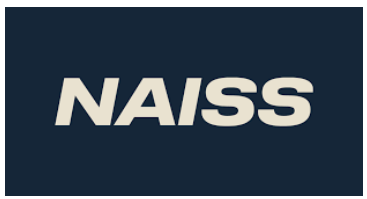
Introduction to HPC2N

Birgitte Brydsø, Pedro Ojeda-May

HPC2N, Umeå University

18 January 2024



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
  - Examples

Projects - compute and storage

Join a project or apply for one.

- Kebnekaise is only open for local project requests.
  - The PI must be affiliated with UmU, LTU, IRF, MiUN, or SLU.
  - You can still add members (join) from anywhere.
- Apply for **compute project** in SUPR

  `https://supr.naiss.se/round/compute`
  - Login to SUPR (create SUPR account if you do not have one).
  - Click "Rounds" in the left menu. Pick "Compute Rounds". Pick "Centre Local Compute".
  - Pick "HPC2N Local Compute YYYY". Choose "Create New Proposal for HPC2N Local Compute YYYY".
  - Create from scratch or use earlier proposal as starting point.
  - Agree to the default storage if 500GB is enough.
  - More information:

    `https://supr.naiss.se/round/open_type/?type=Centre+Local+Compute`
- If the above mentioned default storage is not enough, you will need to apply for a **Local storage project**:

  `https://supr.naiss.se/round/open_type/?type=Centre+Local+Storage`

Projects - compute and storage

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

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		

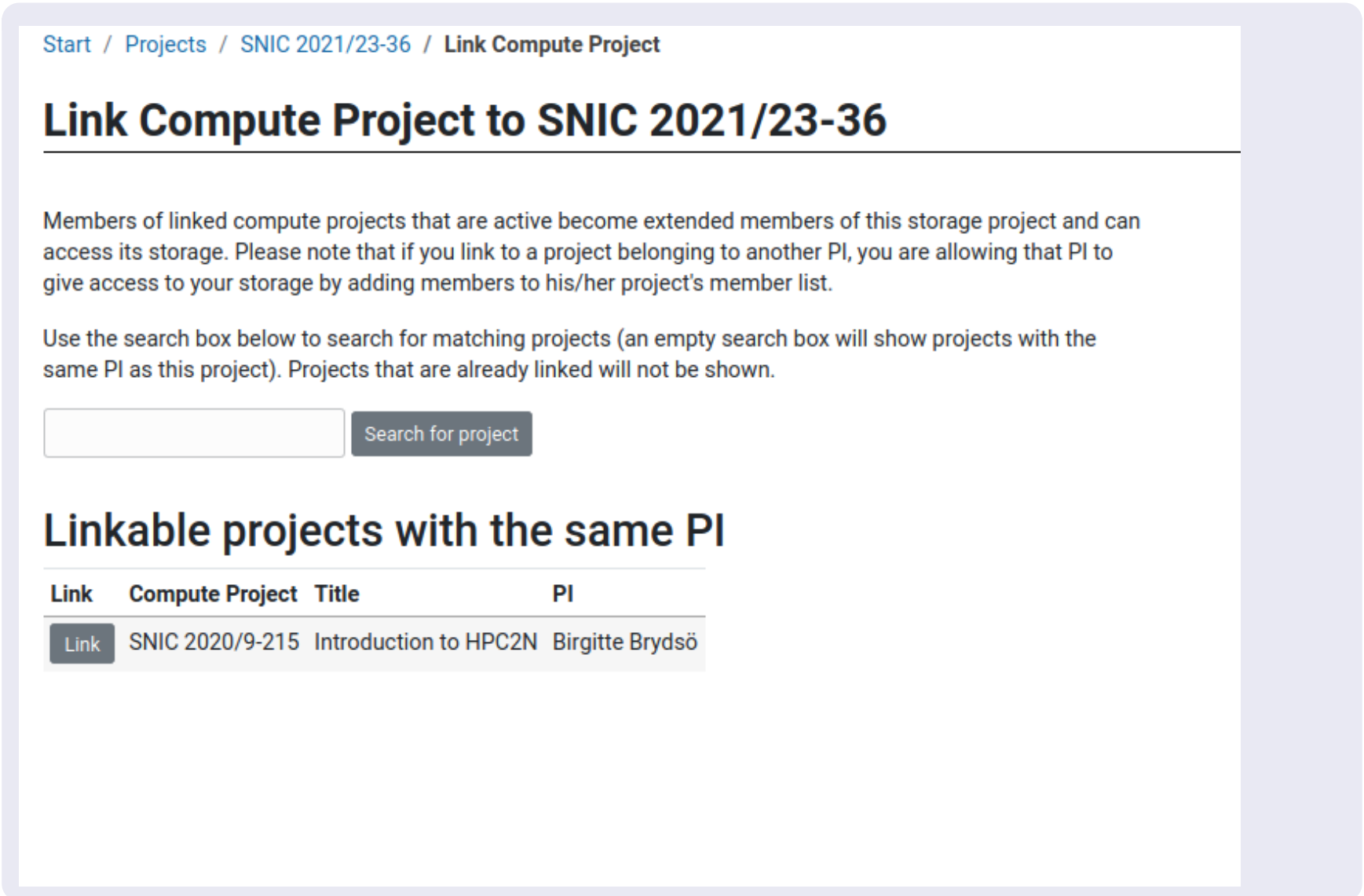
Resource Allocation and Usage

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 - example process

1. Connect to Kebnekaise:
   - **ThinLinc:** `kebnekaise-tl.hpc2n.umu.se`
   - ThinLinc through a browser (less features):
     `https://kebnekaise-tl.hpc2n.umu.se:300/`
   - SSH: `kebnekaise.hpc2n.umu.se`
   - **A100 or AMD Zen3**: `kebnekaise-amd-tl.hpc2n.umu.se` or `kebnekaise-amd.hpc2n.umu.se`
2. Transfer your files and data (if needed)
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 to local/other machine (optionally)

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.

