TensorFlow BKMs for State-Of-The-Art Accuracy & Convergence On Multi-Node Xeon® Processor Clusters

Example: ResNet-50

Valeriu Codreanu & Damian Podareanu, SURFsara, B.V.

Vikram Saletore & Aishwarya Bhandare, Anupama Kurpad, AIPG & DCG, Intel Corp.

April 2018

Overview

Deep learning training convergence to state-of-the-art (SOTA) accuracies is the center piece of any deep learning model. At the same time, achieving the SOTA accuracy with convergence in the best Time-To-Train (TTT) is the metric for performance and accuracy.

In this document, based on joint SURFsara-Intel collaboration, we present the steps needed to achieve SOTA accuracy and convergence for **ResNet-50 on up to 256 nodes of Xeon® Skylake** processor based HPC Cluster. We have been able to achieve **~75% Top-1 accuracy on 256 node Xeon® 8160 Processor cluster** connected over Intel® Omni-Path Architecture (OPA™) 100 Gbit/sec fabric at TACC (Texas Advanced Computing Center, https://www.tacc.utexas.edu/) **in less than 2 hours** using Intel's multi-workers/node best known methodology (BKM) (https://software.intel.com/en-us/articles/boosting-deep-learning-training-inference-performance-on-xeon-and-xeon-phi).

This is **first time that ResNet-50** with TensorFlow 1.6 has been scaled with **81% efficiency and global throughput of 16400 Images/sec** with convergence to SOTA accuracy to **256 Xeon® Platinum 8160 processor** (Skylake) nodes. Also, to the best of our knowledge, to date this is the best convergence and best TTT that we have measured with TensorFlow for distributed training for ResNet-50 model on large scale-out Xeon® cluster.

Following is the set of steps for convergence and performance is recommended:

1. Clone TensorFlow 1.6 from: https://github.com/tensorflow/tensorflow

2. Build Tensorflow with using instructions from: https://www.tensorflow.org/performance/performance_guide

3. Install Horovod: A distributed training framework for TensorFlow based on a data-parallel distributed training paradigm. Horovod is an extension of an MPI-based distributed training framework. Horovod can be installed as a standalone python package as follows from:
    a. https://eng.uber.com/horovod/
    b. `pip install --no-cache-dir --user Horovod`

4. ImageNet2012-1K
    a. Get Training and Validation Raw image dataset from:
        i. http://www.image-net.org/challenges/LSVRC/2012/
        ii. Training raw images: ~148 GB
        iii. Validation raw images: ~6.7 GB

5. An extremely important aspect of convergence with performance is how the dataset is prepared. Dataset needs to be prepared across the specified number of TensorFlow workers across the cluster:
    a. Please Refer to the TPU repository at that we have used to prepare the dataset: https://github.com/tensorflow/tpu/blob/master/tools/datasets/imagenet_to_gcs.py
    b. With TensorFlow tf_cnn_benchmarks (https://github.com/tensorflow/benchmarks/tree/master/scripts/tf_cnn_benchmarks) We were able to achieve 73.5% Top-1 accuracy for ResNet-50 on 2S 256 Xeon® Skylake Platinum nodes with a global batch size of 8K.
    c. However, with TPU repository resnet model we achieved 75%+ Top-1 for global batch size of 8K/24K
    d. Some modifications and optimizations are required for CPUs to use the above TPU repository.
    e. We have developed scripts that enable the shuffling of the images and building the necessary TF Records from raw images.
    f. With this modified tensorflow/tpu repository we have managed to achieve 75+% Top-1 accuracy for ResNet-50 with a global batch size of 24K on 256 Intel® Platnum 8160 nodes using Intel's multi-worker best known methodology with 2 workers/node (BS/worker=48) on Stampede2/TACC in less than 2 hours.
    g. In addition, we have also added Horovod and Intel optimizations on top of the tensorflow/tpu repository.
    h. We have created a set of scripts in (***tpu_dell.tar.gz***) with necessary modifications to the TPU scripts to process and prepare the training dataset.
    i. In order to create the dataset, use the `imagenet_to_gcs.py` script from the ***tpu_dell.tar.gz*** tarball as follows:

```
python imagenet_to_gcs.py --raw_data_dir <path-to-raw-
image-dir> --local_scratch_dir <path-to-tf-records-dir>
```
where
```
path-to-raw-data-dir:      Input directory of raw images
path-to-tf-records-dir:    Output TF-Records directory
```

