Exercise

Run a job on a machine

Login to Barnard

- On your local computer:
  - Start a SSH Client and connect with your useraccount to the HLRS training system (**login2.barnard.hpc.tu-dresden.de**)
  - Command line:
  - `ssh -Xl <youraccount> login2.barnard.hpc.tu-dresden.de`
  - *(replace <youraccount> with your actual account name)*

- Also see ZIH documentation:
  - https://doc.zih.tu-dresden.de/quickstart/getting_started/

Configure SSH

- To connect to the HPC system with SSH
  `ssh -X1 <youraccount> login2.barnard.hpc.tu-dresden.de`

- For easy access, create a local config file
  `~/.ssh/config`
- Add the following text

```
host training
    HostName login2.barnard.hpc.tu-dresden.de
    User <youraccount>
    TCPKeepAlive yes
    ForwardX11 yes
```

Optional, but simplifies further logins

- Now connecting is possible with
  `ssh training`

Create a Workspace on Vulcan

- Connect to HPC system using SSH
  ```
  ssh training
  ```

- Create personal workspace valid for 6 days and save it in MYWS
  ```
  MYWS=`ws_allocate cfd 7`
  ```
- List your workspaces
  ```
  ws_list
  ```

These commands are within the SSH session (on cluster)

- Save workspace variable in bashrc
  ```
  echo "export MYWS=$MYWS" >> $HOME/.bash_profile
  ```

Course Data

- Material for the course can be found on the cluster `/data/horse/ws/nhr420-hpcfd`

- We will refer to this directory on the slides with the variable `$KURS`

These commands are within the SSH session (on cluster)

- To make this available in your shell, use the following commands

```
echo "export KURS=/data/horse/ws/nhr420-hpcfd" >> $HOME/.bash_profile
```

- `source $HOME/.bash_profile`

The Directories

- To summarize: After the previous steps, we have two variables, that refer to our important directories

  1. **$MYWS**: Your workspace to run the exercises in
  2. **$KURS=**`/data/horse/ws/nhr420-hpcfd`
     → The course material

- You can check, that these are defined in your .profile with

```
cat $HOME/.bash_profile
```

Exercise – Get familiar with the terminal

- Print working directory: Shows your location in the folder tree
  `pwd`

These commands are within the SSH session (on cluster)

- List all files
  `ls`
- View manual for list-command and all options
  `man ls`
- Quit manual with:
  `q`

Exercise - Get familiar with the terminal

- Change into workspace
  `cd $MYWS`
- Make a directory
  `mkdir test`
- Go into directory / change directory
  `cd test`
- Move back to parent directory (move one ‘up’)
  `cd ..`
- Rename / move directory
  `mv test exercise`

These commands are within the SSH session (on cluster)

Exercise - Get familiar with the terminal

These commands are within the SSH session (on cluster)

- Create an empty file
  `touch myfile.txt`
- Copy file into directory
  `cp myfile.txt exercise/myfile2.txt`
- Remove file
  `rm myfile.txt`
- Remove directory (attention: -r option is needed for recursive operation)
  `rm -r exercise`

Exercise - Copy Data

- The folder *$KURS/exercises/hpcfdx1* contains
  - The source code *mpiintro.f90* (Fortran code)
  - The executable *mpiintro.sr*
  - The job script *job.slurm*

These commands are within the SSH session (on cluster)

- Copy the files to your personal scratch folder
  `cp -r $KURS/exercises/hpcfdx1 $MYWS`

- Change into the directory
  `cd $MYWS/hpcfdx1`

Running Computations: Jobs

- A single HPC computation is called a **job**
- Job Scheduler SLURM
  - Manages when to run jobs
  - Efficient usage of resources
  - Several commands (each with `-h` for options)
- One job = one command/script
  - Batch job: `sbatch <options> <scriptname>`
  - Monitor jobs: `squeue`, show partitions: `sinfo`
  - Delete job: `scancel <job-id>`

- See also: https://doc.zih.tu-dresden.de/jobs_and_resources/slurm/

Running Computations: Queues

- Jobs are put into queues (called partitions in SLURM)
  - Different runtime
  - Different size
  - Different type of node (e.g. GPU)
- Each queue has default values
- You pick queue, runtime, number of nodes

→ As many resources as necessary, as few as possible (with safety margin)

Other SLURM Commands

- `salloc` allocate compute resources for interactive use
- `srun` can be used outside or inside job
  - Executes the given program in parallel on the compute nodes
- `sinfo` will list available partitions
- `scancel <Job ID>` will kill a job
  - `scancel -u <Your Username>` kills all your jobs
- `scontrol` allows more in-depth information
  - Example: `scontrol show job <Job ID>`

SLURM Terminology and Key Concepts

- "Task": one process (typically)
  - Typically: specify either *tasks* or *nodes* + *tasks per node*
- "CPU": one processor core (important e.g. for multithreading)
  - Multithreading: set CPUs per task

- Typical: `srun --ntasks=$SLURM_NTASKS <whatever MPI-parallel software>`

Exercise - Job script (job.slurm)

```
#!/bin/bash
#SBATCH --time=0:20:00
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --mem 48000

module load GCC/11.3.0
module load OpenMPI/4.1.4

echo "Number of tasks: "
echo $SLURM_NTASKS

mpif90 -o mpiintro.barnard mpiintro.f90
srun ./mpiintro.sr
```

Exercise

- Try different meaningful combinations of the parameters in the job script and observe the effects in the output file
  - nodes, processes per node
    - The compute nodes on the Barnard cluster have 52 cores each

Note on Editors

- On the cluster only text based editors like vim, emacs and nano are usable
- For graphical editors you‘ll have to use your local machine
- MobaXterm on Windows offers the option to edit files in the integrated file viewer (right-click -> edit file)
- We use `$EDITOR` in places you need to invoke the editor

Workflow

- Edit job file
  - `$EDITOR job.slurm`
- Submit job file
  - `sbatch job.slurm`
- Read output
  - `$EDITOR intro.o<<jodID>>`

Reservations

- For our afternoon sessions we have reservations from 12:00 to 19:00 (16:00 on Friday):
  - Mo, 05.02. : **p_nhr_cfd_126**
  - Di, 06.02. : **p_nhr_cfd_127**
  - Mi, 07.02. : **p_nhr_cfd_128**
  - Do, 08.02. : **p_nhr_cfd_129**
  - Fr, 09.02. : **p_nhr_cfd_130**

- Use them with the `--reference=p_nhr_cfd_1??` in your `salloc` and `sbatch` commands