

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

Using Kebnekaise and Abisko

Connecting to HPC2N's systems

- **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

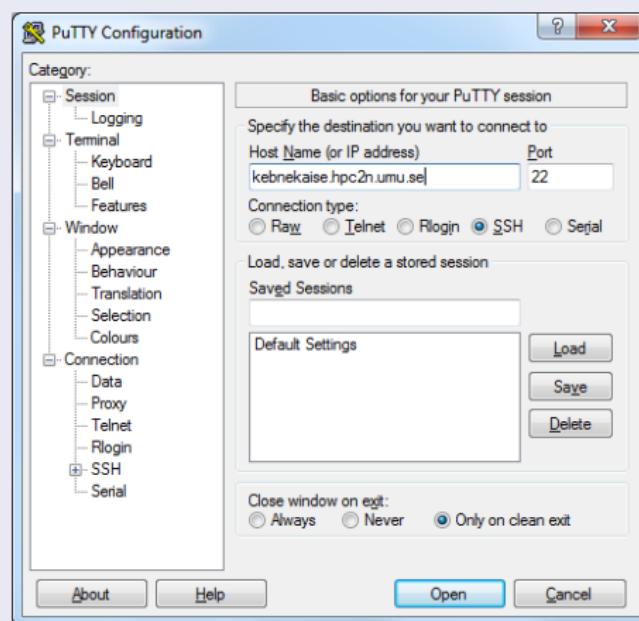
- More information here:

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

Using Kebnekaise and Abisko

Connecting from a Windows System with PuTTY

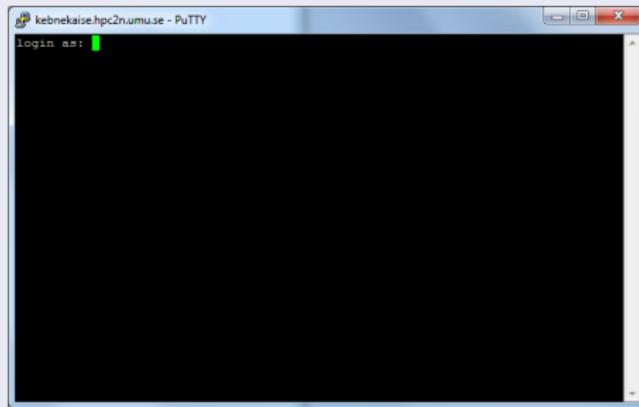
Get the Zip file (<http://www.putty.org/>) with both PuTTY, PSCP, and PSFTP. Unzip, run putty.exe



Using Kebnekaise and Abisko

Connecting from a Windows System with PuTTY

Enter your username and then your password.



Using Kebnekaise and Abisko

Transfer your files and data

- **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>

Using Kebnekaise and Abisko

Editors

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

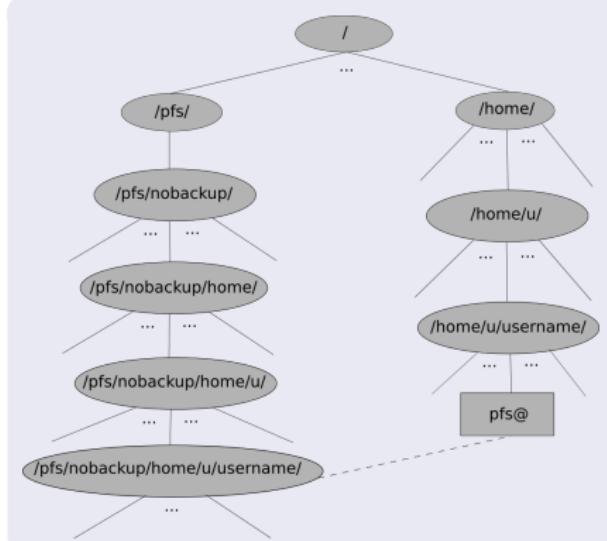
More info here: <http://www.hpc2n.umu.se/filesystems/overview>

- **AFS**

- This is where your home directory is located (`cd $HOME`)
- Regularly backed up
- NOT accessible 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 authentication 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`

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.

