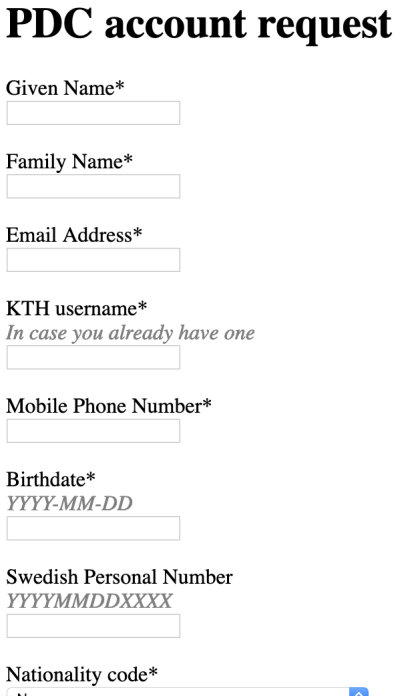
# Hands-on:
# AMD GPU on Dardel supercomputer

# Connect to Dardel

If not yet done, to apply for a PDC user account, you need to fill out the form at:

https://pdc-web.eecs.kth.se/accounts/

For this hands-on session, you will be using project account ‘edu23.summer’ for job allocation



**PDC account request**

Given Name*

Family Name*

Email Address*

KTH username*
*In case you already have one*

Mobile Phone Number*

Birthdate*
*YYYY-MM-DD*

Swedish Personal Number
*YYYYMMDDXXXX*

Nationality code*

# Connect to Dardel - Kerberos

Before connecting to Dardel, the local environment of your laptop must be configured.

Detailed instructions can be found on https://www.pdc.kth.se/support/documents/login/configuration.html.

•If you are using **Mac,** you need to follow a special set of instructions here: https://www.pdc.kth.se/support/documents/login/mac_login.html.

•If you are using **Windows**, we recommend using the WSL2 to use a fully-fledged Ubuntu: https://www.pdc.kth.se/support/documents/login/windows_login.html#wsl-approach.


To configure Kerberos on your local machine, you need to do the following steps:

1) Create .ssh folder if it does not already exist in your home folder.

2) Create a file called krb5.conf in .ssh with the following content

# Connect to Dardel - SSH

1. See https://www.pdc.kth.se/support/documents/login/configuration.html
2. Create a file in .ssh called config with the following content

```
# Hosts we want to authenticate to with Kerberos
Host *.kth.se *.kth.se.
# User authentication based on GSSAPI is allowed
GSSAPIAuthentication yes
# Key exchange based on GSSAPI may be used for server authentication
GSSAPIKeyExchange yes
# Hosts to which we want to delegate credentials. Try to limit this to
# hosts you trust, and were you really have use for forwarded tickets.
Host *.csc.kth.se *.csc.kth.se. *.nada.kth.se *.nada.kth.se. *.pdc.kth.se *.pdc.kth.se.
# Forward (delegate) credentials (tickets) to the server.
GSSAPIDelegateCredentials yes
# Prefer GSSAPI key exchange
PreferredAuthentications gssapi-keyex,gssapi-with-mic
# All other hosts
Host *
```

