Onboarding Exercise

Welcome to the HPC onboarding exercise! This exercise will guide you through the process of getting started with the Noctua 2 Cluster and performing basic tasks. Let's begin!

Login to Noctua 2 Cluster

To start, log in to our training VM using the credentials provided to you via email. Logging in to this VM will give you access to the Noctua 2 Cluster, where you'll be working on various exercises. Open your terminal and enter the following command:

```
ssh <username>@training.pc2.upb.de
```

Replace <username> with your actual username (e.g., usrtr028).

Once you have successfully logged in to the training VM, proceed to log in to Noctua2. Use the following command:

```
ssh n2login2.ab2021.pc2.uni-paderborn.de
```

Create Your Working Directory

Let's create a folder on the shared file system that will serve as your working directory throughout the course. After logging in to Noctua2, navigate to the scratch folder of the course project. Use the following command:

```
cd /scratch/hpc-prf-nhrgs
```

Within the course project folder, create a sub-directory with your username. Replace <username> with your actual username (e.g., usrtr028):

```
mkdir <username>
```

You can store all relevant course materials and submit jobs in this sub-directory.

Find and Load Python Module

Depending on the project, you may need certain software environments to get started. For example, Python is a widely used programming language known for its versatility and extensive ecosystem of libraries and tools. This step ensures that you have the necessary dependencies and environment set up to work with Python effectively.

Let's find and load a suitable Python module on Noctua2. Use the `find_module` command to search for available Python modules:

```
find_module python
```

From the list of available Python modules, choose the one that suits your needs. For example, let's say you decide to use Python 3.10.4. Load the module using the `module load` command:

```
module load lang/Python/3.10.4-GCCcore-11.3.0
```

To verify that Python has been successfully loaded, run the following command:

```
python --version
```

The output should display the version of Python you loaded.

Create and Execute a Simple Python Script

Creating and executing a simple Python script allows you to verify that your Python setup is functioning correctly. This task serves as a quick test to ensure that you can run Python code on the login node of the Noctua 2 Cluster.

Change to your working directory:

```
cd /scratch/hpc-prf-nhrgs/<username>
```

Create a new Python script using your preferred text editor (e.g., vim, nano, etc.). Let's create a file named `hello.py` and open it in the vim editor as an example:

```
vim hello.py
```

In the editor, enter the following Python code:

```
print("Hello World!")
```

Save the file and exit the editor.

To execute the Python script, use the Python interpreter:

```
python hello.py
```

The output should display the message "Hello World!".

You have successfully found and loaded Python on Noctua2, created a simple Python script, and executed it on the login node of Noctua2. However, please note that executing

applications on the login node should be limited to quick tests, small-scale development, and administrative tasks. For production workloads, resource-intensive computations, and long-running jobs, it is essential to submit applications as jobs through the Slurm scheduler.

Submitting a Slurm Job

In an HPC environment, submitting jobs through a job scheduler like Slurm is crucial for managing computational resources efficiently. By submitting a Slurm job, you'll learn how to encapsulate your tasks into a job script, specify resource requirements, and submit your job to the cluster. This step prepares you for running larger and more resource-intensive computations in a controlled and scalable manner.

To submit a Slurm job, you need to create a job script. Open your preferred text editor (e.g., vim, nano) and create a file named `job.sh`:

```
vim job.sh
```

In the editor, enter the following content:

```
#!/bin/bash
#SBATCH -J myjob
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 128
#SBATCH --account=hpc-prf-nhrgs
#SBATCH --partition=normal
#SBATCH -t 00:15:00
#SBATCH --output=python_output.txt

# Load necessary modules
module load lang/Python/3.10.4-GCCcore-11.3.0

# Run program
srun python hello.py
```

This job script requests a single node (`-N 1`) for 15 minutes (`-t 00:15:00`), on which we'll run 1 process (`-n 1`) using 128 CPU cores (`-c 128`). If you need an entire node exclusively for yourself, you can add `#SBATCH --exclusive`.

Save the file and exit the editor.

To submit the job script to the Slurm scheduler, use the following command:

```
sbatch job.sh
```

This will submit your job to the cluster.

To monitor the status of your job, you can use the `squeue` command:

```
squeue
```

This command will display the status of your submitted job, including the job ID and its current state (e.g., “PD” for pending or “R” for running).

Once your job has completed, you can check the output generated by the script. In this case, the output is redirected to a file named `python_output.txt`. Use the `cat` command to view the contents of the output file:

```
cat python_output.txt
```

If you require an additional NVIDIA A100 GPU for your session, you need to change the partition and specify the GPU using the `--gres` parameter in the job script:

```
#!/bin/bash
#SBATCH -J myjob
#SBATCH -N 1
#SBATCH -n 1
#SBATCH -c 128
#SBATCH --account=hpc-prf-nhrgs
#SBATCH --partition=gpu
#SBATCH --gres=gpu:a100:1
#SBATCH -t 00:15:00
#SBATCH --output=python_output.txt

# Load necessary modules
module load lang/Python/3.10.4-GCCcore-11.3.0

# Show that GPU is available
nvidia-smi

# Run program
srun python hello.py
```

However, please note that the availability of GPUs may vary (your job may be stuck in the queue for several days.). However, during the course week, an extra reservation has been created for you to ensure quick access to an additional NVIDIA A100 GPU.

Further information and documentation

More information and documentation about the software and hardware environment at PC2 can be found at doku.pc2.uni-paderborn.de