- **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
```

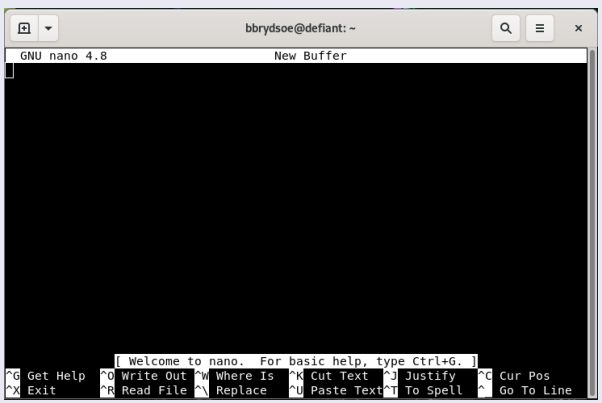
- **Windows:**
  - Download client: WinSCP, FileZilla (sftp), PSCP/PSFTP, ...
  - Transfer with sftp or scp
- https://www.hpc2n.umu.se/documentation/filesystems/filetransfer
- **Mac/OSX:**
  - Transfer with sftp or scp (as for Linux) using Terminal
  - Or download client: Cyberduck, Fetch, ...
- More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

Editing your files with nano

- Starting "nano": Type nano FILENAME and press Enter.
- If FILENAME exists, it will open, otherwise it will be created.
- 

- Notice: many of the commands are listed at the bottom.
- ^ before a letter-command means press CTRL + the letter.
- To exit (and possibly save), press CTRL + x (this is written CTRL-x or ^x). nano asks if you want to save before exiting.

The File System

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

	Project storage	**$HOME**	**/scratch**
Recommended for batch jobs	Yes	No (size)	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 (your PI) will either choose a directory name when you/they apply for the storage project or get the project 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.
