

Tino Meisel


Deep Learning with GPU Cores

Hands on: Submitting a job in an HPC Cluster, Train a Neural Network using GPU Acceleration

Table of contents

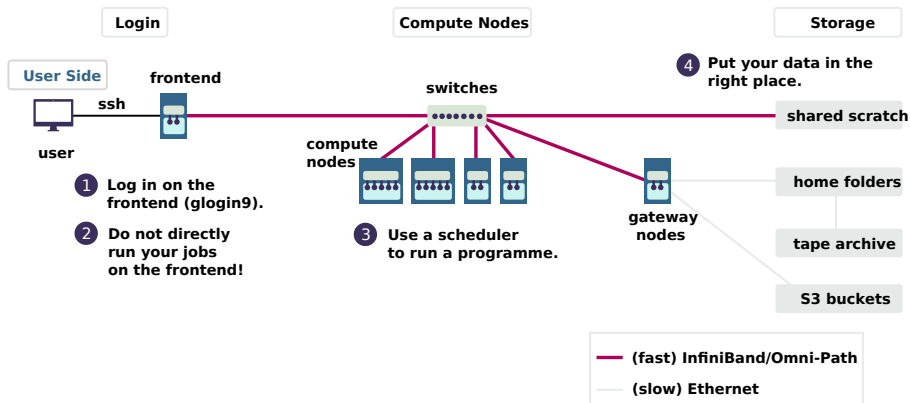
- 1 Access the Cluster
- 2 Job Submission: Slurm
- 3 Code walk through and Job Monitoring

Schedule

	Deep Learning with GPU cores
09.30 - 09.45	Welcome
09.45 - 10.15 (30 min)	Deep Learning and Infrastructure
10.15 - 11.30 (60 min)	Practical: Working on the GPU
11.30 - 11.45	Short break 
11.45 - 12.00 (15 min)	Introduction to Profiling
12.00 - 12.45 (45 min)	Practical: Profiling Jobs
12.45 - 13.00	General Q&A

<https://gitlab-ce.gwdg.de/hpc-team-public/deep-learning-with-gpu-cores>

SSH



```
none - -bash - 80x40  
Mac-User:~ none$ ssh -i .ssh/hlrn YOUR_USER_NAME@glogin9.hlrn.de
```

SSH

- More convenient: add following to your `/.ssh/config`

`$HOME/.ssh/config`

```
1 Host glogin9
2     Hostname glogin9.hlrn.de
3     IdentityFile ~/.ssh/hlrn
4     User YOUR_USER_NAME
5     ForwardAgent yes
```



none – -bash – 80×40

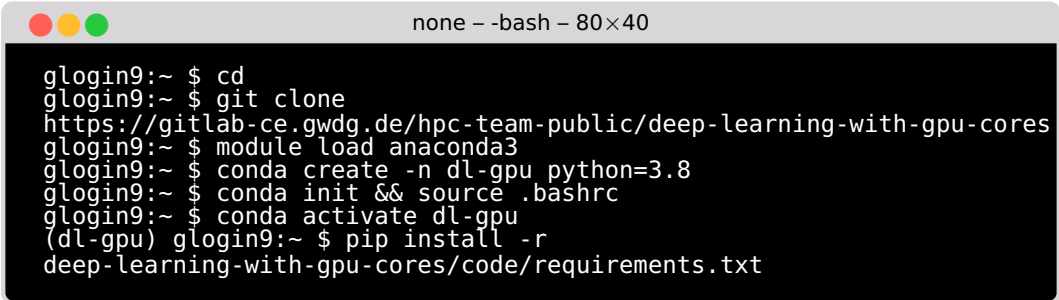
```
Mac-User:~ none$ ssh glogin9
```

Some useful Shell Commands

\$ cd <directory>	... change directory
\$ ls <directory>	... display files and directories
\$ cat <file>	... display the whole content of a file
\$ less <file>	... scroll through the content
\$ tail -f <file>	... display last lines and follow changes
\$ nano <file>	... edit a file
\$ pwd	... show current directory
\$ <command> --help	... provide information on how to use the command

