Introduction to HPC2N and Kebnekaise

Birgitte Brydsø, Jerry Eriksson, and Pedro Ojeda-May

HPC2N, Umeå University

21 February 2017

Overview

- Using Kebnekaise and Abisko
- The File System
- The Module System
  - Overview
  - Compiler Tool Chains
  - Examples
- Compiling/linking with libraries
- The Batch System (SLURM)
  - Overview
  - Simple example
  - More examples

Using Kebnekaise and Abisko

1. Get an account (https://www.hpc2n.umu.se/documentation/access-and-accounts/users)
2. Connect to:

   `kebnekaise.hpc2n.umu.se`
   or
   `abisko.hpc2n.umu.se`

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

- **Linux, OS X:**
  - `ssh username@kebnekaise.hpc2n.umu.se`
    or
    `ssh username@abisko.hpc2n.umu.se`
  - Use `ssh -X ....` if you want to open graphical displays.
- **Windows:**
  - Get an SSH client (PuTTY, Cygwin ...)
  - Get an X11 server if you need graphical displays (Xming, Cygwin ...)
  - Start the client and login to

    `kebnekaise.hpc2n.umu.se`
    or
    `abisko.hpc2n.umu.se`

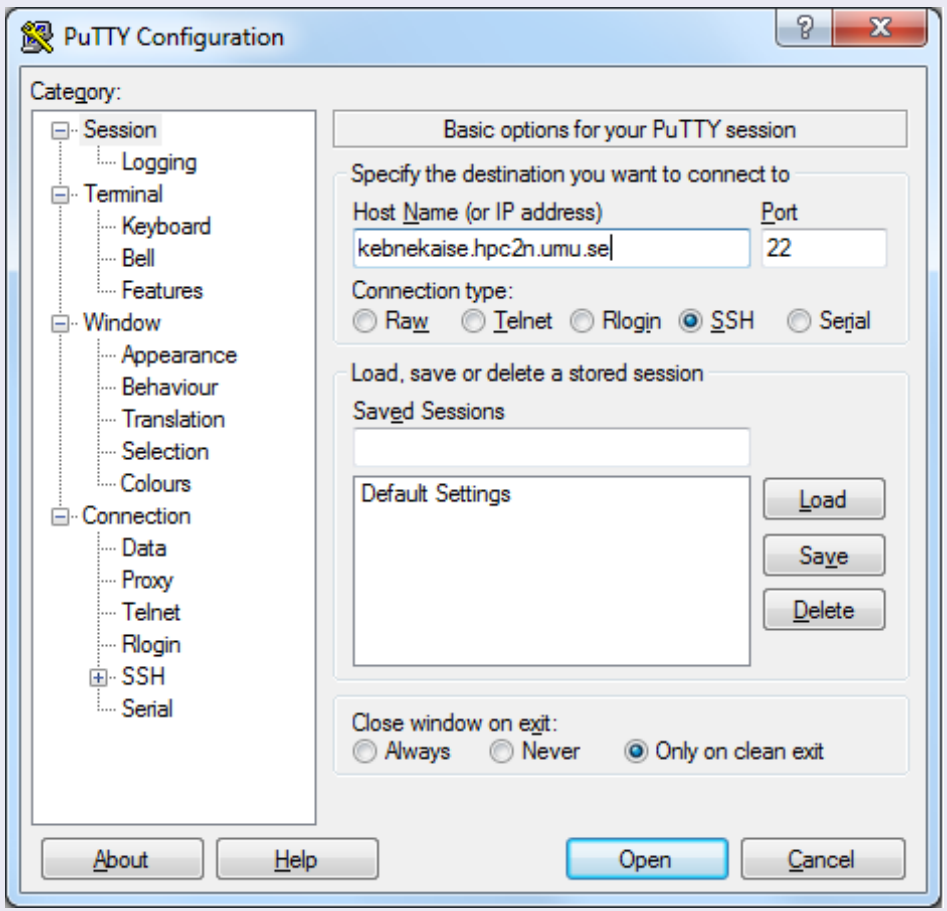
  - More information here:

    https://www.hpc2n.umu.se/documentation/guides/windows-connection

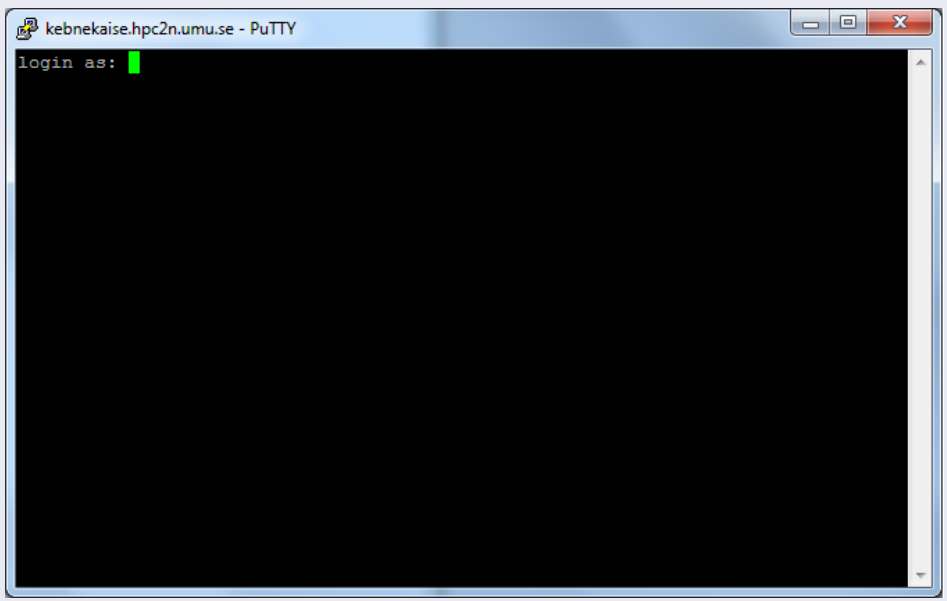
Using Kebnekaise and Abisko

Connecting from a Windows System with PuTTY

Get the Zip file (http://www.putty.org/) with both PuTTY, PSCP, and PSFTP. Unzip, run putty.exe



Enter your username and then your password.



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

    ```
    local> scp username@abisko.hpc2n.umu.se:file .
    local> scp file username@abisko.hpc2n.umu.se:file
    ```
    or
    ```
    local> scp username@kebnekaise.hpc2n.umu.se:file .
    local> scp file username@kebnekaise.hpc2n.umu.se:file
    ```
- **Windows:**
  - Download client: WinSCP, FileZilla (only sftp), PSCP/PSFTP, ...
  - Transfer with sftp or scp
- More information here:

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

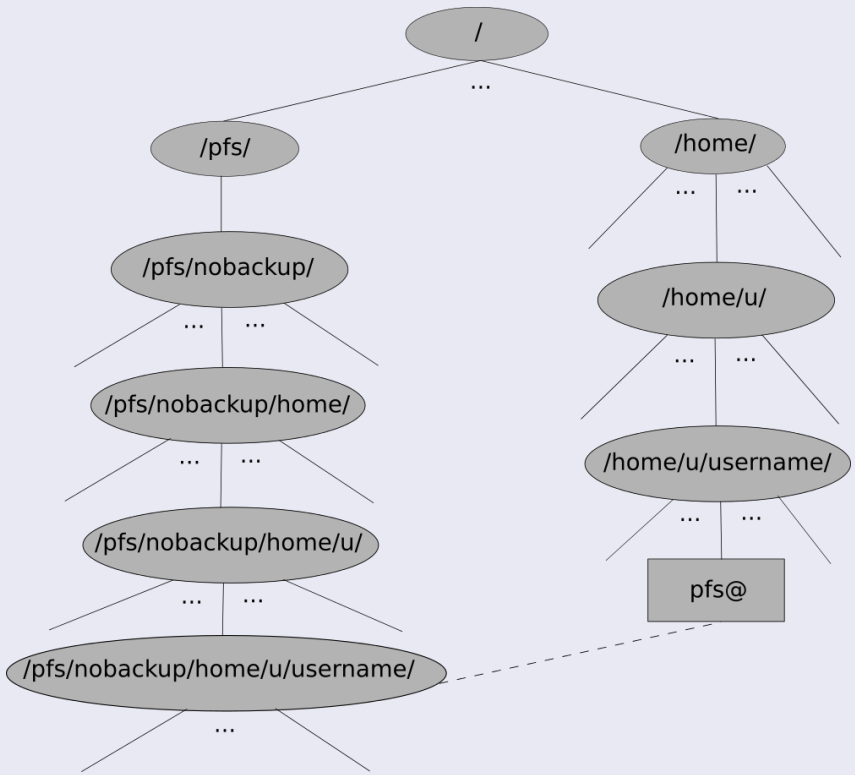
Editing your files

- Various editors: vi, vim, nano, emacs ...
- Example, nano: `nano <filename>`
- Save and exit nano: `Ctrl-x`

The File System

There are 2 file systems
More info here: http://www.hpc2n.umu.se/filesystems/overview

- **AFS**
  - This is where your home directory is located (cd $HOME)
  - Regularly backed up
  - NOT accesseable by the batch system (except the folder Public with the right settings)
- **PFS**
  - Parallel File System
  - NO BACKUP
  - Accessible by the batch system



The File System

AFS

- Your home directory is located in /home/u/username and can also be accessed with the environment variable $HOME
- It is located on the AFS (Andrew File System) file system
- Important! The batch system cannot access AFS since ticket-forwarding to batch jobs do not work
- AFS does secure authentification using Kerberos tickets

The File System

PFS

- The 'parallel' file system, where your 'parallel' home directory is located in /pfs/nobackup/home/u/username (/pfs/nobackup/$HOME)
- Offers high performance when accessed from the nodes
- The correct place to run all your batch jobs
- NOT backed up, so you should not leave files there that cannot easily be recreated
- For easier access, create a symbolic link from your home on AFS to your home on PFS:

  ```
  ln -s /pfs/nobackup/$HOME $HOME/pfs
  ```

  You can now access your pfs with `cd pfs` from your home directory on AFS

The Module System

Abisko

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

- See which modules exists:
  `module avail`
- Different versions of software:
  `module avail <module name>`
- Example: loading the default intel compilers:
  `module load intel`
- Unload the module:
  `module unload intel`

The Module System (Lmod)

