How to run tensorflow on the test cluster

This page provides a description for the procedure to install tensorflow on the test cluster. After log in to phoenix it is possible to access to the following nodes:

- test-u19-n01.test.cluster AMD Epyc
- test-u21-n01.test.cluster Skylake, 4x V100
- test-u23-n01.test.cluster AMD Ryzen
- test-u25-n01.test.cluster Skylake, 2x V100
- test-u25-n02.test.cluster CascadeLake
- test-u36-n01.test.cluster Broadwell, 2x P100

Installation procedure

This procedure will describe the installation for tensorflow 2.0

- Create the virtual environment

```
$virtualenv-3 --system-site-packages -p python3 ~/tensorflow-2.0-gpu-venv
```

- Download cudnn-7, and unzip it. At the end of ~/tensorflow-2.0-gpu-venv/bin/activate add the following:

```
export LD_LIBRARY_PATH=/usr/local/cuda-10.0/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=cudnn_path/cuda/lib64:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=/usr/local/cuda-10.0/extras/CUPTI/lib64:$LD_LIBRARY_PATH
```

- Activate the virtualenv, and install tensorflow

```
$source ~/tensorflow-2.0-gpu/bin/activate
$pip install tensorflow-gpu==2.0
```

Tensorflow versions

In order to use the pip version you need to check the tensorflow version, the cuda library and the cudnn library. More information can be found at https://www.tensorflow.org/install/source#linux

On the test cluster is available cuda-10.0, cudnn can be download and unpack in the user directory.

Tensorboard

TensorBoard is a tool for providing the measurements and visualizations needed during the machine learning workflow. It enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embeddings to a lower dimensional space, and much more.

If you are interested in this tool start from here https://www.tensorflow.org/tensorboard/get_started

All the examples in the guide assume that the user run tensorflow inside jupyter. However, in a production scenario, the user will run either interactively or by using a job script. In such case tensorboard can be invoked like

```
$source ~/tensorflow-2.0-gpu/bin/activate
$tensorboard --bind_all --logdir logs
Tensorboard-2.0.2 at http://test-u36-n01.test.cluster:6006
```

You can now visualize the results in your browser by typing the address http://test-u36-n01.test.cluster:6006. Sometimes it might be required the insert the ip manuallay. For test-u360-n01.test.cluster is 10.91.1.17.
Using shifter

You can also run tensorflow through shifter. For an introduction to shifter please look at [https://scitas-data.epfl.ch/confluence/display/DOC/Running+Docker+images+using+Shifter](https://scitas-data.epfl.ch/confluence/display/DOC/Running+Docker+images+using+Shifter)

You can upload your own image if necessary. However, if you want to experiment with an already available image you can follow the template provided below.

**submission script**

```bash
#!/bin/bash
#SBATCH -p debug
#SBATCH --nodelist=test-u36-n01.test.cluster
export CUDA_VISIBLE_DEVICES=0,1
srun shifter --image tensorflow-gpu/tensorflow/tensorflow:2.0.0-gpu python test_tf.py
```

The example above loads the image tensorflow-gpu/tensorflow/tensorflow:2.0.0-gpu which provides tensorflow 2.0.

**test_tf.py**

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential(
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
)

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

**Related articles**

- Licensed Software on the Clusters
- Data Transfer Nodes
- Compiling codes on different systems
- Connecting to the clusters
- How to compile swak4Foam