

Deep Learning - MAI

Introduction to cluster usage

Dario Garcia Gasulla
dario.garcia@bsc.es

The basics

- ❖ One account per student
 - Access and credentials are private. *Illegal* to share.
 - You are responsible of your own account
 - All data will be lost after the semester
- ❖ BSC clusters downtimes will be notified through Raco
 - Deadlines will be adapted if appropriate

The clusters

- ❖ CTE-Power9: GPU compute clusters (“plogin1.bsc.es”)
 - Run jobs
- ❖ Data Transfer: Cluster for file management (“dt01.bsc.es”)
 - Upload/Download (scp) Copy (dtcp) files, Password change
- ❖ Manuals online (“user guide XXX”)
 - ssh for connecting
 - home dir: /home/nct01/nct01XXX
 - **Change you pass!** “passwd” in dt01

Software

- ❖ Many DL frameworks out there
 - Caffe2 (Berkeley), CNTK (Microsoft), MXNet (Apache), PyTorch (Facebook), TF (Google), PaddlePaddle (Baidu), Keras
- ❖ Use whatever you want. Examples will be provided in Keras
- ❖ P9 software cannot be changed or upgraded
 - Even containers (PowerPC)
 - PyTorch, TF, Keras available (careful, not the latest version)

Managing Software

- ❖ Software is organized in modules. *Load* to use.
 - module list: currently loaded modules
 - module avail: available modules
 - module purge: remove all modules
 - module load X: load module X
- ❖ Order matters!
- ❖ module python/3.7.4_ML
 - TF, PyTorch, Keras, SciKit, Numpy, ...
 - Beware of dependencies

Running jobs

- ❖ Cluster jobs are enqueued and executed in order
 - Resources requested, time length, previous consumption
 - Do not wait until the last week for experimentation
 - Use infrequent times
- ❖ Launcher file should include (see user guide for more detail):
 - queue (see available with “bsc_queues”)
 - “training” (max 48h), “debug” (max 1h)

#SBATCH --qos=debug

Launcher parameters

- ❖ Execution time (hard limit!)

#SBATCH --time=HH:MM:SS

- ❖ Initial execution path

#SBATCH -D pathname

- ❖ Error & Log file (%j means jobId)

#SBATCH --error=file_name_%j.err

#SBATCH --output=file_name_%j.out

- ❖ Resources (40 CPUs per GPU!)

#SBATCH --cpus-per-task=40

#SBATCH --gres gpu:1

Launcher sample

```
#!/bin/bash
#SBATCH --job-name="test_job"
#SBATCH --qos=debug
#SBATCH -D .
#SBATCH --output=test_job_%j.out
#SBATCH --error=test_job_%j.err
#SBATCH --cpus-per-task=40
#SBATCH --gres gpu:1
#SBATCH --time=00:02:00
```

```
module purge; module load gcc/8.3.0 ffmpeg/4.2.1 cuda/10.2 cudnn/7.6.4 nccl/2.4.8 tensorrt/6.0.1
openmpi/4.0.1 atlas/3.10.3 scalapack/2.0.2 fftw/3.3.8 szip/2.1.1 opencv/4.1.1 python/3.7.4_ML
```

```
python some_code.py
```


Managing jobs

- ❖ Launch job

sbatch launcher_file

- ❖ Check status of jobs (the --start flag gives an estimate for entry time)

squeue

- ❖ Kill a job

scancel jobId

- ❖ Interactive jobs (1h limit)

squeue (get jobId)

ssh id_node (from within login node)

Before the first lab...

make sure you can run the following

Testing the environment

1. Download the MNIST dataset:

<https://s3.amazonaws.com/img-datasets/mnist.npz>

Testing the environment

2. Upload it to the cluster:

```
scp mnist.npz nct01XXX@dt01.bsc.es:/home/nct01/nct01XXX/.keras/datasets/
```

you will need to create the .keras and datasets directories first!

3. Write or upload the code:

https://raw.githubusercontent.com/UPC-MAI-DL/UPC-MAI-DL.github.io/master/_codes/1.FNN-CNN/mnist_fnn_example.py

4. Submit job:

```
sbatch launcher.sh
```

Dario Garcia-Gasulla (BSC)
dario.garcia@bsc.es

