Introduction to the module system and compiler toolchains

Birgitte Brydsö, Pedro Ojeda-May, Jerry Eriksson

Umeå University, 2016-11-08

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 - Kebnekaise

- See which modules exists

  `ml spider`

- Modules available without loading a compiler and/or mpi toolchain

  `ml av (or module avail)`

- See which modules are currently loaded

  `ml (or module list)`

- Example: print more information about a module, here iccifort

  `ml show iccifort/2017.1.132-GCC-5.4.0-2.262017`

  `(or module show iccifort/2017.1.132-GCC-5.4.0-2.26)`

The module system - Kebnekaise

- Example: loading the intel compilers, iccifort

  `ml iccifort/2017.1.132-GCC-5.4.0-2.26`

  `(or module load iccifort/2017.1.132-GCC-5.4.0-2.26)`
- Example: Unload the above module

  `ml -iccifort/2017.1.132-GCC-5.4.0-2.26`

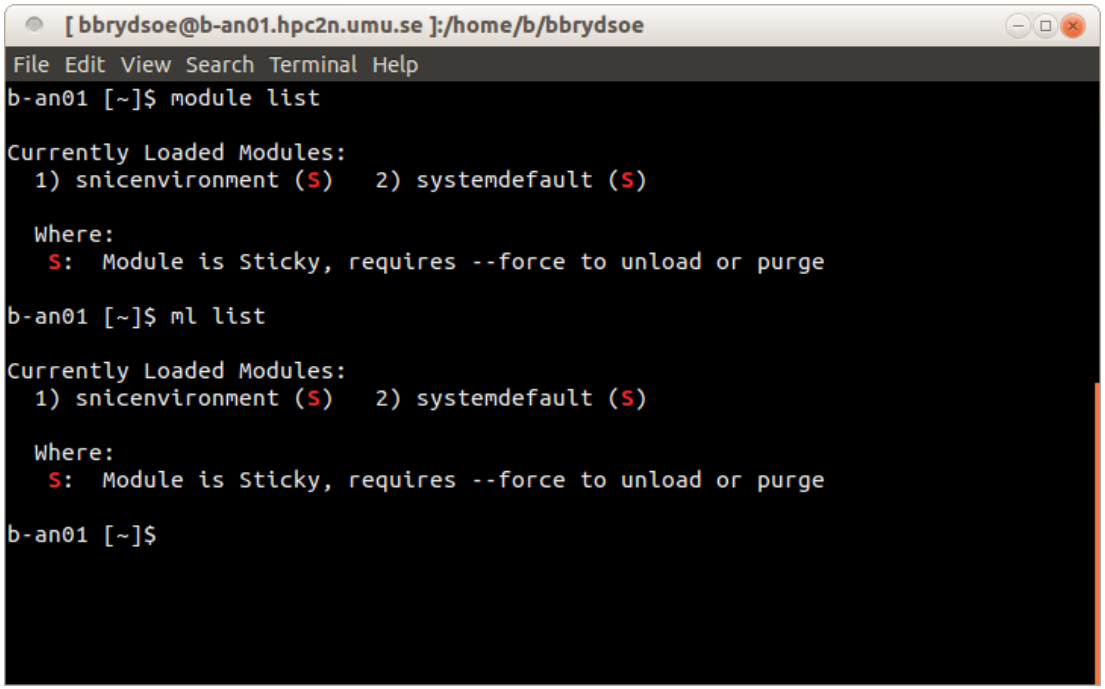
  `(or module unload iccifort/2017.1.132-GCC-5.4.0-2.26)`
- Example: loading a compiler toolchain, here for GCC

