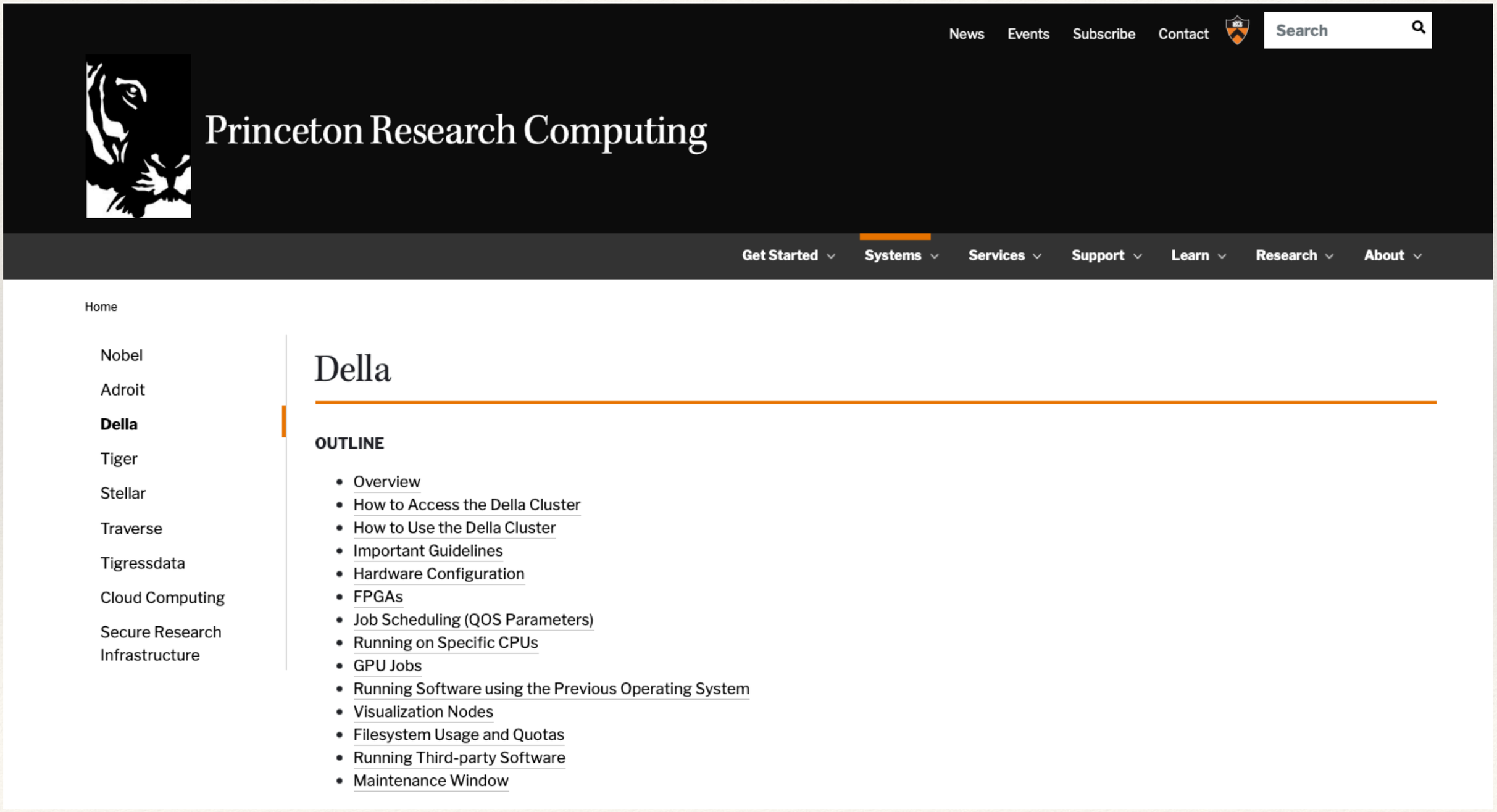
Princeton Computing

https://researchcomputing.princeton.edu/systems/della



Basic commands on Linux

Group resource

https://github.com/YaoGroup/ServerMantaince

https://hackmd.io/dd8wi827SpCLAe8p2Ype6w

Princeton resource

https://github.com/gabeclass/introcmdline

https://github.com/PrincetonUniversity/hpc_beginning_workshop/tree/2021fall

Yao Group Workstation Infrastructure Note

View it on HackMD

This note serves the following purpose:

- As the onboard training for anyone using the workstation for his work. It provides best practices for writing code and using the workstation to run code.
- As the reference note for fundamental technical issues or additional tips for enhancing the personal workflow.
- As the reference note for the future administrator to understand the system setup.

