends libboost-all-dev
sudo apt-get install libopenblas-dev liblapack-dev libatlas-base-dev
sudo apt-get install libgflags-dev libgoogle-glog-dev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

二、驱动安装(Nvidia 384.X版本的驱动)

方式一 在Terminal输入：

```
sudo apt-get update
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
sudo apt-get install nvidia-384
sudo apt-get install mesa-common-dev
sudo apt-getinstall freeglut3-dev
```

方式二 直接在Ubuntu中的System Settings-->Software&Updates中的additional drivers：



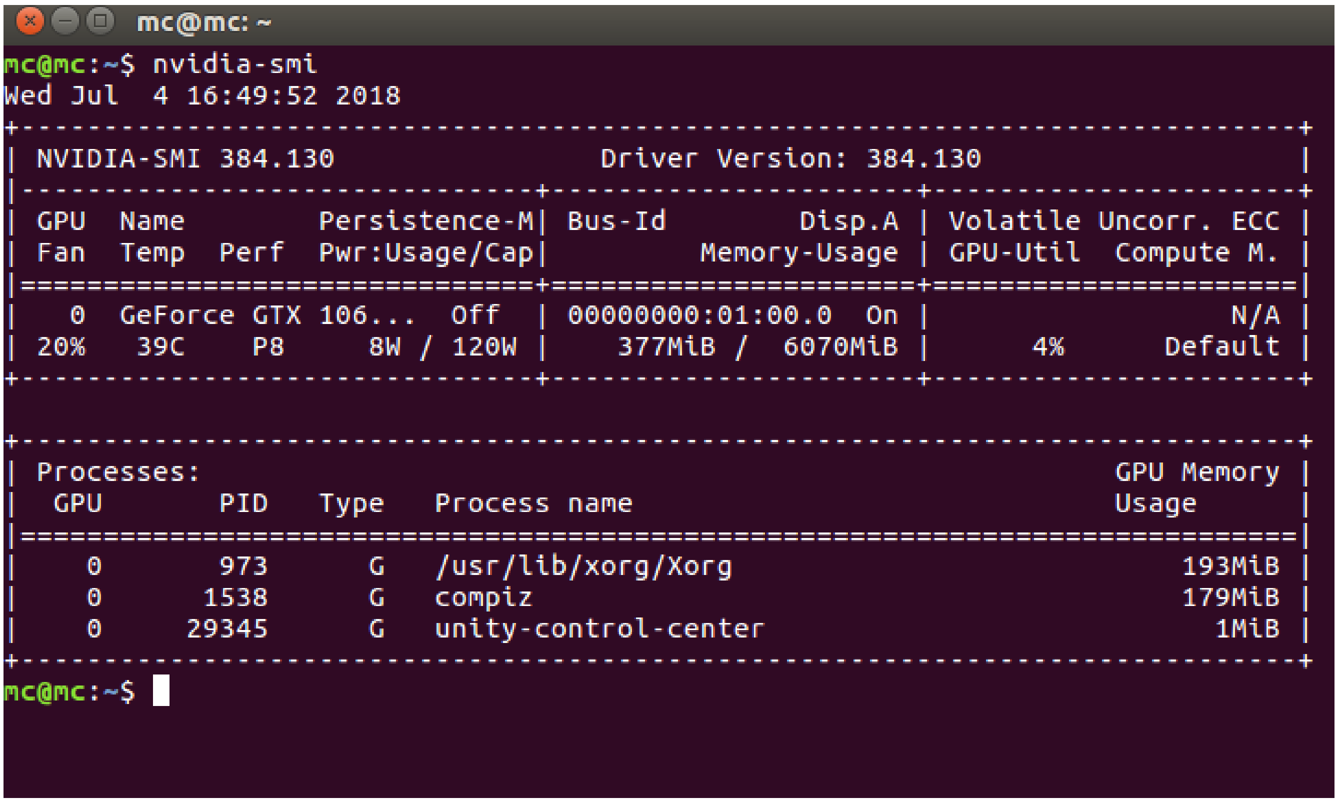
NVIDIA Corporation: Unknown
This device is using the recommended driver.
[✓ Using NVIDIA binary driver - version 384.130 from nvidia-384 (proprietary, tested)]
Using X.Org X server – Nouveau display driver from xserver-xorg-video-nouveau (open source)

Unknown: Unknown
This device is using an alternative driver.
[✓ Using Processor microcode firmware for Intel CPUs from intel-microcode (open source)]
Do not use the device

1 proprietary driver in use.

