Installing PyCUDA on Mac OS X

Contents

1. Installing PyCUDA on Mac OS X
    1. Pre-install Tips
    2. Step 1: Download and unpack PyCUDA
    3. Step 2: Install Numpy
    4. Step 3: Build PyCUDA
    5. Step 4: Test PyCUDA
2. Notes about Snow Leopard
    1. Fatal Python error: Interpreter not initialized (version mismatch?)
    2. Fatal Python error: pytest.py not found
3. Installing PyCUDA on Mac OS X 10.7 Lion with CUDA 4.0
    1. Step 1: Install the CUDA driver and toolkit
    2. Step 2: Get PyCUDA
    3. Step 3: Install PyCUDA
    4. Step 4: Test PyCUDA
    5. Step 5: Use PyCUDA
4. Installing PyCUDA on Mac OS X 10.8 Mountain Lion with CUDA 4.0
5. Installing PyCUDA on Mac OS X 10.9 Mavericks with CUDA 6.0
6. Optional: Install Boost

You're looking at the generic instructions for installing PyCUDA on Mac, involving the system Python. If you'd like to use a different Python version (or are running into trouble with these insructions), check these subpages:

1. **PyCuda/Installation/Mac/EnthoughtPython**

If you're running Snow Leopard (Mac OS 10.6), then you might benefit from following these slightly more up-to-date instructions:

Installing PyCUDA on Snow Leopard

The process of installing PyCUDA on Mac OS X is very similar to the Linux one. In fact, you can use the shell (Terminal) for all operations. Below are the components used to build/use PyCUDA:

- Mac OS 10.5.7 (previous versions should work fine.)
- Nvidia's CUDA toolkit. CUDA 2.2 is the latest one to date and it works fine in OS X.a

- A C++ compiler, preferably a Version 4.x gcc.
- MacOSX10.4u.sdk from Apple Developer Tools package.
- NumPy 1.2.1.
- MacPython 2.6.2. (Python 2.5.1 comes pre-installed on Mac OS X should work just fine.)

(You can install Xcode and not bother installing the C++ compiler and Mac OS SDK separately. However, Xcode will take 2+ GB of your drive.)

Pre-install Tips

- Snow Leopard comes installed with Python 2.6.1. It may be in your best interest to stick with this version for now until someone else can show us how to run a newly installed version of MacPython in 32-bit mode. Read below why we want to run python in 32-bit, but basically it has to do with the fact that CUDA 2.2 currently only supports 32-bit mode, and hence PyCUDA as well.

Step 1: Download and unpack PyCUDA

PyPi: Download PyCUDA and unpack it:

```
$ tar xfz pycuda-VERSION.tar.gz
```

Step 2: Install Numpy

PyCUDA is designed to work in conjunction with numpy, Python's array package. Here's an easy way to install it, if you do not have it already:

```
$ cd pycuda-VERSION
$ sudo "python distribute_setup.py" # this will install distribute
$ sudo "easy_install numpy" # this will install numpy using distribute
```

If you're not sure try `import numpy` at the Python prompt. Don't repeat these commands for the Mac system Python.

Step 3: Build PyCUDA

Next, just type:

```
$ cd pycuda-VERSION # if you're not there already
$ python configure.py --cuda-root=/usr/local/cuda/
```

siteconf.py shoud look like:

Toggle line numbers

```
CUDA_ROOT = '/usr/local/cuda'
CUDA_ENABLE_GL = True
CUDADRV_LIB_DIR = []
CUDADRV_LIBNAME = ['cuda']

# if on Snow Leopard, include these lines:
CXXFLAGS = ["-arch", "x86_64", "-arch", "i386"]
LDFLAGS = ["-arch", "x86_64", "-arch", "i386"]

CXXFLAGS.extend(['-isysroot', '/Developer/SDKs/MacOSX10.6.sdk'])
LDFLAGS.extend(['-isysroot', '/Developer/SDKs/MacOSX10.6.sdk'])
```

Now it's time to build PyCUDA

```
$ sudo make install
```

ensure that there are no warnings around the line of "different architecture"

Once that works, congratulations! You've successfully built PyCUDA.

Note that this step may automatically look for and, if necessary, install a number of Python packages from the Python package index. These packages are listed under `install_requires` in PyCUDA's install control file.