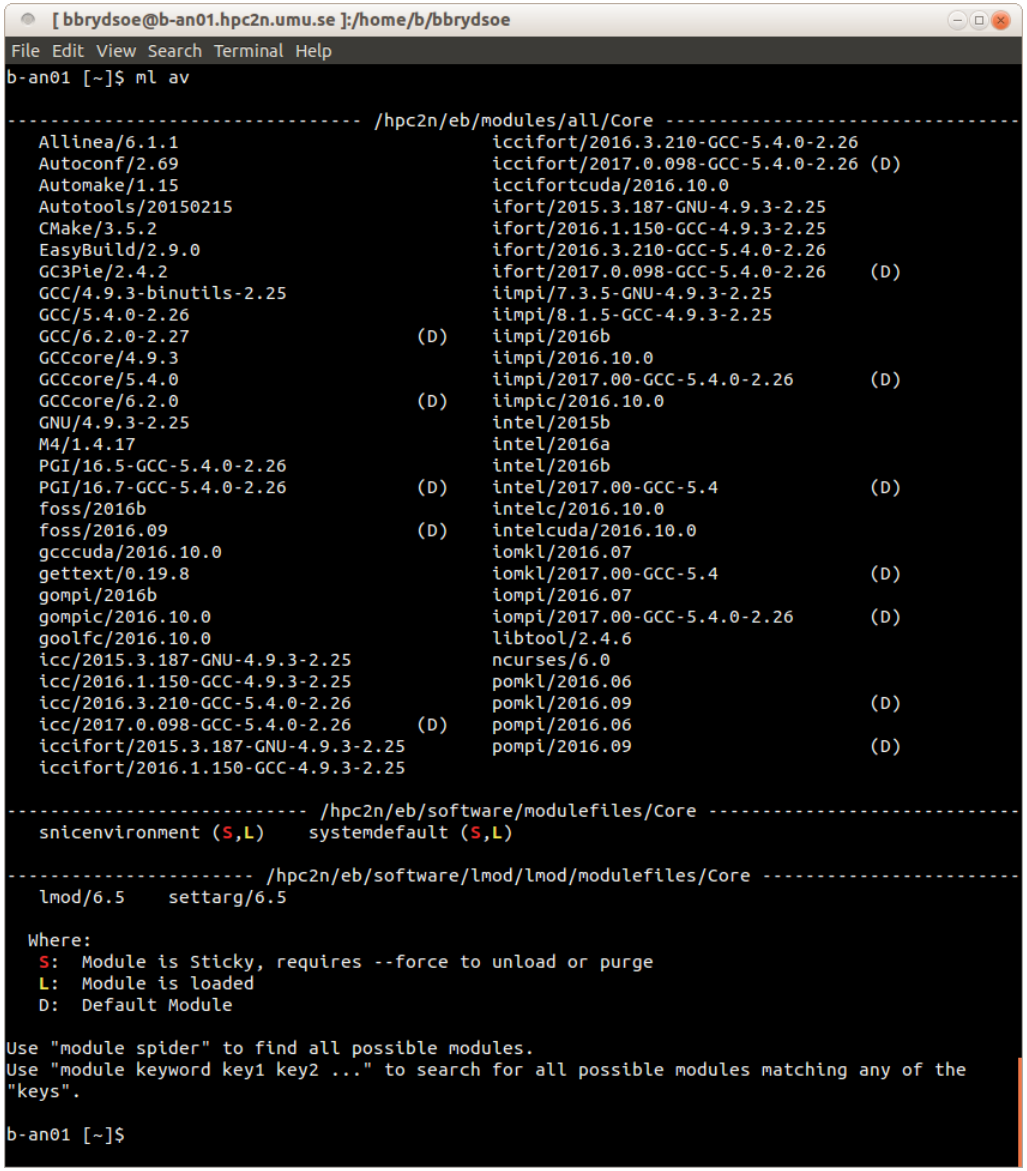
  `ml foss/2016.09 (or module load foss/2016.09)`