2. Set the correct permission on the file

```
$ chmod 644 ~/.ssh/config
```

3. Connect

```
$ ssh YourUsername@dardel.pdc.kth.se
```

# A Supercomputer

Kerberos
+ `ssh`

Login Node

Login Node

slurm workload manager

slurm workload manager

Comp. Node

Comp. Node

Comp. Node

Comp. Node

Comp. Node

Comp. Node

Comp. Node

Comp. Node

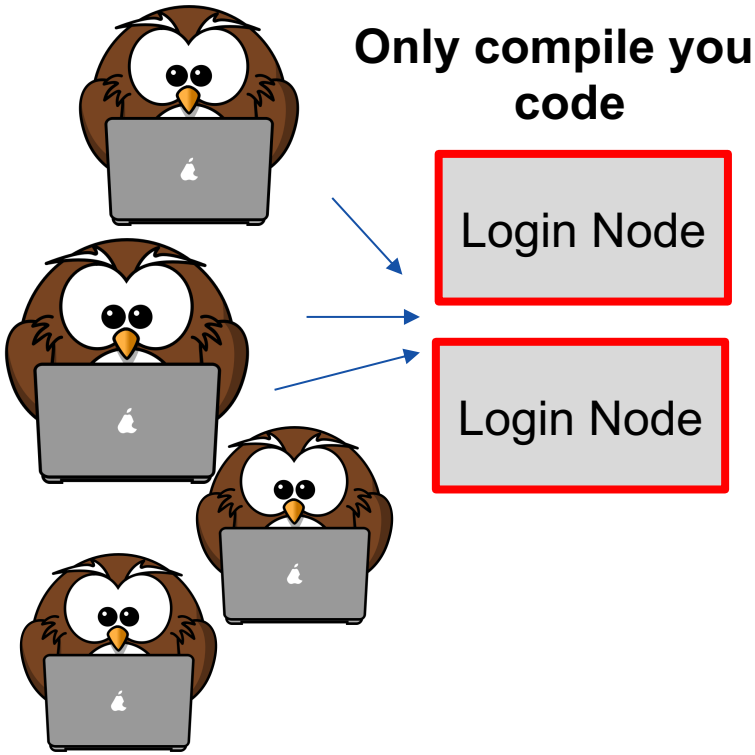
Comp. Node

Comp. Node

Comp. Node

Comp. Node

# Do not run your simulation on the Login Node!




- When connecting to a supercomputer, you are connecting to the **login node**.
- Login nodes are **shared nodes across all users**.
- You can use the `top` command to check who is connected to the login node.
- Use the login nodes only for compiling your code.
  - **Never use the login nodes for running your executable**
  - We use the scheduler slurm to launch the executable on the computing nodes

## Use Slurm to Launch Your Simulation on Computing Nodes

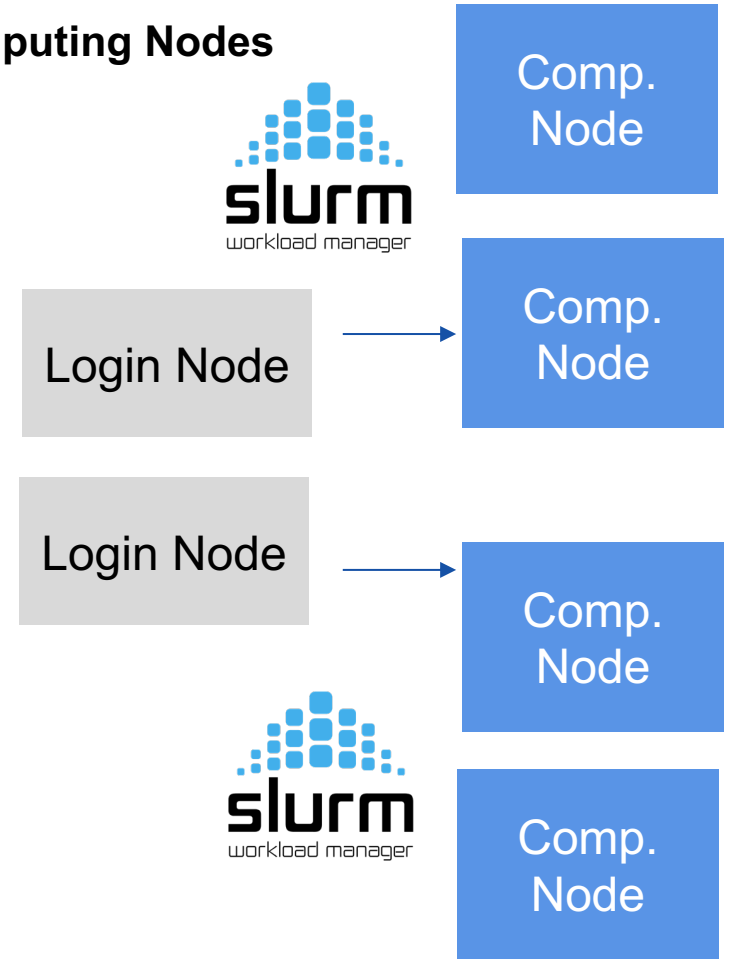
- Slurm allows us to launch our simulation on the computing nodes
- Two ways:
  - **Interactive mode:** ask for a resource, and as soon it becomes available, we can launch your simulations interactively

```
$ salloc --nodes=1 -t 00:30:00 -A edu23.summer -p shared --ntasks-per-node=1 --cpus-per-task=1
```

```
$ salloc --nodes=1 -t 00:30:00 -A edu23.summer -p gpu --ntasks-per-node=1 --cpus-per-task=1
```

```
$ srun -n 1 ./hello.out
```