Step 4: Test PyCUDA

If you'd like to be extra-careful, you can run PyCUDA's unit tests:

```
$ cd pycuda-VERSION/test
$ python test_driver.py
```

If it says "OK" at the end, that means your installation of PyCUDA was successful. Reward yourself by checking out the PyCudaDoc: tutorial.

Notes about Snow Leopard

Python on Snow Leopard defaults to a 64-bit executable. You can check this by typing:

Toggle line numbers

```
import sys
print sys.maxint
```

If it's ~2 billion, you're running Python in 32-bit mode. If it's a huge number, you've got a 64-bit Python. Either should be fine.

If you'd like to, you can force Python on Snow Leopard to run in 32-bit by setting a special environment variable:

```
VERSIONER_PYTHON_PREFER_32_BIT=yes
```

Or you can make this change permanent and global by writing this to the defaults:

```
defaults write com.apple.versioner.python Prefer-32-Bit 1
```

(BryanCatanzaro)

Fatal Python error: Interpreter not initialized (version mismatch?)

If for some reason you are unable to get Boost to link against the correct python, you can use `install_name_tool` to relink it against the correct one. For example, the following command will relink from the System Python (2.6.1) to the MacPython (2.6.4) on Snow Leopard:

```
% install_name_tool -change
/System/Library/Frameworks/Python.framework/Versions/2.6/Python
/Library/Frameworks/Python.framework/Versions/2.6/Python
libboost_python.dylib
```

Fatal Python error: pytest.py not found

You might have gotten this if you install PyCUDA before PyOpenCL. Simply install pytest before running the confidence tests.

```
$ cd pycuda-VERSION/test # if you're not there already
$ sudo easy_install -U pytest
$ python test_driver.py
```

See also: py.test 2.0.0 PyOpenCL Installation for Mac

(BogdanVacaliuc)

Installing PyCUDA on Mac OS X 10.7 Lion with CUDA 4.0

Installing PyCUDA on OS X 10.7 is a bit easier than with 10.6 and other versions. This is because many of the earlier PyCUDA dependencies now come preinstalled with 10.7 Below are the components used to build/use PyCUDA:

- Mac OS X 10.7
- Nvidia's CUDA Developer Drivers here (Version 4.0.50 is current as of this entry)

- Nvidia's CUDA toolkit. here (CUDA 4.0 is current as of this entry)
- Optional - Nvidia's GPU Computing SDK here (Has some nice samples in C++ etc..)
- A C++ compiler, Installing Xcode will take care of this. (I used the i686-apple-darwin11-llvm-gcc-4.2 compiler that came with Xcode)
- Python 2.7 - Included with OS X 10.7
- NumPy 1.5.1 - Included with OS X 10.7
- Boost is not required at all so you don't need the Boost C++ libraries or the Boost.Python libraries. (You can install them if you want but I have not tested it with these instructions.)

Step 1: Install the CUDA driver and toolkit

Do all of the following:

- Install the CUDA Developer Drivers: https://developer.nvidia.com/cuda-downloads
- Install the CUDA Toolkit
- Make sure your path variables are set

You can either type the following in the shell or add it to the ~/.bash_profile file (If ~/.bash_profile does not exist then create it and add the following lines)

```
$ export PATH=/usr/local/cuda/bin:$PATH
$ export DYLD_LIBRARY_PATH=/usr/local/cuda/lib:$DYLD_LIBRARY_PATH
```

Step 2: Get PyCUDA

Download PyCUDA. There are multiple ways to do this. I prefer letting OS X do the hard work by going here and letting the archiver unzip it by doubling clicking the download when it's finished.

Step 3: Install PyCUDA

Next, just type:

```
$ cd pycuda-VERSION # if you're not there already
$ python configure.py --cuda-root=/usr/local/cuda/
```

siteconf.py shoud look like (at least mine did):

