Introduction to HPC2N

Birgitte Brydsø, Pedro Ojeda-May

HPC2N, Umeå University

8 September 2021

UMEÅ UNIVERSITY

Overview

- Projects - compute and storage
- Using our systems
- The File System
- The Module System
  - Overview
  - Compiler Tool Chains
  - Examples
- Compiling/linking with libraries
- The Batch System (SLURM)
  - Overview
  - Simple examples

Projects - compute and storage

Join a project or apply for one.

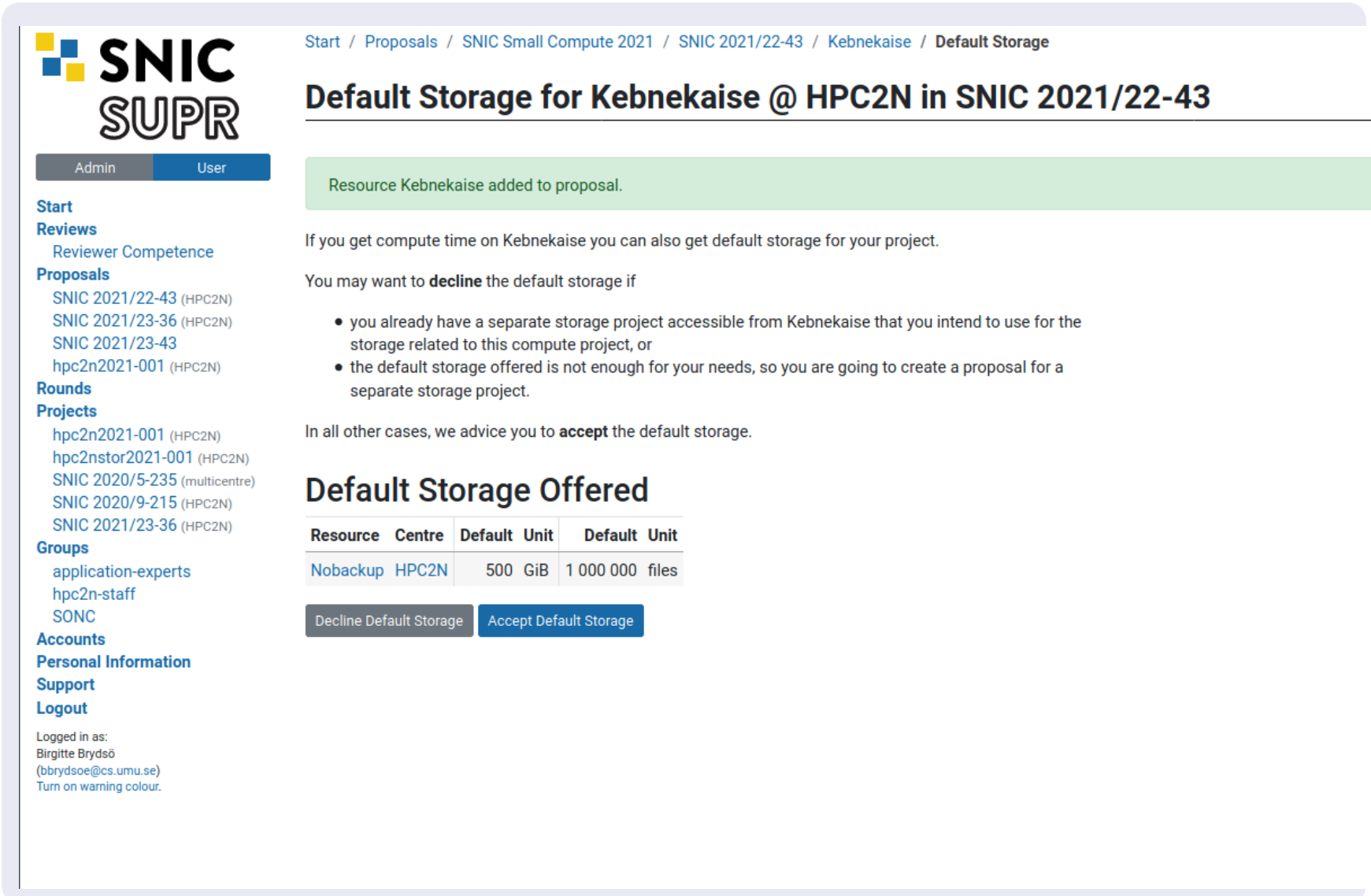
- Apply for a **compute project** in SUPR
  `https://supr.snic.se/round/compute/`
  - Small ($\leq$ 5000 core-h/month, at least PhD student to apply)
  - Medium (monthly rounds, at least assistant professor to apply)
  - Large (bi-annual rounds)
  - When applying for compute projects, you can apply for default storage (small is 500GB)
  - More information:
    `https://snic.se/allocations/compute/`
- If the above mentioned default storage is not enough, you will need to apply for a **storage project**
  `https://supr.snic.se/round/storage/`
  - Small (up to 2 TB, at least PhD student to apply)
  - Medium (monthly rounds)
  - Large (bi-annual rounds)
  - More information:
    `https://snic.se/allocations/storage/`