Kebnekaise

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

- See which modules exists:
  `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` or `ml foss`
- Example: Unload the above module:
  `module unload foss` or `ml -foss`
- More information about a module:
  `ml show <module>`
- Unload all modules except the 'sticky' modules:
  `ml purge`

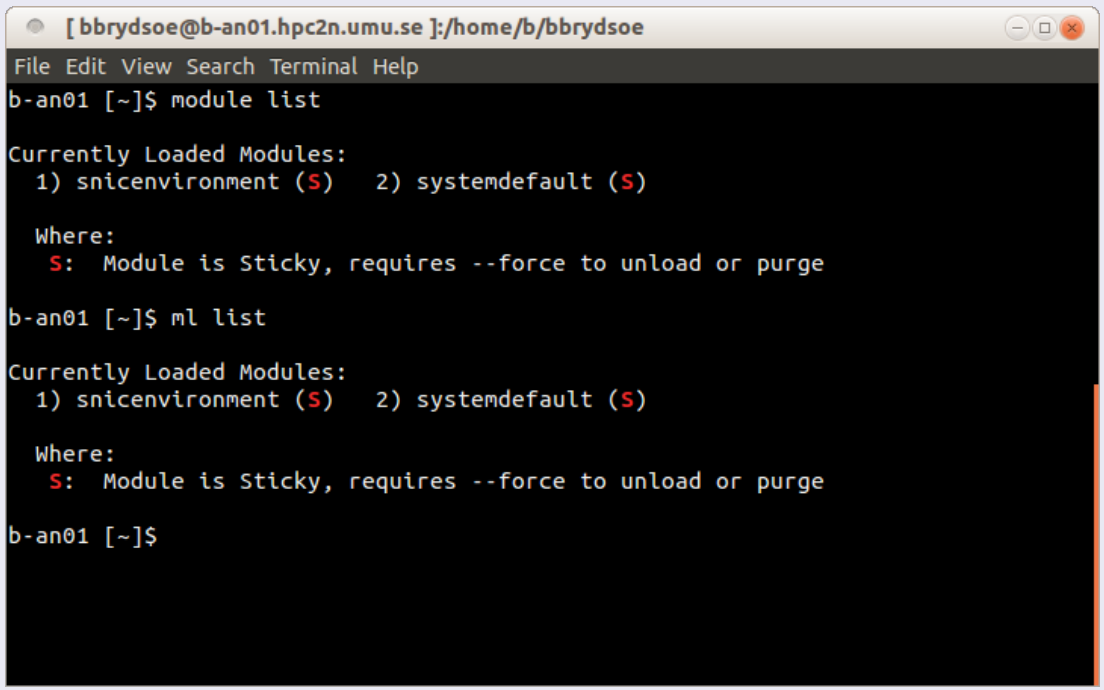
Compiler toolchains load bundles of software making up a complete environment for compiling/using a specific prebuilt software. Includes some/all of: compiler suite, MPI, BLAS, LAPACK, ScaLapack, FFTW, CUDA.

- Currently available toolchains (check `ml av` for versions):
  - **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