Toggle line numbers

```
BOOST_INC_DIR = []
BOOST_LIB_DIR = []
BOOST_COMPILER = 'gcc43'
USE_SHIPPED_BOOST = True
```

```
BOOST_PYTHON_LIBNAME = ['boost_python']
BOOST_THREAD_LIBNAME = ['boost_thread']
CUDA_TRACE = False
CUDA_ROOT = '/usr/local/cuda/'
CUDA_ENABLE_GL = False
CUDA_ENABLE_CURAND = True
CUDADRV_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
CUDADRV_LIBNAME = ['cuda']
CUDART_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
CUDART_LIBNAME = ['cudart']
CURAND_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
CURAND_LIBNAME = ['curand']
CXXFLAGS = []
LDFLAGS = []

# We need to remove the lib64 references from three lines because OSX 10.7 is only 64 bit, the CUDA toolkit only has one version and it is 64bit, it's just not called lib64.

CUDADRV_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
to
CUDADRV_LIB_DIR = ['${CUDA_ROOT}/lib']

CUDART_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
to
CUDART_LIB_DIR = ['${CUDA_ROOT}/lib']

CURAND_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
to
CURAND_LIB_DIR = ['${CUDA_ROOT}/lib']


# We now need to change the CXXFLAGS and LDFLAGS to look like this:
CXXFLAGS = ["-arch", "x86_64", "-arch", "i386"]
LDFLAGS = ["-arch", "x86_64", "-arch", "i386"]
# We also need to add these lines
CXXFLAGS.extend(['-isysroot', '/Developer/SDKs/MacOSX10.6.sdk'])
LDFLAGS.extend(['-isysroot', '/Developer/SDKs/MacOSX10.6.sdk'])
```

You will notice that we just added a line referring to 'MacOSX10.6.sdk'. You should look in the /Developer/SDKs/ folder and see two sdk files. OSX 10.7 should have:

- MacOSX10.6.sdk
- MacOSX10.7.sdk

(Because it's there and works with other builds, I have opted to use the 10.6 sdk)

After editing siteconf.py shoud look like (at least mine did):

Toggle line numbers

```
BOOST_INC_DIR = []
BOOST_LIB_DIR = []
BOOST_COMPILER = 'gcc43'
USE_SHIPPED_BOOST = True
BOOST_PYTHON_LIBNAME = ['boost_python']
BOOST_THREAD_LIBNAME = ['boost_thread']
```

```
CUDA_TRACE = False
CUDA_ROOT = '/usr/local/cuda/'
CUDA_ENABLE_GL = False
CUDA_ENABLE_CURAND = True
CUDADRV_LIB_DIR = ['${CUDA_ROOT}/lib']
CUDADRV_LIBNAME = ['cuda']
CUDART_LIB_DIR = ['${CUDA_ROOT}/lib']
CUDART_LIBNAME = ['cudart']
CURAND_LIB_DIR = ['${CUDA_ROOT}/lib']
CURAND_LIBNAME = ['curand']
CXXFLAGS = ["-arch", "x86_64", "-arch", "i386"]
LDFLAGS = ["-arch", "x86_64", "-arch", "i386"]
CXXFLAGS.extend(['-isysroot', '/Developer/SDKs/MacOSX10.6.sdk'])
LDFLAGS.extend(['-isysroot', '/Developer/SDKs/MacOSX10.6.sdk'])
```

run make

```
$ sudo make
```

(This might take some time)

Now it's time to build PyCUDA

```
$ sudo "make install"
```

(This might also take some time)

ensure that there are no warnings around the line of "different architecture" (yes I copied this)

PyCUDA should now be built

Note that this step may automatically look for and, if necessary, install a number of Python packages from the Python package index. These packages are listed under `install_requires` in PyCUDA's install control file.

Step 4: Test PyCUDA

If you'd like to be extra-careful, you can run PyCUDA's unit tests: (I recommend this)

```
$ cd pycuda-VERSION/test
$ python test_driver.py
```

Step 5: Use PyCUDA

At this point installation should be done you should be able to successfully use PyCUDA. Happy Developing.

Installing PyCUDA on Mac OS X 10.8 Mountain Lion with CUDA 4.0

Here is the siteconf.py that enabled it to compile (upgrade from Lion):

