Installing and Running Tensorflow

CADnetwork
Premium. Performance. Innovation.

DOWNLOAD AND INSTALLATION INSTRUCTIONS

TensorFlow is now distributed under an Apache v2 open source license on GitHub.

STEP 1. INSTALL NVIDIA CUDA

To use TensorFlow with NVIDIA GPUs, the first step is to install the CUDA Toolkit.

STEP 2. INSTALL NVIDIA CUDNN

Once the CUDA Toolkit is installed, download cuDNN v5.1 Library for Linux (note that you will need to register for the Accelerated Computing Developer Program).

Once downloaded, uncompress the files and copy them into the CUDA Toolkit directory (assumed here to be in /usr/local/cuda/):

```
$ sudo tar -xvf cudnn-8.0-linux-x64-v5.1-rc.tgz -C /usr/local
```

STEP 3. INSTALL AND UPGRADE PIP

TensorFlow itself can be installed using the pip package manager. First, make sure that your system has pip installed and updated:

```
$ sudo apt-get install python-pip python-dev
$ pip install --upgrade pip
```

STEP 4. INSTALL BAZEL

To build TensorFlow from source, the Bazel build system must first be installed as follows. Full details are available here.

```
$ sudo apt-get install software-properties-common swig
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
$ echo "deb http://storage.googleapis.com/bazel-apt stable jdk1.8" | sudo tee
/etc/apt/sources.list.d/bazel.list
$ curl https://storage.googleapis.com/bazel-apt/doc/apt-key.pub.gpg | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install bazel
```

STEP 5. INSTALL TENSORFLOW

To obtain the best performance with TensorFlow we recommend building it from source. First, clone the TensorFlow source code repository:

```
$ git clone https://github.com/tensorflow/tensorflow
$ cd tensorflow
$ git reset --hard 70de76e
```

Then run the configure script as follows:

```
$ ./configure
Please specify the location of python. [Default is /usr/bin/python]: [enter]
Do you wish to build TensorFlow with Google Cloud Platform support? [y/N] n
No Google Cloud Platform support will be enabled for TensorFlow
Do you wish to build TensorFlow with GPU support? [y/N] y
GPU support will be enabled for TensorFlow
Please specify which gcc nvcc should use as the host compiler. [Default is /usr/bin/gcc]: [enter]
Please specify the Cuda SDK version you want to use, e.g. 7.0. [Leave empty to use system default]: 8.0
Please specify the location where CUDA 8.0 toolkit is installed. Refer to README.md for more details.
[Default is /usr/local/cuda]: [enter]
Please specify the Cudnn version you want to use. [Leave empty to use system default]: 5
Please specify the location where cuDNN 5 library is installed. Refer to README.md for more details.
[Default is /usr/local/cuda]: [enter]
Please specify a list of comma-separated Cuda compute capabilities you want to build with.
You can find the compute capability of your device at: https://developer.nvidia.com/cuda-gpus.
Please note that each additional compute capability significantly increases your build time and binary size.
[Default is: "3.5,5.2"]: 5.2,6.1 [see https://developer.nvidia.com/cuda-gpus]
Setting up Cuda include
Setting up Cuda lib64
Setting up Cuda bin
Setting up Cuda nvvm
Setting up CUPTI include
Setting up CUPTI lib64
Configuration finished
```

Then call bazel to build the TensorFlow pip package:

```
bazel build -c opt --config=cuda //tensorflow/tools/pip_package:build_pip_package
bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
```

And finally install the TensorFlow pip package

Python 2.7:

```
$ sudo pip install --upgrade /tmp/tensorflow_pkg/tensorflow-0.9.0-*.whl
```

Python 3.4:

```
$ sudo pip install --upgrade /tmp/tensorflow_pkg/tensorflow-0.9.0-*.whl
```

STEP 5. UPGRADE PROTOBUF

Upgrade to the latest version of the protobuf package:

Python 2.7:

```
$ sudo pip install --upgrade https://storage.googleapis.com/tensorflow/linux/cpu/protobuf-3.0.0b2.post2-cp27-none-linux_x86_64.whl
```

Python 3.4:

```
$ sudo pip3 install --upgrade https://storage.googleapis.com/tensorflow/linux/cpu/protobuf-3.0.0b2.post2-cp34-none-linux_x86_64.whl
```

STEP 6. TEST YOUR INSTALLATION

To test the installation, open an interactive Python shell and import the TensorFlow module:

```
$ cd
$ python
...
>>> import tensorflow as tf
I tensorflow/stream_executor/dso_loader.cc:105] successfully opened CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:105] successfully opened CUDA library libcudnn.so locally
I tensorflow/stream_executor/dso_loader.cc:105] successfully opened CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:105] successfully opened CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:105] successfully opened CUDA library libcurand.so locally
```

With the TensorFlow module imported, the next step to test the installation is to create a TensorFlow Session, which will initialize the available computing devices and provide a means of executing computation graphs:

```
>>> sess = tf.Session()
```

This command will print out some information on the detected hardware configuration. For example, the output on a system containing a Tesla M40 GPU is:

```
>>> sess = tf.Session()
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0 with properties:
name: Tesla M40
major: 5 minor: 2 memoryClockRate (GHz) 1.112
pciBusID 0000:04:00.0
Total memory: 11.25GiB
Free memory: 11.09GiB
...
```

To manually control which devices are visible to TensorFlow, set the CUDA_VISIBLE_DEVICES environment variable when launching Python. For example, to force the use of only GPU 0:

```
$ CUDA_VISIBLE_DEVICES=0 python
```

You should now be able to run a Hello World application:

```
>>> hello_world = tf.constant("Hello, TensorFlow!")
>>> print sess.run(hello_world)
Hello, TensorFlow!
>>> print sess.run(tf.constant(123)*tf.constant(456))
56088
```

- See more at: http://www.nvidia.com/object/gpu-accelerated-applications-tensorflow-installation.html#sthash.98tyHyl6.dpuf