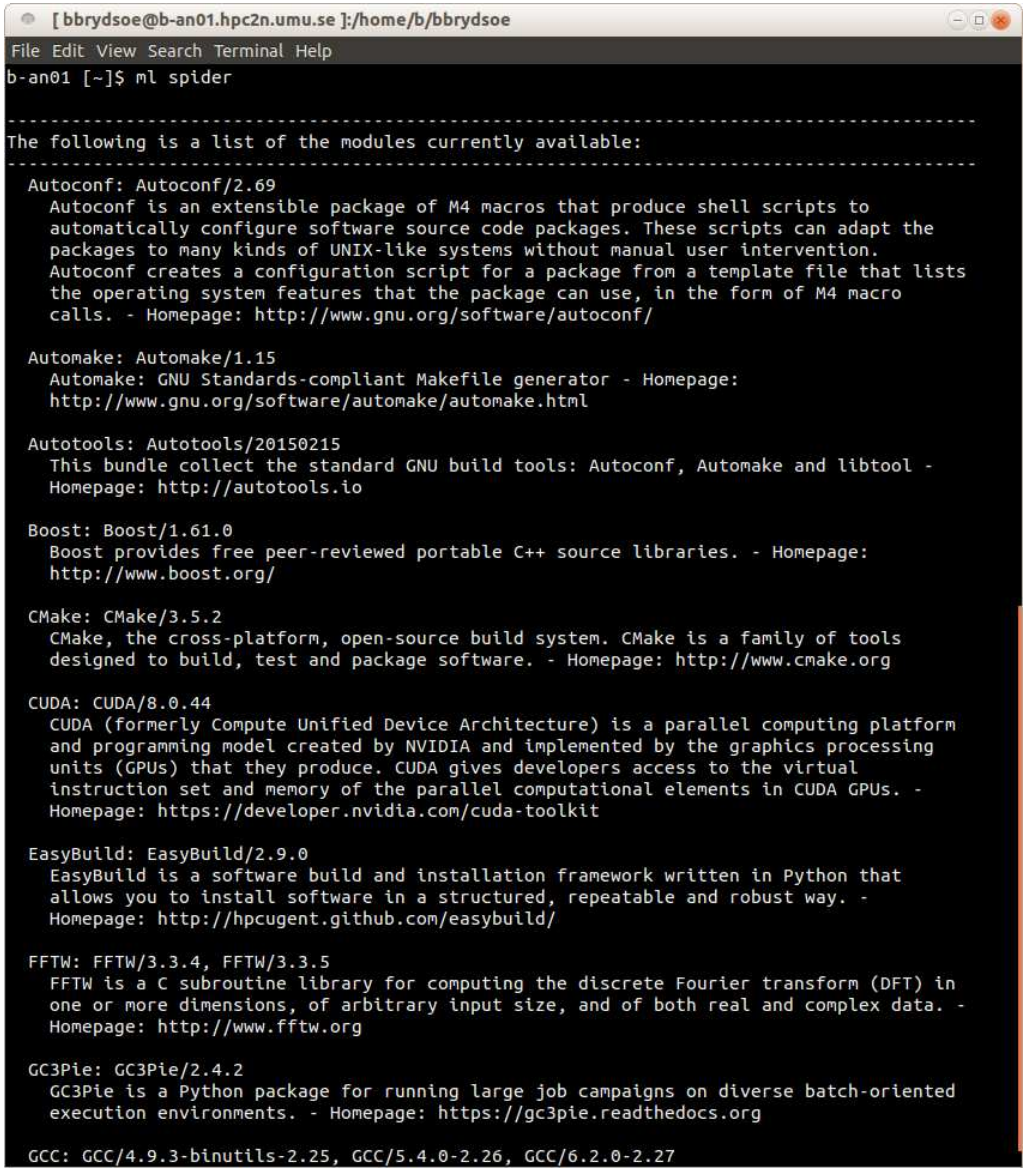
Compiler toolchains - Kebnekaise

Compiler toolchains load bundles of software making up a complete environment for compiling / using a specific prebuilt software. Usually includes a compiler suite, an MPI version, BLAS, LAPACK, ScaLapack, and FFTW versions.
Currently available toolchains:

- **GCC**: GCC only
- **foss**: GCC, OpenMPI, OpenBLAS/LAPACK, FFTW, ScaLAPACK
- **gompi**: GCC, OpenMPI
- **gimpi**: GCC, IntelMPI
- **gimkl**: GCC, IntelMPI, IntelMKL
- **gompic**: GCC, OpenMPI, CUDA
- **goolfc**: gompic, OpenBLAS/LAPACK, FFTW, ScaLAPACK
- **iccifort**: icc, ifort
- **iimpi**: icc, ifort, IntelMPI
- **intel**: icc, ifort, IntelMPI, IntelMKL