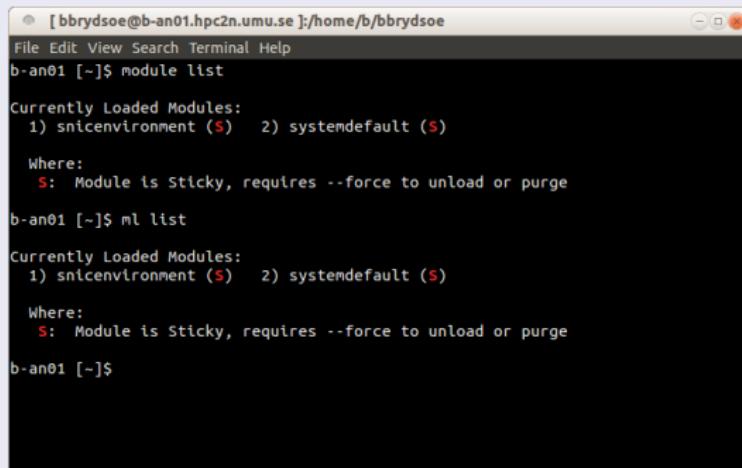
- Currently available toolchains (check `ml av` for versions):

- `GCC`: GCC only
- `gccccuda`: 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`
- `ccifortcuda`: `icc`, `ifort`, CUDA
- `ifort`: Intel Fortran compiler only
- `iimpi`: `icc`, `ifort`, IntelMPI
- `intel`: `icc`, `ifort`, IntelMPI, IntelMKL
- `intetcuda`: 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
```



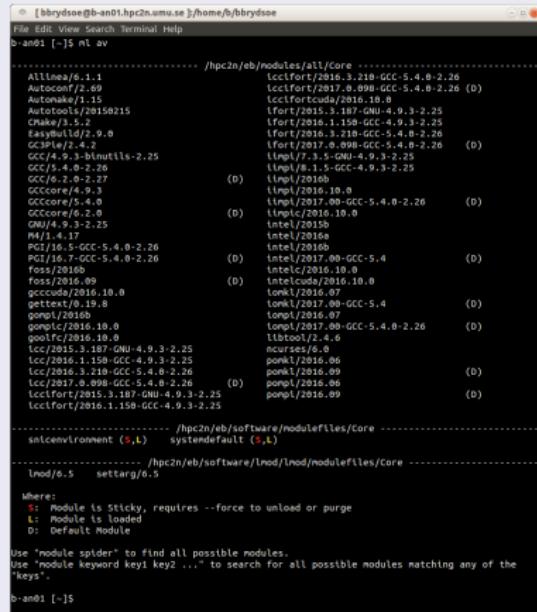
A screenshot of a terminal window titled "[bbrydsoe@b-an01.hpc2n.umu.se :~]:/home/b/bbrydsoe". The window contains two sets of module listing commands:

```
File Edit View Search Terminal Help  
b-an01 [-]$ module list  
  
Currently Loaded Modules:  
 1) snicenvironment ($S)  2) systemdefault ($S)  
  
Where:  
 $S: Module is Sticky, requires --force to unload or purge  
  
b-an01 [-]$ ml list  
  
Currently Loaded Modules:  
 1) snicenvironment ($S)  2) systemdefault ($S)  
  
