

Introduction to HPC2N

Birgitte Brydsø, Jerry Eriksson, and Pedro Ojeda-May

HPC2N, Umeå University

12 September 2017



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

Kebnekaise and Abisko

Abisko



- 1 328 nodes / 15744 cores (10 fat, 318 thin)
- 2 Thin: 4 AMD Opteron 6238, 12 core 2.6 GHz proc.
- 3 Fat: 4 AMD Opteron 6344, 12 core 2.6 GHz proc.
- 4 10 with 512 GB RAM/node, 318 with 128 GB RAM/node
- 5 Interconnect: Mellanox 4X QSFP 40 Gb/s
- 6 Theoretical performance: 163.74 TF
- 7 HP Linpack: 131.9 TF
- 8 Date installed: Fall 2011. Upgraded Jan 2014

Kebnekaise and Abisko

Kebnekaise



- 1 544 nodes / 17552 cores (of which 2448 are KNL)
 - 432 Intel Xeon E5-2690v4, 2x14 cores, 128 GB/node
 - 20 Intel Xeon E7-8860v4, 4x18 cores, 3072 GB/node
 - 32 Intel Xeon E5-2690v4, 2x NVidia K80, 2x14, 2x4992, 128 GB/node
 - 4 Intel Xeon E5-2690v4, 4x NVidia K80, 2x14, 4x4992, 128 GB/node
 - 36 Intel Xeon Phi 7250, 68 cores, 192 GB/node, 16 GB MCDRAM/node
- 2 399360 CUDA “cores” (80 * 4992 cores/K80)
- 3 More than 125 TB memory
- 4 Interconnect: Mellanox 56 Gb/s FDR Infiniband
- 5 Theoretical performance: 728 TF
- 6 HP Linpack: 629 TF
- 7 Date installed: Fall 2016 / Spring 2017

Using our systems

- 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 our systems

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, MobaXterm ...)
- Get an X11 server if you need graphical displays (Xming ...)
- 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>

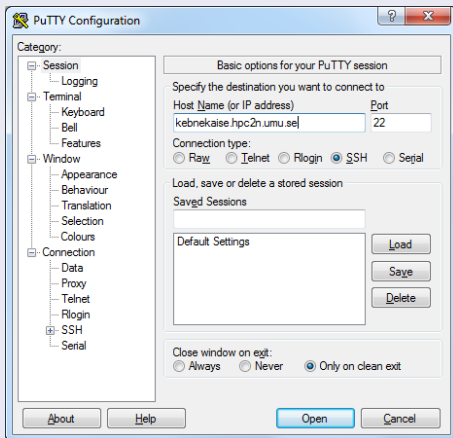
- **Mac/OSX:** Guide here:

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

Using our systems

Connecting from a Windows System with PuTTY

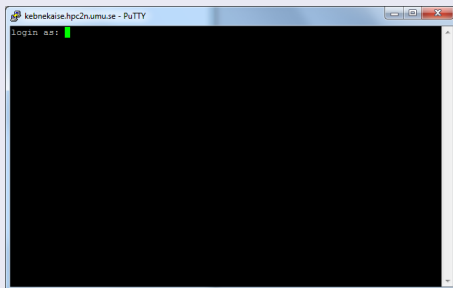
Get the Zip file (<http://www.putty.org/>) with both PuTTY, PSCP, and PSFTP. Unzip, run putty.exe



Using Kebnekaise and Abisko

Connecting from a Windows System with PuTTY

Enter your username and then your password.



Using our systems

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 (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`

The File System

There are 2 file systems

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

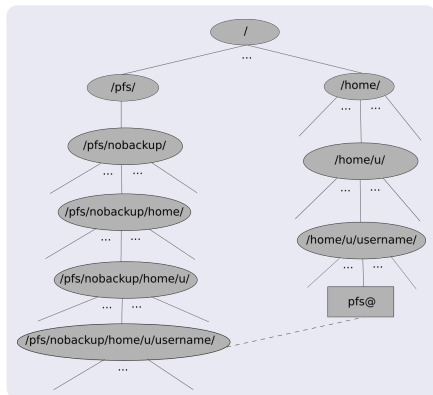
• AFS

- 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 (Lmod)

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.