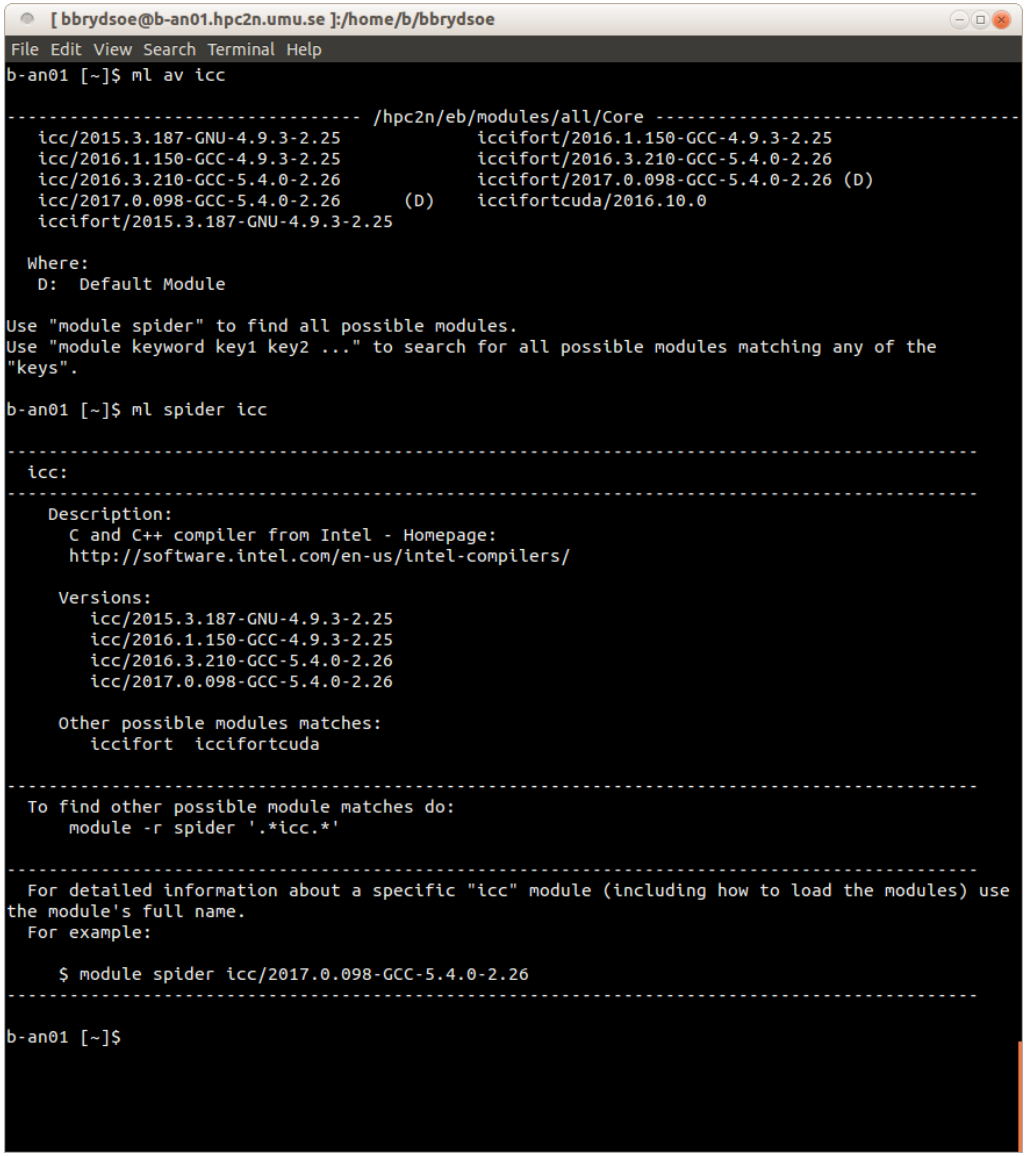

[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
File Edit View Search Terminal Help
b-an01 [~]$ module list
Currently Loaded Modules:
1) snicenvironment (S) 2) systemdefault (S)
Where:
S: Module is Sticky, requires --force to unload or purge
b-an01 [~]$ ml list
Currently Loaded Modules:
1) snicenvironment (S) 2) systemdefault (S)
Where:
S: Module is Sticky, requires --force to unload or purge
b-an01 [~]$