Where:  
 $S: Module is Sticky, requires --force to unload or purge  
b-an01 [-]$
```

The Module System

Examples

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



A screenshot of a terminal window titled "bbrydsoe@b-an01.hpc2n.umu.se". The window displays a list of available modules, organized into several sections:

- Core Modules:** /hpc2n/eb/modules/all/Core
- Allinea/6.1.1
- Autotools/2.0
- AutoMake/1.0
- Autotools/20150215
- CMake/3.5.2
- EasyBuild/2.9.0
- GCC/4.8.2
- GCC/4.9.2-nutils-2.25
- GCC/5.1.0-2.26
- GCC/6.2.0-2.27
- GCCcore/4.9.3
- GCCcore/5.4.0
- GCCcore/6.2.0
- OpenMPI/2.2.25
- MPI/1.4.17
- PGI/16.7-GCC-5.4.0-2.26
- PGI/16.7-GCC-5.4.0-2.26
- foss/2016b
- Fortran/2016.0
- pcccuda/2016.10.0
- gettext/0.19.8
- gomp/2016b
- gomp/c/2016.10.0
- gomp/c/2016.10.0
- lfort/2016.3.187-GNU-4.9.3-2.25
- lcc/2016.3.210-GCC-5.4.0-2.26
- lcc/2017.0.098-GCC-5.4.0-2.26
- tcclFort/2015.3.187-GNU-4.9.3-2.25
- tcclFort/2016.1.158-GCC-4.9.3-2.25
- System Environment:** smicenvironment (\$,) systndefault (\$,)
- lmod/6.5 settarg/6.5
- Help:**
 - S: Module is Sticky, requires --force to unload or purge
 - L: Module is loaded
 - D: Default Module
- Usage Notes:** Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".

The Module System

Examples

```
module spider
ml spider
```



The following is a list of the modules currently available:

```
Autonconf: Autonconf/2.69
Autonconf is a malleable package of M4 macros that produce shell scripts to
automatically configure software source code packages. These scripts can adapt the
packages to many kinds of UNIX-like systems without manual user intervention.
Autonconf creates a configuration script for a package from a template file that lists
the operating system features that the package can use, in the form of M4 macro
calls. - Homepage: http://www.gnu.org/software/autonconf/
```

```
Autonmake: Autonmake/1.15
Autonmake: GNU Standards-compliant Makefile generator - Homepage:
http://www.gnu.org/software/autonmake/autonmake.html
```

```
Autotools: Autotools/20150225
This bundle collect the standard GNU build tools: Autoconf, Automake and libtool -
Homepage: http://autotools.io
```

```
Boost: Boost/1.61.0
Boost provides free peer-reviewed portable C++ source libraries. - Homepage:
http://www.boost.org/
```

```
CMake: CMake/3.5.2
CMake, the cross-platform, open-source build system. CMake is a family of tools
designed to build, test and package software. - Homepage: http://www.cmake.org
```

```
CUDA: CUDA/8.0.44
CUDA (formerly Compute Unified Device Architecture) is a parallel computing platform
and programming model created by NVIDIA and implemented by the graphics processing
units (GPUs) that they produce. CUDA gives developers access to the virtual
instruction set and memory of the parallel computational elements in CUDA GPUs. -
Homepage: https://developer.nvidia.com/cuda-toolkit
```

```
EasyBuild: EasyBuild/2.9.0
EasyBuild is a software build and installation framework written in Python that
allows you to install software in a structured, repeatable and robust way. -
Homepage: https://hpcugent.github.com/easybuild/
```

```
FFTW: FFTW/3.3.4, FFTW/3.3.5
FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in
one or more dimensions, of arbitrary input size, and of both real and complex data. -
Homepage: http://www.fftw.org
```

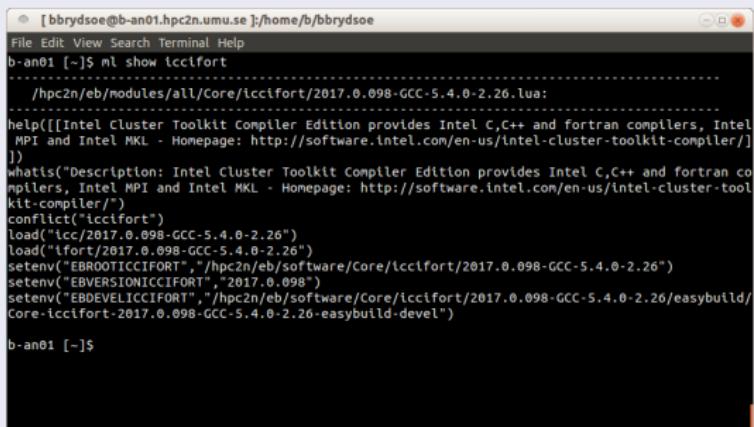
```
G3PLite: G3PLite/2.4.2
G3PLite is a Python package for running large job campaigns on diverse batch-oriented
execution environments. - Homepage: https://g3plite.readthedocs.org
```

