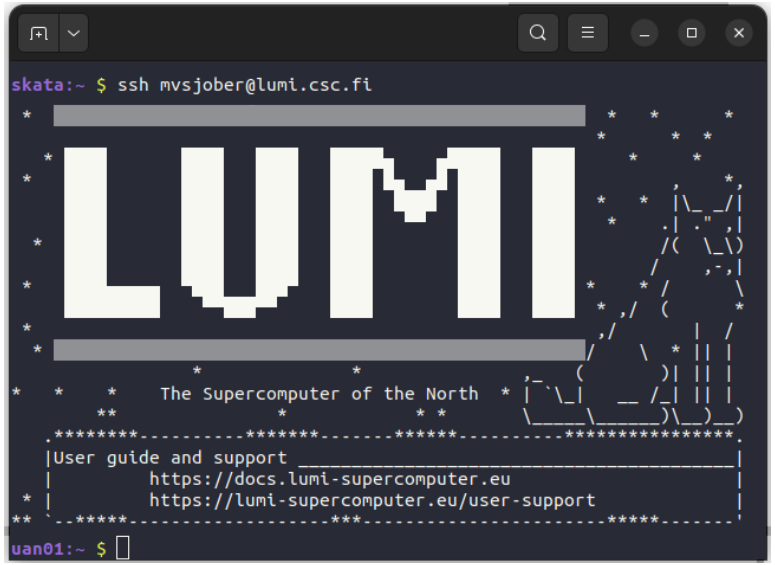
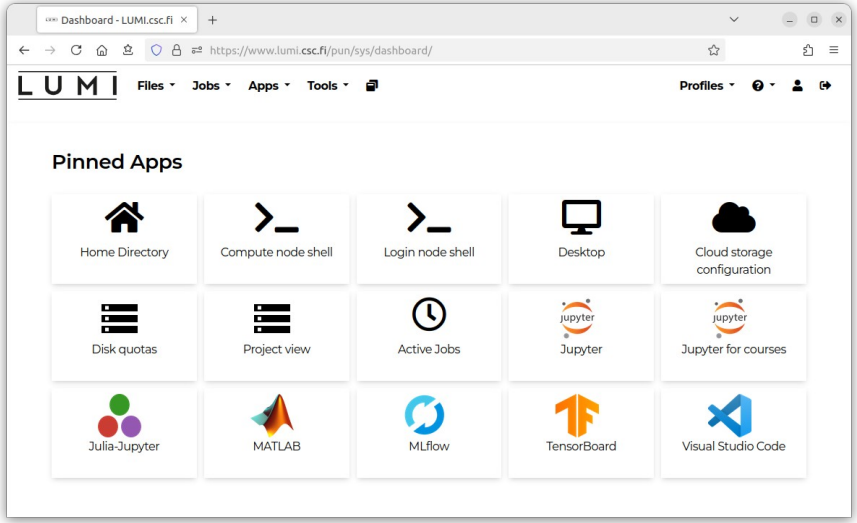
Using the LUMI web-interface

Mats Sjöberg, Oskar Taubert – CSC – IT Center for Science, Finland

Two ways of accessing LUMI

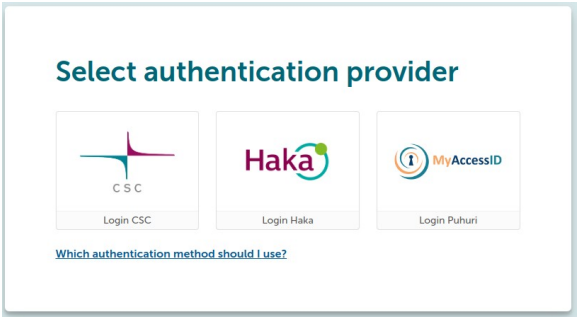
- via SSH connection
  - terminal access only
- via the LUMI web-interface
  - browser based
  - terminal + various apps