A proprietary driver has private code that Ubuntu developers can't review or improve. Security and other updates are dependent on the driver vendor.

驱动安装成功的标志：



```
mc@mc:~$ nvidia-smi
Wed Jul  4 16:49:52 2018
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 384.130                Driver Version: 384.130                   |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  GeForce GTX 106...  Off  | 00000000:01:00.0  On |                  N/A |
| 20%   39C    P8     8W / 120W |    377MiB /  6070MiB |      4%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|    0       973      G   /usr/lib/xorg/Xorg                           193MiB |
|    0      1538      G   compiz                                       179MiB |
|    0     29345      G   unity-control-center                           1MiB |
+-----------------------------------------------------------------------------+
mc@mc:~$
```

三、CUDA9.0安装

请通过官网下载CUDA安装文件（.run文件），运行文件命令如下：

```
# 先CD至.run文件的文件夹，再运行该命令
sudo sh cuda_9.0.176_384.81_linux.run
```

先按q直接跳过阅读协议，然后accept，后面的除了Install NVIDIA Accelerated Graphics Driver for Linux-x86_64 384.81?这样的选n,其它的有y选y，或者直接回车默认 检查一下环境变量

```
gedit ~/.bashr
```

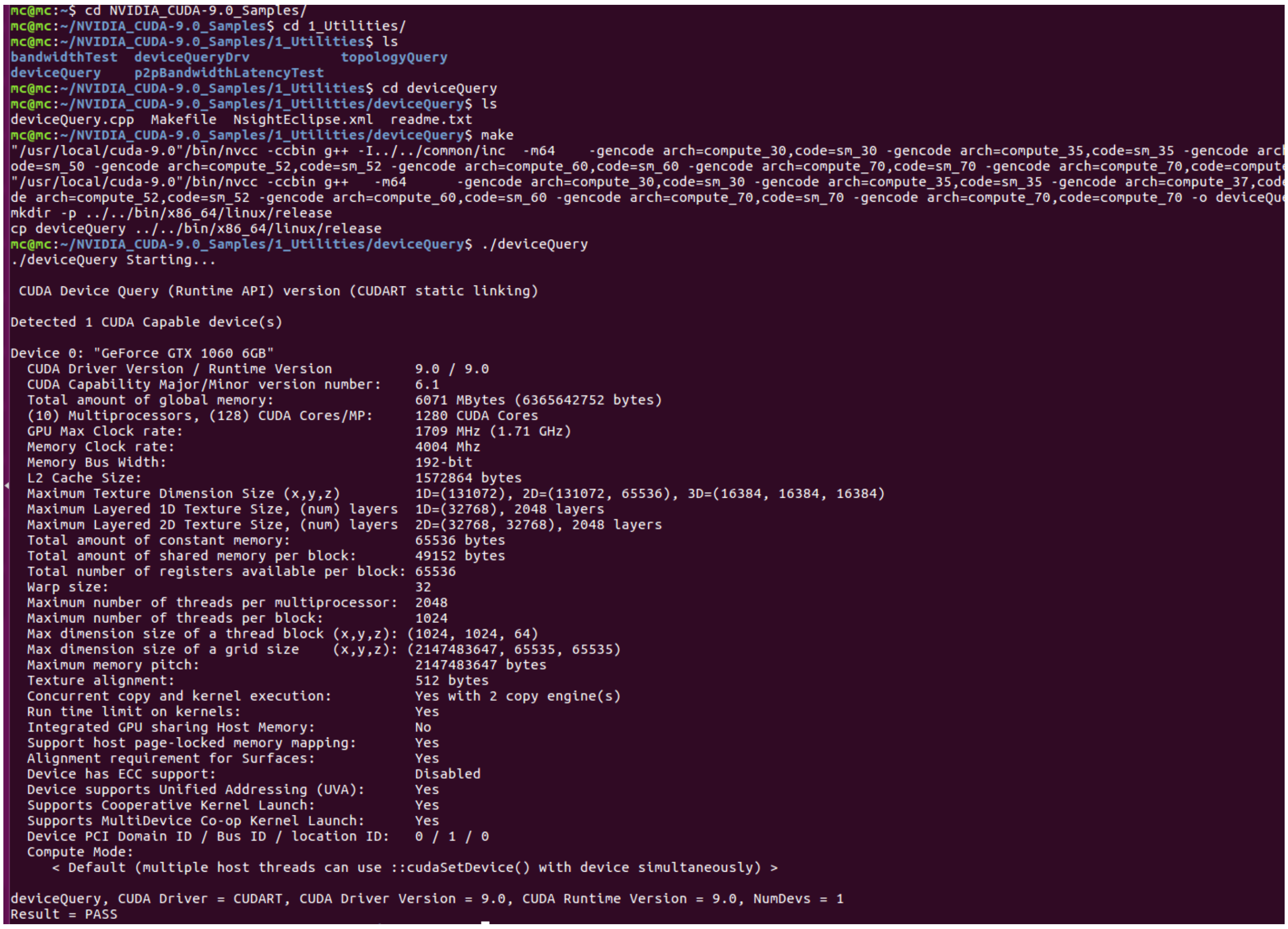
末尾添加

```
#cuda
export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64/:$LD_LIBRARY_PATH
export PATH=/usr/local/cuda-9.0/bin:$PATH
```