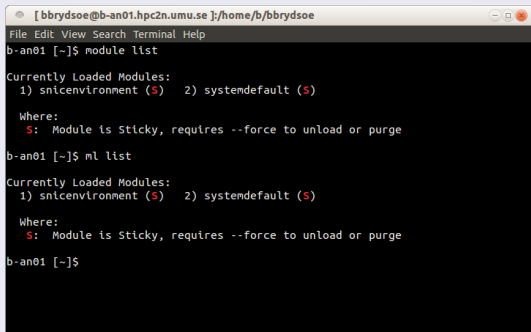
- Currently available toolchains (check `ml av` for versions):

- **GCC**: GCC only
- **gcccuda**: GCC and CUDA
- **foss**: GCC, OpenMPI, OpenBLAS/LAPACK, FFTW, ScaLAPACK
- **gimkl**: GCC, IntelMPI, IntelMKL
- **gimpi**: GCC, IntelMPI
- **gompic**: 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
```



```
[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
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
```

```
b@brydsoe@b-an01.hpc2n.su.se ~/home/h/brydsoe
File Edit View Search Terminal Help
b-an01 [-] $ ml av
----- /hpc2n/eb/modules/all/Core -----
Allinea/6.1.1          lcctfort/2016.3.210-GCC-5.4.0-2.26
Autocore/2.09        lcctfort/2017.0.098-GCC-5.4.0-2.26 (D)
Autonake/1.15        lcctfortcuda/2016.10.0
Autotools/20150215  lfort/2015.3.187-GNU-4.9.3-2.25
CMake/3.5.2          lfort/2016.1.150-GCC-4.9.3-2.25
Easybuild/2.9.0      lfort/2016.3.210-GCC-5.4.0-2.26
GCC/6.2.4.2         lfort/2017.0.098-GCC-5.4.0-2.26 (D)
GCC/4.9.3-Mnutils-2.25  llmpl/7.3.1-GNU-4.9.3-2.25
GCC/5.4.0-2.26      llmpl/8.1.5-GCC-4.9.3-2.25
GCC/6.2.0-2.27      (D) llmpl/2016b
GCCcore/4.9.3       llmpl/2016-10.0
GCCcore/5.4.0       llmpl/2017.00-GCC-5.4.0-2.26 (D)
GCCcore/6.2.0       (D) llmpct/2016.10.0
GNU/4.9.3-2.25      intel/2015b
MH/1.4.17           intel/2016a
PGI/16.5-GCC-5.4.0-2.26  intel/2016b
PGI/16.7-GCC-5.4.0-2.26 (D) intel/2017.00-GCC-5.4 (D)
Foss/2016b          intelc/2016.10.0
Foss/2016.09        intelcuda/2016.10.0
gcccuda/2016.10.0    lomkl/2016.07
gettext/10.10       lomkl/2017.00-GCC-5.4 (D)
gomp/2016b          lompt/2016.07
gomp/2016.10.0      lompt/2017.00-GCC-5.4.0-2.26 (D)
gomp/2016.10.0     llbtool/2.4.6
lcc/2015.3.187-GNU-4.9.3-2.25  mcourse/6.0
lcc/2016.1.150-GCC-4.9.3-2.25  ponkl/2016.06
lcc/2016.3.210-GCC-5.4.0-2.26  ponkl/2016.09 (D)
lcc/2017.0.098-GCC-5.4.0-2.26 (D) ponpl/2016.06
lcctfort/2015.3.187-GNU-4.9.3-2.25  ponpl/2016.09 (D)
lcctfort/2016.1.150-GCC-4.9.3-2.25
----- /hpc2n/eb/software/modulefiles/Core -----
snlcmenvironment (S,L)  systdefaul (S,L)
----- /hpc2n/eb/software/lnod/lnod/modulefiles/Core -----
lnod/0.5  settarg/0.5
Where:
S: Module is Sticky, requires --force to unload or purge
L: Module is loaded
D: Default Module
Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the
"keys".
b-an01 [-] $
```

The Module System

Examples

```
module spider
ml spider
```

```
[bbrydsoe@b-an01.hpc2n.umn.se]~/home/bbrydsoe
File Edit View Search Terminal Help
b-an01 [-] $ ml spider

-----
The following is a list of the modules currently available:
-----

Autoconf: Autoconf/2.68
Autoconf is an extensible 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.
Autoconf 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/autoconf/