Clone Git Repository and Prepare your Environment

All the necessary Python packages will be provided in the following conda environment:



```
glogin9:~ $ cd
glogin9:~ $ git clone
https://gitlab-ce.gwdg.de/hpc-team-public/deep-learning-with-gpu-cores
glogin9:~ $ module load anaconda3
glogin9:~ $ conda create -n dl-gpu python=3.8
glogin9:~ $ conda init && source .bashrc
glogin9:~ $ conda activate dl-gpu
(dl-gpu) glogin9:~ $ pip install -r
deep-learning-with-gpu-cores/code/requirements.txt
```

Additional Information in the README file:

<https://gitlab-ce.gwdg.de/hpc-team-public/deep-learning-with-gpu-cores>

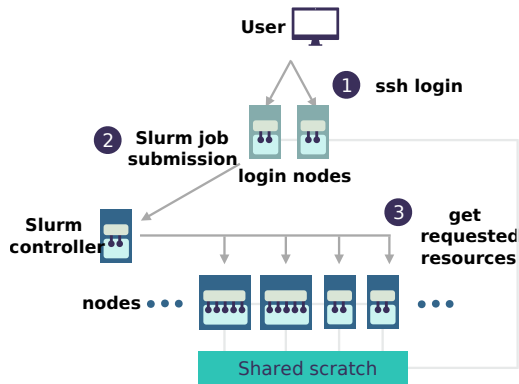
Access the Cluster

Let's get started.

Slurm

Slurm in 1 minute

Job Script Submission



Slurm job submission:

```
$ srun
```

```
$ sbatch
```

```
$ salloc
```

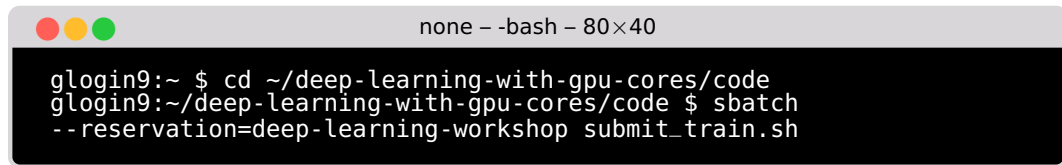
```
$ scancel
```

Some useful Slurm Commands

\$ squeue --me	... display your job postions in the queue
\$ sinfo -p <partiton>	... available partitions
\$ sacct	... info about your job
\$ srun	... run a scheduled job
\$ sbatch	... run a scheduled batch scrpted job
\$ salloc	... allocate a node
\$ scancel <job_id>	... cancel/stop a certain Slurm job
\$ <command> --help	... provide information on how to use the command

Start a Training

Start a training session:

A terminal window with a title bar that says "none - -bash - 80x40". The window has three colored window control buttons (red, yellow, green) on the left. The terminal text shows a user logging in as 'glogin9' and running a series of commands to start a training session using 'sbatch' with specific reservation and script parameters.

```
glogin9:~ $ cd ~/deep-learning-with-gpu-cores/code
glogin9:~/deep-learning-with-gpu-cores/code $ sbatch
--reservation=deep-learning-workshop submit_train.sh
```

You can check your submitted jobs via:

```
squeue - -me
```

PLEASE SUBMIT max 2 JOBS in parallel!

(scancel JOB_ID if you have to cancel your submitted jobs)

Batch Scripting using Slurm

Setting Slurm Parameters:

/code/submit_train.sh

```
1  #!/bin/bash
2  #SBATCH --job-name=train-nn-gpu
3  #SBATCH -t 05:00:00                # estimated time # TODO: adapt to your needs
4  #SBATCH -p grete:shared            # the partition you are training on (i.e., whi
5  #SBATCH -G A100:1                  # requesting GPU slices, see https://docs.hpc.
6  #SBATCH --nodes=1                  # total number of nodes
7  #SBATCH --ntasks=1                 # total number of tasks
8  #SBATCH --cpus-per-task 4          # number of CPU cores per task
9  #SBATCH --mail-type=all            # send mail when job begins and ends
10 #SBATCH --mail-user=username@gwdg.de # TODO: change this to your mailaddress!
11 #SBATCH --output=./slurm_files/slurm-%x-%j.out    # where to write output, %x give
12 #SBATCH --error=./slurm_files/slurm-%x-%j.err     # where to write slurm error
13
```

