- Important Links:
  - judoor.fz-juelich.de - SSH Keys, Rechenzeitverbrauch
  - llview.fz-juelich.de - Auslastung, Warteschlange, Performance
  - https://apps.fz-juelich.de/jsc/hps/jureca/index.html

HPC Access

- Per SSH username@jureca.fz-juelich.de (per putty oder WSL)
- Add SSH-Key in Judoor
  - From Clause: Aktueller IP-Adresse oder die IP-Adress-Range des Providers abgreifen
  - FZJ-VPN?

Setup an Environment

- Most Stable: Docker
- Fastest: module load + python virtualenv or module load + conda
- /p/project/cjinm17/pierschke1 <- Software und Environments

```
$ cat load.sh
module load Stages/2023
module load GCC
module load CUDA
module load ParaStationMPI
module load HDF5
module load LibTIFF/.4.3.0
module load Python/3.10.4
module load PyCUDA/2022.1
module load mpi4py
```

Run A Job

- srun - Schickt einen Job ab (im Vordergrund, die Verbindung muss aktiv bleiben)
- salloc - Allokiert eine gewissen Rechenkapazität und gibt eine interaktive Shell zurück
- sbatch - Schickt einen Job oder mehrere im Hintergrund ab

https://pytorch.org/tutorials/intermediate/ddp_tutorial.html

```
✔ jrlogin08 kropp1 /p/fastdata/bigbrains/projects/cellgeneration/experiments/flat_dim1024_steps250_res1_global 0.378s
$ cat train.sh
#! /bin/sh
#SBATCH --account=cjinm17
#SBATCH --partition=dc-gpu
##SBATCH --account=hai_fzj_bda
##SBATCH --partition=booster
#SBATCH --nodes=16
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4
#SBATCH --time=02:00:00


export MODEL_FLAGS="--image_size 1024 --num_channels 64 --attention_resolutions 32,16,8 --learn_sigma=True --use_fp16=True --num_heads 4 --num_res_blocks 2 --re
export DIFFUSION_FLAGS="--diffusion_steps 256 --noise_schedule cosine"
export TRAIN_FLAGS="--lr 1e-5 --batch_size 2 --resume_checkpoint ./model088000.pt"
export OPENAI_LOGDIR="."

srun python3 ../../code/celldiffusion/scripts/training/image_train.py --data_dir ../../data/global/flat/1024/training $MODEL_FLAGS $DIFFUSION_FLAGS $TRAIN_FLAGS
```

Rechenzeit - Kontingent

- Jede Node hat 4 GPUs und 512 GB Ram
  - Idealerweise benutzt du alles, Datensatz kann wahrscheinlich vollständig in den RAM, dann spart man sich I/O
- 128 * Anzahl Nodes * Stunden
  - Beispiel: 4 Nodes für 3 Stunden -> 1,536 Core-h
- Ist das Rechenzeitprojekt von Eric und mir
- Cap bei 20,000 Core-h pro Monat