



LUMI

First AI training
job on LUMI

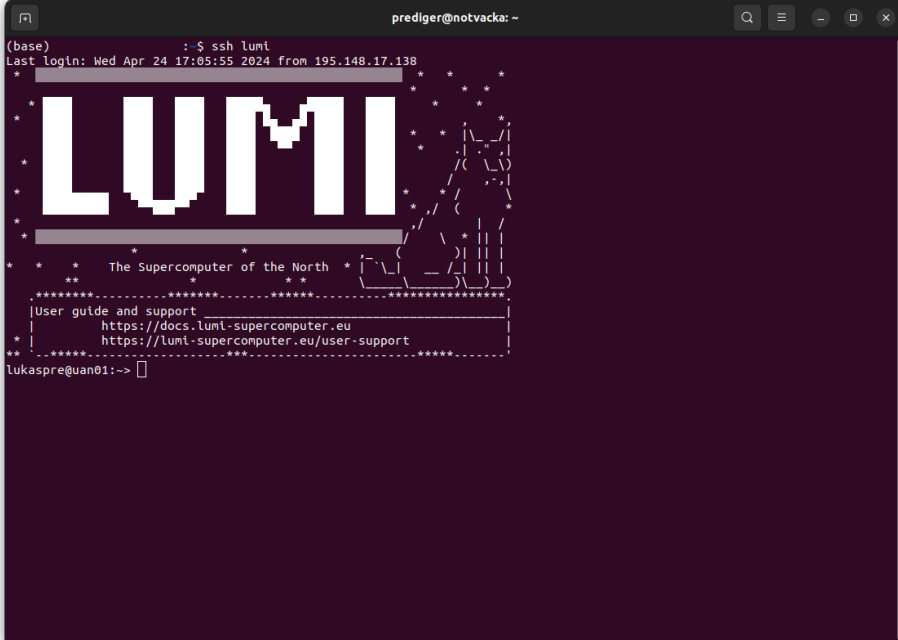
Mats Sjöberg, Lukas Prediger – CSC – IT Center for Science, Finland

Accessing LUMI via SSH

- No passwords
→ SSH key pair **required**
- register your public key in your MyAccessID user profile

```
ssh -i <path-to-private-key> \  
    <username>@lumi.csc.fi
```
















<https://docs.lumi-supercomputer.eu/firststeps/>

A terminal window showing the LUMI login banner. The banner features the word 'LUMI' in large, stylized white letters on a dark purple background. Below the logo, it says 'The Supercomputer of the North' and provides links for user guides and support. The terminal prompt shows the user 'lukaspre@uan01' is ready for input.

```
prediger@notvacka: ~  
(base) : $ ssh lumi  
Last login: Wed Apr 24 17:05:55 2024 from 195.148.17.138  
*  
* L U M I *  
*  
* The Supercomputer of the North *  
*  
* User guide and support *  
* https://docs.lumi-supercomputer.eu *  
* https://lumi-supercomputer.eu/user-support *  
*  
*  
lukaspre@uan01:~$
```

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

Setting up the Software Environment

<https://pytorch.org/>

PyTorch Build	Stable (2.5.1)			Preview (Nightly)	
Your OS	Linux		Mac	Windows	
Package	Conda	Pip		LibTorch	Source
Language	Python			C++/Java	
Compute Platform	CUDA 11.8	CUDA 12.1	CUDA 12.4	ROCm 6.2	CPU
Run this Command:	<pre>pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/rocm6.2</pre>				

```
! environment-minimal.yml
1 channels:
2   - pytorch
3   - conda-forge
4   - defaults
5 dependencies:
6   - accelerate=0.29.1
7   - datasets=2.18.0
8   - python=3.10.14
9   - pytorch=2.2.2
10  - transformers=4.39.3
11
```