6. ResNet-50 Benchmark: We have modified the official model from TensorFlow:
    a. https://github.com/tensorflow/models/tree/master/official/resnet

7. We provide a modified set of scripts for ResNet-50 in the attached ***tpu_dell.tar.gz*** using TPU official model.
    a. Go to the models/official/resnet directory after you untar the ***tpu_dell.tar.gz:***
    b. Modify Line 290 `in models/official/resnet/resnet_main.py` to suit your cluster environment:

Example: Convergence run on 2Skt 24C/Skt, 256 SKX nodes on 100Gbit OmniPath Fabric:

i. `learning_rate = gradual_warmup_then_dec(0.1, 260 , 4.8, global_step, FLAGS.train_steps, name="gradual_warmup_then_dec")`
ii. The above options include:
    1. StartingLR (Learning Rate): 0.1 (for BS=256)
    2. 5 Epoch warm-up iterations:
        a. BS/Worker = 48
        b. Workers/Node = 2
        c. BS/Node = 96
        d. NumNodes = 256
        e. Num Iterations/Epoch = 1280000/(96*256) = 52.08
        f. Warm Up of 5 Epochs Iterations = 52.08 * 5 ~= 260
    3. train_batch_size: 24576
        a. BS/Node * NumNodes = 96 * 256 = 24576
    4. Learning after warmup: 4.8
        a. LR post warmup= StartingLR*train_batch_size/BS(StartingLR)
        b. LR post warmup = 0.1 * 24576/256 = 9.6. We half this to 4.8 as we've found out empirically.
    5. train_steps: 4680
        a. Training Epochs: 90
        b. Training iterations = 52.08 * 90 = 4680

2. Details of modifications to `tensorflow/tpu` repository: Three (3) files modified in order to improve validation accuracy. The modifications are as follows:

a. tools/datasets/imagenet_to_gcs.py

i. The only modifications are deleting the GCS-related things, namely the error raising for lack of providing FLAGS.project and FLAGS.gcs_output_path. Also, `upload_to_gcs` is left out.
ii. Using this script is very important, as compared to the Inception-provided. It also does shuffling of categories across TF-Records.
iii. **Failing to shuffle the categories may decrease accuracy performance of ResNet-50 by 2-3%**.

b. models/official/resnet/resnet_model.py

i. The only modification is changing the `BATCH_NORM_DECAY` (moving_average_fraction in Caffe terminology) `to 0.95`

c. models/official/resnet/resnet_main.py:

i. L36: `import horovod.tensorflow as hvd (`MKL and horovod support)
ii. L88-105. Added `MKL` and `OpenMP` flags
iii. L116. Set `eval_batch_size` to lower value
iv. L119. Set `steps_per_eval` to high value. Evaluate only at the end.
v. L158. Set `WEIGHT_DECAY to 5e-5`. In practice we see better convergence with this value
vi. L191: Added function:

1. `gradual_warmup_then_dec(learning_rate, warmup_steps, end_learning_rate, global_step, steps, name=None)`
2. This function increases the learning rate linearly from *learning_rate* to *end_learning_rate* for *warmup_steps* iterations.
3. Afterwards, it decreases the learning rate with a linear rate (power-1 polynomial) from *end_learning_rate* to 0 for *steps-warmup_steps* iterations.
4. Different decays can be explored by changing L212

vii. L290. Call to `gradual_warmup_then_dec` function. At the moment the LR schedule is optimized for a global batch size of 24576.
viii. L300. `optimizer = hvd.DistributedOptimizer(optimizer)`. This adds Horovod distributed optimizer.
ix. L344. Save summaries only on HVD rank 0.
x. L420. Initialize Horovod (hvd.init())
xi. L421-424. Set `KMP/OMP` environment variables
xii. L429-430 and L437. Save checkpoints and summaries/logs only on rank 0
xiii. L438. Very important. Shard across all hvd ranks.
xiv. L444. Split global batch among number of workers (hvd.size())
xv. L447. Horovod broadcast original model parameters from rank 0 to all other ranks
xvi. L471. Add broadcast hook to the train loop.

