Like this page? Share with your friends Share

THE LAZY LOG

HOME ABOUT

email address

Subscribe

04 Sep 2016 on debugging • deep learning • tensorflow

Install TensorFlow with GPU support on a RedHat (supercluster)

I am working on a deep learning model for text summarization and I use TensorFlow as my main framework. It is a great framework and contains many built-in functions to ease the implementation. However, when I trained my model, it was too slow. It took about 7 seconds to train a batch containing 10 documents. Considering that my dataset contains about 150,000 documents, it would take around 29 hours to train all documents per epoch, which is way too slow. Generally it took more than 10 epochs for the model to converge, so I had to figure out how to resolve this issue. Beside optimizing the neural network, a preferred option is to train my model on GPU to take advantage of the fast matrix computation.

Unfortunately, the TensorFlow package installed using `pip` on the supercluster at my school did not support GPU. The reason is because it requires both CUDA 7.5 and CuDNN to be installed, which they do not support on the server. They have CUDA but its version is 7.0 and there are no CuDNN at all. Hence, according to TensorFlow tutorial, my best option was to build TensorFlow from source. Even though the guideline on TensorFlow website is simple, getting it to work took me a lot of efforts. But it was totally worth it and I am happy with the result. In this post, I want to walk you through the steps I did to successfully compile TensorFlow with GPU. It took me about 1 day to fix issues occurred during the compiling process and I hope it would take you less than 1 hour following my tutorial.

1. Environment Information

Server: RedHat (CentOS release 6.5 (Final))
Python: 2.7.11 :: Anaconda custom (64-bit)
Default `gcc` (at `/usr/bin/gcc`): gcc version 4.4.7 20120313 (Red Hat 4.4.7-4) (GCC)
Default `ld` (at `/usr/bin/ld`): GNU ld version 2.20.51.0.2-5.36.el6 20100205

2. Compiling Bazel

`glibc 2.14`. Of course we could not upgrade `glibc` since that I did not have root permission. So as you expected, we have to build it from source. (I found great help from this link: https://github.com/bazelbuild/bazel/issues/418#issuecomment-156147911)

Because of an issue related to `tmp` folder, we have to build bazel directly from within this folder.

```
cd /tmp
git clone https://github.com/bazelbuild/bazel.git
cd /tmp/bazel
```

Before compiling, I had to switch to another `gcc`. The default `gcc` (on `/usr/bin/gcc`) on server was 4.4.7 and I could not compile `bazel` with it. I instead activated gnu_4.9.2 module:

```
module load gnu_4.9.2
which gcc
/opt/rh/devtoolset-3/root/usr/bin/gcc
```

Next, replace all references to old `gcc` with the new `gcc`

```
setenv GCC `which gcc`
sed -i -e "s=/usr/bin/gcc=$GCC=g" tools/cpp/CROSSTOOL
```

Noted that my OS was RedHat, that's why I used `setenv` command, if you are using Linux then you should use `export` command (ex: `` export GCC=`which gcc` ``)

Then, prepare the `bazelrc` file on `/tmp` to pass options when compiling Bazel.

```
cd /tmp
cat >/tmp/bazelrc <<EOF
startup --batch
build --spawn_strategy=standalone --genrule_strategy=standalone
EOF
```

Let's build it

```
cd /tmp/bazel
BAZELRC=/tmp/bazelrc ./compile.sh
```

You should see the following output:

Copy `bazel` binary to your own directory, I'll refer the path to `bazel` binary is BAZEL_PATH

3. Compiling TensorFlow

Activate cuda 7.0 and Java 1.8 module first

```
module unload cuda/6.5
module load cuda/7.0
module load java/java8
```

Download CuDNN from Nvidia website (you might need to register for download), extract it into a folder, for example: `~/software/cudnn`. Update `$LD_LIBRARY_PATH` environment variable (you should put it in ~/.bash_profile or ~/.cshrc):

```
setenv LD_LIBRARY_PATH ${HOME}/software/cudnn/lib64:${LD_LIBRARY_PATH}
```

