TensorFlow 0.10.0 Installation Best Practices

HPC ADVISORY COUNCIL
NETWORK OF EXPERTISE
BEST PRACTICES

1. Introduction:

The following best practices document is provided as courtesy of the HPC Advisory Council.

2. Application Description

TensorFlow is an open source software library for machine learning across a range of tasks. It is developed by Google to meet their needs for systems capable of building and training neural networks, which is used to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. It is used currently for both research and production at Google.

3. Version Information:

Download TensorFlow version 0.10.0:
https://www.tensorflow.org/

4. Prerequisites:

The instructions from this best practice have been tested on the following configuration:
Hardware:

- Colfax CX2660s-X6 2U 4-node "Odin" cluster
- Dual-Socket 16-Core Intel E5-2697v4 @ 2.60 GHz CPUs
- Mellanox ConnectX-4® EDR InfiniBand and 100Gb/s Ethernet VPI adapters
- Mellanox Switch-IB SB7700 36-Port 100Gb/s EDR InfiniBand switches
- GPU: NVIDIA Kepler K80 GPUs
- Memory: 64GB DDR4 2133MHz RDIMMs per node
- NVIDIA Kepler K80 and Pascal P100 GPUs

OS and software:

- OS: Ubuntu 16.04
- InfiniBand driver: MLNX_OFED_LINUX-3.4-1.0.0.0 InfiniBand SW stack
- MPI: Mellanox HPC-X v1.7.0-406
- Compilers: GNU compilers 4.8.4
- CUDA Library: 7.5, CUDNN version 5.1.5
- Application:
  - TensorFlow 0.10.0
- Benchmarks:
  - ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)

5. Installation

5.1 Installation of dependencies

If you are starting from a vanilla Ubuntu OS, you need to install these packages in order to run TensorFlow. First you need to prepare for the packages:

```
apt remove -y ufw resolveconf
apt -y install unzip lrzsz
```

Install packages required for MLNX_OFED installation

```
apt-get install -y swig libgfortran3 flex graphviz make dkms gfortran
gcc debhelper autotools-dev m4 tcl autoconf chrpath python-libxml2 pkg-
config tk quilt libltdl-dev dpatch bison libnl-route-3-200 automake
```

Install the 32 bit libraries

```
apt install -y gcc-multilib g++-multilib
```

Install packages required for TensorFlow python support

```
apt-get install python-numpy swig python-dev python-wheel python-pip
pip install --upgrade pip
```

Put Path and JAVA_HOME in ~/.bashrc

```
# vim ~/.bashrc
export PATH+=:/usr/local/cuda-7.5/bin
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

# cat /etc/ld.so.conf
include /etc/ld.so.conf.d/*.conf
/usr/local/cuda-7.5/lib64
/usr/lib
/usr/lib64
/usr/local/lib
/usr/local/lib64
```

5.1 Installation of NVIDIA drivers and cuDNN libraries

Download cuda_7.5.18_linux.run from https://developer.nvidia.com/cuda-downloads
Download cudnn-7.5-linux-x64-v5.1.tgz from https://developer.nvidia.com/cudnn
Download NVIDIA-Linux-x86_64-352.99.run from the NVIDIA web site.

Install NVIDIA drivers:

```
# ./NVIDIA-Linux-x86_64-352.99.run
```

Install CUDA libraries:

```
# ./cuda_7.5.18_linux.run –override
```

Install cuDNN libraries:

```
# tar xvfz cudnn-7.5-linux-x64-v5.1.tgz -C /usr/local/
```

Run "`nvidia-smi`" to make sure the driver can detect the NVIDIA GPUs

5.2 Install Bazel

1. Install bazel following location below. Bazel version 0.2.2 is what is used here.
   a. https://www.bazel.io/versions/master/docs/install.html
2. Install oracle JDK 1.8

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

3. Add the apt source repo for Bazel:

```
echo "deb [arch=amd64] 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 -
```

4. Install Bazel

```
sudo apt-get update
sudo apt-get install bazel
sudo apt-get upgrade bazel
```

Oracle JDK 1.8 is needed for building TensorFlow project, not for installing bazel