Autonake: Autonake/1.15
Autonake: GNU Standards-compliant makefile generator - Homepage:
http://www.gnu.org/software/autonake/autonake.html

Autotools: Autotools/28158215
This bundle collect the standard GNU build tools: Autoconf, Autonake 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: http://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

GC3Ple: GC3Ple/2.4.2
GC3Ple is a Python package for running large job campaigns on diverse batch-oriented
execution environments. - Homepage: https://gc3ple.readthedocs.org

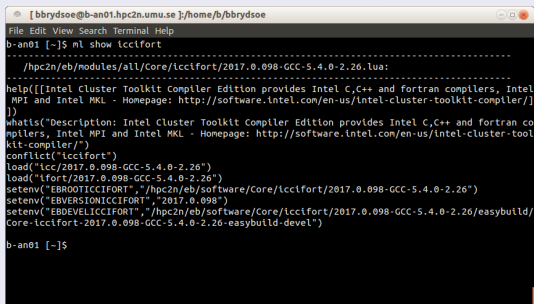
GCC: GCC/4.9.3-binutils-2.25, GCC/5.4.0-2.26, GCC/6.2.0-2.27
```

The Module System

Examples

```
module show <module>
```

```
ml show <module>
```

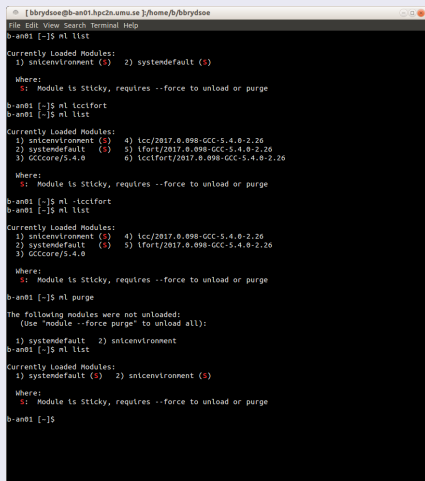


```
[bbrydsoe@b-an01.hpc2n.umu.se ~]:/home/b/brydsoe
File Edit View Search Terminal Help
b-an01 [-]$ ml show iccfort
-----
/hpc2n/eb/modules/all/Core/iccfort/2017.0.098-GCC-5.4.0-2.26.lua:
-----
help([[Intel Cluster Toolkit compiler Edition provides Intel C,C++ and fortran compilers, Intel
MPI and Intel MKL - Homepage: http://software.intel.com/en-us/intel-cluster-toolkit-compiler/
]])
whats("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-toolkit-compiler/
")
conflict("iccfort")
load("icc/2017.0.098-GCC-5.4.0-2.26")
load("ifort/2017.0.098-GCC-5.4.0-2.26")
setenv("EBROOTICCFORT", "/hpc2n/eb/software/Core/iccfort/2017.0.098-GCC-5.4.0-2.26")
setenv("EBVERSIONICCFORT", "2017.0.098")
setenv("EBDEVELICCFORT", "/hpc2n/eb/software/Core/iccfort/2017.0.098-GCC-5.4.0-2.26/easybuild/
Core-iccfort-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>
```



```
[bbrydsoe@b-an01.hpc2n.ums.se] /home/b/bbrydsoe  
File Edit View Search Terminal Help  
b-an01 [-] $ ml list  
Currently Loaded Modules:  
1) snicenvironment (s) 2) systendefault (s)  
Where:  
s: Module is Sticky, requires --force to unload or purge  
b-an01 [-] $ ml tccifort  
b-an01 [-] $ ml list  
Currently Loaded Modules:  
1) snicenvironment (s) 4) tcc/2017.0.098-GCC-5.4.0-2.26  
2) systendefault (s) 5) lfport/2017.0.098-GCC-5.4.0-2.26  
3) GCCcore/5.4.0 6) tccifort/2017.0.098-GCC-5.4.0-2.26  
Where:  
s: Module is Sticky, requires --force to unload or purge  
b-an01 [-] $ ml -tccifort  
b-an01 [-] $ ml list  
Currently Loaded Modules:  
1) snicenvironment (s) 4) tcc/2017.0.098-GCC-5.4.0-2.26  
2) systendefault (s) 5) lfport/2017.0.098-GCC-5.4.0-2.26  
3) GCCcore/5.4.0  
Where:  
s: 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):  
1) systendefault 2) snicenvironment  
b-an01 [-] $ ml list  
Currently Loaded Modules:  
1) systendefault (s) 2) snicenvironment (s)  
Where:  
s: Module is Sticky, requires --force to unload or purge  
b-an01 [-] $
```

Some examples

- MPI C program:
 - Intel compilers, Intel MPI:
ml iimpi
mpicc <program.c> -o <outfile>
 - GCC compilers, OpenMPI:
ml gomp
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/version`

`-L${MKLR00T}/lib/intel64 -lmkl_intel_ilp64 \
-lmkl_sequential -lmkl_core -lpthread -lm -ldl`
 - GCC compilers, OpenBLAS/LAPACK:
`ml foss/version`
`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/version`
`gcc -o program program.o -lscalapack -lopenblas`
 - Intel, MKL, Intel MPI:
`ml intel/version`
`-L${MKLRROOT}/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 (add relevant version):