The note is the main page and contains links to additional references. The main page aims at onboard new comber, so they can grasp the essential concepts and start using the workstation ASAP. Other references teach one how to achieve specific things in a step-by-step way.

:::info Before you started, check you already get an account and a password for the workstation. And if you are outside Princeton, make sure you are using VPN.

carolinarr Merge branch '2021fall' of https://github.com/PrincetonUniversity/hp

01_cluster_what	Merge branch 'croeray' of https://githu
02_cluster_why	Minor updates
03_parallel_programming_primer	adding 03 images and alt text notes
04_clusters_getting_access	incorporating feedback
05_connect_byweb	Update README.md
06_connect_byssh	changed snodes to shownodes

Della access

Della has both GPU nodes and CPU nodes

For CPU or GPUs jobs using the Springdale Linux 8 operating system

```
$ ssh <YourNetID>@della.princeton.edu
```

CPU nodes log-in

For GPU jobs (VPN required from off-campus)

```
$ ssh <YourNetID>@della-gpu.princeton.edu
```

GPU nodes log-in

1000 times faster on Della GPU for tensorflow

Additional requirement

(VPN required from off-campus)

Load the Conda environment

```
$ module avail anaconda3
```

```
(base) [yw1705@della-gpu ~]$ module avail anaconda3
------------------------------------------ /usr/licensed/Modules/modulefiles --
-------------------------------
anaconda3/2018.12  anaconda3/2019.10  anaconda3/2020.7   anaconda3/2021.5
anaconda3/2019.3   anaconda3/2020.2   anaconda3/2020.11  anaconda3/2021.11
(base) [yw1705@della-gpu ~]$
```

```
$ module load anaconda3/2021.11
```

```
(base) [yw1705@della-gpu ~]$ module load anaconda3/2021.11
(base) (base) [yw1705@della-gpu ~]$
```

https://researchcomputing.princeton.edu/support/knowledge-base/modules

Load the Conda environment

$ conda list (Still no Tensorflow yet)

```
(base) (base) [yw1705@della-gpu ~]$ conda list
# packages in environment at /usr/licensed/anaconda3/2021.11:
#
# Name                    Version                   Build  Channel
_ipyw_jlab_nb_ext_conf    0.1.0           py39h06a4308_0
_libgcc_mutex             0.1                        main
_openmp_mutex             4.5                        1_gnu
alabaster                 0.7.12             pyhd3eb1b0_0
anaconda                  2021.11                 py39_0
anaconda-client           1.9.0           py39h06a4308_0
anaconda-navigator        2.1.1                   py39_0
anaconda-project          0.10.1             pyhd3eb1b0_0
anyio                     2.2.0           py39h06a4308_1
appdirs                   1.4.4              pyhd3eb1b0_0
argh                      0.26.2          py39h06a4308_0
```

https://researchcomputing.princeton.edu/support/knowledge-base/modules

Create virtual environments

Install virtual environment with tensorflow (latest version)

```
$ module load anaconda3/2021.11
# copy and paste the next 2 lines
$ CONDA_OVERRIDE_CUDA="11.2" conda create --name tf2-gpu "tensorflow==2.7.0=cuda112*" \
  matplotlib pandas --channel conda-forge
$ conda activate tf2-gpu
```

Install other package: pip install pyDOE

Environmental file for the group:

https://github.com/YaoGroup/IceShelf2D/tree/main/env

(Uploaded the environmental file under your della account)

```
$  module load anaconda3/2021.5
$  conda env create -f environment.yml
$  conda activate tf24
```

Recommended

19 lines (18 sloc) | 363 Bytes

```yaml
name: tf24
channels:
    - defaults
    - conda-forge
dependencies:
    - python=3.8.10
    - cudatoolkit=11
    - cudnn=8
    - defaults::scipy=1.6.2
    - defaults::ipykernel=5.3.4
    - defaults::matplotlib=3.4.2
    - tensorflow-probability=0.12.2
    - pydoe=0.3.8
    - jupyter=1.0.0
    - pytest=6.2.4
    - pip
    - pip:
        - tensorflow-gpu==2.4.1
```

