Beginner's Guide to UT HPC

Fall 2025

In this guide, we will learn how to use the University of Tartu's
High Performance Computing Center servers. Later we will use GPUs there
to train our machine translation models.

Login

To connect to the server, use SSH.

> **Note:**
> If you only use Windows, please set up a Linux virtual machine
> (I recommend Ubuntu for Windows).
> Another option is to install PuTTY.
>
> In macOS and Linux terminals, you do not need to do anything extra.

Log in using your university username and password:

```
ssh your_username@rocket.hpc.ut.ee
```

For added security and convenience, you can set up an SSH keypair

Finding your way around

If you are not comfortable with Linux commands, check out, for example, this guide.

Once you have logged in, you can create a directory where you will keep all the data for your experiments:

```
mkdir transformers-course
```

Move into the new directory:

```
cd transformers-course
```

Create directories in which to store your data and scripts:

```
mkdir data
mkdir scripts
```

Environment

First, load python:

```
module load python/3.10.10
```

Then create a new Python virtual environment:

```
python -m venv transformers-venv
```

Activate the environment:

```
source transformers-venv/bin/activate
```

Re-activate it later with

```
module load python/3.10.10
source transformers-venv/bin/activate
```

Upgrade build tools (helps avoid weird wheel issues):

```
python -m pip install --upgrade pip wheel setuptools
```

Install packages.
torch:

```
pip install torch --index-url https://download.pytorch.org/whl/cu121
```

NumPy:

```
pip install -U numpy
```

Hugging Face:

```
pip install -U transformers datasets accelerate huggingface_hub tokenizers
sentencepiece
```

SLURM

The Rocket cluster uses SLURM (a scheduling system for running jobs).
We will cover all the basic things that you will need in this document,
but you can check out HPC's docs
as well.

> **Note:**
> **DO NOT** execute commands that require considerable resources (preprocessing, model training, etc.)
> directly on the head node! **ALWAYS USE SLURM,** or your access to HPC
> may be suspended. You can do some small and quick stuff on the nead node,
> like managing your virtual environments, installing packages, copying or
> removing files, etc.
>
> Generally, be responsible and considerate of others. For example, if you are holding up a GPU
> and not using it, people who are waiting for it will not be happy. Don't do things like that.

To submit your jobs to SLURM, you will need scripts like this one. There is also a course project, which is mainly for lab materials: `/gpfs/space/projects/transformers2025`

```
#!/bin/bash

# The name of the job is test_job
#SBATCH -J test_job

# Format of the output filename: slurm-jobname.jobid.out
#SBATCH --output=slurm-%x.%j.out

# The job requires 1 compute node
#SBATCH -N 1

# The job requires 1 task per node
#SBATCH --ntasks-per-node=1

# The maximum walltime of the job is 5 minutes
#SBATCH -t 00:05:00
```

```
#SBATCH --mem=5G

# If you keep the next two lines, you will get an e-mail notification
# whenever something happens to your job (it starts running, completes or
fails)
#SBATCH --mail-type=ALL
#SBATCH --mail-user=your_email@here.com

# Keep this line if you need a GPU for your job
#SBATCH --partition=gpu

# Indicates that you need one GPU node
#SBATCH --gres=gpu:tesla:1

# Commands to execute go below

# Load CUDA module (for jobs that need GPU)
module load cuda/12.1

# Load Python
module load python/3.10.10

# Activate your environment
source transformers-venv/bin/activate
```

I recommend to save the scripts that you use for every step of your work. This way, you can always go back and check what you did. Having all your scripts makes it easier to find mistakes.

Do not ask for GPU nodes when you perform preprocessing steps (e.g. cleaning, SentencePiece). It will not make them faster and you will block valuable resources. Only use GPUs for training models.

To send your script to the queue:

```
sbatch path/to/your/script.sh
```

You will see output like:

```
Submitted batch job XXX
```

`XXX` will be the ID of your job. If you want to cancel it:

```
scancel XXX
```

Once your job starts running, a file named `slurm-test_job.XXX.out` will be created in your current working directory (where you executed `sbatch`).
Your log and output will be written into this file.

Queue

You can view your jobs that are pending or running:

```
squeue -u your_username
```

Or all the GPU jobs on the cluster:

```
squeue -p gpu
```

Or simply all the jobs on the cluster (the list will be very long):

```
squeue
```

Run a script

Task. Create your first SLURM script (you can call it `test_env.sh`) and create a Python file

```
import torch

print("Torch:", torch.__version__)
print("CUDA version baked in:", torch.version.cuda)
print("GPU available:", torch.cuda.is_available())
if torch.cuda.is_available():
    print("GPU name:", torch.cuda.get_device_name(0))

from huggingface_hub import whoami
from transformers import AutoTokenizer
```

```
print("Transformers import OK")
# This will download a small tokenizer into HF_HOME cache:
tok = AutoTokenizer.from_pretrained("bert-base-uncased")
print("Vocab size:", tok.vocab_size)
```

Your goal is to check that the training tools are installed and get access to GPU.

4. Make it run and check the contents of your output file to see if everything works correctly.