```
ml foss/version  
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

```
[brydsoe@b-an01.hpc2n.umu.se ~]$ cat /home/b/brydsoe/pfs
file Edit View Search Terminal Help
setenv("CXXFLAGS", "-O2 -marchnative")
setenv("F77", "gfortran")
setenv("F90", "gfortran")
setenv("F90FLAGS", "-O2 -marchnative")
setenv("FC", "gfortran")
setenv("FCFLAGS", "-O2 -marchnative")
setenv("FFLAGS", "-O2 -marchnative")
setenv("FFTW_INC_DIR", "/hpc2n/eb/software/MPICC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/Include")
setenv("FFTW_LIB_DIR", "/hpc2n/eb/software/MPICC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/lib")
setenv("FFTW_STATIC_LIBS", "libfftw3.a")
setenv("FFTW_STATIC_LIBS_MT", "libfftw3-threads")
setenv("FFT_INC_DIR", "/hpc2n/eb/software/MPICC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/Include")
setenv("FFT_LIB_DIR", "/hpc2n/eb/software/MPICC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/lib")
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("LODLFLAGS", "-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/MPICC/6.3.0-2.27/OpenMPI/2.0.2/FFTW/3.3.6/lib")
setenv("LIBBLAS", "-lopenblas -lgfortran")
setenv("LIBBLAS_MT", "-lopenblas -lgfortran")
setenv("LIBFFT", "-lfftw3")
setenv("LIBFFT_MT", "-lfftw3 -lpthread")
setenv("LIBLAPACK", "-lopenblas -lgfortran")
setenv("LIBLAPACK_MT", "-lopenblas -lgfortran")
setenv("LIBLAPACK_ONLY", "-lopenblas -lgfortran")
setenv("LIBLAPACK_MT_ONLY", "-lopenblas -lgfortran")
setenv("LIBS", "-ln -lpthread")
setenv("LIBSCALAPACK", "-lscalapack -lopenblas -lgfortran")
setenv("LIBSCALAPACK_MT", "-lscalapack -lopenblas -lpthread -lgfortran")
setenv("LIBSCALAPACK_MT_ONLY", "-lscalapack -lgfortran")
setenv("LIBSCALAPACK_ONLY", "-lscalapack -lgfortran")
setenv("MPICC", "mpicc")
setenv("MPICXX", "mpicxx")
setenv("MPIF77", "mpifort")
setenv("MPIF90", "mpifort")
setenv("MPIC", "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_FC", "gfortran")
setenv("OPTFLAGS", "-O2 -marchnative")
setenv("PREFLAGS", "")
setenv("SCALAPACK_INC_DIR", "/hpc2n/eb/software/MPICC/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/MPICC/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")
b-an01 ~ [~/pfs]$
```

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, Abisko

- Physically, a socket is 12 cores, but for SLURM allocation purposes a socket is 6 cores (a NUMA node)
- Thus allocation is in groups of 6 cores (one NUMA island). This also means 6 cores is the smallest unit you can allocate.
- This is how your project is charged, depending on how many cores you ask for:

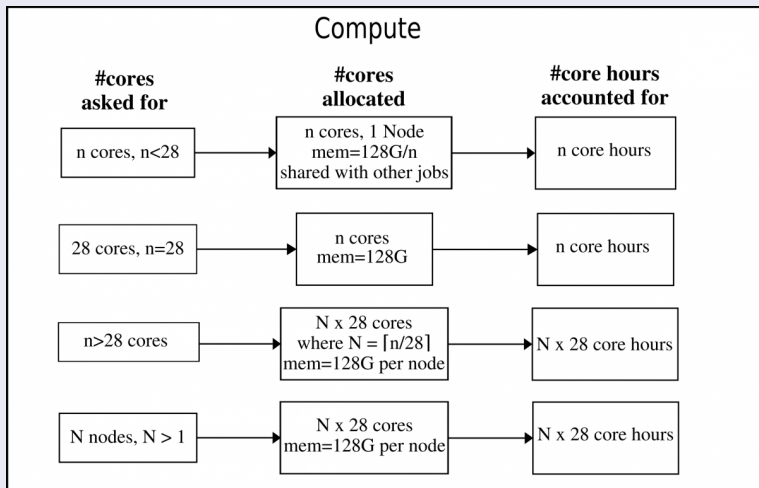
You ask for	Cores you get	Project is charged
1 core	6 cores	6 cores
6 cores	6 cores	6 cores
7 cores	12 cores	12 cores
c cores	$\text{ceil}(c/6)$ cores	$\text{ceil}(c/6)$ cores

If you request resources using **#SBATCH -c** you request c cores per task, and SLURM only allocates cores on a single node.

If you request resources using **#SBATCH -n** you request tasks which can be allocated on multiple nodes.

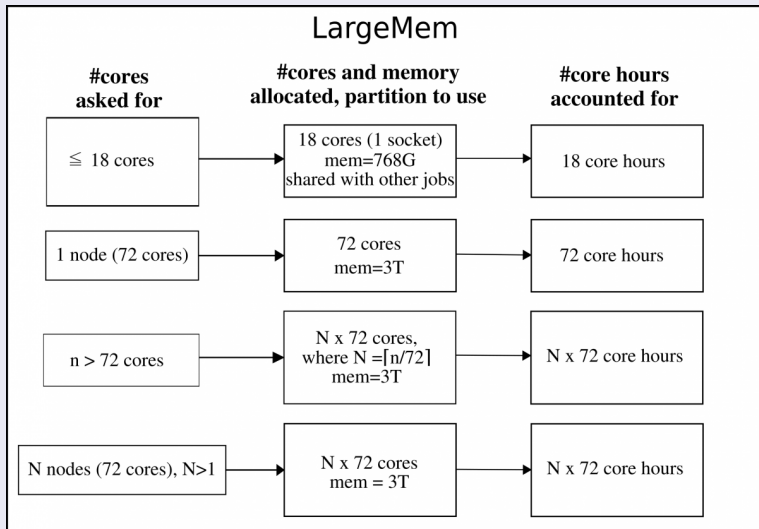
The Batch System

Accounting, Compute nodes, Kebnekaise



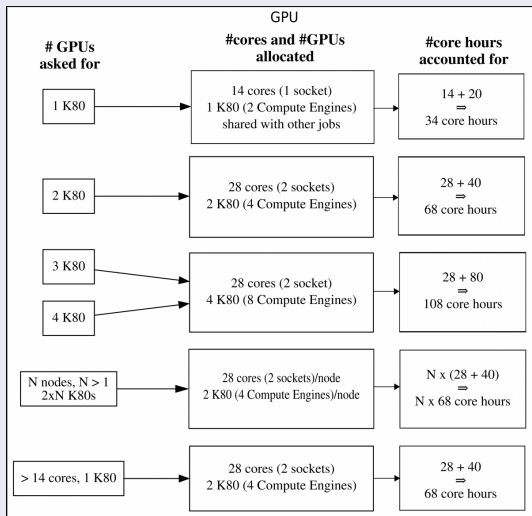
The Batch System

Accounting, largemem nodes, Kebnekaise



The Batch System

Accounting, GPU nodes, Kebnekaise



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 SNIC2017-3-81
# 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.  module purge
ml foss/2017b

./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 SNIC2017-3-81
#SBATCH -n 14
#SBATCH --time=00:05:00
##SBATCH --exclusive
#SBATCH --reservation=hpc2n-intro

module purge
ml foss/2017b

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 [~/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 SNIC2017-3-81
#SBATCH -n 14
#SBATCH --time=00:05:00
#SBATCH --reservation=hpc2n-intro

module purge
ml foss/2017b

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 srun hostname:"
srun /bin/hostname

srun ./mpi_hello
```