The Module System

Examples

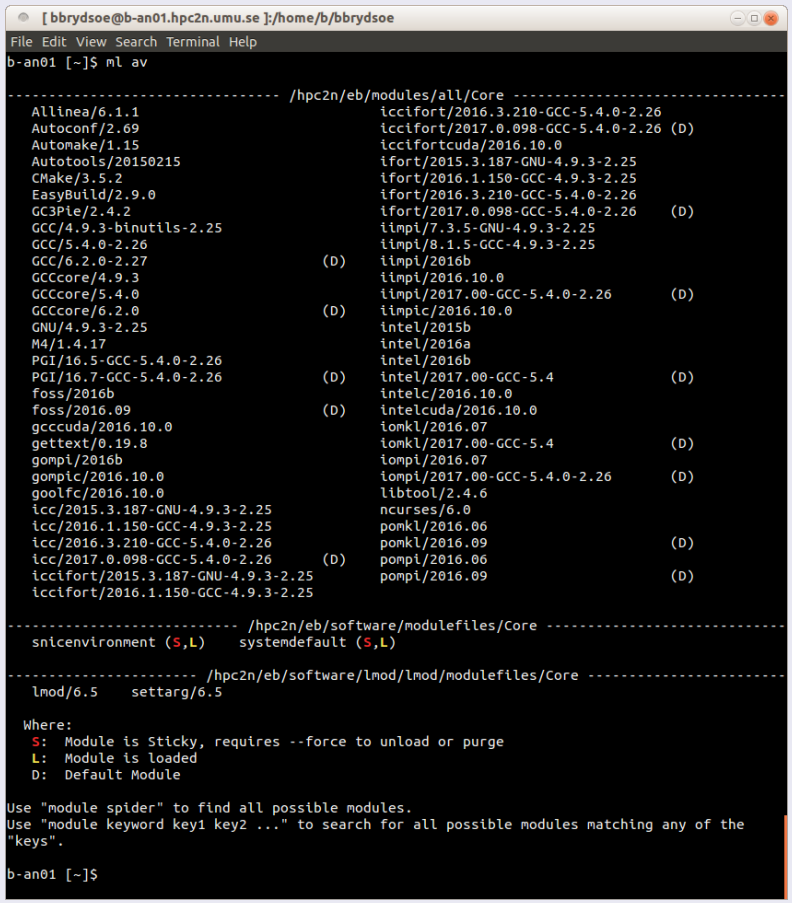
```
module list
ml list
ml
```



The Module System

Examples

```
module avail
ml avail
ml av
```



The Module System

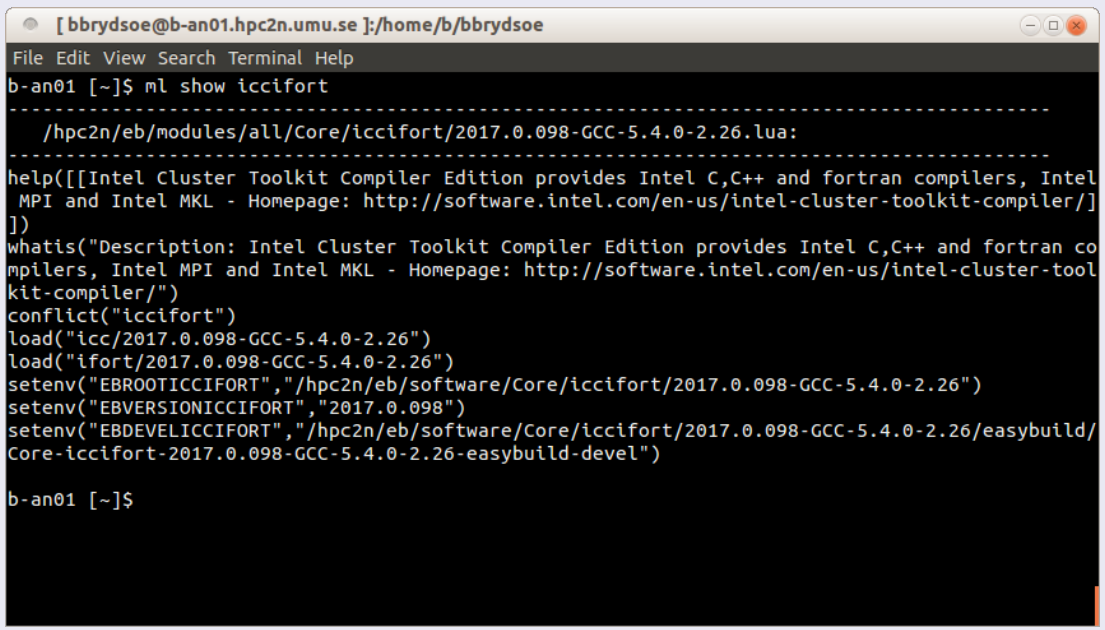
Examples

```
module spider
ml spider
```

The Module System

Examples

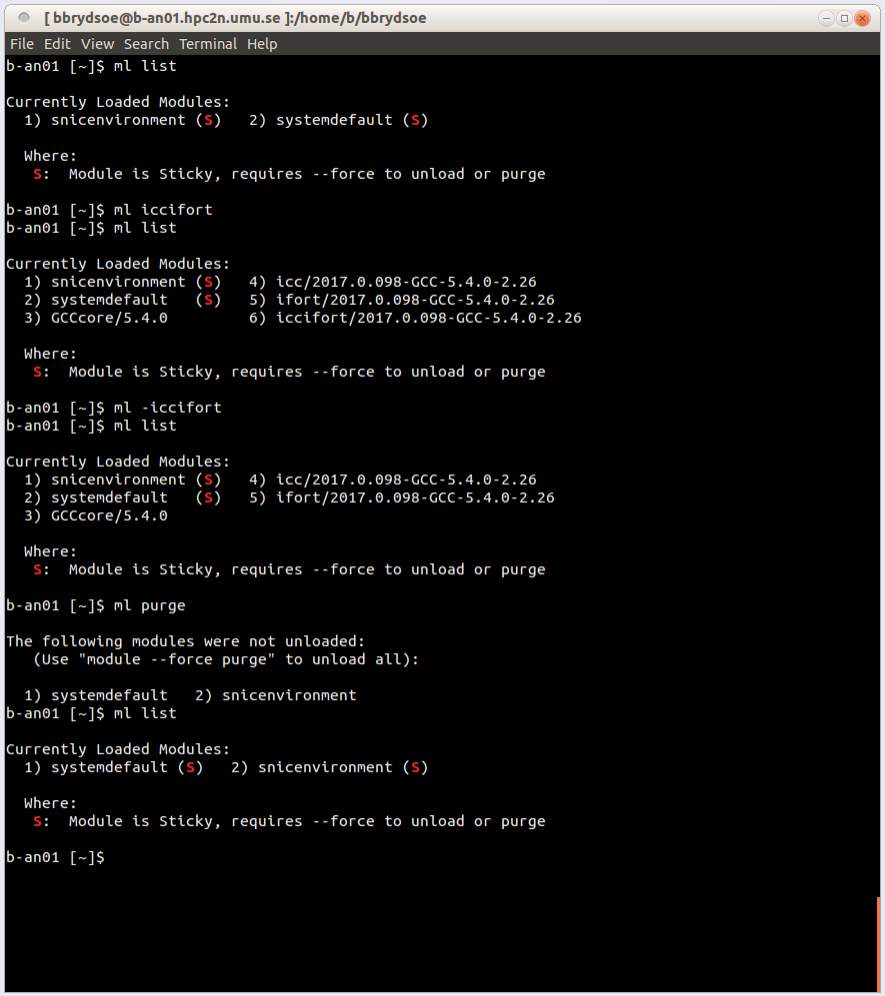
```
module show <module>
ml show <module>
```



