GPU Course Cheat Sheet

Workstation

Logging in

Username: `train0XX`
Password: *See slip of paper*

Execution

We are doing most of the work in the command line. Start a terminal window from your desktop using `ALT+F2` and entering `konsole`

Editing

Vim can be launched in directly in a shell. Tip: Open individual shell windows for editing, compilation, and running!
kate provides remote editing. Use `sftp://jureca/` to access your files via JURECA and edit them locally.
Alternative: Mount your remote home directory via `fish://jureca` typed into Dolphin application

Some Commands

`cd` *`dir`* Changes working directory to *`dir`*
`ls` Lists files in the current directory
`ls -l` Like above, but gives more detail
`mkdir` *`dir`* Creates a new subdirectory named *`dir`*
`rm` *`file`* Removes file *`file`* (Can not be undone!)
`less` *`file`* Shows the content of *`file`*

Supercomputers

We will be working on **JURECA**.

Logging In

- Start SSH agent: ``eval `ssh-agent` ``
- Add SSH key to agent: `ssh-add` → enter password
- Login: `ssh jureca.fz-juelich.de`

Environment

JURECA uses a module system to provide different software. All required modules are already loaded into your environment. List available modules with `module avail`
On JURECA, CUDA can be loaded with
`module load CUDA`
The PGI compilers can be loaded with
`module load PGI`

JURECA Compute Nodes

On JURECA, only certain compute nodes are equipped with GPUs. Allocate of resources on one of the compute nodes is done with a string like the following:

```
salloc --reservation=openacc1 -p gpus
--nodes=1 --gres=gpu:4 --time=8:0:0
[--cpus-per-task=4]
```

A default allocation string is saved into an environment variable; see `echo $JSC_SUBMIT_CMD`.
Run your program on a GPU node with `srun` *`prog`*
Open an interactive Bash shell with
`srun --pty /bin/bash -i`

OpenACC

Parallel Constructs

```
#pragma acc parallel
#pragma acc kernels

#pragma acc loop
#pragma acc reduction(+:x)
```

Data Regions

```
#pragma acc data
#pragma acc enter data
```

See `www.openacc.org/specification/`

MPI

```
int MPI_Init(int *argc, char ***argv)
int MPI_Comm_rank(MPI_Comm comm, int *rank)
int MPI_Comm_size(MPI_Comm comm, int *size)

int MPI_Sendrecv(
void *sendbuf, int sendcount,
↪  MPI_Datatype sendtype, int dest,
↪  int sendtag,
void *recvbuf, int recvcount,
↪  MPI_Datatype recvtype, int source,
↪  int recvtag,
MPI_Comm comm, MPI_Status *status
)
```

See `www.open-mpi.org/doc/`