- **For this course the storage is in** `/proj/nobackup/hpc2n2023-132`

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
  - **gompi**: GCC, OpenMPI
  - **gompic**: GCC, OpenMPI, CUDA
  - **gomkl**: GCC, OpenMPI, MKL
  - **iccifort**: icc, ifort
  - **iccifortcuda**: icc, ifort, CUDA
  - **iimpi**: icc, ifort, IntelMPI
  - **intel**: icc, ifort, IntelMPI, IntelMKL
  - **intelcuda**: intel and CUDA
  - **iompi**: iccifort and 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!

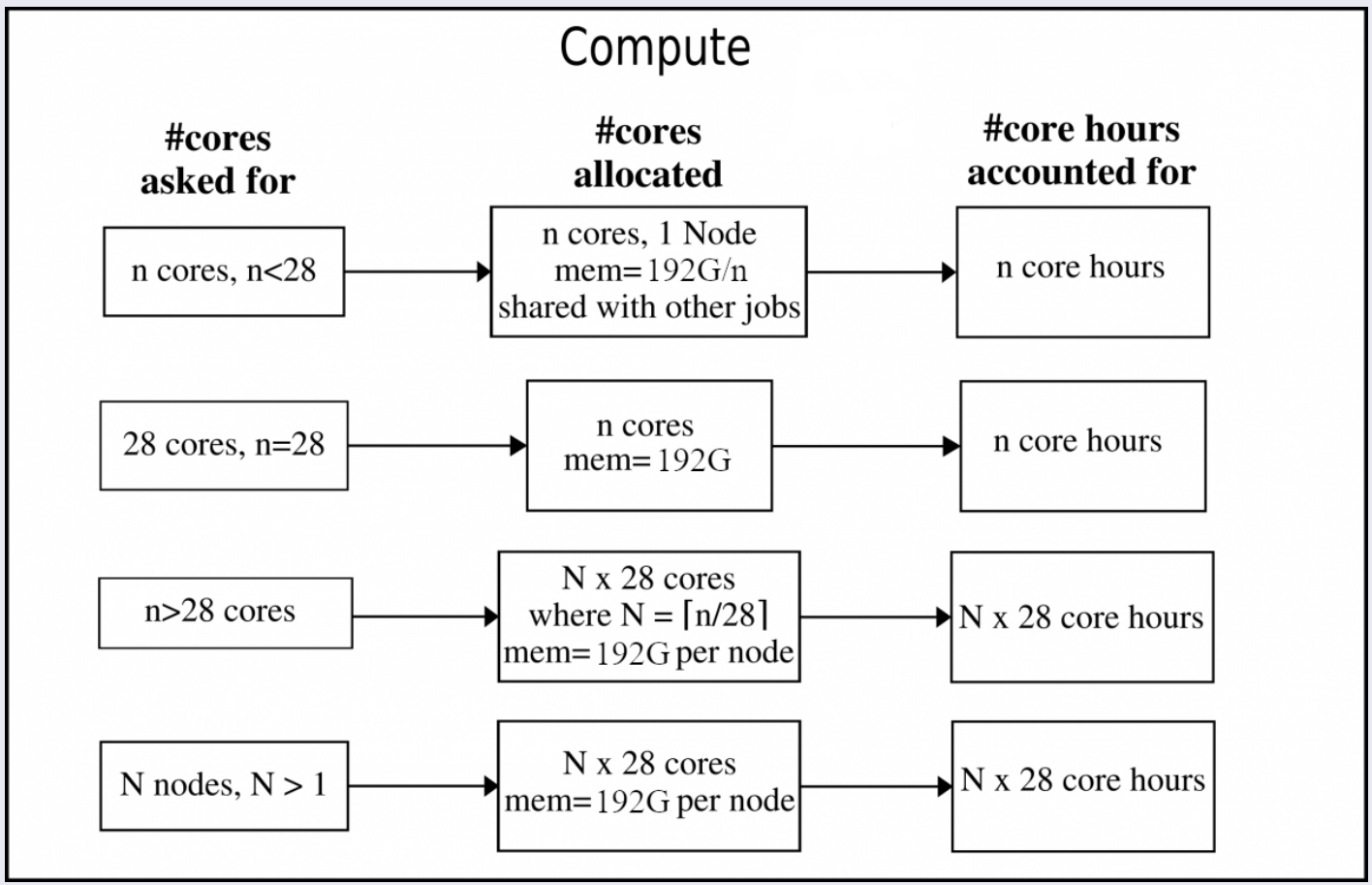
The Batch System (SLURM)

- 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

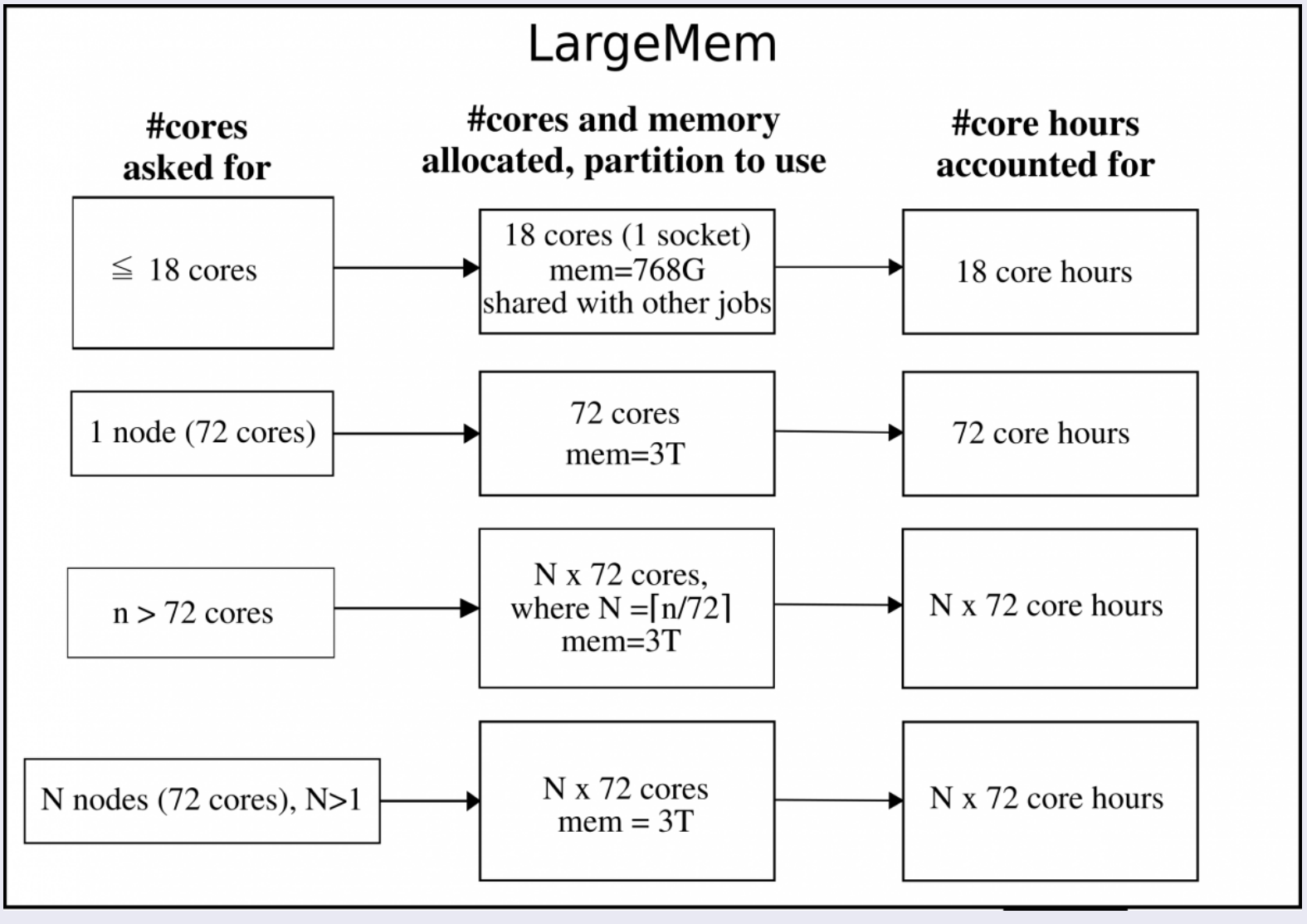
Accounting, Compute nodes, Kebnekaise

Here the Skylake nodes are used as an example. The only difference for the Broadwell nodes is that it would say 128G instead of 192G per node.



The Batch System

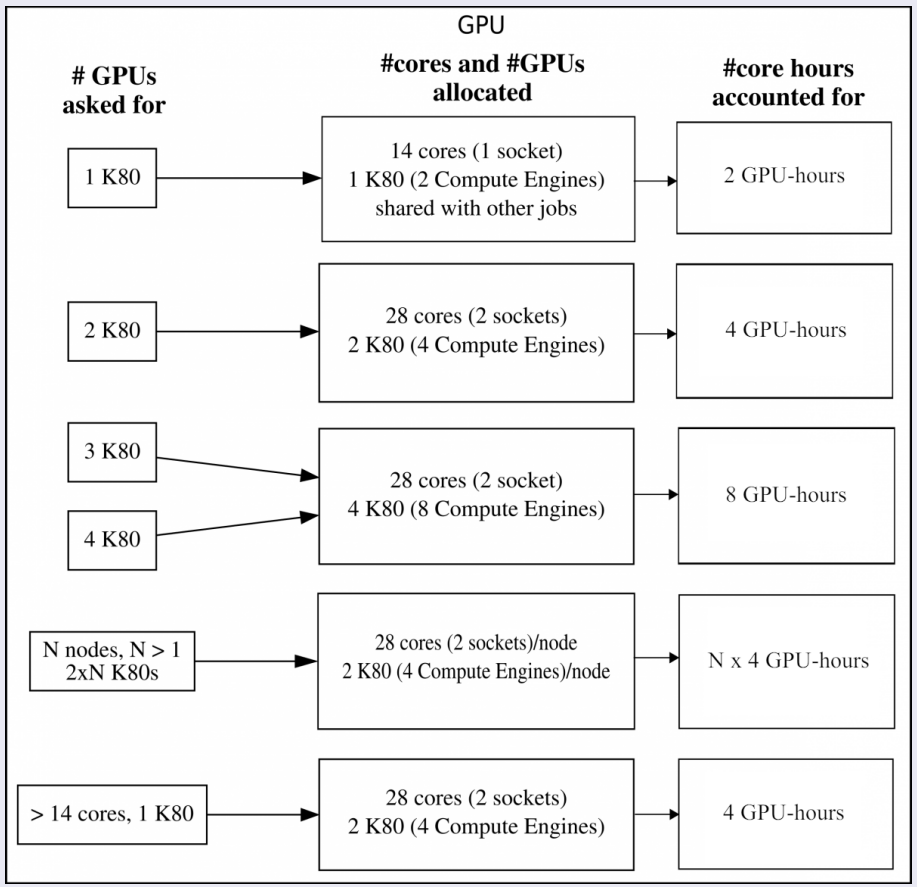
Accounting, largemem nodes, Kebnekaise



The Batch System

Accounting, K80 GPU nodes, Kebnekaise.

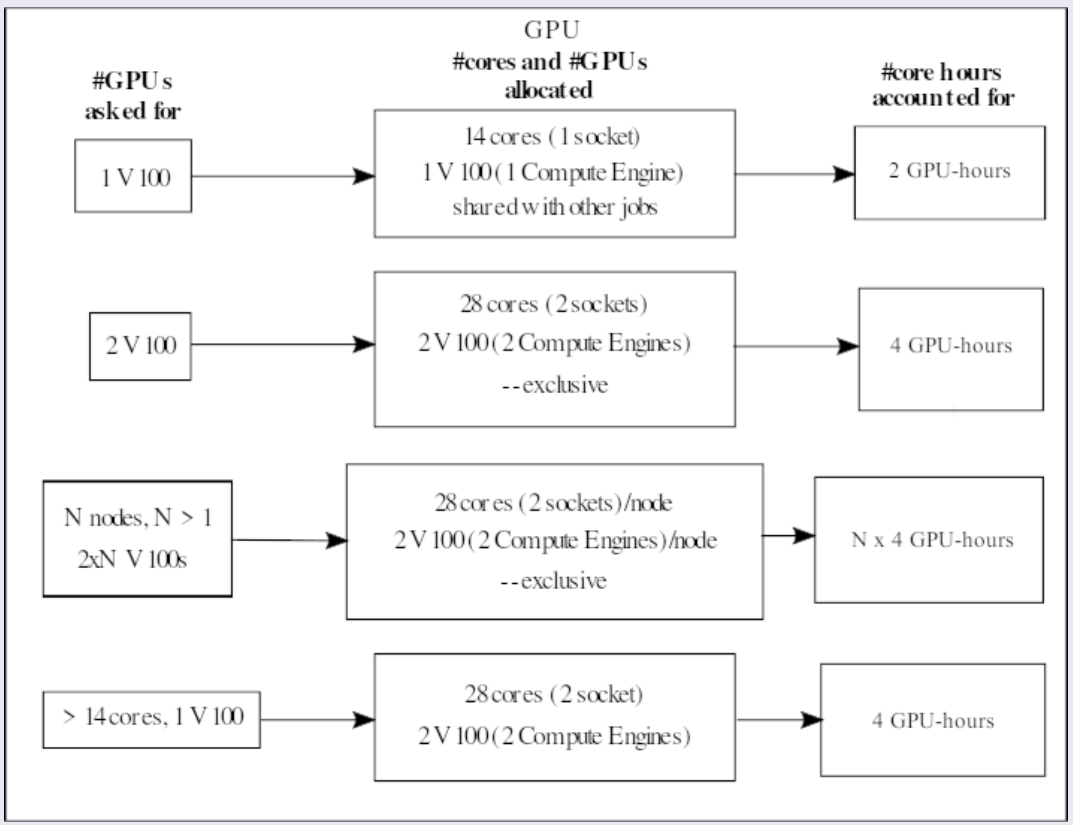