Toggle line numbers

```
BOOST_INC_DIR = []
BOOST_LIB_DIR = []
BOOST_COMPILER = 'gcc43'
USE_SHIPPED_BOOST = True
BOOST_PYTHON_LIBNAME = ['boost_python']
BOOST_THREAD_LIBNAME = ['boost_thread']
CUDA_TRACE = False
CUDA_ROOT = '/usr/local/cuda/'
CUDA_ENABLE_GL = False
CUDA_ENABLE_CURAND = True
CUDADRV_LIB_DIR = ['${CUDA_ROOT}/lib']
CUDADRV_LIBNAME = ['cuda']
CUDART_LIB_DIR = ['${CUDA_ROOT}/lib']
CUDART_LIBNAME = ['cudart']
CURAND_LIB_DIR = ['${CUDA_ROOT}/lib']
CURAND_LIBNAME = ['curand']
CXXFLAGS = ["-arch", "x86_64", "-arch", "i386", "-mmacosx-version-min=10.7"]
LDFLAGS = ["-F/Library/Frameworks","-framework CUDA","-arch",
"x86_64", "-arch", "i386", "-mmacosx-version-min=10.7"]
CXXFLAGS.extend(['-isysroot',
'/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.7.sdk'])
LDFLAGS.extend(['-isysroot',
'/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.7.sdk'])
```

Installing PyCUDA on Mac OS X 10.9 Mavericks with CUDA 6.0

Here is the siteconf.py that enabled it to compile:

Toggle line numbers

```
BOOST_INC_DIR = []
BOOST_LIB_DIR = []
BOOST_COMPILER = 'gcc43'
USE_SHIPPED_BOOST = True
BOOST_PYTHON_LIBNAME = ['boost_python-py27']
BOOST_THREAD_LIBNAME = ['boost_thread']
CUDA_TRACE = False
CUDA_ROOT = '/usr/local/cuda'
CUDA_ENABLE_GL = False
```

```
CUDA_ENABLE_CURAND = True
CUDADRV_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
CUDADRV_LIBNAME = ['cuda']
CUDART_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
CUDART_LIBNAME = ['cudart']
CURAND_LIB_DIR = ['${CUDA_ROOT}/lib', '${CUDA_ROOT}/lib64']
CURAND_LIBNAME = ['curand']
CXXFLAGS = ["-arch", "x86_64", "-arch", "i386"]
LDFLAGS = ["-F/Library/Frameworks", "-arch", "x86_64", "-arch", "i386"]
CXXFLAGS.extend(['-isysroot', '/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.9.sdk'])
LDFLAGS.extend(['-isysroot', '/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX10.9.sdk'])
```

Optional: Install Boost

As of version 0.94, the Boost C++ libraries are *no longer* a dependency of PyCUDA.

If you would like to use multiple packages that use Boost.Python (e.g. both PyCUDA and PyOpenCL), you *can* still build PyCUDA against a separate Boost installation. If so, follow the BoostInstallationHowto. Continue here when you're done.

- If you are installing boost in /usr/local instead of $HOME/pool, you will need to run the commands as 'sudo', otherwise you will get permission denied errors.
- LD_LIBRARY_PATH in Mac OS X is called DYLD_LIBRARY_PATH.
- Make sure your bash shell startup file '.bash_profile' has an entry for the library path. Something like:

```
export
DYLD_LIBRARY_PATH=/usr/local/cuda/lib:$HOME/pool/lib:$DYLD_LIBRARY_PATH
```

$HOME/pool/lib is where you installed Boost to. Change it if installed elsewhere.

Note that gcc42 is a compiler tag that depends on the compiler with which you built boost. Check the contents of your boost library directory to find out what the correct tag is. The instructions above assume that the boost libraries were installed in $HOME/pool.

Continue here when you're done installing Boost.

This warning was added at some point--I'm not sure if it's still relevant now:

> Stick with Boost 1.39 instead of the latest (as of Jan 23, 2010 is 1.41). Follow the directions to install 1.39 at BoostInstallationHowto and pay attention to the Mac notes. The issue with running either the earlier or later versions of Boost are the build options you can specify. Again it goes back to the fact that we want to build 32-bit versions of all libraries we use on Snow Leopard so it is CUDA compatible. I also vaguely recall seeing numerous Boost compile warnings when using 1.41 and building PyCUDA. PyCUDA is probably preferring a particular version of Boost

with given signatures, so 1.39 worked best for now.

-- AndreasKloeckner 2010-09-21 00:16:32

If you are getting the following error:
`ld: library not found for -lboost_python-xgcc40-mt` and installed in `/usr/lib`, try to re-install boost somewhere else or just "cp -vf /usr/lib/libboost* ." and use --boost-lib-dir=./ with configure.py

CategoryPyCuda

PyCuda/Installation/Mac (last edited 2014-11-07 18:16:01 by ::ffff:98)