Cluster Usage Tutorial

WireGuard Activation

If you are using your hotspot, you can update the DNS resolver settings to ensure proper connectivity:

1. Edit the DNS configuration file:

   ```
   sudo nano /etc/resolv.conf
   ```

2. Replace the existing content with the following:

   ```
   nameserver 8.8.8.8
   nameserver 1.1.1.1
   ```

3. Save the file and exit.

Connecting to the Cluster

Use the following command to connect to the cluster (adjust based on your SSH configuration):

```
ssh cluster
```

Module Management

1. Load the conda environment:

   ```
   ml conda  # or
   module load conda
   ```

2. Check available modules:

   ```
   module avail
   ```

Conda Environment Setup

1. Create a new Conda environment:

   ```
   conda create -n my_env  # Add specific Python version if needed
   ```

2. Activate the environment:

```
conda activate my_env
```

Cloning and Installing Requirements

1. Clone your repository:

```
git clone <repository_url>
cd <repository_name>
```

2. Install dependencies:

```
conda install -r requirements.txt  # Or
pip install -r requirements.txt
```

Handling Disk Space Issues (Spoiler Alert: You Will Encounter Them)

If you encounter a disk space error during installation:

1. Create a temporary directory:

```
mkdir ~/tmp
```

2. Use the temporary directory for installation:

```
TMPDIR=$HOME/tmp pip install
# Or
TMPDIR=$HOME/tmp conda install
```

GPU and CUDA Setup

1. Check the CUDA version:

```
module load cuda/xx.x
```

2. Verify the compatible PyTorch version from the PyTorch website.

Using VS Code for Remote Development

1. Install the **Remote - SSH** extension in VS Code.
2. Connect to the cluster using the GUI or terminal.

Checking GPU Availability

1. General cluster information:

   ```
   sinfo
   sinfo -Nl
   ```

2. Filter GPU nodes:

   ```
   sinfo -o "%25N %5c %10m %32f %10G %18P" | grep gpu
   ```

3. Detailed node information:

   ```
   scontrol show node <node_name>
   ```

Running Interactive Jobs

Use one of the following commands to start an interactive job:

1. Specific resource request:

   ```
   srun --partition=mundus --gres=gpu:a100-5:1 --mem=32G --cpus-per-
   task=4 --time=1:00:00 --pty bash
   ```

2. Default interactive session:

   ```
   srun --pty bash
   ```

Submitting Batch Jobs

1. Create a job file (e.g., `file.sh`):

   ```
   #!/bin/bash

   #SBATCH --job-name=Train_USIS
   #SBATCH --partition=mundus            # Use the mundus partition
   ```

```
#SBATCH --nodelist=mundus-mir-1       # Use a specific node # mir-1 or mir-2 or mir-3
#SBATCH --gres=gpu:a100-5:1          # Request 1 GPU of type a100-5
#SBATCH --ntasks=1                    # Single task
#SBATCH --cpus-per-task=4             # 4 CPU threads per task
#SBATCH --mem=32G                     # Request 32 GB of memory
#SBATCH --time=27:00:00               # Time limit: 27 hours
#SBATCH --output=output/job_output_%j.log    # Output log
#SBATCH --error=output/job_error_%j.log      # Error log

# Load necessary modules
module purge
module load cuda/12.1
module load conda
conda activate my_env

# Execute the desired script
python file.py
```

2. Submit the job:

```
sbatch file.sh
```

3. Track the job status:

```
squeue -j <job_id>
```

4. Monitor output and error logs:

```
tail -f output/job_output_<job_id>.log
tail -f output/job_error_<job_id>.log
```

File Transfer

Use `rsync` to transfer files between your local machine and the cluster:

```
rsync -r <source_path> <destination_path>
```

Example:

```
rsync -r /home/data_folder user@sms.lis-lab.fr:/home/mundus/user
```

Notes

- Always ensure your configurations (e.g., partitions, resources) align with the cluster policies.
- Adjust time limits and memory requests based on your specific job requirements.
- For further assistance, I accept donationns.