```
GCC: GCC/4.9.3-binutils-2.25, GCC/5.4.0-2.26, GCC/6.2.0-2.27
```

The Module System

Examples

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



A screenshot of a terminal window titled "[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe". The window contains the following text:

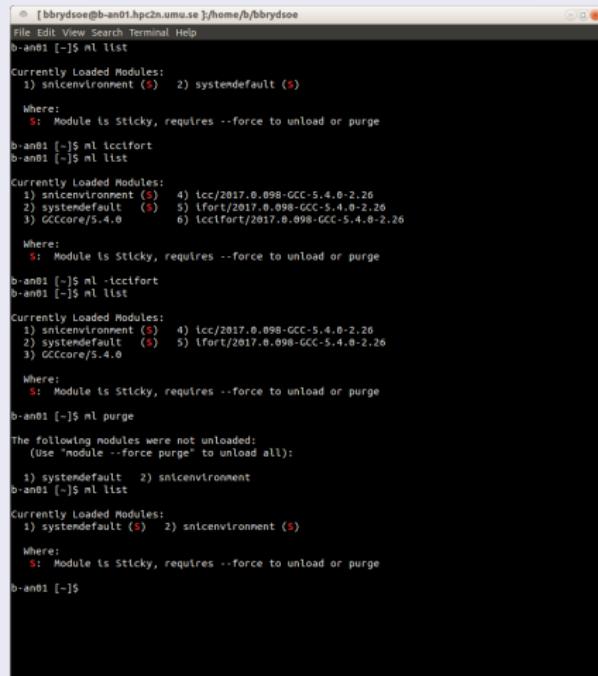
```
File Edit View Search Terminal Help
b-an01 [-]$ ml show iccifort
-----
/hpc2n/eb/modules/all/Core/iccifort/2017.0.098-GCC-5.4.0-2.26.lua:
-----
help([[Intel Cluster Toolkit Compiler Edition provides Intel C,C++ and fortran compilers, Intel
MPI and Intel MKL - Homepage: http://software.intel.com/en-us/intel-cluster-toolkit-compiler]])
whatis("Description: Intel Cluster Toolkit Compiler Edition provides Intel C,C++ and fortran co
mpilers, Intel MPI and Intel MKL - Homepage: http://software.intel.com/en-us/intel-cluster-tool
kit-compiler/")
conflict("iccifort")
load("icc/2017.0.098-GCC-5.4.0-2.26")
load("ifort/2017.0.098-GCC-5.4.0-2.26")
setenv("EBROOTICCIFORT","/hpc2n/eb/software/Core/iccifort/2017.0.098-GCC-5.4.0-2.26")
setenv("EBVERSIONICCIFORT","2017.0.098")
setenv("EBDEVELICCIFORT","/hpc2n/eb/software/Core/iccifort/2017.0.098-GCC-5.4.0-2.26/easybuild/
Core-iccifort-2017.0.098-GCC-5.4.0-2.26-easybuild-devel")

b-an01 [-]$
```

The Module System

Examples

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



A screenshot of a terminal window titled "[bbrydsoe@b-an01.hpc2n.umu.se ~]/home/b/bbrydsoe". The window displays a series of commands related to module loading and unloading:

```
File Edit View Search Terminal Help
b-an01 [-]$ ml list
Currently Loaded Modules:
  1) snicenvironment ($)
  2) systemdefault ($)

  Where:
    $: Module is Sticky, requires --force to unload or purge

b-an01 [-]$ ml iccifort
b-an01 [-]$ ml list

Currently Loaded Modules:
  1) snicenvironment ($)
  4) icc/2017.0.098-GCC-5.4.0-2.26
  2) systemdefault ($)
  5) ifort/2017.0.098-GCC-5.4.0-2.26
  3) GCCcore/5.4.0
  6) iccifort/2017.0.098-GCC-5.4.0-2.26

  Where:
    $: Module is Sticky, requires --force to unload or purge

b-an01 [-]$ ml -iccifort
b-an01 [-]$ ml list

Currently Loaded Modules:
  1) snicenvironment ($)
  4) icc/2017.0.098-GCC-5.4.0-2.26
  2) systemdefault ($)
  5) ifort/2017.0.098-GCC-5.4.0-2.26
  3) GCCcore/5.4.0

