How to run tf2-cosmoflow on Summit

Sunwoo Lee sunwoolee1.2014@u.northwestern.edu

On Summit, IBM Watson's Machine Learning module should be loaded to use TensorFlow. The module contains TensorFlow, caffe, pyTorch, and other packages for ML applications. However, the module does not support mpi4py by default. So, users should manually install mpi4py to use MPI.

1. Clone the code from github repository.

```
[slz839@login2.summit cosmo]$ git clone https://swblaster@github.com/NU-CUCIS/tf2-cosmoflow
Cloning into 'tf2-cosmoflow'...
Password for 'https://swblaster@github.com':
remote: Enumerating objects: 631, done.
remote: Total 631 (delta 0), reused 0 (delta 0), pack-reused 631
Receiving objects: 100% (631/631), 118.76 KiB | 0 bytes/s, done.
Resolving deltas: 100% (443/443), done.
```

2. Load the IBM Watson module.

```
[slz839@login2.summit tf2-cosmoflow]$ module load ibm-wml-ce/1.7.0-3
(ibm-wml-ce-1.7.0-3) [slz839@login2.summit tf2-cosmoflow]$
```

3. Install mpi4py in the user space.

```
(ibm-wml-ce-1.7.0-3) login4:slz839 vgv$ pip install --user mpi4py
Collecting mpi4py
Downloading mpi4py-3.0.3.tar.gz (1.4 MB)
|################################################| 1.4 MB 1.8 MB/s
Building wheels for collected packages: mpi4py
…
Successfully installed mpi4py-3.0.3
```

4. Modify main.py (comment out line 55 and switch to line 54). Summit allows each process views the assigned GPUs only. So, each process should always access gpus[0] only.

```
51    if gpus:
52        # On Summit, each resource set can view its own GPUs only.
53        # So, the visible devices should be set to gpu:0 for every process.
54        tf.config.experimental.set_visible_devices(gpus[0], 'GPU')
55        #tf.config.experimental.set_visible_devices(gpus[hvd.local_rank()], 'GPU')
```

5. Specify path to the input files and a few parameters in ***test_summit.yaml*** file. The below is an example of the ***test_summit.yaml*** file.

```
frameCnt: 128
numPar: 4
sourceDir: {
  prj: /gpfs/alpine/ast153/scratch/slz839/
}
subDir: 1/multiScale_tryG
fnameTmpl: PeterA_2019_05_4parE-rec*.h5
splitIdx:
  val: [100, 101, 102, 103, 104, 105, 106, 107]
  train: [20, 21, 22, 23, 24, 25, 26, 110,
          30, 31, 32, 33, 34, 35, 36, 111,
          40, 41, 42, 43, 44, 45, 46, 112,
          50, 51, 52, 53, 54, 55, 56, 113,
          60, 61, 62, 63, 64, 65, 66, 114,
          70, 71, 72, 73, 74, 75, 76, 115,
          80, 81, 82, 83, 84, 85, 86, 116,
```

6. Submit a job using ***myjob.lsf***. The below is an example script.

```
#!/bin/bash
#BSUB -P AST153
#BSUB -W 00:10
#BSUB -nnodes 11
#BSUB -J sunwoo
#BSUB -o sunwoo.%J
#BSUB -e sunwoo.%J

jsrun -n64 -a1 -c4 -g1 python3 main.py --epochs=3 \
                       --batch_size=4 \
                       --overlap=1 \
                       --checkpoint=0 \
                       --cache_size=0 \
                       --file_shuffle=1 \
                       --buffer_size=8 \
                       --record_acc=0 \
                       --config="test_summit.yaml" \
                       --evaluate=0 \
                       --async_io=1
```

7. The output log looks like the below.

```
Epoch: 0 lr: 0.002
Epoch 1 waiting time = 0 training loss = 1.8676238 training timing: 30.020379295921884 sec
Epoch: 1 lr: 0.002
Epoch 2 waiting time = 0 training loss = 0.31578225 training timing: 9.695642819046043 sec
Epoch: 2 lr: 0.002
Epoch 3 waiting time = 0 training loss = 0.2627585 training timing: 9.665014030993916 sec
All done!
```