Batch Scripting using Slurm

Load your Environment:

/code/submit_train.sh

```
15 module load cuda
16 source activate dl-gpu # Or whatever you called your environment.
17
```

Batch Scripting using Slurm

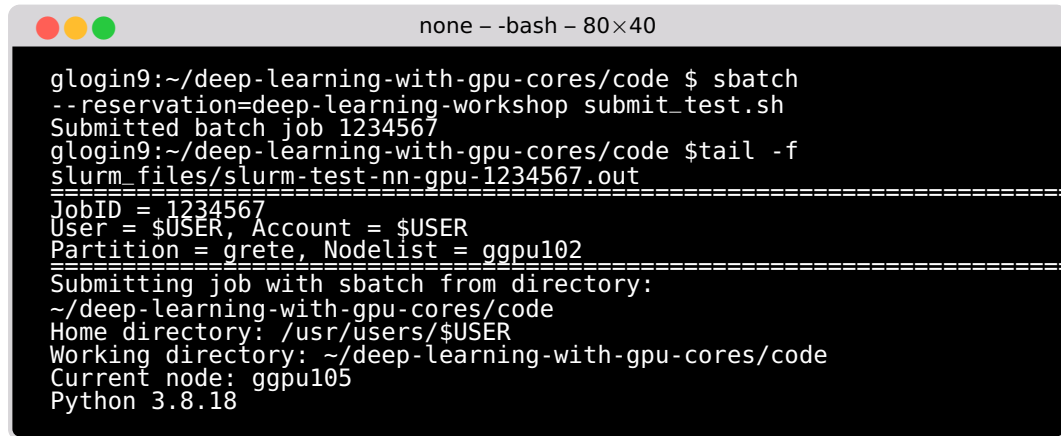
Debug and Execute your Code:

/code/submit_train.sh

```
19 echo "Submitting job with sbatch from directory: ${SLURM_SUBMIT_DIR}"
20 echo "Home directory: ${HOME}"
21 echo "Working directory: $PWD"
22 echo "Current node: ${SLURM_NODELIST}"
23
24 # For debugging purposes.
25 python --version
26 python -m torch.utils.collect_env
27 nvcc -V
28
29 # Run the script:
30 python -u train.py
31
```

Train and Test

Test a model and check the outputfile ('CTRL + C' to stop the viewer)



```
glogin9:~/deep-learning-with-gpu-cores/code $ sbatch
--reservation=deep-learning-workshop submit_test.sh
Submitted batch job 1234567
glogin9:~/deep-learning-with-gpu-cores/code $ tail -f
slurm_files/slurm-test-nn-gpu-1234567.out
=====
JobID = 1234567
User = $USER, Account = $USER
Partition = grete, Nodelist = ggpu102
=====
Submitting job with sbatch from directory:
~/deep-learning-with-gpu-cores/code
Home directory: /usr/users/$USER
Working directory: ~/deep-learning-with-gpu-cores/code
Current node: ggpu105
Python 3.8.18
```

Train and Test

Some fundamental vocabulary:

- accuracy = number of correct classification predictions divided by the total number of predictions
- one epoch = one forward pass and one backward pass of all the training examples
- batch size = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- number of iterations = number of passes, each pass using [batch size] number of examples.
- More on <https://developers.google.com/machine-learning/glossary>

Train and Test

Let's take a quick view on the train and test python scripts.