Configure tensorflow environments for building

```
./configure
```

This script will ask for several pieces of information, it is important to enter them correctly. The answer on each system might be different. Empty answer mean using the default option.

```
Please specify the location of python. [Default is /home/s1510032/anaconda2/bin/python]:
Do you wish to build TensorFlow with Google Cloud Platform support? [y/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 should be used by nvcc as the host compiler. [Default is

/opt/rh/devtoolset-3/root/usr/bin/gcc]:
Please specify the Cuda SDK version you want to use, e.g. 7.0. [Leave empty to use system
default]:
Please specify the location where CUDA  toolkit is installed. Refer to README.md for more
details. [Default is /usr/local/cuda]: /opt/cuda/7.0
Please specify the Cudnn version you want to use. [Leave empty to use system default]:
Please specify the location where cuDNN  library is installed. Refer to README.md for more
details. [Default is /usr/local/cuda]: ~/software/cuda
libcudnn.so resolves to libcudnn.4
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-
```

```
time and binary size.
[Default is: "3.5,5.2"]:
```

Now come the hardest part. The next step is to build a pip package for tensorflow by using Bazel. On TensorFlow website, it's written simply in a single command like this:

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

But things are not as easy as like that since we are on a shared hosting, there is no root access and installed libraries can be different. We need to update several files before running the above command

3.1. Modify CROSSTOOL

By default, the script to build tensorflow assumes that our `gcc` is at `/usr/bin/gcc` and our libraries are at `/usr/lib/gcc`. This information is hard-coded in the file third_party/gpus/crosstools/CROSSTOOL.tpl. But remember that we have switched to using `gnu_4.9.2`, so we need to update the CROSSTOOL file with new `gcc` path. In addition, `ld` command is no longer `/usr/bin/ld` so we need to update it as well. Open CROSSTOOL file using your favorite editor (vi, nano,...) and do the following:

- replace all references to `/usr/bin/gcc` with `/opt/rh/devtoolset-3/root/usr/bin/gcc` (using the command `which gcc` to find this path on your system)
- replace all `/usr/bin/ld` with '/opt/rh/devtoolset-3/root/usr/bin/ld`
- replace `linker_flag: "-B/usr/bin"` with `linker_flag: "-B/opt/rh/devtoolset-3/root/usr/bin"`

In addition, we need to change the paths to header files which will be used when compiling tensorflow. Paths to header files are specified with option `cxx_builtin_include_directory` in CROSSTOOL file. Here are old attributes:

```
cxx_builtin_include_directory: "/usr/lib/gcc/"
cxx_builtin_include_directory: "/usr/local/include"
cxx_builtin_include_directory: "/usr/include"
```

After updating:

```
cxx_builtin_include_directory: "/opt/rh/devtoolset-3/root/usr/lib/gcc/x86_64-redhat-
linux/4.9.2/include"
cxx_builtin_include_directory: "/opt/rh/devtoolset-3/root/usr/include/c++/4.9.2"
cxx_builtin_include_directory: "/opt/cuda/7.0/include"
cxx_builtin_include_directory: "/usr/include"
```

This file is located at `third_party/gpus/crosstool/clang/bin/crosstool_wrapper_driver_is_not_gcc.tpl`, open this file and replace the `LLVM_HOST_COMPILER_PATH` with new `gcc` path like this:

```
LLVM_HOST_COMPILER_PATH = ('/opt/rh/devtoolset-3/root/usr/bin/gcc')
```

You should also pay attention to the first line in this file, it is something like: `#!/usr/bin/env python`. By default, this file will be run by the default Python interpreter of the system. Since I was using Anaconda and my python was different, I had to change this as well. You should replace this line with your own python path, like this:

```
#!/home/s1510032/anaconda2/bin/python
```

If you do not do this, you might get an error about `argparse` module when building tensorflow. This module might not available in your default system Python interpreter so it is better to use your own interpreter, in which you can install any package you like.

3.3. Create `~/.bazelrc`

This file is different from the file we created to build Bazel, this file is for building TensorFlow. Create this file at your home directory with the content like this:

```
build --verbose_failures --linkopt=-Wl,-rpath,/opt/rh/devtoolset-3/root/usr/lib64 --linkopt=-Wl,-rpath,/u/drspeech/opt/jdks/jdk1.8.0_25/lib --linkopt=-lz --linkopt=-lrt --linkopt=-lm --genrule_strategy=standalone --spawn_strategy=standalone --linkopt=-Wl,-rpath,/opt/cuda/7.0/lib64
```

3.4 Build TensorFlow

Okay, it seems we are almost done. Try building tensorflow with GPU support:

```
bazel build -c opt --config=cuda --genrule_strategy=standalone --spawn_strategy=standalone //tensorflow/tools/pip_package:build_pip_package
```

It would take several minutes for it to compile, and then it might stop with an error like this:

```
Undefined reference to symbol 'ceil@@GLIBC_2.2.5' at build time
```

```
LINK_OPTS = select({
    ":android": [],
    "//conditions:default": ["-lpthread","-lrt","-lm"],
})
```

(Some people said that we need to add extra linker flags in `google/protobuf/BUILD` but I have no idea where this file is.