然后激活

```
source ~/.bashrc
```

检验安装是否完整：



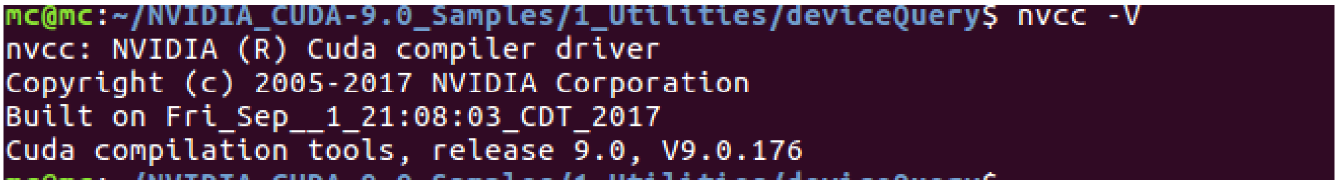
```
mc@mc:~$ cd NVIDIA_CUDA-9.0_Samples/
mc@mc:~/NVIDIA_CUDA-9.0_Samples$ cd 1_Utilities/
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities$ ls
bandwidthTest  deviceQueryDrv         topologyQuery
deviceQuery    p2pBandwidthLatencyTest
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities$ cd deviceQuery
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities/deviceQuery$ ls
deviceQuery.cpp  Makefile  NsightEclipse.xml  readme.txt
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities/deviceQuery$ make
"/usr/local/cuda-9.0"/bin/nvcc -ccbin g++ -I../../common/inc  -m64    -gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -gencode arc
ode=sm_50 -gencode arch=compute_52,code=sm_52 -gencode arch=compute_60,code=sm_60 -gencode arch=compute_70,code=sm_70 -gencode arch=compute_70,code=compute
"/usr/local/cuda-9.0"/bin/nvcc -ccbin g++   -m64      -gencode arch=compute_30,code=sm_30 -gencode arch=compute_35,code=sm_35 -gencode arch=compute_37,code
de arch=compute_52,code=sm_52 -gencode arch=compute_60,code=sm_60 -gencode arch=compute_70,code=sm_70 -gencode arch=compute_70,code=compute_70 -o deviceQue
mkdir -p ../../bin/x86_64/linux/release
cp deviceQuery ../../bin/x86_64/linux/release
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities/deviceQuery$ ./deviceQuery
./deviceQuery Starting...

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

Detected 1 CUDA Capable device(s)

Device 0: "GeForce GTX 1060 6GB"
  CUDA Driver Version / Runtime Version          9.0 / 9.0
  CUDA Capability Major/Minor version number:    6.1
  Total amount of global memory:                 6071 MBytes (6365642752 bytes)
  (10) Multiprocessors, (128) CUDA Cores/MP:     1280 CUDA Cores
  GPU Max Clock rate:                            1709 MHz (1.71 GHz)
  Memory Clock rate:                             4004 Mhz
  Memory Bus Width:                              192-bit
  L2 Cache Size:                                 1572864 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 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:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 2 copy engine(s)
  Run time limit on kernels:                     Yes
  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
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      Yes
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 9.0, CUDA Runtime Version = 9.0, NumDevs = 1
Result = PASS
```