The K80 GPU cards have 2 onboard compute engines (GK210 chips). Most GPU nodes have 2 K80s, placed together as 14 cores + 1 K80/socket. 4 GPU nodes have 4 K80 GPU cards.



The Batch System

Accounting, V100 GPU nodes, Kebnekaise.

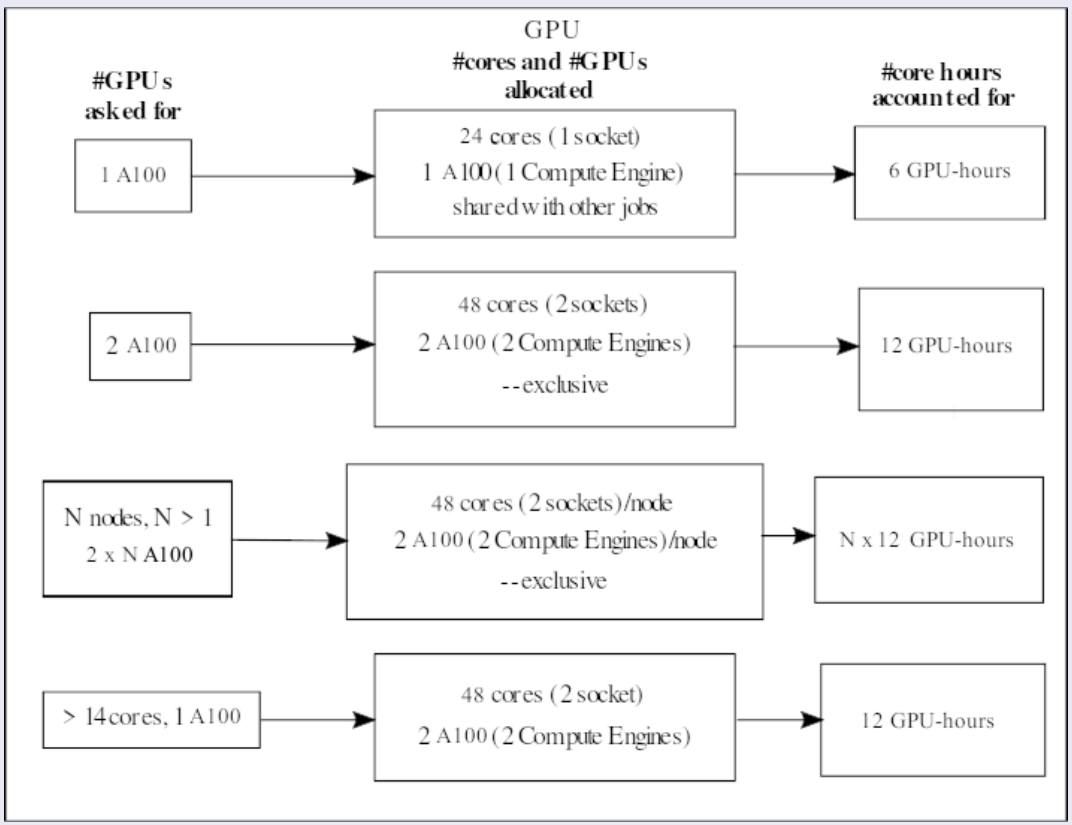
Each V100 GPU accelerator card has 1 onboard compute engine (GV100 chip). They are placed together as 14 cores + 1 V100 on a socket (28 cores, 2 V100s per node).



The Batch System

Accounting, A100 GPU nodes, Kebnekaise.

Each A100 GPU accelerator card has 1 onboard compute engine. The AMD Zen3 nodes have 2 CPUs sockets with 24 cores each, for a total of 48 cores, and 2 NVidia A100 GPUs. They are placed together as 24 cores + 1 A100 on a socket.



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>`
- Delete all your own jobs: `scancel -u <user>`
- 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. To view, use
    `sacct -l -j ..... | less -S`
    (makes it sideways scrollable, using the left/right arrow key)
- Web url with graphical info about a job: `job-usage <job-id>`

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

- 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

- Use the 'fat' nodes by adding this flag to your script:
  `#SBATCH -p largemem` (separate resource)
- Specifying Intel Broadwell, Intel Skylake, or AMD Zen3 CPUs:
  `#SBATCH --constraint=broadwell`
  or
  `#SBATCH --constraint=skylake`
  or
  `#SBATCH --constraint=zen3`
- Using the GPU nodes (separate resource):
  `#SBATCH --gres=gpu:<type-of-card>:x` where <type-of-card> is either k80, v100, or a100 and x = 1, 2, or 4 (4 only for K80).
  - In the case of the A100 GPU nodes, you also need to add a partition
    `#SBATCH -p amd_gpu`
- Use the AMD login node for correct modules and compilers for AMD Zen3 and A100 nodes:
  `kebnekaise-amd-tl.hpc2n.umu.se` or
  `kebnekaise-amd.hpc2n.umu.se`

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 hpc2n2023-132
# 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 > /dev/null 2>&1
ml foss/2021b

./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 hpc2n2023-132
#SBATCH -n 14
#SBATCH --time=00:05:00
##SBATCH --exclusive
#SBATCH --reservation=intro-gpu

module purge > /dev/null 2>&1
ml foss/2021b

srun ./my_parallel_program
```