Projects - compute and storage

- As default, you have 25GB in your home directory.
- If you need more, you can accept the “default storage” that you will be offered after applying for compute resources. them together. It is done from the **storage** project.
- The default storage is 500GB.
- If you need more than that, you will have to apply for a storage project

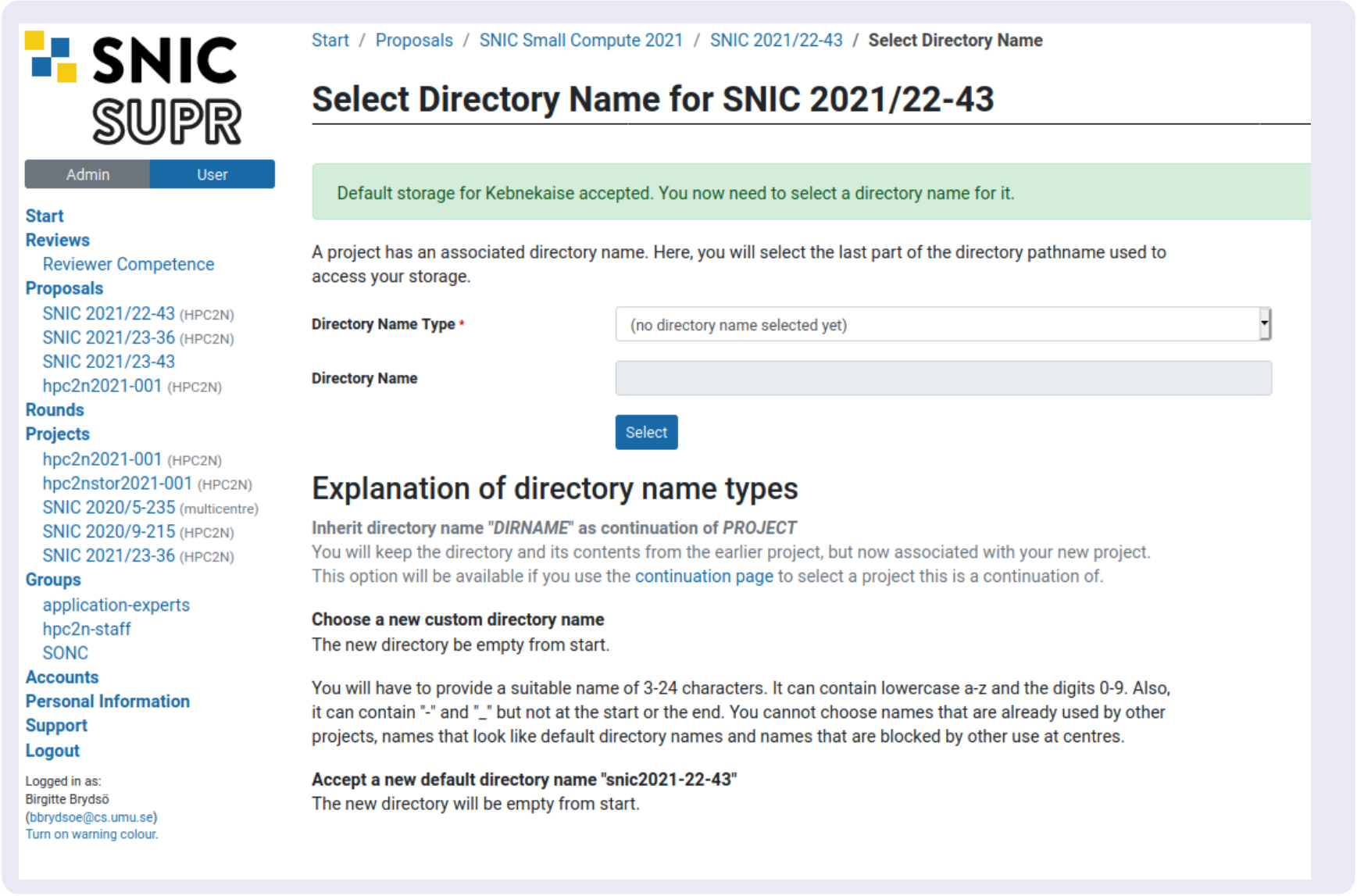
Projects - compute and storage

After indicating the amount of compute resources you want, you will be asked if you want the default amount of storage:



Projects - compute and storage

You will also be asked what the default storage directory should be called:



Projects - compute and storage

- After applying on SUPR, the project(s) will be reviewed.
- When (if) the projects are approved, the PI needs to link the compute and storage projects together if they applied for both.
- Linking them together is done from the **storage** project.
- This way all members of the compute project also becomes members of the storage project.

Projects - compute and storage

Linking a compute project to a storage project:

Membership in this project can also be requested using the Request Membership in Project function. The Principal Investigator and the proxy will then get an email with a link in it, that is used to approve or deny the membership request.

Compute projects linked to this storage project

Members of linked compute projects that are active become extended members of this storage project and can access its storage.

Search for Compute Project to Link

Resources

Allocation shows the current allocation.

Storage

Percentage field to the right, compares **Usage** with the allocation. **Last Updated** shows the time at which the usage was last updated.