The Module System

Examples

`module load <module>` / `module unload <module>`
`ml <module>` / `ml -<module>`



Compiling and Linking with Libraries

Some examples

- MPI C program:
  - Intel compilers, Intel MPI:
    ```
    ml iimpi
    mpicc <program.c> -o <outfile>
    ```
  - GCC compilers, OpenMPI:
    ```
    ml gompi
    mpicc <program.c> -o <outfile>
    ```
- OpenMP Fortran program:
  - Intel compilers:
    ```
    ml iccifort
    ifort -qopenmp <program.f90> -o <outfile>
    ```
  - GCC compilers:
    ```
    ml GCC
    gfortran -fopenmp <program.f90> -o <outfile>
    ```

Compiling and Linking with Libraries

Continued

Examples

- C program, BLAS, LAPACK:
  - Intel compilers, Intel MKL:

    ```
    ml intel

    -L${MKLROOT}/lib/intel64 -lmkl_intel_ilp64 \
    -lmkl_sequential -lmkl_core -lpthread -lm -ldl
    ```

  - GCC compilers, OpenBLAS/LAPACK:

    ```
    ml foss
    gcc -o program.c program.o -lopenblas
    ```

Compiling and Linking with Libraries

Continued

Examples

- Fortran program, ScaLAPACK, OpenMPI:
  - GCC, OpenBLAS/LAPACK, ScaLAPACK, OpenMPI:

    ```
    ml foss
    gcc -o program program.o -lscalapack -lopenblas
    ```
  - Intel, MKL, Intel MPI:

    ```
    ml intel
    -L${MKLROOT}/lib/intel64 -lmkl_scalapack_ilp64 \
    -lmkl_intel_ilp64 -lmkl_sequential -lmkl_core \
    -lmkl_blacs_intelmpi_ilp64 -lpthread -lm -ldl
    ```
- C program, OpenMPI, CUDA:
  - GCC:

    ```
    ml goolfc
    -lcuda -lcudart
    ```

    or `nvcc program.cu -o program`

Figuring out how to link

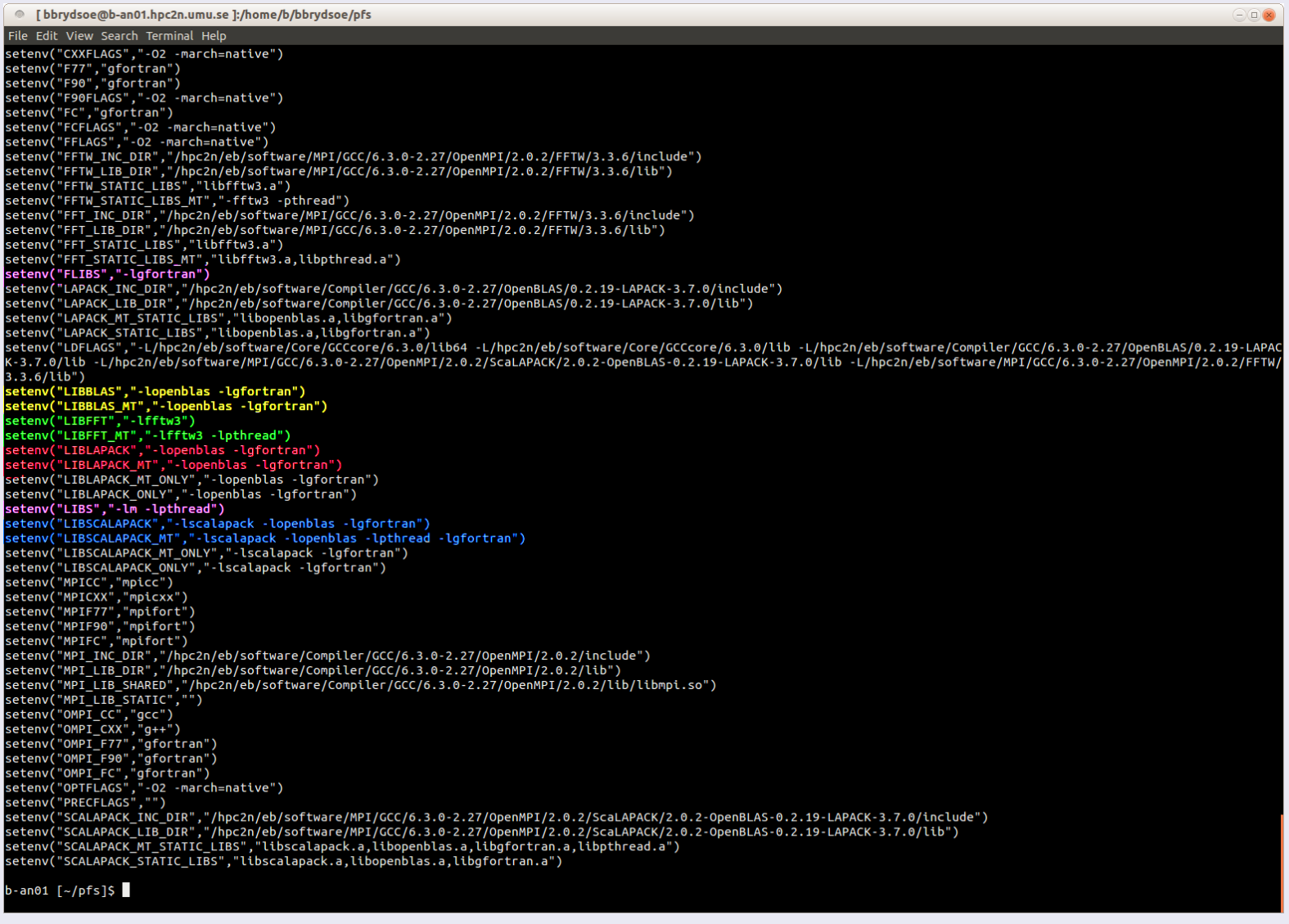
- Intel and Intel MKL linking:

  `https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor`
- Buildenv
  - After loading a compiler toolchain, load `'buildenv'` and use `'ml show buildenv'` to get useful linking info
  - Example, foss:

    ```
    ml foss
    ml buildenv
    ml show buildenv
    ```
  - Using the environment variable (prefaced with $) is highly recommended!

Compiling and Linking with Libraries

Example: ml foss, ml buildenv, ml show buildenv

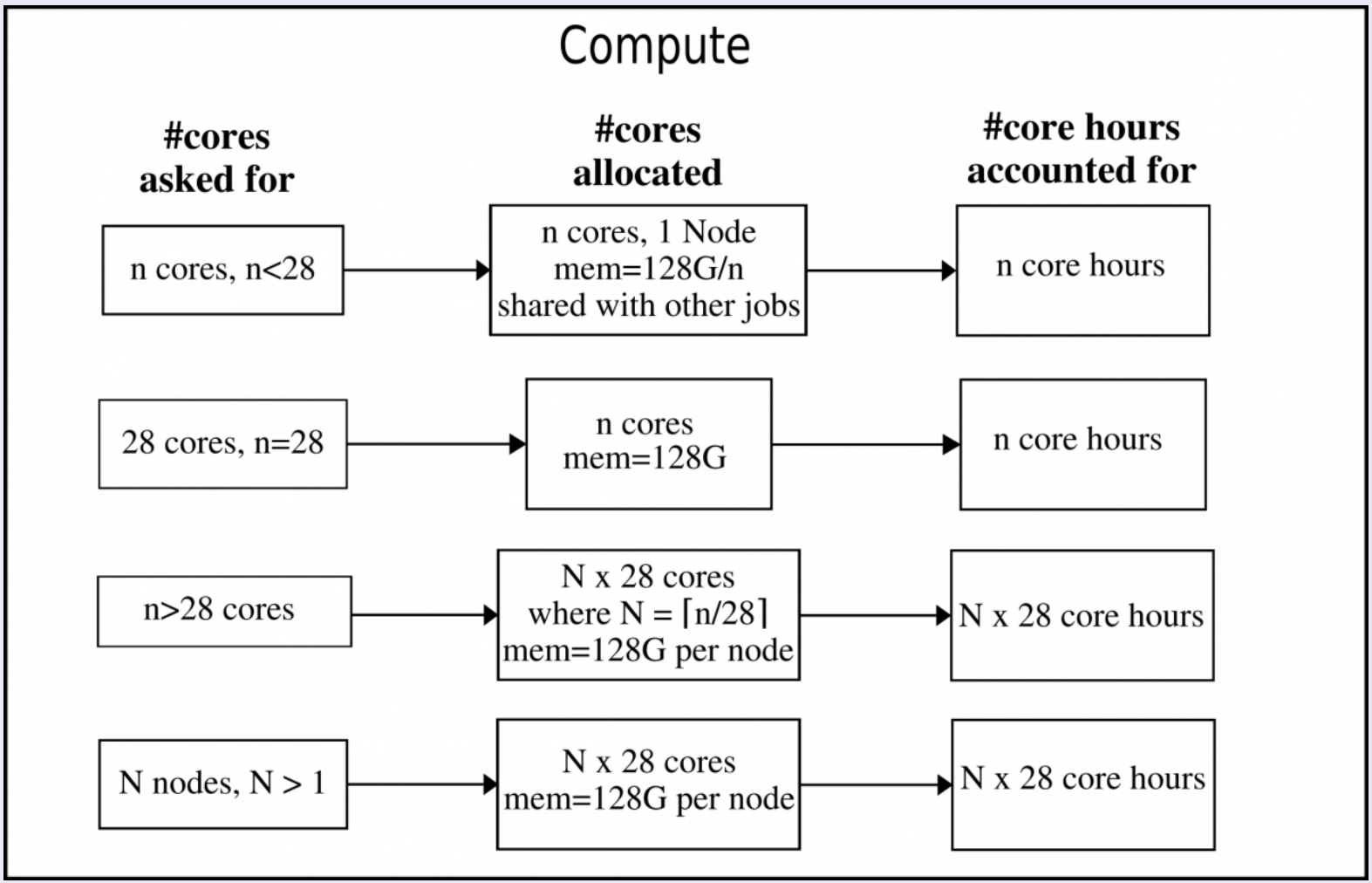


The Batch System (SLURM)