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

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

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 hpc2n2023-132
#SBATCH -n 5
#SBATCH --time=00:15:00

module purge > /dev/null 2>&1
ml foss/2021b

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 hpc2n2023-132
#SBATCH -c 28
# Remember to ask for enough time for all jobs to complete
#SBATCH --time=02:00:00

module purge > /dev/null 2>&1
ml foss/2021b

# 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 hpc2n2023-132
# 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 > /dev/null 2>&1
ml foss/2021b

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 - V100

```
#!/bin/bash
#SBATCH -A hpc2n2023-132
# Expected time for job to complete
#SBATCH --time=00:10:00
# Number of GPU cards needed.  Here asking for 2 V100 cards
#SBATCH --gres=v100:2

module purge > /dev/null 2>&1
# Change to modules needed for your program
ml fosscuda/2021b

./my-cuda-program
```

The Batch System (SLURM)

GPU Job - A100

```
#!/bin/bash
#SBATCH -A hpc2n2023-132
# Expected time for job to complete
#SBATCH --time=00:10:00
# Adding the partition for the A100 GPUs
#SBATCH -p amd_gpu
# Number of GPU cards needed.  Here asking for 2 A100 cards
#SBATCH --gres=a100:2

module purge > /dev/null 2>&1
# Change to modules needed for your software - remember to login
# to kebnekaise-amd.hpc2n.umu.se or
# kebnekaise-amd-tl.hpc2n.umu.se login node to see availability
ml CUDA/11.7.0

./my-cuda-program
```

Important information

- The course project has the following project ID: hpc2n2023-132
- In order to use it in a batch job, add this to the batch script:
  - #SBATCH -A hpc2n2023-132
- There is a reservation with one A100 GPU node reserved for the course, in order to let us run small GPU examples without having to wait for too long. The reservation also is for one Broadwell CPU node.
- The reservation is ONLY valid during the course:
  - intro-gpu
    (add with #SBATCH –reservation=intro-gpu)
  - To use the reservation with the A100 GPU node, also add `#SBATCH -p amd_gpu` and `#SBATCH --gres=a100:x (for x=1,2)`.
- We have a storage project linked to the compute project. It is hpc2n2023-132. You find it in /proj/nobackup/hpc2n2023-132. Remember to create your own directory under it.

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.naiss.se/support/` or `support@hpc2n.umu.se`