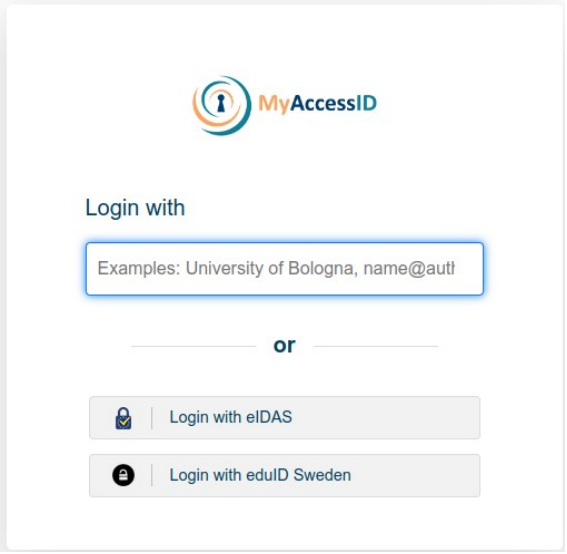


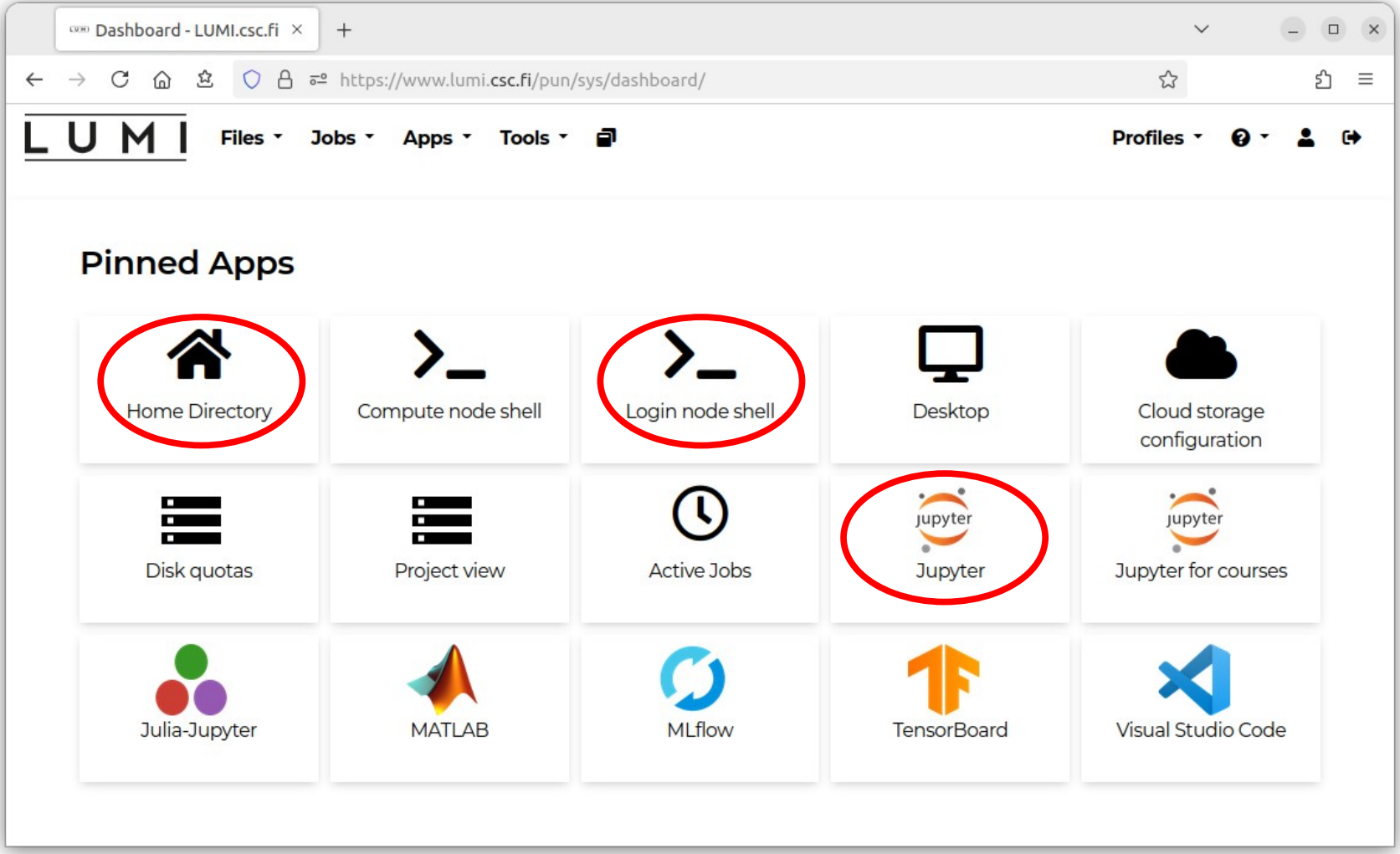


Accessing the LUMI web-interface

- Go to https://www.lumi.csc.fi
- Click “Go to login”
- Select login method
  - In most cases you should select MyAccessID / Puhuri
  - Finnish users: if you have linked your account to MyCSC, you can also use Haka
- MyAccessID: type the name of your institution / university
- You should be directed to your institution’s login page