5.3 Install TensorFlow from source

Sometimes Tensorflow git master might not as stable. In this example, we are using TensorFlow v0.10.0. You can download source from:

https://github.com/tensorflow/tensorflow/releases

Make a copy of CROSSTOOL from the template (tensorflow/third_party/gpus/crosstool/CROSSTOOL.tpl)

```
$ cp /third_party/gpus/crosstool/CROSSTOOL.tpl
/third_party/gpus/crosstool/CROSSTOOL
```

Then add these 3 lines under the line cxx_flag: "-std=c++11" due to these issues:

```
cxx_flag: "-D_FORCE_INLINES"
cxx_flag: "-D_MWAITXINTRIN_H_INCLUDED"
cxx_flag: "-D__STRICT_ANSI__"
```

https://github.com/tensorflow/tensorflow/issues/1066
https://github.com/tensorflow/tensorflow/issues/2143
https://github.com/tensorflow/tensorflow/issues/1157

Add a line for "cxx_builtin_include_directory" for CUDA:

```
55 # and the device compiler to use "-std=c++11".
56 cxx_flag: "-std=c++11"
57 cxx_flag: "-D_FORCE_INLINES"
58 cxx_flag: "-D_MWAITXINTRIN_H_INCLUDED"
59 cxx_flag: "-D__STRICT_ANSI__"
60 cxx_builtin_include_directory: "/usr/local/cuda-7.5/include"
```

Under the other 3 cxx_builtin_include_directory lines, for cuda and cudnn headers

```
68   cxx_builtin_include_directory: "/usr/lib/gcc/"
69   cxx_builtin_include_directory: "/usr/local/include"
70   cxx_builtin_include_directory: "/usr/include"
71   cxx_builtin_include_directory: "/usr/local/cuda-7.5/include"
```

Comment out the #error directive in line 115 of file:

```
/usr/local/cuda-7.5/include/host_config.h
```

There maybe a warning for “unsupported compiler” warning, you can mute that warning by commenting out this line below:

```
113 #if __GNUC__ > 4 || (__GNUC__ == 4 && __GNUC_MINOR__ > 9)
114
115 //#error -- unsupported GNU version! gcc versions later than 4.9
are not supported!
```

Run ‘config’ for the installation:

```
$ ./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] <enter>
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 /usr/bin/gcc]: <enter>
Please specify the Cuda SDK version you want to use, e.g. 7.0. [Leave
empty to use system default]: <enter>
Please specify the location where CUDA  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]: <enter>
Please specify the location where cuDNN  library is installed. Refer to
README.md for more details. [Default is /usr/local/cuda]: <enter>
libcudnn.so resolves to libcudnn.5
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"]: 3.7
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
```

5.3 Install Bezel

Build the target:

```
bazel build -c opt --config=cuda
//tensorflow/cc:tutorials_example_trainer --verbose_failures
```

Run the example:

```
bazel-bin/tensorflow/cc/tutorials_example_trainer --use_gpu
```

Build the pip package with CUDA:

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

Create the pip package

```
bazel-bin/tensorflow/tools/pip_package/build_pip_package
/tmp/tensorflow_pkg
~/tf/tensorflow-0.10.0rc0
```

Install the pip package

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

6.0 Verifying TensorFlow Installation

Verify TensorFlow installation by importing TensorFlow in a python shell

```
$ cd
$ python
Python 2.7.12 (default, Jul  1 2016, 15:12:24)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcudnn.so.5 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcurand.so locally
>>>
```

7.0 Downloading Input Data for TensorFlow

Some background information on training for distributed TensorFlow:

https://github.com/tensorflow/models/tree/master/inception#how-to-train-from-scratch-in-a-distributed-setting

https://www.tensorflow.org/versions/r0.10/how_tos/distributed/index.html#distributed-tensorflow

The ImageNet data can be downloaded at the “Download Original Images”:

http://image-net.org/download

For the ImageNet ILSVRC2012 training and validation data, they are about 142GB in size:

```
# du -sh *
138G       ILSVRC2012_img_train.tar
6.3G        ILSVRC2012_img_val.tar
```

Data for the the preprocessed images is about 292GB

```
# du -sh imagenet-data/
292G       imagenet-data/
```

8.0 Training the ImageNet ILSVRC 2012 for TensorFlow

Each node has a process serves as a parameter server. We have 1 NVIDIA K80 GPU for each node, therefore we spawn 2 processes for the workers. The script uses 2 nodes to run. We use the "CUDA_VISIBLE_DEVICES" to control which GPUs are used by the processes. Parameter process does not bind to a GPU. The "max_step" parameter limits the number of iteration to run to only 500 iterations. You may want to pay attention to the highlighted letters in the script.