四、CUDNN7.1安装

请通过官网下载CUDNN安装文件（.tgz文件），直接在Terminal中cd至所在文件夹，运行以下命令：

```
tar -zxvf cudnn-9.0-linux-x64-v7.1.tgz
sudo cp cuda/include/cudnn.h /usr/local/cuda/include/
sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64/ -d
sudo chmod a+r /usr/local/cuda/include/cudnn.h
sudo chmod a+r /usr/local/cuda/lib64/libcudnn*
```

检验是否安装完整：



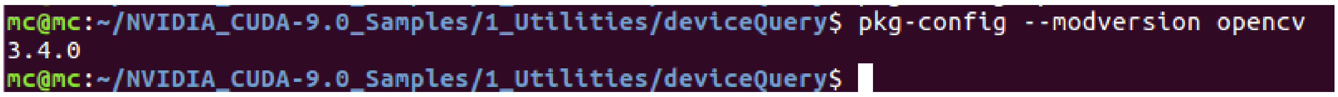
```
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities/deviceQuery$ nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri_Sep__1_21:08:03_CDT_2017
Cuda compilation tools, release 9.0, V9.0.176
```

五、Opencv源码编译安装

将下载好的opencv源码（.zip文件）解压缩至home文件夹下，然后在Terminal中输入：

```
cd ~/opencv-3.4.1
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=Release ..
sudo make -j8
sudo make install
```

安装完后检验:



```
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities/deviceQuery$ pkg-config --modversion opencv
3.4.0
mc@mc:~/NVIDIA_CUDA-9.0_Samples/1_Utilities/deviceQuery$
```

六、Caffe安装

此处我直接安装到home目录，执行：

```
cd ~
git clone https://github.com/BVLC/caffe.git #开始clone
```

等待下载结束，下载结束后在你的home路径下会存在，caffe文件夹。接下来进入caffe并开始配置caffe，配置如下:

```
sudo cp Makefile.config.example Makefile.config
sudo gedit Makefile.config #或者sudo vim Makefile.config
```

修改Makefile.config内容：

```
将:
#USE_CUDNN := 1
修改为:
USE_CUDNN := 1

将:
#OPENCV_VERSION := 3
修改为:
OPENCV_VERSION := 3

将:
#WITH_PYTHON_LAYER := 1
修改为
WITH_PYTHON_LAYER := 1

将:
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib
修改为:
INCLUDE_DIRS := $(PYTHON_INCLUDE) /usr/local/include /usr/include/hdf5/serial
LIBRARY_DIRS := $(PYTHON_LIB) /usr/local/lib /usr/lib /usr/lib/x86_64-linux-gnu
/usr/lib/x86_64-linux-gnu/hdf5/serial

将
# CUDA architecture setting: going with all of them.
# For CUDA &lt; 6.0, comment the *_50 through *_61 lines for compatibility.
# For CUDA &lt; 8.0, comment the *_60 and *_61 lines for compatibility.
# For CUDA &gt;= 9.0, comment the *_20 and *_21 lines for compatibility.
CUDA_ARCH := -gencode arch=compute_20,code=sm_20 \
           -gencode arch=compute_20,code=sm_21 \
           -gencode arch=compute_30,code=sm_30 \
           -gencode arch=compute_35,code=sm_35 \
           -gencode arch=compute_50,code=sm_50 \
           -gencode arch=compute_52,code=sm_52 \
           -gencode arch=compute_60,code=sm_60 \
           -gencode arch=compute_61,code=sm_61 \
           -gencode arch=compute_61,code=compute_61
```

改为:

```
# CUDA architecture setting: going with all of them.
# For CUDA &lt; 6.0, comment the *_50 through *_61 lines for compatibility.
# For CUDA &lt; 8.0, comment the *_60 and *_61 lines for compatibility.
# For CUDA &gt;= 9.0, comment the *_20 and *_21 lines for compatibility.
CUDA_ARCH := -gencode arch=compute_30,code=sm_30 \
           -gencode arch=compute_35,code=sm_35 \
           -gencode arch=compute_50,code=sm_50 \
           -gencode arch=compute_52,code=sm_52 \
           -gencode arch=compute_60,code=sm_60 \
           -gencode arch=compute_61,code=sm_61 \
           -gencode arch=compute_61,code=compute_61
```

修改Makefile文件:

将:

```
NVCCFLAGS +=-ccbin=$(CXX) -Xcompiler-fPIC $(COMMON_FLAGS)
```

替换为:

```
NVCCFLAGS += -D_FORCE_INLINES -ccbin=$(CXX) -Xcompiler -fPIC $(COMMON_FLAGS)
```