~47k files



**bad for the
shared network
filesystem**

Setting up the Software Environment

- Singularity/Apptainer containers recommended for ML applications on LUMI
- Similar to Docker, better suited for multi-user supercomputers
 - Isolated environment
 - Easier to manage complex software dependencies
- For now we'll provide a pre-installed container
 - Later lectures will discuss how to create your own containers

python



```
srun singularity exec $CONTAINER python
```

Setting up the Software Environment

Set up the software environment

```
$ module purge  
$ module use /appl/local/training/modules/AI-20240529/  
$ module load singularity-userfilesystems singularity-CPEbits
```

Launch the PyTorch container

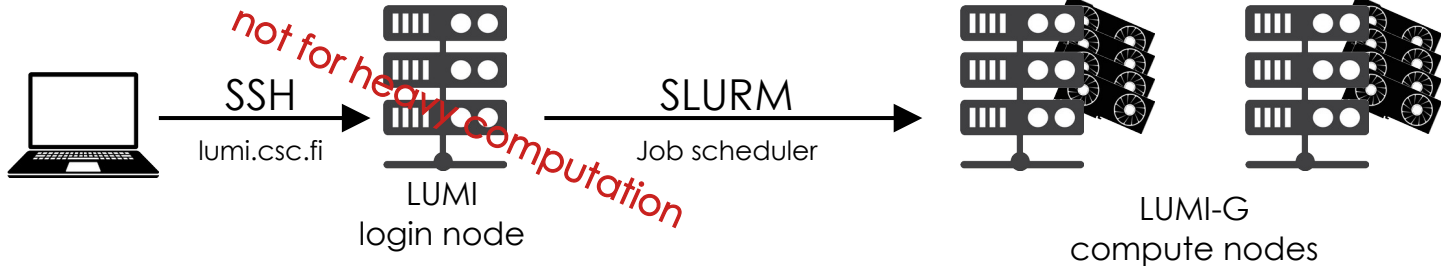
```
$ CONTAINER=/scratch/project_465001363/containers/pytorch_transformers.sif  
$ srun singularity exec $CONTAINER python
```

Don't run computation on the login node
– use the Slurm job scheduler!