On the first node, start this script:

```
$ cat TRAIN_distributed_odin001.sh

#!/bin/bash
rm /tmp/imagenet_train -fr
export LD_LIBRARY_PATH+=:/usr/local/cuda/lib64
export CUDA_HOME=/usr/local/cuda
export PSHOST="odin001:2001,odin002:2002"
export WKHOST="odin001:3001,odin001:3011,odin002:3002,odin002:3012"

export DISTTRAIN="/home/tensorflow/models/inception/bazel-
bin/inception/imagenet_distributed_train"
export DATA="/home/tensorflow/imagenet-data"
export LOGDIR="/home/tensorflow"
export BATCHSIZE=64
export LD_LIBRARY_PATH+=:/usr/local/cuda/lib64
export CUDA_HOME=/usr/local/cuda

export PSO1=$LOGDIR/log.tf.pso1.${HOSTNAME}.txt
export PSE1=$LOGDIR/log.tf.pse1.${HOSTNAME}.txt
export WKO1=$LOGDIR/log.tf.wko11.${HOSTNAME}.txt
export WKE1=$LOGDIR/log.tf.wke11.${HOSTNAME}.txt
export WKO2=$LOGDIR/log.tf.wko12.${HOSTNAME}.txt
export WKE2=$LOGDIR/log.tf.wke12.${HOSTNAME}.txt

CUDA_VISIBLE_DEVICES='' $DISTTRAIN \
--num_gpus=0 \
--job_name='ps' \
--task_id=0 \
--ps_hosts=$PSHOST \
--worker_hosts=$WKHOST 1>$PSO1 2>$PSE1 &

CUDA_VISIBLE_DEVICES='0' nohup $DISTTRAIN \
--max_steps=500 \
```

```
--num_gpus=1 \
--batch_size=$BATCHSIZE \
--data_dir=$DATA \
--job_name='worker' \
--task_id=2 \
--ps_hosts=$PSHOST \
--worker_hosts=$WKHOST 1>$WKO1 2>$WKE1 &

CUDA_VISIBLE_DEVICES='1' nohup $DISTTRAIN \
--max_steps=500 \
--num_gpus=1 \
--batch_size=$BATCHSIZE \
--data_dir=$DATA \
--job_name='worker' \
--task_id=3 \
--ps_hosts=$PSHOST \
--worker_hosts=$WKHOST 1>$WKO2 2>$WKE2 &
```

On the second node, start this script:

```
$ cat TRAIN_distributed_odin002.sh

#!/bin/bash
rm /tmp/imagenet_train -fr
export LD_LIBRARY_PATH+=:/usr/local/cuda/lib64
export CUDA_HOME=/usr/local/cuda
export PSHOST="odin001:2001,odin002:2002"
export WKHOST="odin001:3001,odin001:3011,odin002:3002,odin002:3012"

export DISTTRAIN="/home/tensorflow/models/inception/bazel-
bin/inception/imagenet_distributed_train"
export DATA="/home/tensorflow/imagenet-data"
export LOGDIR="/home/tensorflow"
export BATCHSIZE=64
export LD_LIBRARY_PATH+=:/usr/local/cuda/lib64
export CUDA_HOME=/usr/local/cuda

export PSO1=$LOGDIR/log.tf.pso1.${HOSTNAME}.txt
export PSE1=$LOGDIR/log.tf.pse1.${HOSTNAME}.txt
export WKO1=$LOGDIR/log.tf.wko11.${HOSTNAME}.txt
export WKE1=$LOGDIR/log.tf.wke11.${HOSTNAME}.txt
export WKO2=$LOGDIR/log.tf.wko12.${HOSTNAME}.txt
export WKE2=$LOGDIR/log.tf.wke12.${HOSTNAME}.txt

CUDA_VISIBLE_DEVICES='' $DISTTRAIN \
--num_gpus=0 \
--job_name='ps' \
--task_id=1 \
--ps_hosts=$PSHOST \
--worker_hosts=$WKHOST 1>$PSO1 2>$PSE1 &

CUDA_VISIBLE_DEVICES='0' nohup $DISTTRAIN \
--max_steps=500 \
--num_gpus=1 \
--batch_size=$BATCHSIZE \
--data_dir=$DATA \
```

```
--job_name='worker' \
--task_id=2 \
--ps_hosts=$PSHOST \
--worker_hosts=$WKHOST 1>$WKO1 2>$WKE1 &

CUDA_VISIBLE_DEVICES='1' nohup $DISTTRAIN \
--max_steps=500 \
--num_gpus=1 \
--batch_size=$BATCHSIZE \
--data_dir=$DATA \
--job_name='worker' \
--task_id=3 \
--ps_hosts=$PSHOST \
--worker_hosts=$WKHOST 1>$WKO2 2>$WKE2 &
```