Resource	Allocation	Usage	Unit	Percentage	Allocation	Usage	Unit	Percentage	Last Updated
Nobackup @ HPC2N	2 000		GiB		999 999		files		

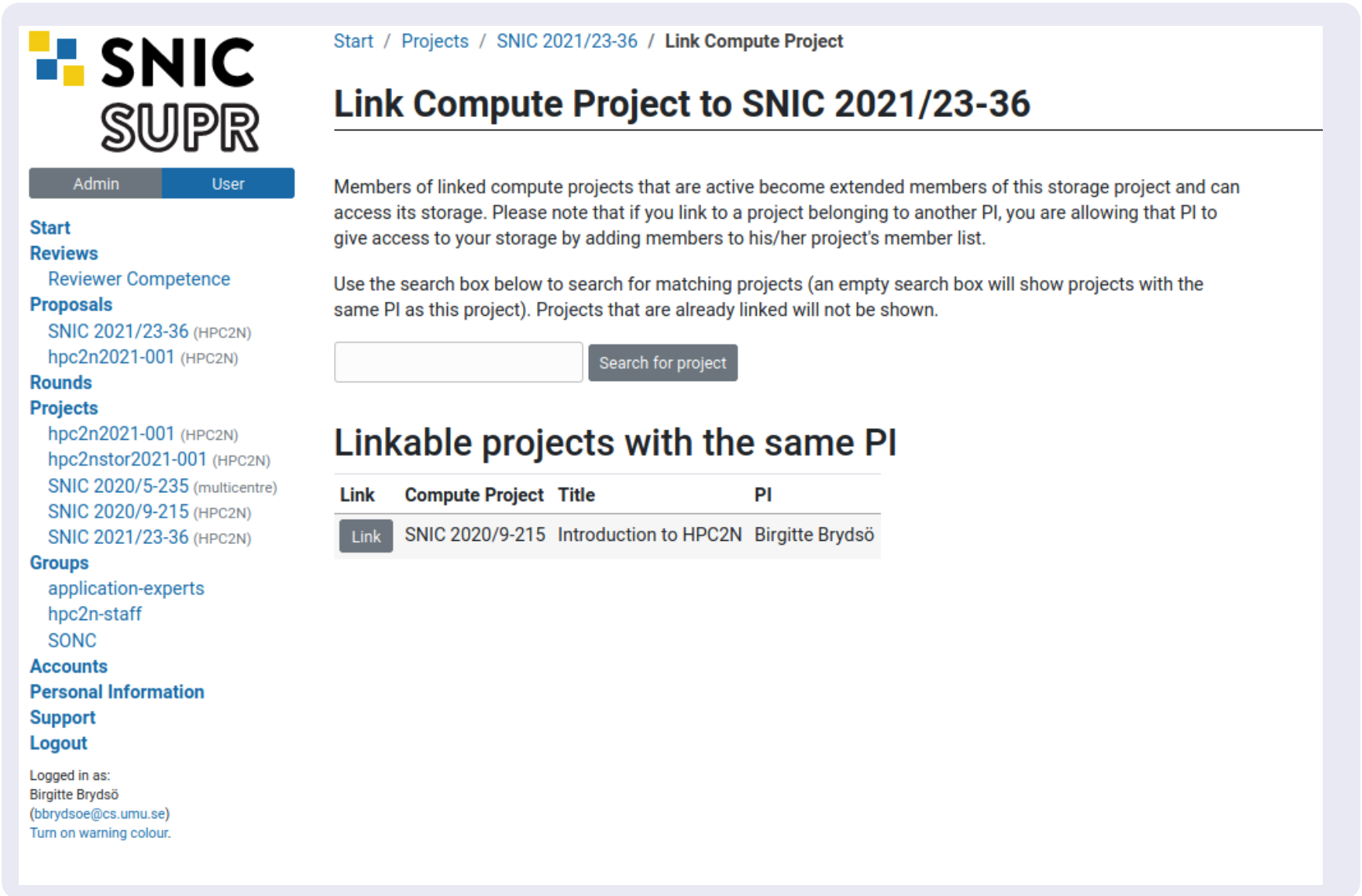
Resource Allocation and Usage

Storage

Show Storage Allocation and Usage on Nobackup @ HPC2N

Projects - compute and storage

Pick a compute project to link:



Projects - compute and storage

Showing linked projects:

Compute projects linked to this storage project

Members of linked compute projects that are active become extended members of this storage project and can access its storage.

Compute Project	Title	PI	Remove
SNIC 2020/9-215	Introduction to HPC2N	Birgitte Brydsö	Remove

Search for Compute Project to Link

Resources

Allocation shows the current allocation.

Storage

Percentage field to the right, compares **Usage** with the allocation. **Last Updated** shows the time at which the usage was last updated.

Resource	Allocation	Usage	Unit	Percentage	Allocation	Usage	Unit	Percentage	Last Updated
Nobackup @ HPC2N	2 000		GiB		999 999		files		

Projects - compute and storage

Members of the storage project after linking:

Project Members

These persons are direct members of this project and do not depend on linked compute projects for their access to the storage.