Upload your python codes

Go to the folder where your python files are

```
$ cd Downloads/SSA_ice
```

Upload the file to your Della account (using *scp* command)

```
$ scp SSA_main.py yw1705@della-gpu.princeton.edu:/home/yw1705/SSA_ice
```

File to upload — account ID — Folder path for your della account

Or upload the entire folder to Della (using *scp -r* command)

```
$ scp -r SSA_ice yw1705@della-gpu.princeton.edu:/home/yw1705/
```

Folder to upload — account ID — Folder path for your della account

Slurm file

https://researchcomputing.princeton.edu/support/knowledge-base/slurm

```
#!/bin/bash
#SBATCH --job-name=ssa              # create a short name for your job
#SBATCH --output=ssa_%A.out         # stdout file
#SBATCH --error=ssa_%A.out          # stderr file
#SBATCH --nodes=1                   # node count
#SBATCH --ntasks=1                  # total number of tasks across all nodes
#SBATCH --cpus-per-task=1           # cpu-cores per task (>1 if multi-threaded tasks)
#SBATCH --mem-per-cpu=8G            # memory per cpu-core (4G is default)
#SBATCH --gres=gpu:1                # number of gpus per node
#SBATCH --time=30:00:00             # total run time limit (HH:MM:SS)
#SBATCH --mail-type=begin           # send email when job begins
#SBATCH --mail-type=end             # send email when job ends
#SBATCH --mail-user=yw1705@princeton.edu

module purge
module load anaconda3/2021.5
conda activate tf24

python3 /home/yw1705/SSA_ice/Final_IceInverse_Sys_main.py
~
~
```

Slurm file - job array

https://researchcomputing.princeton.edu/support/knowledge-base/slurm

```
#!/bin/bash
#SBATCH --job-name=array-job     # create a short name for your job
#SBATCH --output=slurm-%A.%a.out # stdout file
#SBATCH --error=slurm-%A.%a.err  # stderr file
#SBATCH --nodes=1                # node count
#SBATCH --ntasks=1               # total number of tasks across all nodes
#SBATCH --cpus-per-task=1        # cpu-cores per task (>1 if multi-threaded tasks)
#SBATCH --mem-per-cpu=4G         # memory per cpu-core (4G is default)
#SBATCH --time=00:01:00          # total run time limit (HH:MM:SS)
#SBATCH --array=0-4              # job array with index values 0, 1, 2, 3, 4
#SBATCH --mail-type=all          # send email on job start, end and fault
#SBATCH --mail-user=<YourNetID>@princeton.edu

module purge
module load anaconda3/2020.11
conda activate myenv

python myscript.py
```

the first few lines of a Python script (`myscript.py`) might look like this

```
import os
idx = int(os.environ["SLURM_ARRAY_TASK_ID"])
parameters = [2.5, 5.0, 7.5, 10.0, 12.5]
myparam = parameters[idx]
# execute the rest of the script using myparam
```

Submit the Slurm file

Submit the Slurm file

```
$ sbatch job.slurm
```

Check the status of ongoing jobs

```
$ squeue -u <YourNetID>
```

Cancel the ongoing jobs

```
$ scancel 40097615
```

```
(base) [yw1705@della-gpu ~]$ sbatch job_SSA.slurm
Submitted batch job 40097615
```

```
(base) [yw1705@della-gpu ~]$ ls
environment_new.yml  ssa_39676063.out  ssa_39683012.out  ssa_39785974.out  ssa_39786306.out
job_SSA_cpu.slurm    ssa_39676067.out  ssa_39683014.out  ssa_39785982.out  ssa_39786310.out
job_SSA.slurm        ssa_39676069.out  ssa_39683018.out  ssa_39786302.out  ssa_40097615.out
ondemand             ssa_39676073.out  ssa_39683022.out  ssa_39786303.out  SSA_ice
```

```
(base) [yw1705@della-gpu ~]$ squeue -u yw1705
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
          40097615       gpu      ssa   yw1705 PD       0:00      1 (None)
(base) [yw1705@della-gpu ~]$
```

