Guide to using GPUs in HTCondor

Honey Gupta

This doc contains instructions on how to run a python script possibly containing a training routine for a deep neural network on the batch service: HTCondor. We will consider a sample script from here and the following instructions will show how to queue the code for training on HTCondor.

Reference guide: https://batchdocs.web.cern.ch/tutorial/exercise10.html
For queries or more details, contact: Honey Gupta (hn.gpt1@gmail.com)

Before starting

Check the quick start guide and follow the instructions to ensure you have appropriate licenses in place. The guide will also give you a brief introduction on how to queue codes.

Training a network on HTCondor

Few pointers:

- We use the AFS service to store the data, scripts as well as the virtual environment containing the required packages for running the training code.

Where to start?

Start with preparing the **python training script**. A sample python script for training can be found here.

You'll have to include scripts to save the training logs within the script in order to be able to debug/diagnose later. You can include plot and save commands to save the training/validation logs, save intermediate outputs and checkpoints.

You can either hardcode the arguments such as data path and hyperparameters or mention them as input arguments in the bash file. The python script that we have considered has hardcoded parameters and paths. *Note that when mentioning paths to AFS locations, mention the full-path as the working directory location changes when you login to the node.*

Installing the required packages

Once you have the python script ready, you need to make a list of the additional packages required for running the python script.

Pro-tip: Determining the required additional packages could be difficult as you might need to submit the job multiple times just to see if it is sufficient and that could take many trials and a lot of your time. To make debugging easier, you could use the ***interactive mode*** *once you have a draft of the scripts ready. More information on the interactive mode can be found here: https://batchdocs.web.cern.ch/tutorial/exercise10.html#interactive-jobs*

There are two ways in which you can load the required packages to run your code on the assigned node

1. You can include the installation instructions in the bash file and mention all the installation details before issuing the run command to start the training.
2. Or you can preinstall the packages in a virtual environment at a location in AFS and then activate the environment through the bash service.

I prefer to use the 2nd option as it is

- easier,
- saves time as it avoids the installation of packages every time you submit the code, and
- easy to debug and fix

Let's name the bash file to be submitted to the batch services as ***train.sh***

1. If you choose the first option, your *train.sh* script could be something like this:

```
#!/bin/bash
python -m virtualenv -p python3 myvenv
source myvenv/bin/activate
pip install pandas
pip install fastai
pip install scikit-learn
cd /afs/cern.ch/work/h/h*/public/AE-Compression-pytorch/examples/27D/
python 27D_train.py
```

2. If you choose the second option, you can create a virtual environment at an accessible (public) AFS location and install the packages. This can simply be done by running the first part of the above *train.sh* script in the desired installation location.
   For *e.g.* let's say my installation location is
   /afs/cern.ch/work/h/h*/public/AE-Compression-pytorch/

   Then I can create the virtual-env and packages by running an *install_libs.sh* bash file that contains the following:

```
cd /afs/cern.ch/work/h/h*/public/AE-Compression-pytorch/
python -m virtualenv -p python3 myvenv
source myvenv/bin/activate
pip install pandas
pip install fastai
pip install scikit-learn
```

And then run:

```
$ chmod +x install_libs.sh
$ ./install_libs.sh
```

This creates a virtual-env at the required location, with all the required packages and now you can just cd to the location from the HTCondor nodes.

Now your *train.sh* file to be submitted to the HTCondor node can be something like the following:

```
#!/bin/bash
cd /afs/cern.ch/work/h/h*/public/AE-Compression-pytorch/
source myvenv/bin/activate
cd examples/27D/
python 27D_train.py
```

Pointers:

- `source myvenv/bin/activate` activates you virtual-env with the name “myvenv”
- `cd examples/27D/` is to go to the AFS location that contains your python script
- The link line of the above code runs the python script. You can add your input argument here if any.

Job submit file

Let’s name the submit file to be submitted to the batch services as ***train.sub***

Here is a sample submit file for training the network.

```
executable              = train.sh
arguments               = $(ClusterId)$(ProcId)
output                  = train.$(ClusterId).$(ProcId).out
```

```
error                   = train.$(ClusterId).$(ProcId).err
log                     = train.$(ClusterId).log
request_GPUs = 1
request_CPUs = 4
+JobFlavour = "tomorrow"
queue
```

Pointers:

- The first 5 lines of the above code are same as mentioned in the quick-start guide.
- `request_GPUs = 1` says that the node to be assigned for running this job file should have a GPU
- `request_CPUs = 4` says that 4 CPUs are required. The number of CPUs is also related to the memory (RAM) and disk allotment, the details of which can be found here: https://batchdocs.web.cern.ch/local/submit.html#resources-and-limits
- `+JobFlavour = "tomorrow"` mentions a suggested maximum runtime of the job. Jobs which exceed the maximum runtime will be terminated. The runtime is the wall time of the job (the elapsed actual time) rather than a calculated cpu time. Here, “tomorrow” is equal to a maximum runtime of 1 day. More details regarding the job flavors and max-time can be found here: https://batchdocs.web.cern.ch/local/submit.html#job-flavours

Finally, you can submit the job by running the following command from your lxplus terminal window:

```
$ condor_submit train.sub
```