  - **Batch mode:** put your simulation into a queue



# AMD GPU – Exercise 1

Query your AMD GPU on the Dardel supercomputer

- Request a job in the GPU partition in interactive mode

```
$ salloc --nodes=1 -t 00:30:00 -A edu23.summer -p gpu --ntasks-per-node=1 --cpus-per-task=1
```

- Run the *rocm-smi* command

```
> srun -n 1 rocm-smi
======================= ROCm System Management Interface =======================
================================== Concise Info ==================================
GPU  Temp   AvgPwr  SCLK    MCLK     Fan  Perf  PwrCap  VRAM%  GPU%
0    27.0c  91.0W   800Mhz  1600Mhz  0%   auto  560.0W    0%   0%
1    28.0c  N/A     800Mhz  1600Mhz  0%   auto  0.0W      0%   0%
```

**Question**: how many GPUs do you see?

# AMD GPU – Exercise 2

- Compile a simple HIP code
- Go to the folder ./1_helloWorld

```
> ls
HelloWorld.cpp Makefile
```

- Compile the code with

```
1_helloWorld> make
hipcc --offload-arch=gfx90a   -c -o HelloWorld.o HelloWorld.cpp
hipcc HelloWorld.o -o HelloWorld
```

- Run on AMD GPU in interactive mode

**Question**: what the output string do you see?

# AMD GPU – Exercise 3

- Compile a simple 1D vectorAdd HIP code
- Go to the folder ./2_vectorAdd

```
2_vectorAdd > ls
Makefile  vectoradd_hip.cpp
```

- Compile the code with

```
2_vectorAdd> make
hipcc --offload-arch=gfx90a   -c -o vectoradd_hip.o vectoradd_hip.cpp
hipcc vectoradd_hip.o -o vectoradd_hip.exe
```

- Run on AMD GPU in interactive mode

**Question**: what output message do you see?

# Convert a CUDA code into HIP code

**Hipify** our vectorAdd.cu example in yesterday's hands-on exercise

- Ensure rocm/5.3.3 module is loaded

```
/3_hipify> module load rocm/5.3.3
```

- Call hipify script

```
/3_hipify> hipify-perl -print-stats vectorAdd.cu > vectorAdd.cpp

[HIPIFY] info: file 'vectorAdd.cu' statisitics:
  CONVERTED refs count: 14
  TOTAL lines of code: 100
  WARNINGS: 0
[HIPIFY] info: CONVERTED refs by names:
  cudaDeviceSynchronize => hipDeviceSynchronize: 1
  cudaFree => hipFree: 3
  cudaMalloc => hipMalloc: 3
  cudaMemcpy => hipMemcpy: 3
  cudaMemcpyDeviceToHost => hipMemcpyDeviceToHost: 1
```

# Convert a CUDA code into HIP code

- Compile your hipified code

```
> hipcc --offload-arch=gfx90a -o vectoradd vectorAdd.cpp
```

- Run on AMD GPU in interactive mode

```
> srun -n 1 ./vectoradd 8192
The input length is 8192
valid
```

# Reduction Operation

The example HIP code takes a user input array length N and sum up their values

- Compile the code using hipcc

```
4_reduction> make
hipcc -std=c++11 -O3 --offload-arch=gfx90a -o reduction reduction.cpp
```

- Run on AMD GPU in interactive mode

```
/4_reduction> srun -n 1 ./reduction
Usage: ./reduction num_of_elems
using default value: 52428800
ARRAYSIZE: 52428800
Array size: 200 MB
The average performance of reduction is 374.525 GBytes/sec
VERIFICATION: result is CORRECT
```

# Reduction Operation – Compare different kernels

Use the run.sh batch script to test reduction of different array length

```
4_reduction> bash run.sh
./reduction 1024*1024*4
ARRAYSIZE: 1024
Array size: 0.00390625 MB
The average performance of reduction is 0.0488957 GBytes/sec
VERIFICATION: result is CORRECT

./reduction 8388608
ARRAYSIZE: 8388608
Array size: 32 MB
The average performance of reduction is 83.1296 GBytes/sec
VERIFICATION: result is CORRECT
```

# Reduction Operation – Exercise

Switch between the three reduction kernels,
*atomic_reduction_kernel(),*

*atomic_reduction_kernel2(),*

*atomic_reduction_kernel3()*

- What performance difference do you observe in them?
- Try with different input length N

# Reduction Operation – Exercise

For an array of length N:

- How many atomic operations are used in `atomic_reduction_kernel()` ?

# Q & A