After fixing, run the command again to build tensorflow:

```
bazel build -c opt --config=cuda --genrule_strategy=standalone --spawn_strategy=standalone
//tensorflow/tools/pip_package:build_pip_package
```

You should get no error messages this time. After running this successfully, run the command to build Wheel (python distribution)

```
bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
```

And install this new package with `pip` (make sure you already removed old `tensorflow` before installing)

```
pip install /tmp/tensorflow_pkg/tensorflow-0.10.0rc0-py2-none-any.whl
```

4. Testing

You can check whether your new tensorflow works as you expected or not by running the following script in python. If there are no errors, it means tensorflow has been installed correctly.

```
import tensorflow as tf
# Creates a graph.
with tf.device('/gpu:0'):
  a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
  b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
  c = tf.matmul(a, b)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print sess.run(c)
```

Like this page? Share with your friends Share

While building tensorflow, you might get this error:

```
ERROR: no such package '@local_config_cuda//crosstool': BUILD file not found on package
path.
```

This seems to be an error with the Bazel process. You need to kill bazel process (ps aux | grep bazel) and re-run the above procedure (from `./configure` step). It is a pain in the ass but I have no idea of a better way to fix this issue

Also, do your search on Issues page of tensorflow's github, I got many help and guidelines from people got similar error messages. For examples:

- https://github.com/bazelbuild/bazel/issues/418#issuecomment-156147911
- https://github.com/tensorflow/tensorflow/issues/110#issuecomment-201834137
- https://github.com/bazelbuild/bazel/issues/934#issuecomment-193474914

Do you have any other issues when building TensorFlow with GPU support? Does this post help you? Leave your comment and feedback. Good luck!

SHARE THIS POST

Like this page? Share with your friends Share

Recommend Share Sort by Best

Join the discussion…

rohan • 7 months ago
Hi Chien, I tried following this post to install tensorflow with gpu support.

However, a) i did not find any crosstool_wrapper_driver_is_not_gcc.tpl file so i skipped that step
b) i get this error when i try to build tensorflow:
ERROR: The specified --crosstool_top '//third_party/gpus/crosstool:crosstool' is not a valid cc_toolchain_suite rule.

Have you ever encountered this error message? Thanks!
Reply • Share ›

Aditya Chaganti → rohan • 6 months ago
Hey Rohan, did you get over this?
Reply • Share ›

Chien Tran Mod → rohan • 7 months ago
Are you sure there is no crosstool_wrapper_driver_is_not_gcc.tpl file? As I've just checked the latest code of TensorFlow on Github, it is still there. So I think you should recheck again, or you need to redownload the tensorflow code.
Here is the file on github: https://github.com/tensorfl...
Reply • Share ›

rohan → Chien Tran • 7 months ago
Do you happen to have any idea about this error though?:

ERROR: The specified --crosstool_top '//third_party/gpus/crosstool:crosstool' is not a valid cc_toolchain_suite rule.

the error message is the same as the one here, but I'm not sure if this is related:
https://github.com/tensorfl....
Reply • Share ›

Chien Tran Mod → rohan • 7 months ago
I did not see that error when compiling the code on my side. Perhaps you should follow that issue post on Github, it seems related to yours. If you are using HEAD of tensorflow, it might contain many changes since the last time I posted this.
You can also try pulling a specific "release" like r0.11 and see if it works better than HEAD (it is definitely more stable).
https://github.com/tensorfl...
1 • Reply • Share ›

rohan → Chien Tran • 7 months ago
Oh you're right, thanks!
Reply • Share ›

ALSO ON **THE LAZY LOG**

UNIX commands for daily tasks
6 comments • 10 months ago•
Siraset Jirapatchandej — I like $ tail -f log and $ less +F log , too :)

How to create private git server
1 comment • 2 years ago•
mcchots — Thanks for this writeup.With regards to the clone command git clone ubuntu@example.com:test.git should also work just fine.

Deploying Rails application with Nginx, Puma and Mina
26 comments • 2 years ago•
Chien Tran — According to your error message, I think you should update rake in your local machine to a different version,

Install python and pip as local user on shared Linux
29 comments • a year ago•
Chien Tran — I don't know if this would help but please try: http://stackoverflow.com/a/...