将:

```
LIBRARIES += glog gflags protobuf boost_system boost_filesystem m hdf5_hl hdf5
```

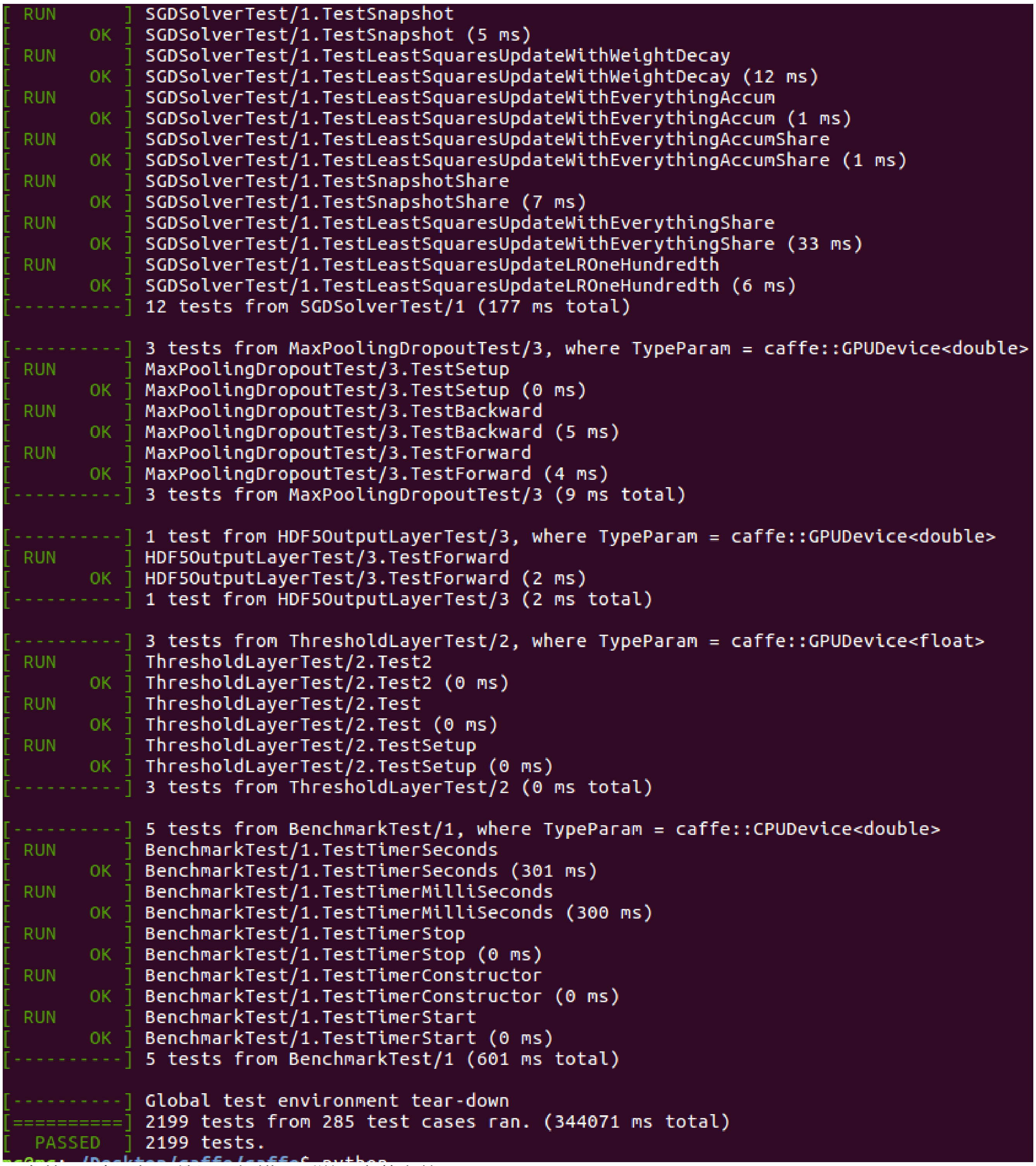
改为:

```
LIBRARIES += glog gflags protobuf boost_system boost_filesystem m hdf5_serial_hl hdf5_serial
```

配置完好之后开始编译:

```
cd caffe
sudo make clean
sudo make all #或者make all -j4(代表4核，或者j8)
sudo make test
sudo make runtest #或者sudo make runtest -j8
sudo make pycaffe
```