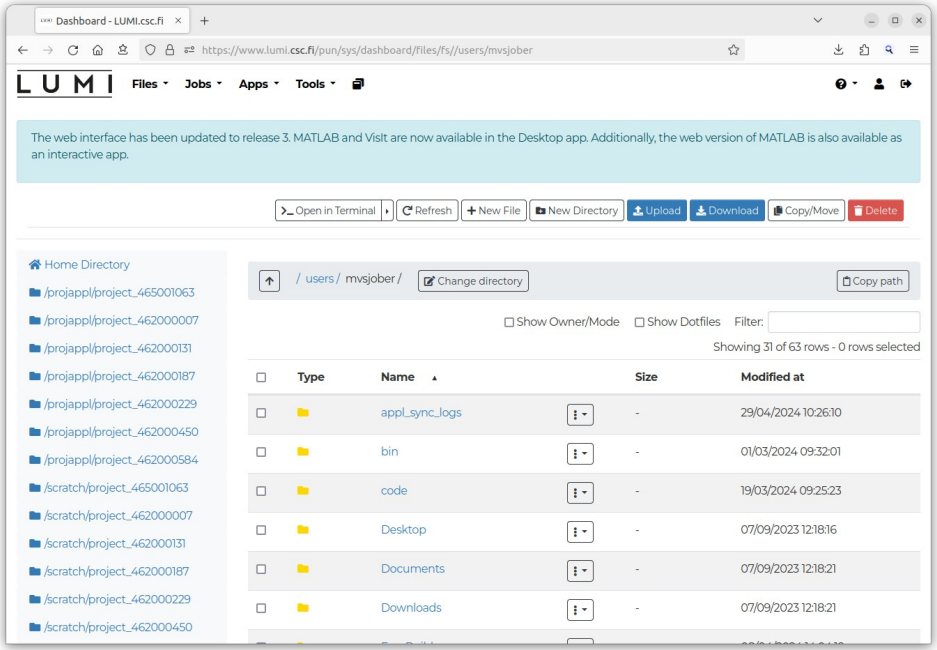


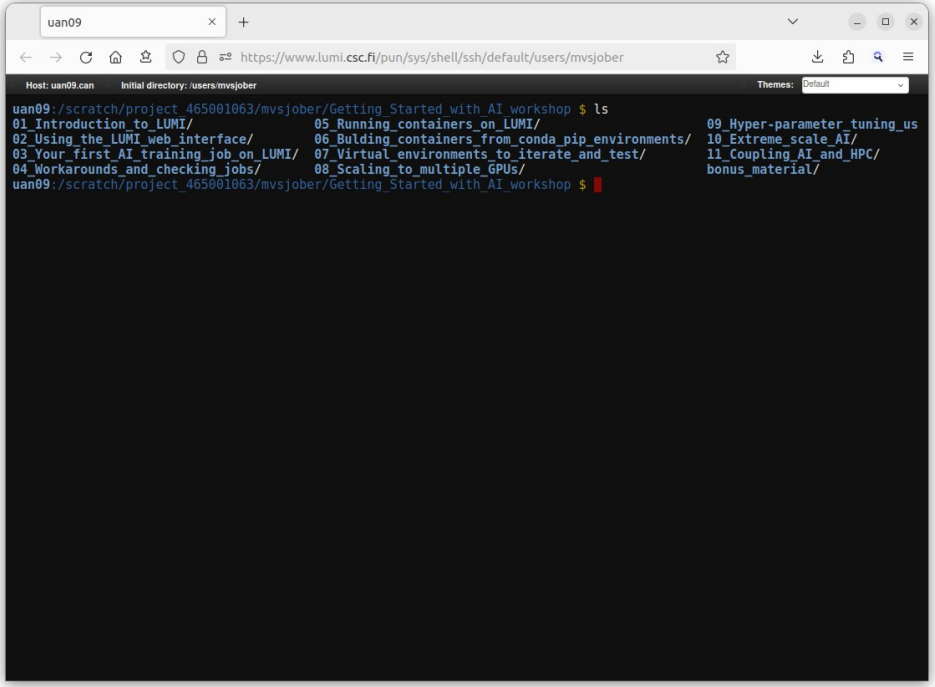

Dashboard - LUMI.csc.fi
https://www.lumi.csc.fi/pun/sys/dashboard/
LUMI
Files
Jobs
Apps
Tools
Profiles
Pinned Apps
Home Directory
Compute node shell
Login node shell
Desktop
Cloud storage configuration
Disk quotas
Project view
Active Jobs
Jupyter
Jupyter for courses
Julia-Jupyter
MATLAB
MLflow
TensorBoard
Visual Studio Code