Name	Email	Organization / Department	Remove
Birgitte Brydsö	bbrydsoe@cs.umu.se	Umeå universitet / HPC2N	*PI cannot be removed*
Mirko Myllykoski	mirkom@cs.umu.se	Umeå universitet / Department of Computing Science and HPC2N	Remove
Pedro Ojeda	pedro.ojeda-may@umu.se	Umeå universitet / High Performance Computing Center North (HPC2N)	Remove

Add Member | Add Members from Other Project

Membership in this project can also be requested using the Request Membership in Project function. The Principal Investigator and the proxy will then get an email with a link in it, that is used to approve or deny the membership request.

Extended Project Members

These persons are members of this project or of active compute projects linked to this storage project. All extended members can access the storage belonging to this storage project.

Name	Email	Organization / Department

Using our systems

1. Connect to:

   `kebnekaise.hpc2n.umu.se`

2. Transfer your files and data (optionally)
3. Load modules (if needed)
4. Compile own code, install software, or run pre-installed software
5. Create batch script, submit batch job
6. Download data/results

Using our systems

Connecting to HPC2N's systems - ThinLinc

ThinLinc: a cross-platform remote desktop server from Cendio AB. Especially useful when you need software with a graphical interface.

- We recommend ThinLinc if you don't have a preferred SSH client.
- Download the client from `https://www.cendio.com/thinlinc/download`. Install it.
- Start the client. Enter the name of the server: **kebnekaise-tl.hpc2n.umu.se**. Enter your username.
- Go to "Options" $->$ "Security". Check that authentication method is set to password.
- Go to "Options" $->$ "Screen". Uncheck "Full screen mode".
- Enter your HPC2N password. Click "Connect"
- Click "Continue" when you are being told that the server's host key is not in the registry. Wait for the ThinLinc desktop to open.

Using our systems

Transfer your files and data

- **Linux, OS X:**
  - Use scp for file transfer:

```
local> scp username@kebnekaise.hpc2n.umu.se:file .
local> scp file username@kebnekaise.hpc2n.umu.se:file
```

- **Windows:**
  - Download client: WinSCP, FileZilla (sftp), PSCP/PSFTP, ...
  - Transfer with sftp or scp
- https://www.hpc2n.umu.se/documentation/filesystems/filetransfer
- **Mac/OSX:**
  - Transfer with sftp or scp (as for Linux) using Terminal
  - Or download client: Cyberduck, Fetch, ...
- More info in guides (see previous slide) and here:

  https://www.hpc2n.umu.se/documentation/filesystems/filetransfer

Editing your files

- Various editors: vi, vim, nano, emacs ...
- Example, nano:
  - `nano <filename>`
  - Save and exit nano: `Ctrl-x`
- Example, Emacs:
  - Start with: emacs
  - Open (or create) file: Ctrl-x Ctrl-f
  - Save: Ctrl-x Ctrl-s
  - Exit Emacs: Ctrl-x Ctrl-c
  - (If you want to run in an a separate emacs window, and with full functionality, you need to login with ThinLinc, ssh -Y or similar, for X11 forwarding)

The File System

More info here: http://www.hpc2n.umu.se/filesystems/overview

	Project storage	**$HOME**	**/scratch**
Recommended for batch jobs	Yes	No	Yes
Backed up	No	Yes	No
Accessible by batch system	Yes	Yes	Yes (node only)
Performance	High	High	Medium
Default readability	Group only	Owner	Owner
Permissions management	chmod, chgrp, ACL	chmod, chgrp, ACL	N/A for batch jobs
Notes	Storage your group get allocated through the storage projects	Your home-directory	Per node

Using project storage

- If you have a storage project, you should use that to run your jobs.
- You will either choose a directory name when you apply for the storage project or get the SNIC id as default name.
- The location of the storage project in the file system is `/proj/nobackup/<name-you-picked>`.
- Since the storage project is shared between all users of the project, you should go to that directory and create a subdirectory for your things, which you will then be using.

The Module System (Lmod)

Most programs are accessed by first loading them as a 'module'

Modules are

- used to set up your environment (paths to executables, libraries, etc.) for using a particular (set of) software package(s)
- a tool to help users manage their Unix/Linux shell environment, allowing groups of related environment-variable settings to be made or removed dynamically
- allows having multiple versions of a program or package available by just loading the proper module
- are installed in a hierarchial layout. This means that some modules are only available after loading a specific compiler and/or MPI version.

The Module System (Lmod)

Useful commands (Lmod)

- See which modules exists:
  `module spider` or `ml spider`
- Modules depending only on what is currently loaded:
  `module avail` or `ml av`
- See which modules are currently loaded:
  `module list` or `ml`
- Example: loading a compiler toolchain, here for GCC:
  `module load foss/version` or `ml foss/version`
- Example: Unload the above module:
  `module unload foss` or `ml -foss`
- More information about a module:
  `ml show <module>` or `module show <module>`
- Unload all modules except the 'sticky' modules:
  `ml purge`

The Module System

Compiler Toolchains

Compiler toolchains load bundles of software making up a complete environment for compiling/using a specific prebuilt software. Includes some/all of: compiler suite, MPI, BLAS, LAPACK, ScaLapack, FFTW, CUDA.

