Kejie Wang

An introduction to package tensorflow c++ library

2018-03-01 | ml | 0 comments | 832

Currently, building a c++ tensorflow program requires creating your project in the tensorflow source code tree and compile it with bazel togather with tensorflow source code. But in some cases, we may want to use tensorflow as a bazel-indepent library and can be linked with. The official tensorflow has not provided a tutorial about how to build a standalone tensorflow c++ library. This blog will introduce how to compile tensorflow source code and package tensorflow c++ library as a bazel-independent use.

Prerequisite

Using bazel, an open-source build and test tools developed by google, to build tensorflow is officially suppoorted. Follow the instruction to install the bazel. Here are a simple note on how to install bazel on Ubunut.

- **Install required packages**

```
1 sudo apt-get install pkg-config zip g++ zlib1g-dev unzip python
```

- **Download bazel from [Github releases page**](https://github.com/bazelbuild/bazel/releases)

```
1 wget https://github.com/bazelbuild/bazel/releases/download/0.5.4/bazel-0.5.4-installer·
```

- **Run the installer**

If you want to use tensorflow gpu version, cuda toolkit and cudnn are required. There are many

tutorials about how to install cuda and cudnn and therefore here will not introduce them.

Compile tensorflow

- **Download tensorflow code from Github releases page.**

```
# Download the source code (Take 1.5.0 version as example)
$ wget https://github.com/tensorflow/tensorflow/archive/v1.5.0.tar.gz
# Extract the code
$ tar -zxf v1.5.0.tar.gz
```

Using git clone to get the latest code on Github maybe ok. Get a copy of tensorflow source code anyway.

- **Configure your compile.**

```
cd tensorflow-1.5.0 && ./configure
```

If you do not want to an interactive configure, you can export the configuration using environment variables. Here is an example configuration.

```
export CC_OPT_FLAGS="-march=native"
export TF_NEED_GCP=0
export TF_NEED_HDFS=0
export TF_NEED_OPENCL=0
export TF_NEED_OPENCL_SYCL=0
export TF_NEED_TENSORRT=0
export TF_NEED_JEMALLOC=1
export TF_NEED_VERBS=0
export TF_NEED_MKL=1
export TF_DOWNLOAD_MKL=1
export TF_NEED_MPI=0
export TF_ENABLE_XLA=1
export TF_NEED_S3=0
export TF_NEED_GDR=0
export TF_CUDA_CLANG=0
export TF_SET_ANDROID_WORKSPACE=0
export TF_NEED_KAFKA=0
export PYTHON_BIN_PATH="$(which python)"
export PYTHON_LIB_PATH="$($PYTHON_BIN_PATH -c 'import site; print(site.getsitepac
```

```
export TF_NEED_CUDA=1
export TF_CUDA_VERSION=8.0
export CUDA_TOOLKIT_PATH=/usr/local/cuda
export TF_CUDA_COMPUTE_CAPABILITIES="3.5,5.2,6.1,6.2"
export TF_CUDNN_VERSION=5
export CUDNN_INSTALL_PATH=/usr/lib/x86_64-linux-gnu
export GCC_HOST_COMPILER_PATH=/usr/bin/gcc
export TF_CUDA_CLANG=0
export TF_SET_ANDROID_WORKSPACE=''
```

- **Compile tensorflow**

```
bazel build -c opt --config=cuda --copt=-march=native tensorflow:libtensorflow_cc.so
```

- **Installation**

Take the /usr/local/tensorflow as example installation prefix.

  - **Create the installation directory.**

```
sudo mkdir -p /usr/local/tensorflow/include /usr/local/tensorflow/lib
```

  - **Copy the header files**

```
sudo cp -r tensorflow /usr/local/tensorflow/include
sudo cp -r third_party /usr/local/tensorflow/include
sudo cp bazel-genfiles/tensorflow /usr/local/tensorflow/include
# Delete the files that are not header files
sudo find /usr/local/tensorflow/include -type f ! -name "*.h" -delete
```

  - **Copy the third party header files**

The header files of tensorflow may include some header files of third party library, e.g. **Eigen**. And thus we must also copy there header files otherwise it will can not find these header files. But there are so many third party libraries that tensorflow depends on and there are maintained by bazel and finding these files can be troublesome. Luckily, tensorflow contrib provides a script to download these dependencies.

```
# Download dependencies
./tensorflow/contrib/makefile/download_dependencies.sh
# Copy to include
 sudo cp -r tensorflow/contrib/makefile/downloads /usr/local/tensorflow/include/tensor
```

- **Add tensorflow library directory to library path**

```
sudo echo /usr/local/tensorflow/lib > /ect/ld.so.conf.d/tensorflow.conf
sudo ldconfig
```

For those non-sudo user, add the path into LD_LIBRARY_PATH can be ok.

```
export LD_LIBRARY_PATH=/usr/local/tensorflow/lib:$LD_LIBRARY_PATH
```

Compile your tensorflow code

For exmaple, there are a c++ source code using tensorflow called example.cc .

```
g++ -std=c++11 \
  -I /usr/local/tensorflow/include \
  -I /usr/local/tensorflow/include/third_party \
  -I /usr/local/tensorflow/include/tensorflow/contrib/makefile/downloads/eigen \
  -I /usr/local/tensorflow/include/tensorflow/contrib/makefile/downloads/nsync \
  -I /usr/local/tensorflow/include/tensorflow/contrib/makefile/downloads/nsync/public \
  -L /usr/local/tensorflow/lib \
  -ltensorflow_cc \
  -ltensorflow_framework \
  exmaple.cc
```

You can also use make, cmake or bazel to build your code.

tensorflow

‹ Run GUI app in docker on a remote server

Creating a trusted CA and self signed SAN certificate by Openssl ›