LUMI

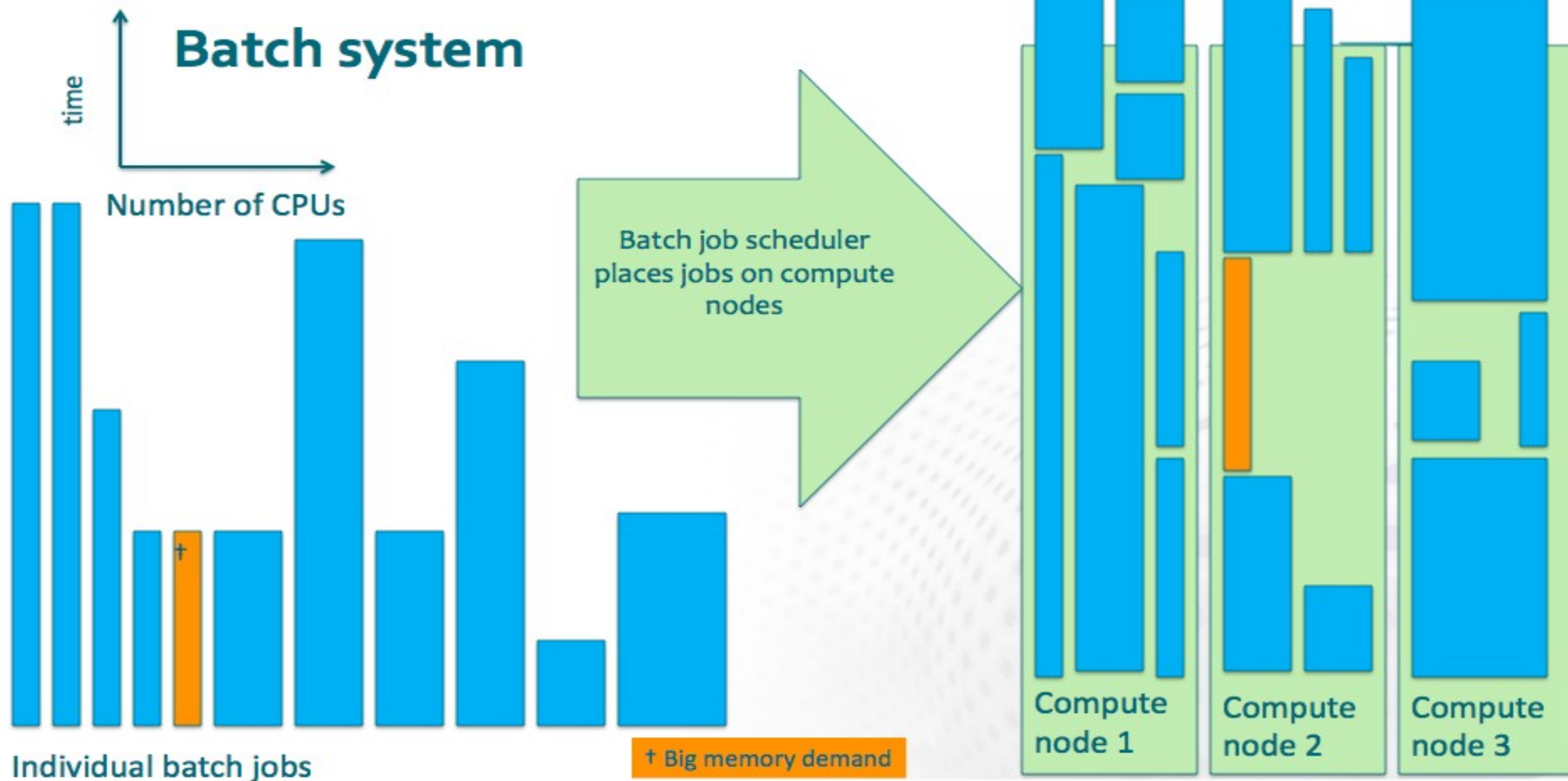


```
$ python my_pytorch_script.py
```



SLURM is used to reserve resources and submit scripts for running on the compute nodes

Batch system



Training on a single Graphics Compute Die

LUMI

SLURM batch script (run.sh)

```
#!/bin/bash
```

```
#SBATCH --account=project_123456
```

```
#SBATCH --partition=small-g
```

```
#SBATCH --gpus-per-node=1
```

```
#SBATCH --ntasks-per-node=1
```

```
#SBATCH --cpus-per-task=7
```

```
#SBATCH --mem-per-gpu=60G
```

```
#SBATCH --time=1:00:00
```

What resources requested?

- 1 GPU = 1/8 of node
- Use also $\leq 1/8$ of CPU cores and memory

```
module purge
```

What software to load?

```
module use /appl/local/training/modules/AI-20240529/
```

```
module load singularity-userfilesystems singularity-CPEbits
```

```
CONTAINER=/scratch/project_465001363/containers/pytorch_transformers.sif
```

```
srun singularity exec $CONTAINER python my_pytorch_script.py
```

Available GPU partitions

- **standard-g**
≤ 48h, whole nodes only, max 1024 nodes/job
- **small-g**
≤ 72h, individual GCDs, max 4 nodes/job
- **dev-g**
≤ 3h, individual GCDs, max 32 nodes/job, *max 1 job running*

Useful SLURM commands

Submit the SLURM batch script

```
$ sbatch run.sh  
Submitted batch job 987654
```

Check the SLURM queue

```
$ squeue --me  
  
JOBID    PARTITION NAME    ST TIME    NODES NODELIST  
987654   small-g   run.sh  R   0:18      1 compute_node
```

Cancel a job

```
$ scancel 987654
```

Useful SLURM commands

```
Read job outputs  
$ tail -f slurm-987654.out
```

```
Access compute node for diagnostics  
$ srun --interactive --pty -jobid=987654 bash  
[compute_node] $ rocm-smi
```