- Some currently available toolchains (check `ml av` for versions and full, updated list):
  - **GCC**: GCC only
  - **gcccuda**: GCC and CUDA
  - **foss**: GCC, OpenMPI, OpenBLAS/LAPACK, FFTW, ScaLAPACK
  - **gimkl**: GCC, IntelMPI, IntelMKL
  - **gimpi**: GCC, IntelMPI
  - **gompi**: GCC, OpenMPI
  - **gompic**: GCC, OpenMPI, CUDA
  - **goolfc**: gompic, OpenBLAS/LAPACK, FFTW, ScaLAPACK
  - **icc**: Intel C and C++ only
  - **iccifort**: icc, ifort
  - **iccifortcuda**: icc, ifort, CUDA
  - **ifort**: Intel Fortran compiler only
  - **iimpi**: icc, ifort, IntelMPI
  - **intel**: icc, ifort, IntelMPI, IntelMKL
  - **intelcuda**: intel and CUDA
  - **iomkl**: icc, ifort, Intel MKL, OpenMPI
  - **pomkl**: PGI C, C++, and Fortran compilers, IntelMPI
  - **pompi**: PGI C, C++, and Fortran compilers, OpenMPI

Compiling and Linking with Libraries

Linking

Figuring out how to link

- Intel and Intel MKL linking:

  `https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor`

- GCC, etc. **Use buildenv**
  - After loading a compiler toolchain, load `'buildenv'` and use `'ml show buildenv'` to get useful linking info
  - Example, foss (add relevant version):

    ```
    ml foss/version
    ml buildenv
    ml show buildenv
    ```

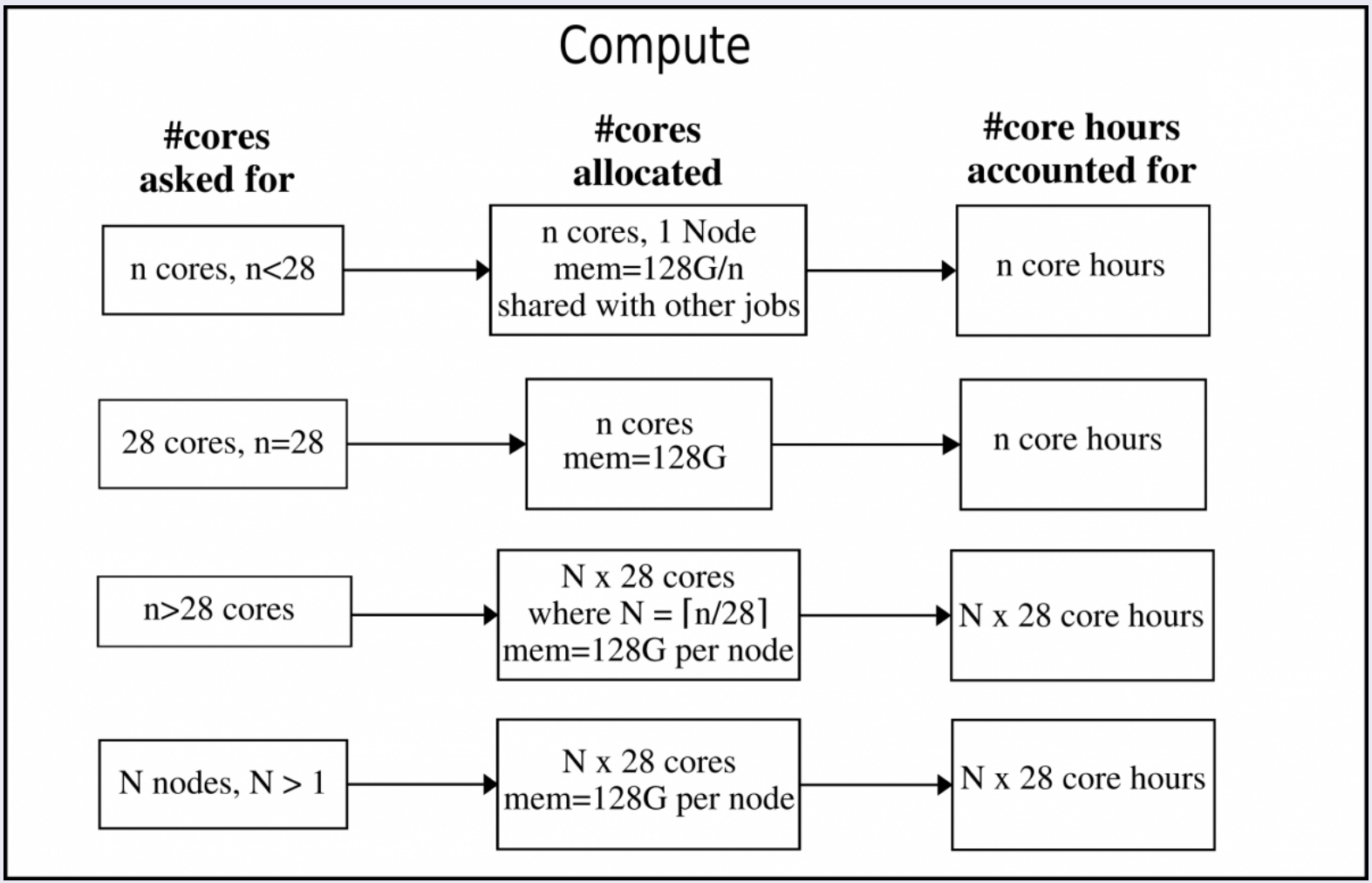
  - Using the environment variable (prefaced with $) for linking is highly recommended!
  - You have to load the buildenv module in order to use the environment variable for linking!