3. **Intel's BKMs for Multi-Worker/Node & Multi-Node Training**:
   a. Please refer to: https://software.intel.com/en-us/articles/boosting-deep-learning-training-inference-performance-on-xeon-and-xeon-phi

4. **Measuring TTT (Time-To-Train) Convergence Performance**:
   a. To run convergence tests with 512 workers, 2 workers/node on 256 Nodes 2S Xeon® SKX with 24C/Socket, global batch size of 24576 (i.e. BS of 48/worker or 96/node), inter_op=2, intra_op=22 for each worker, on with use the following ***mpirun*** command:

```
mpirun -np 512 -ppn 2 python <path-to>/resnet_main.py \
--train_batch_size 24576 \
--train_steps 4680 \
--num_intra_threads 22 --num_inter_threads 2 \
--data_dir=<path-to-local-scratch-train-dir> \
--model_dir <path-to-model-dir>/model_batch_24k_90ep \
--use_tpu=False --kmp_blocktime 1
```

5. **Measuring Top-1 & Top-5 Accuracy: Using ImageNet-1K Validation Images**
   a. To run validation tests on the single 2S Xeon® SKX with 24C/Socket using the saved model with an evaluation batch of 200, thus 250 evaluation batches (50,000 eval images in total)

```
OMP_NUM_THREADS=46 python <path-to>/resnet_eval.py \
--eval_batch_size 200 \
--num_intra_threads 46 --num_inter_threads 2 \
--data_dir=<path-to-local-scratch-validation-dir> \
--model_dir=<path-to-model-dir>/model_batch_24k_90ep \
--use_tpu=False --kmp_blocktime 1
```

b. Partial output log from the convergence & accuracy test:

```
I0415 09:17:18.300860 140247695472448 tf_logging.py:116] Evaluation [25/250]
I0415 09:18:29.873797 140247695472448 tf_logging.py:116] Evaluation [50/250]
I0415 09:19:35.788666 140247695472448 tf_logging.py:116] Evaluation [75/250]
I0415 09:20:41.003854 140247695472448 tf_logging.py:116] Evaluation [100/250]
I0415 09:21:45.633120 140247695472448 tf_logging.py:116] Evaluation [125/250]
I0415 09:22:49.952527 140247695472448 tf_logging.py:116] Evaluation [150/250]
I0415 09:23:54.425580 140247695472448 tf_logging.py:116] Evaluation [175/250]
I0415 09:24:59.644886 140247695472448 tf_logging.py:116] Evaluation [200/250]
I0415 09:26:04.271167 140247695472448 tf_logging.py:116] Evaluation [225/250]
I0415 09:27:08.750324 140247695472448 tf_logging.py:116] Evaluation [250/250]
I0415 09:27:08.821882 140247695472448 tf_logging.py:116] Finished evaluation at
2018-04-15-16:27:08
I0415 09:27:08.822196 140247695472448 tf_logging.py:116] Saving dict for global
step 14075: Top-1 accuracy = 0.75232, Top-5 accuracy = 0.9241, global_step =
14075, loss = 1.9024274
I0415 09:27:09.251455 140247695472448 tf_logging.py:116] Eval results: {'Top-1
accuracy': 0.75232, 'loss': 1.9024274, 'Top-5 accuracy': 0.9241, 'global_step':
14075}
```

c. Top-1 Accuracy: 75.23% & Top-5 Accuracy: 92.41%