  Where:
    $: Module is Sticky, requires --force to unload or purge

b-an01 [-]$ ml purge
The following modules were not unloaded:
  (Use "module --force purge" to unload all):

  3) systemdefault  2) snicenvironment
b-an01 [-]$ ml list

Currently Loaded Modules:
  1) systemdefault ($)
  2) snicenvironment ($)

  Where:
    $: Module is Sticky, requires --force to unload or purge

b-an01 [-]$
```

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 -fopenmp <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
```

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>

- Buildenv

- After loading a compiler toolchain, load 'buildenv' and use 'ml show buildenv' to get useful linking info
 - Example, foss:

ml foss

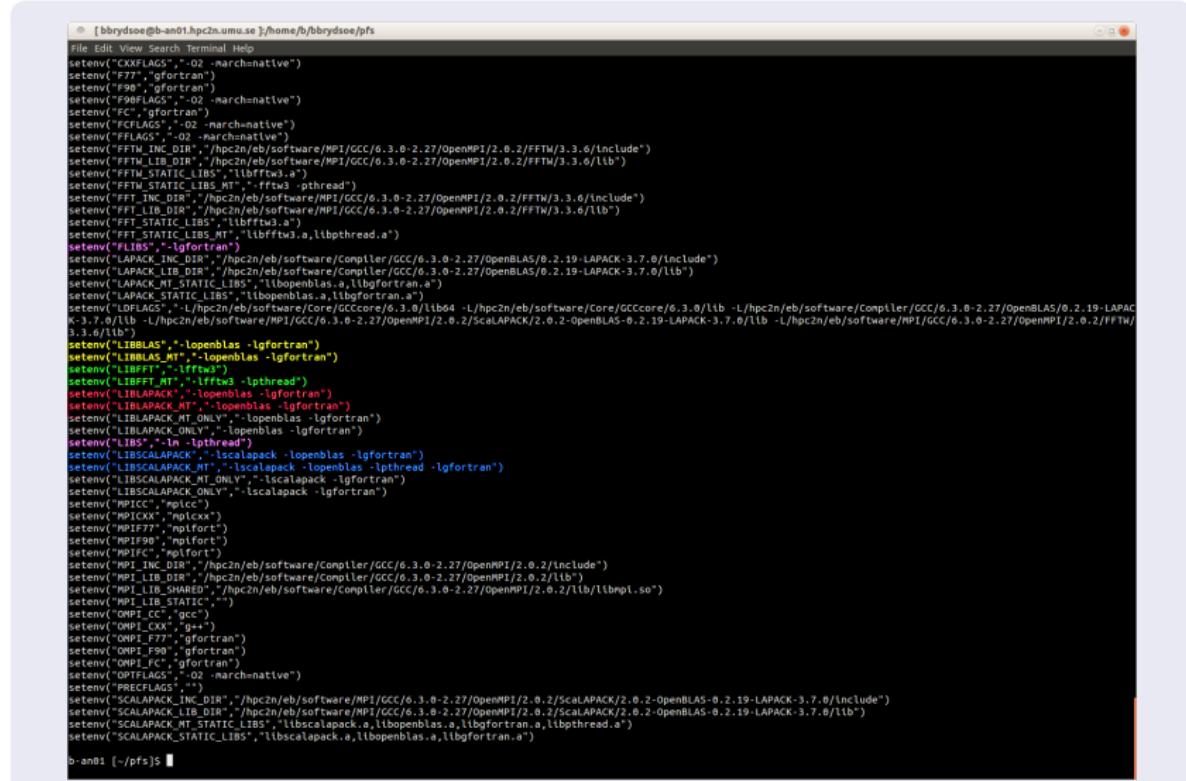
ml buildenv

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 screenshot shows a terminal window with a light gray background and a dark gray title bar. The title bar contains the text "[bbyrdsoe@b-an01.hpc2n.umu.se]:/home/b/bbyrdsoe/pfs". The main area of the terminal displays a large amount of text representing environment variables. The text is organized into several sections, each starting with 'setenv' or 'unsetenv'. The variables include paths to MPI, FFTW, LAPACK, BLAS, and SCALAPACK libraries, as well as various compiler flags like '-O2' and '-fPIC'. The text is mostly black with some color-coded segments, such as 'MPI_CXX' and 'MPI_F77'. The terminal prompt at the bottom is 'b-an01 [~/pfs]\$'.