- 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
- Same batch system on Abisko and Kebnekaise. The differences are that there are GPUs and KNLs which can be allocated on Kebnekaise
- Guides and documentation at: http://www.hpc2n.umu.se/support

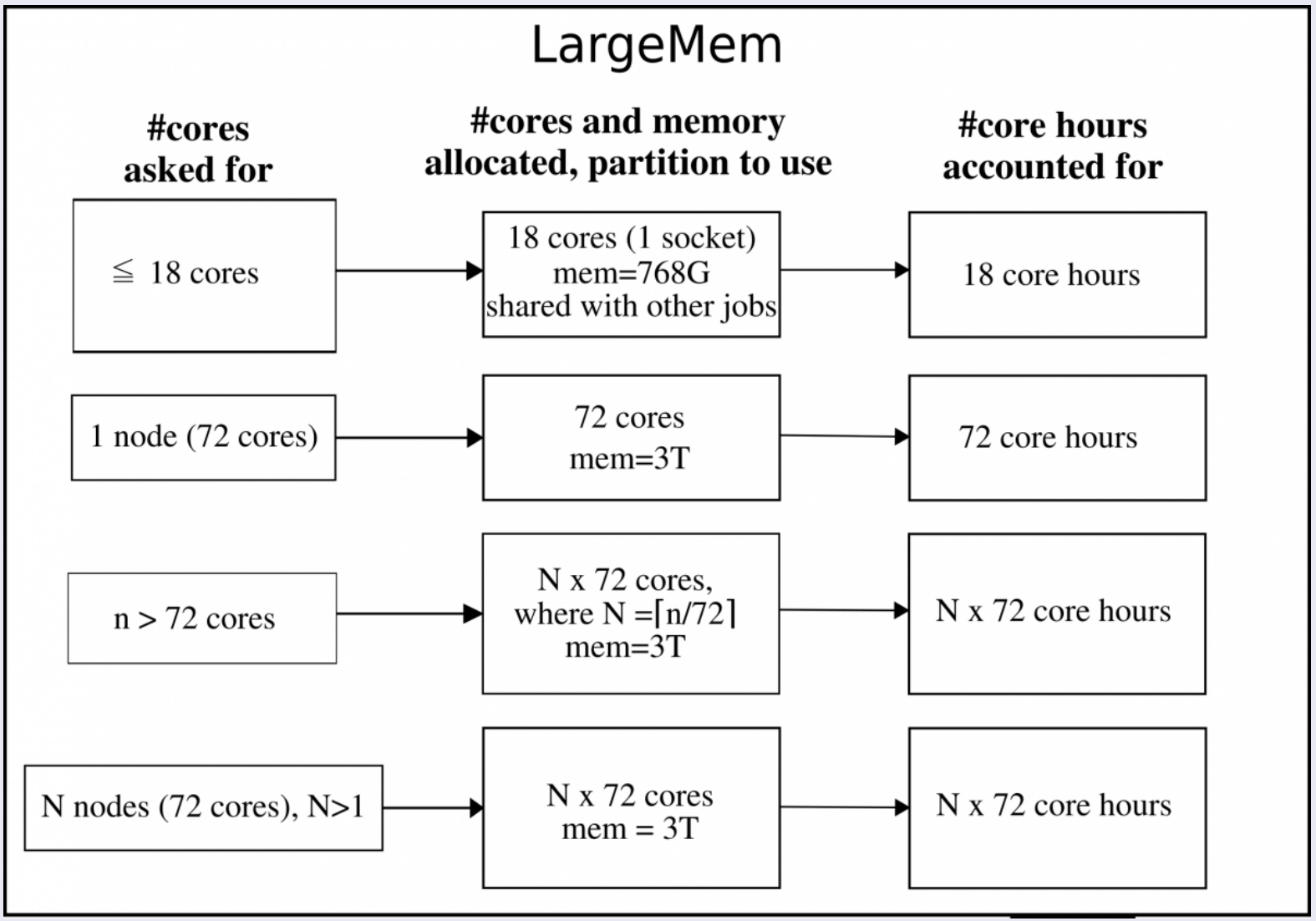
The Batch System

Accounting, Compute nodes



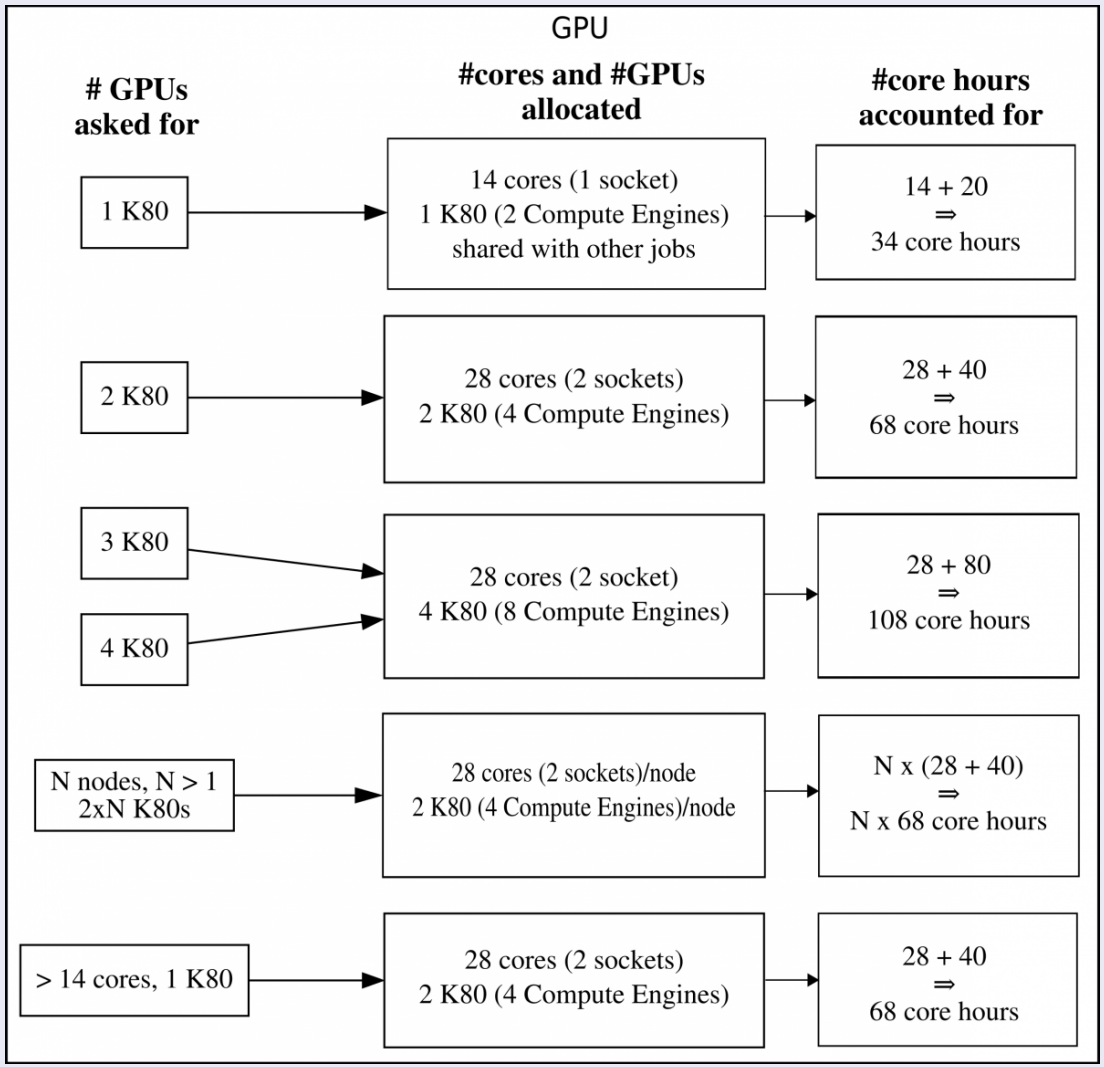
The Batch System

Accounting, largemem nodes



The Batch System

Accounting, GPU nodes



The Batch System (SLURM)

Useful Commands

- Submit job: `sbatch <jobscript>`
- Get list of your jobs: `squeue -u <username>`
- `srun <commands for your job/program>`
- `salloc <commands to the batch system>`
- Check on a specific job: `scontrol show job <job id>`
- Delete a specific job: `scancel <job id>`

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`
- To run on the 'fat' nodes, add this flag to your script:
  `#SBATCH -p largemem` (Kebnekaise - largemem does not have general access) `#SBATCH -p bigmem` (Abisko)

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 SNIC2017-3-22
# Asking for 1 core
#SBATCH -n 1
# Asking for a walltime of 5 min
#SBATCH --time=00:05:00

# Always purge modules before loading new ones in a
script.  Note, this is only on Kebnekaise module purge