The Batch System (SLURM)

- Large/long/parallel jobs **must** be run through the batch system
- SLURM is an Open Source job scheduler, which provides three key functions
  - Keeps track of available system resources
  - Enforces local system resource usage and job scheduling policies
  - Manages a job queue, distributing work across resources according to policies
- In order to run a batch job, you need to create and submit a SLURM submit file (also called a batch submit file, a batch script, or a job script).
- Guides and documentation at: http://www.hpc2n.umu.se/support

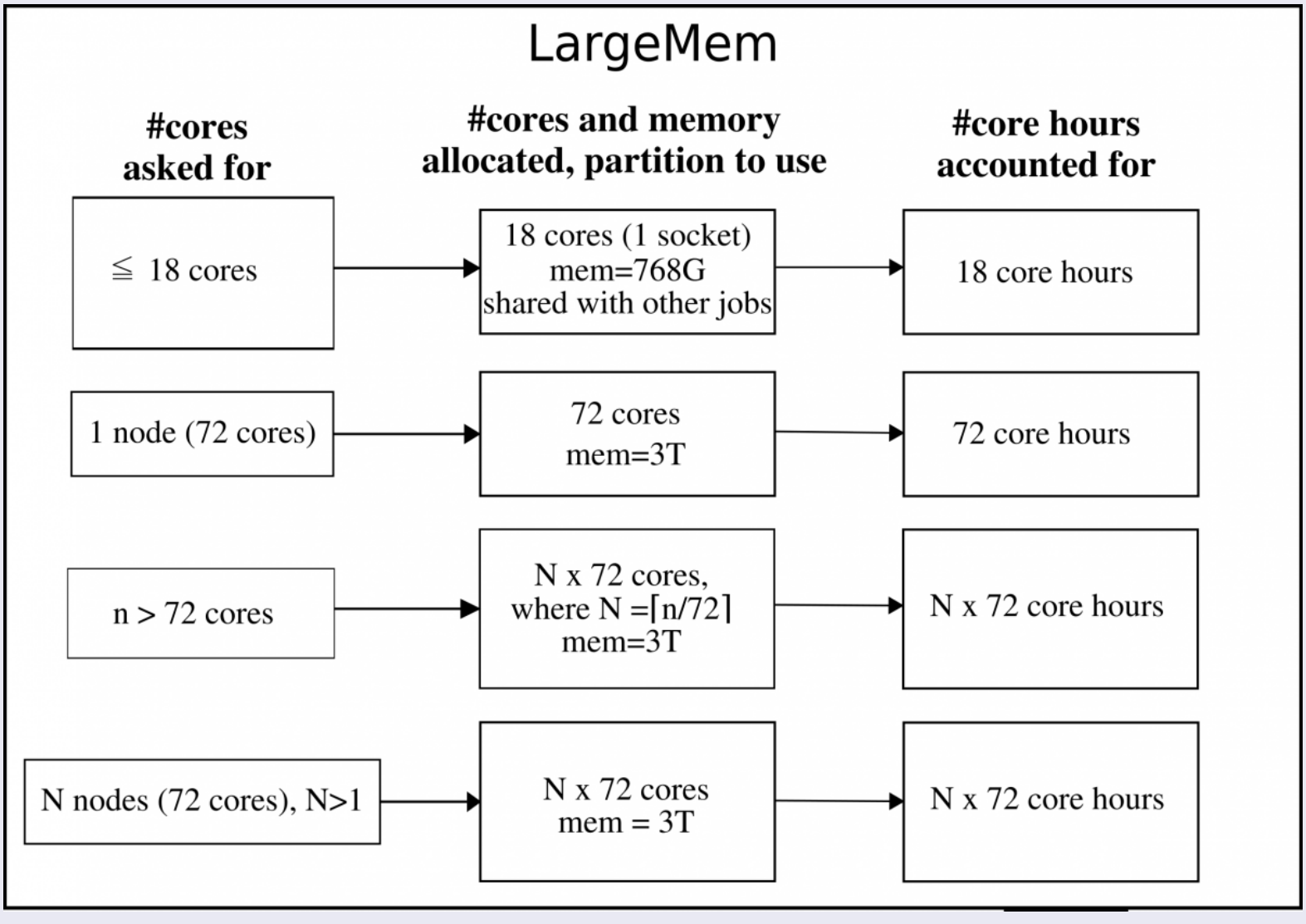
The Batch System

Accounting, Compute nodes, Kebnekaise



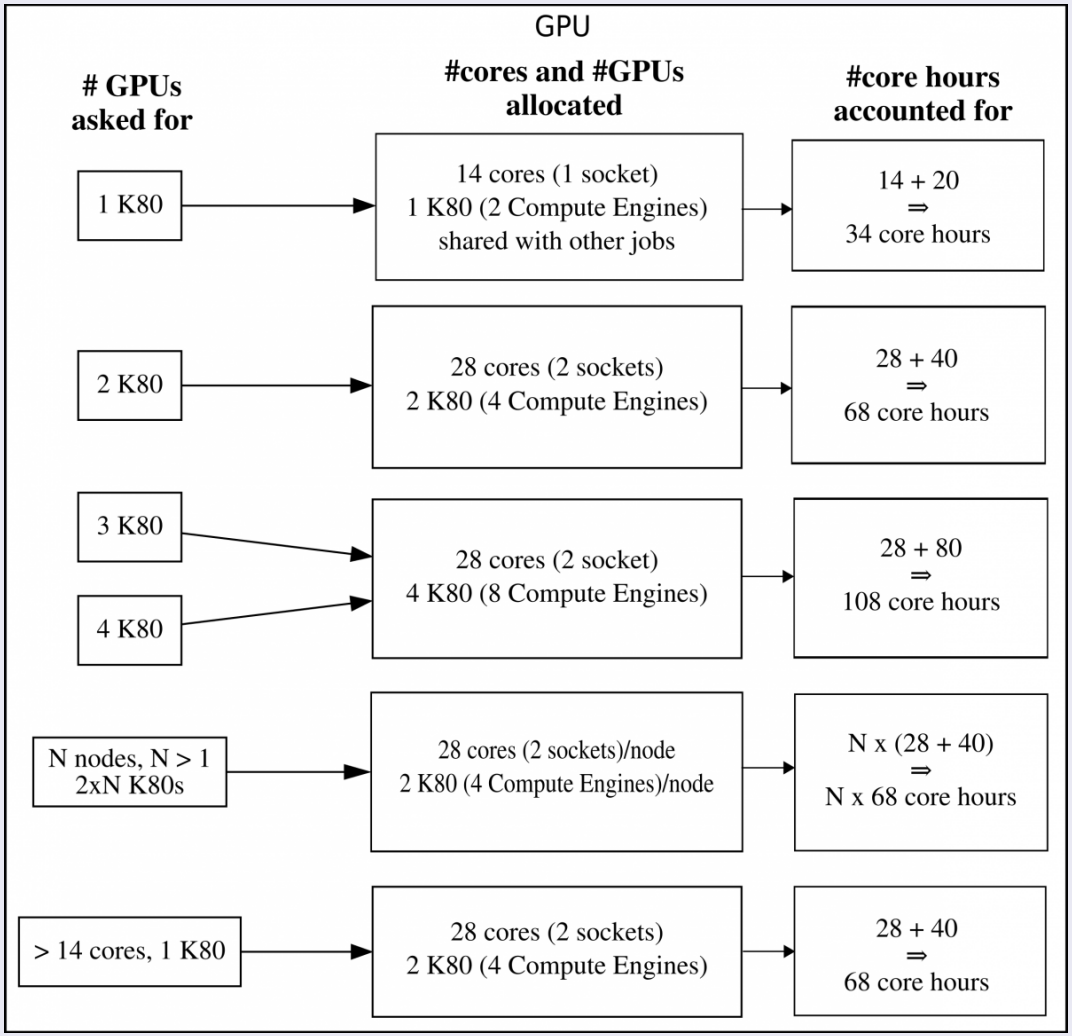
The Batch System

Accounting, largemem nodes, Kebnekaise



The Batch System

Accounting, GPU nodes, Kebnekaise. Same for the V100 as for the K80.



Note: V100s accounts like K80s and have **one** engine per card.

The Batch System (SLURM)

Useful Commands

- Submit job: `sbatch <jobscript>`
- Get list of your jobs: `squeue -u <username>`
- `srun <commands for your job/program>`
- Check on a specific job: `scontrol show job <job id>`
- Delete a specific job: `scancel <job id>`