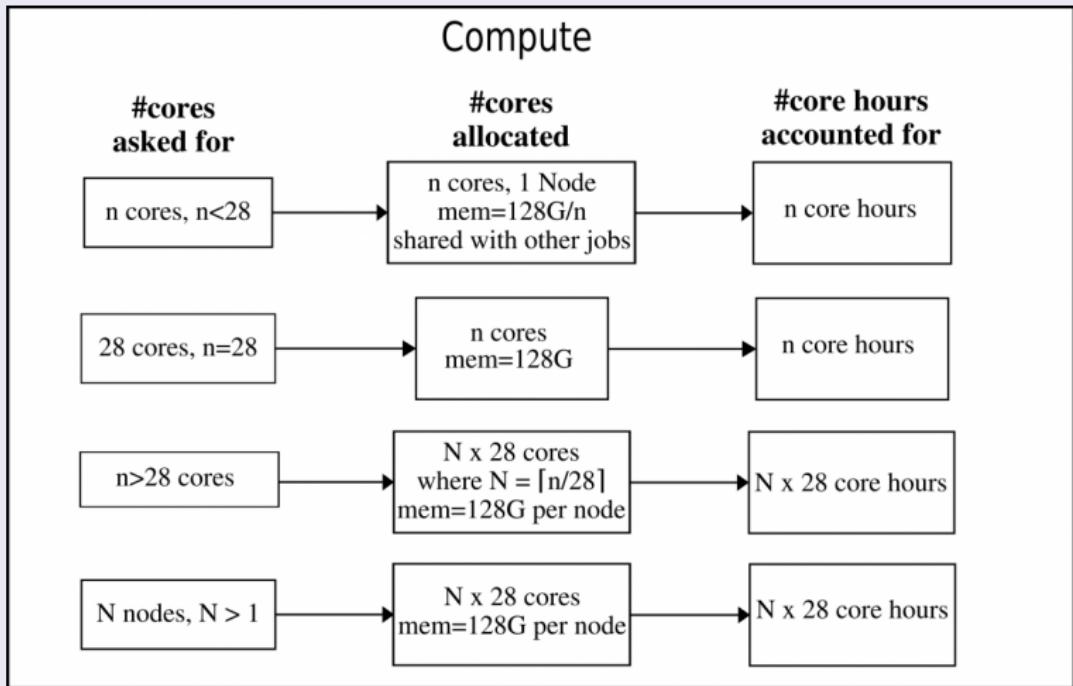
```
setenv("CXXFLAGS","-O2 -farch=native")
setenv("F77","gfortran")
setenv("F90","gfortran")
setenv("F90FLAGS","-O2 -farch=native")
setenv("FC","gfortran")
setenv("FCFLAGS","-O2 -farch=native")
setenv("FLANG","-O2 -farch=native")
setenv("FFTW_INC_DIR","/hpc2n/eb/software/MPI/GCC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/include")
setenv("FFTW_STATIC_LIBS","libfftw3.a")
setenv("FFTW_STATIC_LIBS_MT","-fftw -pthread")
setenv("FFT_INC_DIR","/hpc2n/eb/software/MPI/GCC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/include")
setenv("FFT_STATIC_LIBS","libfftw3.a")
setenv("FFT_STATIC_LIBS_MT","-libfftw3.a,libpthread.a")
setenv("FLIBS","lgfortran")
setenv("LAPACK_INC_DIR","/hpc2n/eb/software/Compiler/GCC/6.3.0-2.27/OpenBLAS/0.2.19-LAPACK-3.7.0/include")
setenv("LAPACK_LIB_DIR","/hpc2n/eb/software/Compiler/GCC/6.3.0-2.27/OpenBLAS/0.2.19-LAPACK-3.7.0/lib")
setenv("LAPACK_MT_STATIC_LIBS","libopenblas.a,libgfortran.a")
setenv("LAPACK_STATIC_LIBS","libopenblas.a,libgfortran.a")
setenv("LDFLAGS","-L/hpc2n/eb/software/Core/GCCcore/6.3.0/lib64 -L/hpc2n/eb/software/Core/GCCcore/6.3.0/lib -L/hpc2n/eb/software/Compiler/GCC/6.3.0-2.27/OpenBLAS/0.2.19-LAPACK-3.7.0/lib -L/hpc2n/eb/software/MPI/GCC/6.3.0-2.27/OpenMPI/2.0.2/OpenBLAS-0.2.19-LAPACK-3.7.0/lib -L/hpc2n/eb/software/MPI/GCC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/lib")
setenv("LIBBLAS","openblas.lgfortran")
setenv("LIBBLAS_MT","openblas.lgfortran")
setenv("LIBEFT","-lfifftw3")
setenv("LIBEFT_MT","-lfifftw3 -lpthread")