Monitoring: nvidia



none – -bash – 80×40

```
glogin9:~/deep-learning-with-gpu-cores/code $ sbatch
--reservation=deep-learning-workshop submit_train.sh
Submitted batch job 1234567
glogin9:~/deep-learning-with-gpu-cores/code $ squeue --me
###
JOBID   PARTITION  NAME USER ACCOUNT   STATE TIME NODES NODELIST
1234567 grete  train-nn-gpu $USER $USER  RUNNING 00:05 1  ggpu102
###
glogin9:~/deep-learning-with-gpu-cores/code $ ssh ggpu102
ggpu102:~ $ module load nvidia
ggpu102:~ $ nvidia
```

Monitoring: nvitop

Mon Apr 03 14:40:02 2023 (Press **h** for help or **q** to quit)

NVITOP 1.0.0				Driver Version: 530.30.02		CUDA Driver Version: 12.1	
GPU Fan Temp Perf Pwr:Usg/Cap		Memory-Usage		GPU-Util Compute M.			
0 N/A 22C P0 52W / 400W		2487MiB / 40.00GiB		N/A Default		MEM: <div><div></div></div> 6.1% UTL: <div><div></div></div> N/A	
0:0	1g.5gb @ GI/CI: 7/0	2412MiB / 4864MiB		BAR1: 2MiB / 0%		MEM: <div><div></div></div> 49.6%	
0:1	1g.5gb @ GI/CI: 8/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
0:2	1g.5gb @ GI/CI: 9/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
0:3	1g.5gb @ GI/CI:10/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
0:4	1g.5gb @ GI/CI:11/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
0:5	1g.5gb @ GI/CI:12/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
0:6	1g.5gb @ GI/CI:13/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1 N/A 20C P0 45W / 400W		88MiB / 40.00GiB		N/A Default		MEM: <div><div></div></div> 0.2% UTL: <div><div></div></div> N/A	
1:0	1g.5gb @ GI/CI: 7/0	12MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1:1	1g.5gb @ GI/CI: 8/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1:2	1g.5gb @ GI/CI: 9/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1:3	1g.5gb @ GI/CI:10/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1:4	1g.5gb @ GI/CI:11/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1:5	1g.5gb @ GI/CI:12/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
1:6	1g.5gb @ GI/CI:13/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2 N/A 20C P0 41W / 400W		88MiB / 40.00GiB		N/A Default		MEM: <div><div></div></div> 0.2% UTL: <div><div></div></div> N/A	
2:0	1g.5gb @ GI/CI: 7/0	12MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2:1	1g.5gb @ GI/CI: 8/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2:2	1g.5gb @ GI/CI: 9/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2:3	1g.5gb @ GI/CI:10/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2:4	1g.5gb @ GI/CI:11/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2:5	1g.5gb @ GI/CI:12/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
2:6	1g.5gb @ GI/CI:13/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3 N/A 21C P0 45W / 400W		88MiB / 40.00GiB		N/A Default		MEM: <div><div></div></div> 0.2% UTL: <div><div></div></div> N/A	
3:0	1g.5gb @ GI/CI: 7/0	12MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3:1	1g.5gb @ GI/CI: 8/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3:2	1g.5gb @ GI/CI: 9/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3:3	1g.5gb @ GI/CI:10/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3:4	1g.5gb @ GI/CI:11/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3:5	1g.5gb @ GI/CI:12/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	
3:6	1g.5gb @ GI/CI:13/0	13MiB / 4864MiB		BAR1: 64KiB / 0%		MEM: <div><div></div></div> 0.3%	

[CPU: 1.4%] (Load Average: 1.41 1.39 1.01)
[MEM: 2.2%] [SWP: 0.0%]
(Press ^C(INT)/T(TERM)/K(KILL) to send signals)

Processes:								gzadmtneis@ggpu103	
GPU	PID	USER	GPU-MEM	%SM	%CPU	%MEM	TIME	COMMAND	
0:0	8210	C gzadmt+	2392MiB	0	89.8	0.8	14:07	python -u train.py	

Coffee Break

	Deep Learning with GPU cores
09.30 - 09.45	Welcome
09.45 - 10.15 (30 min)	Deep Learning and Infrastructure
10.15 - 11.30 (60 min)	Practical: Working on the GPU
11.30 - 11.45	Short break ☕
11.45 - 12.00 (15 min)	Introduction to Profiling
12.00 - 12.45 (45 min)	Practical: Profiling Jobs
12.45 - 13.00	General Q&A