Files browser and shell (login node)

Home Directory

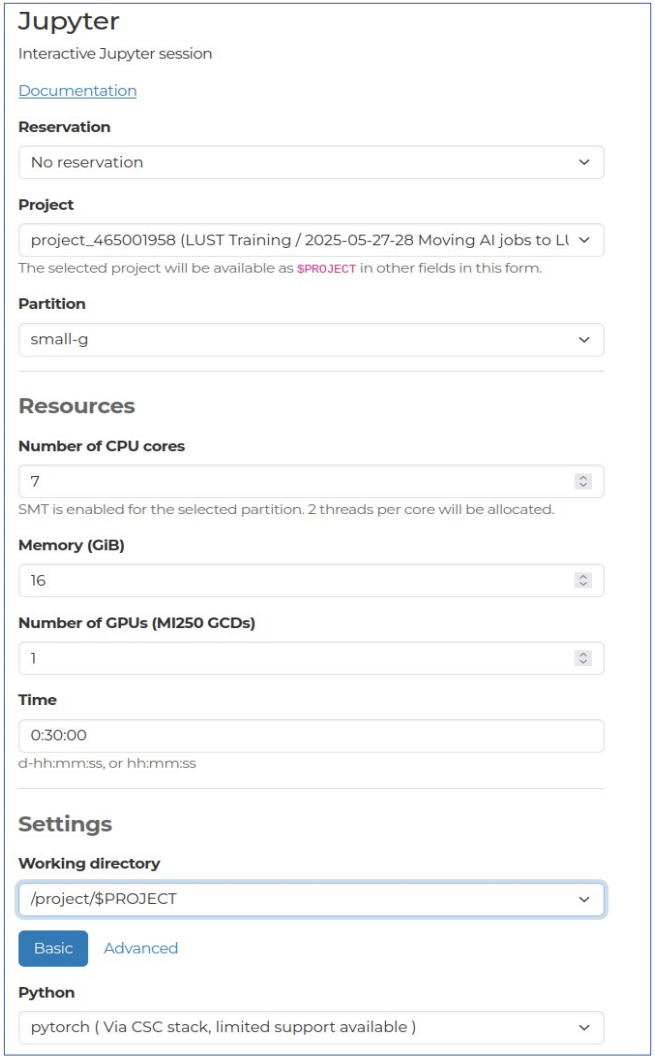
Login node shell





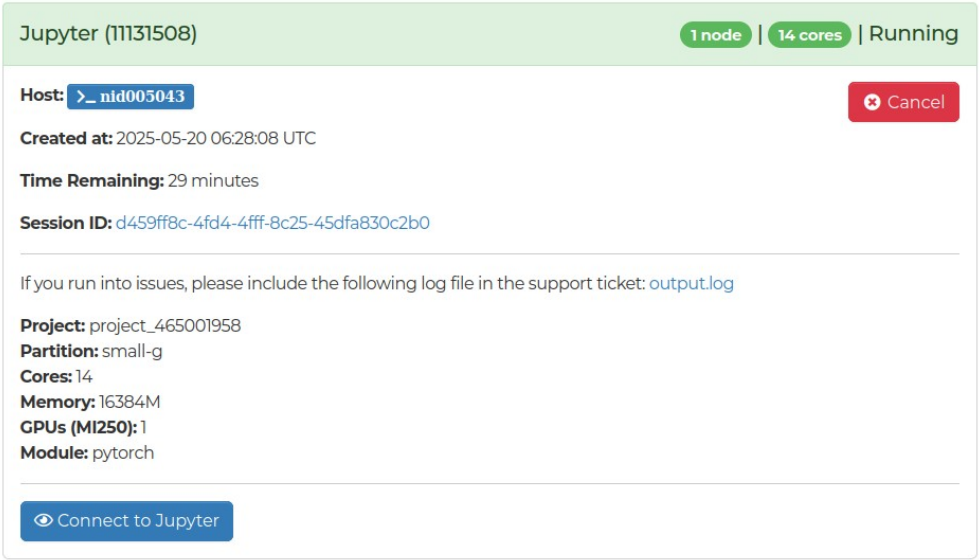
Jupyter notebook

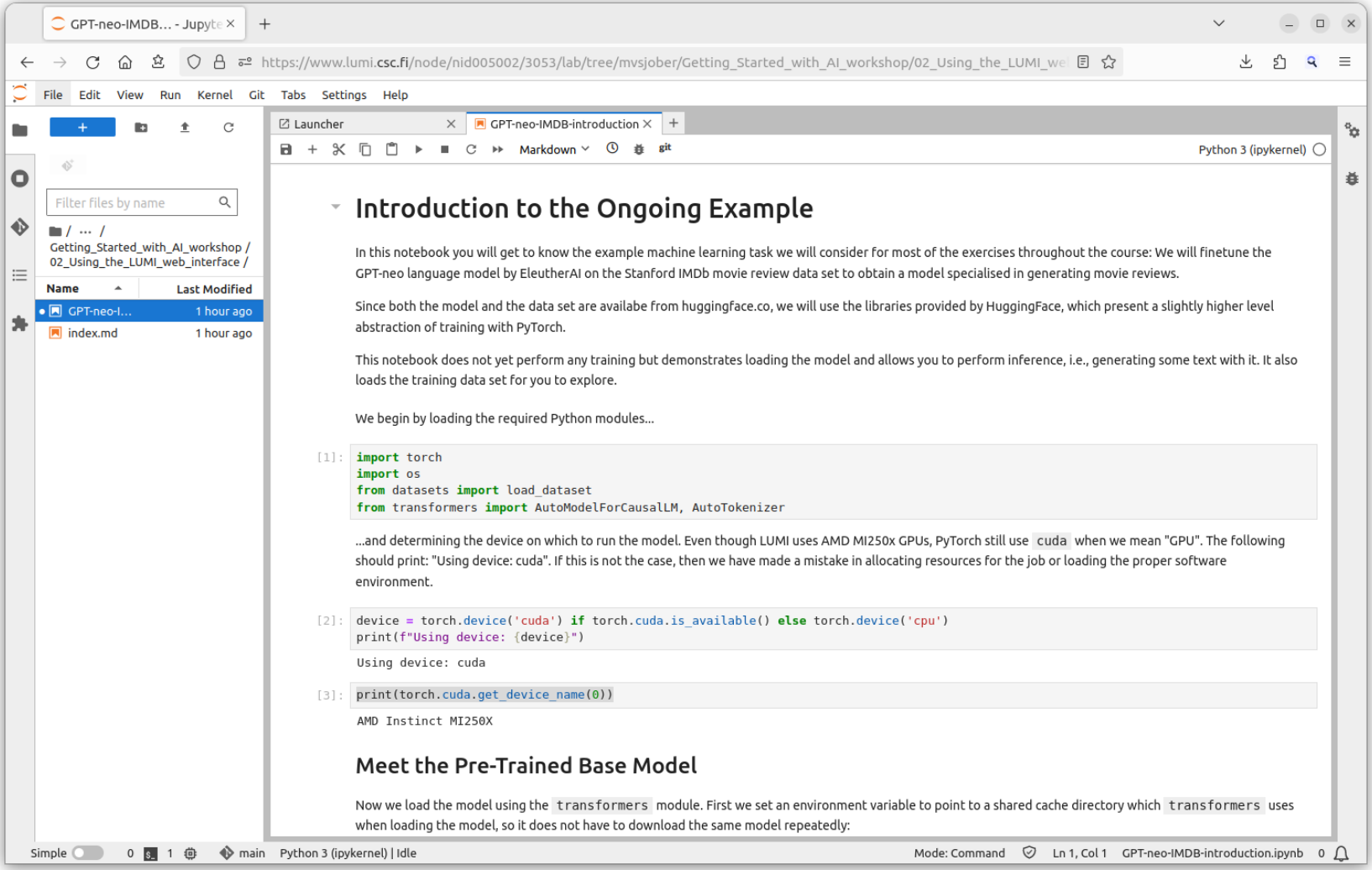
- Launching Jupyter actually launches a job in the cluster (more on that in the next lecture)
- You need to fill in some things:
  - Project
  - Partition - **not needed with course reservation!**
  - Resources
  - What Python installation to use
- Max resources for a single GPU job – 1/8 of a node:
  - Number of CPU cores: 7
  - Memory: 60 GB