setenv("LIBBLAPACK","openblas.lgfortran")
setenv("LIBBLAPACK_MT","openblas.lgfortran")
setenv("LIBBLAPACK_MT_ONLY","-openblas -lgfortran")
setenv("LIBBLAPACK_ONLY","-openblas -lgfortran")
setenv("LIBBLAPACK_P","-lpthread")
setenv("LIBBLAPACK_P_L","-lscalapack -openblas -lgfortran")
setenv("LIBBLAPACK_MT","-lscalapack -openblas -lpthread -lgfortran")
setenv("LIBBLAPACK_MT_ONLY","-lscalapack -lgfortran")
setenv("LIBINTC","_picc")
setenv("MPI_CXX","_piccxx")
setenv("MPI_F77","_mpifort")
setenv("MPI_F90","_mpifort")
setenv("MPI_F90P","_mpifort")
setenv("MPI_F90P_P","_mpifort")
setenv("MPI_INC_DIR","/hpc2n/eb/software/Compiler/GCC/6.3.0-2.27/OpenMPI/2.0.2/include")
setenv("MPI_LIB_DIR","/hpc2n/eb/software/Compiler/GCC/6.3.0-2.27/OpenMPI/2.0.2/lib")
setenv("MPI_LIB_SHARED","/hpc2n/eb/software/Compiler/GCC/6.3.0-2.27/OpenMPI/2.0.2/lib/libmpi.so")
setenv("MPI_LIB_STATIC","");
setenv("OMPI_CC","gcc")
setenv("OMPI_CXX","g++")
setenv("OMPI_F77","gfortran")
setenv("OMPI_F90","gfortran")
setenv("OMPI_F90P","gfortran")
setenv("OMPI_F90P_P","gfortran")
setenv("OMPIFLAGS","-O2 -farch=native")
setenv("PRECLFLAGS","");
setenv("SCALAPACK_INC_DIR","/hpc2n/eb/software/MPI/GCC/6.3.0-2.27/OpenMPI/2.0.2/ScalAPACK/2.0.2-OpenBLAS-0.2.19-LAPACK-3.7.0/include")
setenv("SCALAPACK_LIB_DIR","/hpc2n/eb/software/MPI/GCC/6.3.0-2.27/OpenMPI/2.0.2/ScalAPACK/2.0.2-OpenBLAS-0.2.19-LAPACK-3.7.0/lib")
setenv("SCALAPACK_MT_STATIC_LIBS","libscalapack.a,libopenblas.a,libgfortran.a,libpthread.a")
setenv("SCALAPACK_STATIC_LIBS","libscalapack.a,libopenblas.a,libgfortran.a")
```