检验是否安装完整:



```
[ RUN      ] SGDSolverTest/1.TestSnapshot
[       OK ] SGDSolverTest/1.TestSnapshot (5 ms)
[ RUN      ] SGDSolverTest/1.TestLeastSquaresUpdateWithWeightDecay
[       OK ] SGDSolverTest/1.TestLeastSquaresUpdateWithWeightDecay (12 ms)
[ RUN      ] SGDSolverTest/1.TestLeastSquaresUpdateWithEverythingAccum
[       OK ] SGDSolverTest/1.TestLeastSquaresUpdateWithEverythingAccum (1 ms)
[ RUN      ] SGDSolverTest/1.TestLeastSquaresUpdateWithEverythingAccumShare
[       OK ] SGDSolverTest/1.TestLeastSquaresUpdateWithEverythingAccumShare (1 ms)
[ RUN      ] SGDSolverTest/1.TestSnapshotShare
[       OK ] SGDSolverTest/1.TestSnapshotShare (7 ms)
[ RUN      ] SGDSolverTest/1.TestLeastSquaresUpdateWithEverythingShare
[       OK ] SGDSolverTest/1.TestLeastSquaresUpdateWithEverythingShare (33 ms)
[ RUN      ] SGDSolverTest/1.TestLeastSquaresUpdateLROneHundredth
[       OK ] SGDSolverTest/1.TestLeastSquaresUpdateLROneHundredth (6 ms)
[----------] 12 tests from SGDSolverTest/1 (177 ms total)

[----------] 3 tests from MaxPoolingDropoutTest/3, where TypeParam = caffe::GPUDevice<double>
[ RUN      ] MaxPoolingDropoutTest/3.TestSetup
[       OK ] MaxPoolingDropoutTest/3.TestSetup (0 ms)
[ RUN      ] MaxPoolingDropoutTest/3.TestBackward
[       OK ] MaxPoolingDropoutTest/3.TestBackward (5 ms)
[ RUN      ] MaxPoolingDropoutTest/3.TestForward
[       OK ] MaxPoolingDropoutTest/3.TestForward (4 ms)
[----------] 3 tests from MaxPoolingDropoutTest/3 (9 ms total)

[----------] 1 test from HDF5OutputLayerTest/3, where TypeParam = caffe::GPUDevice<double>
[ RUN      ] HDF5OutputLayerTest/3.TestForward
[       OK ] HDF5OutputLayerTest/3.TestForward (2 ms)
[----------] 1 test from HDF5OutputLayerTest/3 (2 ms total)

[----------] 3 tests from ThresholdLayerTest/2, where TypeParam = caffe::GPUDevice<float>
[ RUN      ] ThresholdLayerTest/2.Test2
[       OK ] ThresholdLayerTest/2.Test2 (0 ms)
[ RUN      ] ThresholdLayerTest/2.Test
[       OK ] ThresholdLayerTest/2.Test (0 ms)
[ RUN      ] ThresholdLayerTest/2.TestSetup
[       OK ] ThresholdLayerTest/2.TestSetup (0 ms)
[----------] 3 tests from ThresholdLayerTest/2 (0 ms total)

[----------] 5 tests from BenchmarkTest/1, where TypeParam = caffe::CPUDevice<double>
[ RUN      ] BenchmarkTest/1.TestTimerSeconds
[       OK ] BenchmarkTest/1.TestTimerSeconds (301 ms)
[ RUN      ] BenchmarkTest/1.TestTimerMilliSeconds
[       OK ] BenchmarkTest/1.TestTimerMilliSeconds (300 ms)
[ RUN      ] BenchmarkTest/1.TestTimerStop
[       OK ] BenchmarkTest/1.TestTimerStop (0 ms)
[ RUN      ] BenchmarkTest/1.TestTimerConstructor
[       OK ] BenchmarkTest/1.TestTimerConstructor (0 ms)
[ RUN      ] BenchmarkTest/1.TestTimerStart
[       OK ] BenchmarkTest/1.TestTimerStart (0 ms)
[----------] 5 tests from BenchmarkTest/1 (601 ms total)

[----------] Global test environment tear-down
[==========] 2199 tests from 285 test cases ran. (344071 ms total)
[  PASSED  ] 2199 tests.
```

所有的test中，如果编译不报错，则说明安装完整。