- More detailed info about jobs:
  `sacct -l -j <jobid> -o jobname,NTasks,nodelist,MaxRSS,MaxVMSize...`
  - More flags can be found with `man sacct`
  - The output will be **very** wide. Use something like
    `sacct -l -j ..... | less -S`
    to view (makes it sideways scrollable, using the left/right arrow key)

Use `man sbatch`, `man srun`, `man ....` for more information

The Batch System (SLURM)

Job Output

- Output and errors in:
  `slurm-<job id>.out`
- Look at it with vi, nano, emacs, cat, less...
- To get output and error files split up, you can give these flags in the submit script:
  `#SBATCH --error=job.%J.err`
  `#SBATCH --output=job.%J.out`

The Batch System (SLURM)

Using different parts of Kebnekaise

- To run on the 'fat' nodes, add this flag to your script: `#SBATCH -p largemem` (Kebnekaise - largemem does not have general access)
- Specifying Intel Broadwell or Skylake CPUs only on Kebnekaise:
  `#SBATCH --constraint=broadwell`
  or
  `#SBATCH --constraint=skylake`
- Using the GPU nodes on Kebnekaise
  `#SBATCH --gres=gpu:<type-of-card>:x` where <type-of-card> is either k80 or v100 and x = 1, 2, or 4 (4 only for the K80 type).