Useful SLURM commands for monitoring job progress

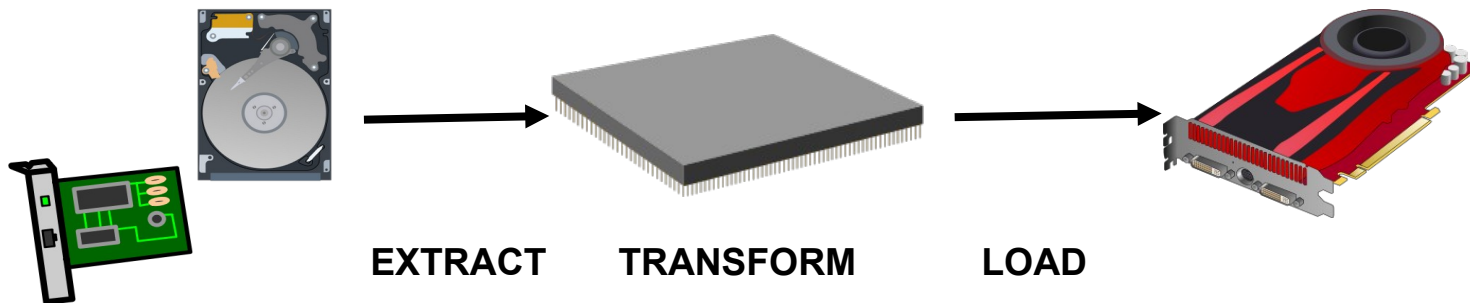
Check the status of partitions

```
$ sinfo -s
PARTITION  AVAIL  TIMELIMIT  NODES(A/I/O/T)  nid[002595-002597]
debug      up     30:00      8/0/0/8         nid[002595-002597]
interactive up     8:00:00    4/0/0/4         nid[002502,002507,002594,002599]
q_fiqci    inact  15:00      0/1/0/1         nid002598
q_industry up     15:00      0/1/0/1         nid002598
q_nordiq   up     15:00      0/1/0/1         nid002503
small      up 3-00:00:00 280/6/20/306   nid[002280-002499,002508-002593]
standard   up 2-00:00:00 1493/111/124/172 nid[001000-002279,002600-003047]
dev-g      up     3:00:00    30/17/1/48     nid[005002-005025,007954-007977]
small-g    up 3-00:00:00 193/2/3/198   nid[005026-005123,007852-007951]
standard-g up 2-00:00:00 2555/23/150/2728 nid[005124-007851]
largemem   up 1-00:00:00 1/5/0/6        nid[000101-000106]
lumid      up     4:00:00    1/6/1/8        nid[000016-000023]
```

A = allocated
I = idle
O = other states
T = total

Using multiple CPUs for ETL

LUMI



- Reserve enough CPU cores per GPU, 7 cores/GPU on LUMI

```
#SBATCH --cpus-per-task=7
```

- Use multiple workers (processes) in PyTorch DataLoader:

```
train_loader = torch.utils.data.DataLoader(... ,  
num_workers=N)
```

Use checkpointing!

Checkpointing = save the model weights every now and then

- E.g. after each epoch, or every N batches
- If your run crashes, simply restart from the latest checkpoint
- Train for longer than the “short” 2-3 days limit on LUMI partitions
- Exploit scheduling “backfilling”

HuggingFace Trainer example

```
training_args =  
    TrainingArguments(  
        ...,  
        overwrite_output_dir=False,  
        ...  
    )  
  
trainer = Trainer(...)  
trainer.train(  
    resume_from_checkpoint=True  
)
```

Monitoring Progress with TensorBoard

LUMI



- HuggingFace trainer automatically writes logs for TensorBoard
- You can start the UI in the LUMI web interface
- For the exercises use path:
`/scratch/project_465001363/USERNAME/runs/`

https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html