If the run is successful, you will see these messages during training. The metric for training is shown as number of examples/sec. Below is the output for training on 4 nodes, with 1 K80 on each node.

```
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcublas.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcudnn.so.5 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcufft.so locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:108] successfully opened
CUDA library libcurand.so locally
INFO:tensorflow:PS hosts are: ['odin001-100ge:2001', 'odin002-
100ge:2002', 'odin003-100ge:2003', 'odin004-100ge:2004']
INFO:tensorflow:Worker hosts are: ['odin001-100ge:3001', 'odin001-
100ge:3011', 'odin002-100ge:3002', 'odin002-100ge:3012', 'odin003-
100ge:3003', 'odin003-100ge:3013', 'odin004-100ge:3004', 'odin004-
100ge:3014']
I tensorflow/core/common_runtime/gpu/gpu_init.cc:102] Found device 0
with properties:
name: Tesla K80
major: 3 minor: 7 memoryClockRate (GHz) 0.8235
pciBusID 0000:07:00.0
Total memory: 11.92GiB
Free memory: 11.86GiB
I tensorflow/core/common_runtime/gpu/gpu_init.cc:126] DMA: 0
I tensorflow/core/common_runtime/gpu/gpu_init.cc:136] 0:   Y
I tensorflow/core/common_runtime/gpu/gpu_device.cc:838] Creating
TensorFlow device (/gpu:0) -> (device: 0, name: Tesla K80, pci bus id:
0000:07:00.0)
I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:206]
Initialize HostPortsGrpcChannelCache for job ps -> {odin001-100ge:2001,
odin002-100ge:2002, odin003-100ge:2003, odin004-100ge:2004}
I tensorflow/core/distributed_runtime/rpc/grpc_channel.cc:206]
Initialize HostPortsGrpcChannelCache for job worker -> {odin001-
100ge:3001, odin001-100ge:3011, odin002-100ge:3002, odin002-100ge:3012,
localhost:3003, odin003-100ge:3013, odin004-100ge:3004, odin004-
100ge:3014}
I tensorflow/core/distributed_runtime/rpc/grpc_server_lib.cc:202]
Started server with target: grpc://localhost:3003
```

```
INFO:tensorflow:SyncReplicas enabled: replicas_to_aggregate=8;
total_num_replicas=8
INFO:tensorflow:2016-12-30 11:02:02.018130 Supervisor
INFO:tensorflow:Started 3 queues for processing input data.
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:244]
PoolAllocator: After 3546 get requests, put_count=2540
evicted_count=1000 eviction_rate=0.393701 and unsatisfied allocation
rate=0.593909
I tensorflow/core/common_runtime/gpu/pool_allocator.cc:256] Raising
pool_size_limit_ from 100 to 110
INFO:tensorflow:Worker 4: 2016-12-30 11:02:48.085282: step 0, loss =
13.06(1.9 examples/sec; 33.802  sec/batch)
INFO:tensorflow:Worker 4: 2016-12-30 11:02:51.032279: step 0, loss =
13.08(21.7 examples/sec; 2.946  sec/batch)
INFO:tensorflow:Worker 4: 2016-12-30 11:02:53.700298: step 0, loss =
13.08(24.0 examples/sec; 2.668  sec/batch)
…
```