More on
https://www.hpc2n.umu.se/documentation/guides/using_kebnekaise

The Batch System (SLURM)

Simple example, serial

Example: Serial job on Kebnekaise, compiler toolchain 'foss'

```
#!/bin/bash
# Project id - change to your own after the course!
#SBATCH -A SNIC2021-22-514
# Asking for 1 core
#SBATCH -n 1
# Asking for a walltime of 5 min
#SBATCH --time=00:05:00

# Purge modules before loading new ones in a script.
ml purge
ml foss/2019b

./my_serial_program
```

Submit with:

`sbatch <jobscript>`

The Batch System (SLURM)

Example, MPI C program

```
#include <stdio.h>
#include <mpi.h>

int main (int argc, char *argv[])

int myrank, size;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
MPI_Comm_size(MPI_COMM_WORLD, &size);

printf("Processor %d of %d:  Hello World!\n", myrank,
size);

MPI_Finalize();
```

The Batch System (SLURM)

Simple example, parallel

Example: MPI job on Kebnekaise, compiler toolchain 'foss'

```
#!/bin/bash
#SBATCH -A SNIC2021-22-514
#SBATCH -n 14
#SBATCH --time=00:05:00
##SBATCH --exclusive
#SBATCH --reservation=intro-hpc

module purge
ml foss/2019b

srun ./my_parallel_program
```

The Batch System (SLURM)

Simple example, output

Example: Output from a MPI job on Kebnekaise, run on 14 cores (one NUMA island)

```
b-an01 [~/slurm]$ cat slurm-15952.out

The following modules were not unloaded:
   (Use "module --force purge" to unload all):

  1) systemdefault   2) snicenvironment
Processor 12 of 14: Hello World!
Processor 5 of 14: Hello World!
Processor 9 of 14: Hello World!
Processor 4 of 14: Hello World!
Processor 11 of 14: Hello World!
Processor 13 of 14: Hello World!
Processor 0 of 14: Hello World!
Processor 1 of 14: Hello World!
Processor 2 of 14: Hello World!
Processor 3 of 14: Hello World!
Processor 6 of 14: Hello World!
Processor 7 of 14: Hello World!
Processor 8 of 14: Hello World!
Processor 10 of 14: Hello World!
```

The Batch System (SLURM)

Starting more than one serial job in the same submit file

```
#!/bin/bash
#SBATCH -A SNIC2021-22-514
#SBATCH -n 5
#SBATCH --time=00:15:00

module purge
ml foss/2019b

srun -n 1 ./job1.batch &
srun -n 1 ./job2.batch &
srun -n 1 ./job3.batch &
srun -n 1 ./job4.batch &
srun -n 1 ./job5.batch
wait
```

The Batch System (SLURM)

Multiple Parallel Jobs Sequentially

```
#!/bin/bash
#SBATCH -A SNIC2021-22-514
#SBATCH -c 28
# Remember to ask for enough time for all jobs to complete
#SBATCH --time=02:00:00

module purge
ml foss/2019b

# Here 14 tasks with 2 cores per task.  Output to file.
# Not needed if your job creates output in a file
# I also copy the output somewhere else and then run
# another executable...

srun -n 14 -c 2 ./a.out > myoutput1 2>&1
cp myoutput1 /pfs/nobackup/home/u/username/mydatadir
srun -n 14 -c 2 ./b.out > myoutput2 2>&1
cp myoutput2 /pfs/nobackup/home/u/username/mydatadir
srun -n 14 -c 2 ./c.out > myoutput3 2>&1
cp myoutput3 /pfs/nobackup/home/u/username/mydatadir
```

The Batch System (SLURM)

Multiple Parallel Jobs Simultaneously

Make sure you ask for enough cores that all jobs can run at the same time, and have enough memory. Of course, this will also work for serial jobs - just remove the srun from the command line.

```
#!/bin/bash
#SBATCH -A SNIC2021-22-514
# Total number of cores the jobs need
#SBATCH -n 56
# Remember to ask for enough time for all of the jobs to
# complete, even the longest
#SBATCH --time=02:00:00

module purge
ml foss/2019b

srun -n 14 --cpu_bind=cores ./a.out &
srun -n 28 --cpu_bind=cores ./b.out &
srun -n 14 --cpu_bind=cores ./c.out &
...
wait
```

The Batch System (SLURM)

GPU Job

```
#!/bin/bash
#SBATCH -A SNIC2021-22-514
# Expected time for job to complete
#SBATCH --time=00:10:00
# Number of GPU cards needed.  Here asking for 2 K80 cards
#SBATCH --gres=k80:2

module purge
ml fosscuda/2019b

./my-program
```

Important information

- The course project has the following SNIC ID: SNIC2021-22-514
- In order to use it in a batch job, add this to the batch script:
  - #SBATCH -a SNIC2021-22-514
- There are two nodes reserved for the course, in order to let us run small examples without having to wait for too long. One node is a regular CPU node, the other a GPU node.
- These reservations are ONLY valid during the course:
  - intro-hpc
    (add with #SBATCH –reservation=intro-hpc)
  - intro-hpc-gpu
    (add with #SBATCH –reservation=intro-hpc-gpu)
- We have a storage project linked to the compute project. It is snic2021-22-514. You find it in /proj/nobackup/snic2021-22-514

Questions and support

Questions? Now: Ask me or one of the other support or application experts present.

OR

- Documentation: `https://www.hpc2n.umu.se/support`
- Support questions to: `https://supr.snic.se/support/` or `support@hpc2n.umu.se`