Jupyter notebook

- The job is submitted to the normal Slurm queue
- Once it has started "Connect to Jupyter" button will appear




GPT-neo-IMDB... - Jupyte
https://www.lumi.csc.fi/node/nid005002/3053/lab/tree/mvsjober/Getting_Started_with_AI_workshop/02_Using_the_LUMI_we
File Edit View Run Kernel Git Tabs Settings Help
Launcher
GPT-neo-IMDB-introduction
Markdown
Python 3 (ipykernel)
Filter files by name
/ ... /
Getting_Started_with_AI_workshop /
02_Using_the_LUMI_web_interface /
Name
Last Modified
GPT-neo-I...
1 hour ago
index.md
1 hour ago
Introduction to the Ongoing Example
In this notebook you will get to know the example machine learning task we will consider for most of the exercises throughout the course: We will finetune the GPT-neo language model by EleutherAI on the Stanford IMDb movie review data set to obtain a model specialised in generating movie reviews.
Since both the model and the data set are availabe from huggingface.co, we will use the libraries provided by HuggingFace, which present a slightly higher level abstraction of training with PyTorch.
This notebook does not yet perform any training but demonstrates loading the model and allows you to perform inference, i.e., generating some text with it. It also loads the training data set for you to explore.
We begin by loading the required Python modules...
[1]:
import torch
import os
from datasets import load_dataset
from transformers import AutoModelForCausalLM, AutoTokenizer
...and determining the device on which to run the model. Even though LUMI uses AMD MI250x GPUs, PyTorch still use cuda when we mean "GPU". The following should print: "Using device: cuda". If this is not the case, then we have made a mistake in allocating resources for the job or loading the proper software environment.
[2]:
device = torch.device('cuda') if torch.cuda.is_available() else torch.device('cpu')
print(f"Using device: {device}")
Using device: cuda
[3]:
print(torch.cuda.get_device_name(0))
AMD Instinct MI250X
Meet the Pre-Trained Base Model
Now we load the model using the transformers module. First we set an environment variable to point to a shared cache directory which transformers uses when loading the model, so it does not have to download the same model repeatedly:
Simple 0 1 main Python 3 (ipykernel) | Idle
Mode: Command Ln 1, Col 1 GPT-neo-IMDB-introduction.ipynb 0

Limitations of interactive Jupyter runs

LUMI

- Using GPUs interactively is *inefficient usage of resources*
  - Most of the time, when you are editing code or grabbing a coffee , the GPU is idle (but nobody else can use it!)
  - Because of this interactive use is limited to a single node
- Running multiple copies of your job (e.g., hyperparameter search) is not possible
- Solution:
  - Use Jupyter for development and experimentation
  - Use terminal interface and batch jobs for real runs

more on this in the next session

Our running example for this course

LUMI

- Finetuning GPT-Neo LLM for generating movie reviews on the IMDb data set
- Using Hugging Face's `datasets` and `transformers` on top of PyTorch as training library

GPT-Neo 1.3B
- 1370 M parameters
- BF16
- ~2.67 GB

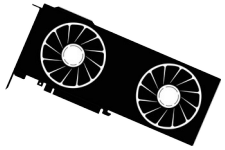
Stanford IMDb data set
- 100 000 movie reviews
- Varying lengths (low hundreds of words)
- 25 000 reserved for testing

PyTorch

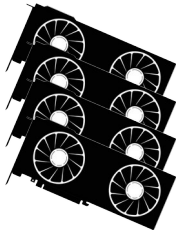
Our running example for this course

LUMI

02: Get familiar with LUMI web interface and the example



03: Using Slurm scheduler to train on a single GPU
04: Checking on training jobs and some common problems



08: Training on multiple GPUs (on a single GPU node)
09: Training across multiple nodes

Icons: vecteezy.com

Course practicalities: Reservations

LUMI

- A portion of the cluster is reserved for the course
- When starting jobs you need to give the reservation names, otherwise you apply for resources in the general queues
  - in the web-interface, select from the drop down menu
- Day 1: `AI_workshop_1`
- Day 2: `AI_workshop_2`
- However, if you use a reservation that is not active, your job will not run until the reservation becomes active.