Looking the output log file

```
$ tail -f ssa_40097615.out
```

Read the updated log file (update every 20 sec)

```
(base) [yw1705@della-gpu ~]$ tail -f ssa_40097615.out
Iter: 1560, loss: 3.6924e-03, loss_d: 3.6098e-03, loss_e: 8.2609e-03
Iter: 1570, loss: 3.6870e-03, loss_d: 3.6001e-03, loss_e: 8.6845e-03
Iter: 1580, loss: 3.6865e-03, loss_d: 3.6012e-03, loss_e: 8.5361e-03
Iter: 1590, loss: 3.6847e-03, loss_d: 3.5997e-03, loss_e: 8.4996e-03
Iter: 1600, loss: 3.6870e-03, loss_d: 3.6005e-03, loss_e: 8.6542e-03
Iter: 1610, loss: 3.6833e-03, loss_d: 3.5963e-03, loss_e: 8.6905e-03
Iter: 1620, loss: 3.6821e-03, loss_d: 3.5987e-03, loss_e: 8.3467e-03
Iter: 1630, loss: 3.6810e-03, loss_d: 3.5961e-03, loss_e: 8.4981e-03
Iter: 1640, loss: 3.6803e-03, loss_d: 3.5934e-03, loss_e: 8.6856e-03
Iter: 1650, loss: 3.6792e-03, loss_d: 3.5947e-03, loss_e: 8.4466e-03
Iter: 1660, loss: 3.6780e-03, loss_d: 3.5961e-03, loss_e: 8.1980e-03
Iter: 1670, loss: 3.6767e-03, loss_d: 3.5978e-03, loss_e: 7.8827e-03
Iter: 1680, loss: 3.6756e-03, loss_d: 3.5948e-03, loss_e: 8.0779e-03
Iter: 1690, loss: 3.6770e-03, loss_d: 3.5967e-03, loss_e: 8.0376e-03
Iter: 1700, loss: 3.6735e-03, loss_d: 3.5907e-03, loss_e: 8.2827e-03
Iter: 1710, loss: 3.6722e-03, loss_d: 3.5898e-03, loss_e: 8.2464e-03
```

Output file saving

https://researchcomputing.princeton.edu/systems/della#overview

Filesystem Usage and Quotas

/home (shared via NFS to all the compute nodes) is intended for scripts, source code, executables and small static data sets that may be needed as standard input/configuration for codes.

/scratch/network (shared via NFS to all the compute nodes) is intended for dynamic data that doesn't require high bandwidth i/o such as storing final output for a compute job. You may a create a directory /scratch/network/myusername, and use this to place your temporary files. Files are NOT backed up so this data should be moved to persistent storage once it is no longer needed for continued computation.

/scratch/gpfs (shared via GPFS to all the compute nodes, 800 TB) is intended for dynamic data that requires higher bandwidth i/o. Files are NOT backed up so this data should be moved to persistent storage as soon as it is no longer needed for computations.

Examples

```
from pathlib import Path

mdic = {"w": w, "b": b, "lossEnd": lossEnd, "n": n, "B0": B0, "dcen": dcen, "dsize": dsize}
FileName = 'Ice_Ross1_infer_%.4d.mat' % (l+1)
output_dir = '/scratch/gpfs/yw1705'
FilePath = str(Path(output_dir).joinpath(FileName))
savemat(FilePath, mdic)
```

Computational speed test

Systemic comparison between laptop, workstation and cluster

Test: SSA equation 10000 collo. point 8000 data point

	Full training (200k)	Per 1000 iteration
Laptop		520 sec
Workstation		79 sec
Della GPU	80 min	24 sec

Systemic comparison between Della CPU and GPU

https://hackmd.io/BLBm6v2CRHGe5xsfPGPN_g?view

CPU Result:

# of Collocation/Data Points, CPU Cores	Raw Time (secs)	Unit Time (secs per $pt \times 10^3$ iter)
1024 pts, 1 core	835.4102	0.4079
1024 pts, 8 core	473.1401	0.2310
1024 pts, 16 cores	439.2834	0.2145

GPU Result:

# of Collocation/Data Points	Raw Time (secs)	Unit Time (secs per $pt \times 10^3$ iter)
1024 pts (32 * 32)	31.6597	0.0154

GPU is significantly faster (> 50x) than CPU.