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>

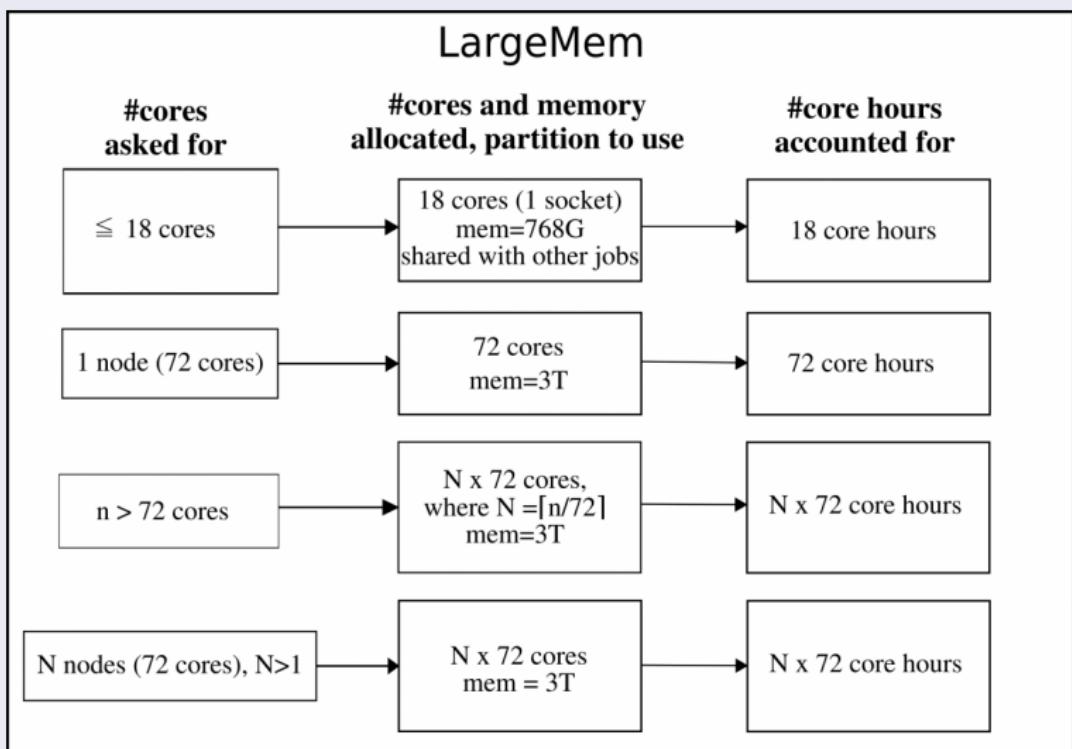
The Batch System

Accounting, Compute nodes



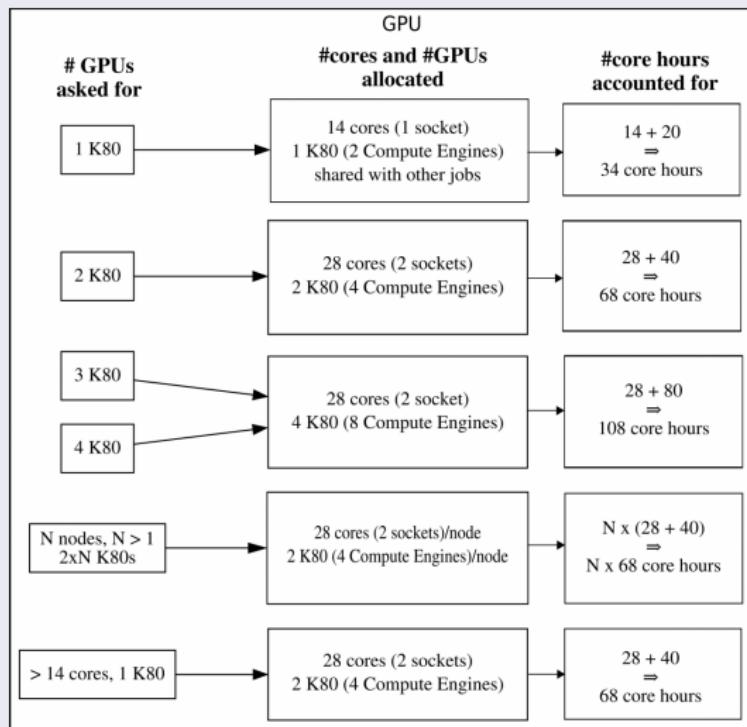
The Batch System

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
```