ml foss

./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 SNIC2017-3-22
#SBATCH -n 14
#SBATCH --time=00:05:00
##SBATCH --exclusive
#SBATCH --reservation=SNIC2017-3-22

module purge
ml foss

mpirun ./my_parallel_program
```

Note: On Abisko you use 'srun' instead of 'mpirun'.

The Batch System (SLURM)

Simple example, output

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

```
b-an01 [~/pfs/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)

Requesting GPU nodes

Currently there is no separate queue for the GPU nodes

- You request GPU nodes by adding the following to your batch script:
  `#SBATCH --gres=gpu:k80:x` where `x=1, 2, 4`
- x = the number of K80 cards, each with 2 GPU engines
- There are 32 nodes with dual K80 cards and 4 nodes with quad K80 cards

Note: This is only valid on Kebnekaise. Abisko has no GPUs.

The Batch System (SLURM)

Longer example

```
#!/bin/bash
#SBATCH -A SNIC2017-3-22
#SBATCH -n 14
#SBATCH --time=00:05:00
#SBATCH --reservation=SNIC2017-3-22

module purge
ml foss

echo "Running on hosts:  $SLURM_NODELIST"
echo "Running on $SLURM_NNODES nodes."
echo "Running on $SLURM_NPROCS processors."
echo "Current working directory is `pwd`"

echo "Output of mpirun hostname:"
mpirun /bin/hostname

mpirun ./mpi_hello
```