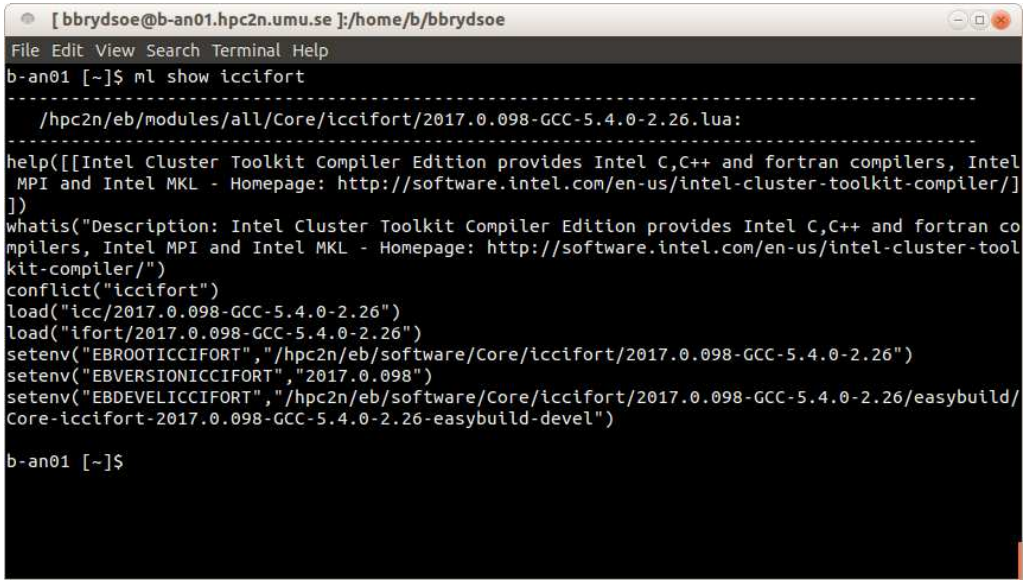

[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
File Edit View Search Terminal Help
b-an01 [~]$ ml av
/hpc2n/eb/modules/all/Core
Allinea/6.1.1
Autoconf/2.69
Automake/1.15
Autotools/20150215
CMake/3.5.2
EasyBuild/2.9.0
GC3Pie/2.4.2
GCC/4.9.3-binutils-2.25
GCC/5.4.0-2.26
GCC/6.2.0-2.27 (D)
GCCcore/4.9.3
GCCcore/5.4.0
GCCcore/6.2.0 (D)
GNU/4.9.3-2.25
M4/1.4.17
PGI/16.5-GCC-5.4.0-2.26
PGI/16.7-GCC-5.4.0-2.26 (D)
foss/2016b
foss/2016.09 (D)
gcccuda/2016.10.0
gettext/0.19.8
gompi/2016b
gompic/2016.10.0
goolfc/2016.10.0
icc/2015.3.187-GNU-4.9.3-2.25
icc/2016.1.150-GCC-4.9.3-2.25
icc/2016.3.210-GCC-5.4.0-2.26
icc/2017.0.098-GCC-5.4.0-2.26 (D)
iccifort/2015.3.187-GNU-4.9.3-2.25
iccifort/2016.1.150-GCC-4.9.3-2.25
iccifort/2016.3.210-GCC-5.4.0-2.26
iccifort/2017.0.098-GCC-5.4.0-2.26 (D)
iccifortcuda/2016.10.0
ifort/2015.3.187-GNU-4.9.3-2.25
ifort/2016.1.150-GCC-4.9.3-2.25
ifort/2016.3.210-GCC-5.4.0-2.26
ifort/2017.0.098-GCC-5.4.0-2.26 (D)
iimpi/7.3.5-GNU-4.9.3-2.25
iimpi/8.1.5-GCC-4.9.3-2.25
iimpi/2016b
iimpi/2016.10.0
iimpi/2017.00-GCC-5.4.0-2.26 (D)
iimpic/2016.10.0
intel/2015b
intel/2016a
intel/2016b
intel/2017.00-GCC-5.4 (D)
intelc/2016.10.0
intelcuda/2016.10.0
iomkl/2016.07
iomkl/2017.00-GCC-5.4 (D)
iompi/2016.07
iompi/2017.00-GCC-5.4.0-2.26 (D)
libtool/2.4.6
ncurses/6.0
pomkl/2016.06
pomkl/2016.09 (D)
pompi/2016.06
pompi/2016.09 (D)
/hpc2n/eb/software/modulefiles/Core
snicenvironment (S,L)
systemdefault (S,L)
/hpc2n/eb/software/lmod/lmod/modulefiles/Core
lmod/6.5
settarg/6.5
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 [~]$

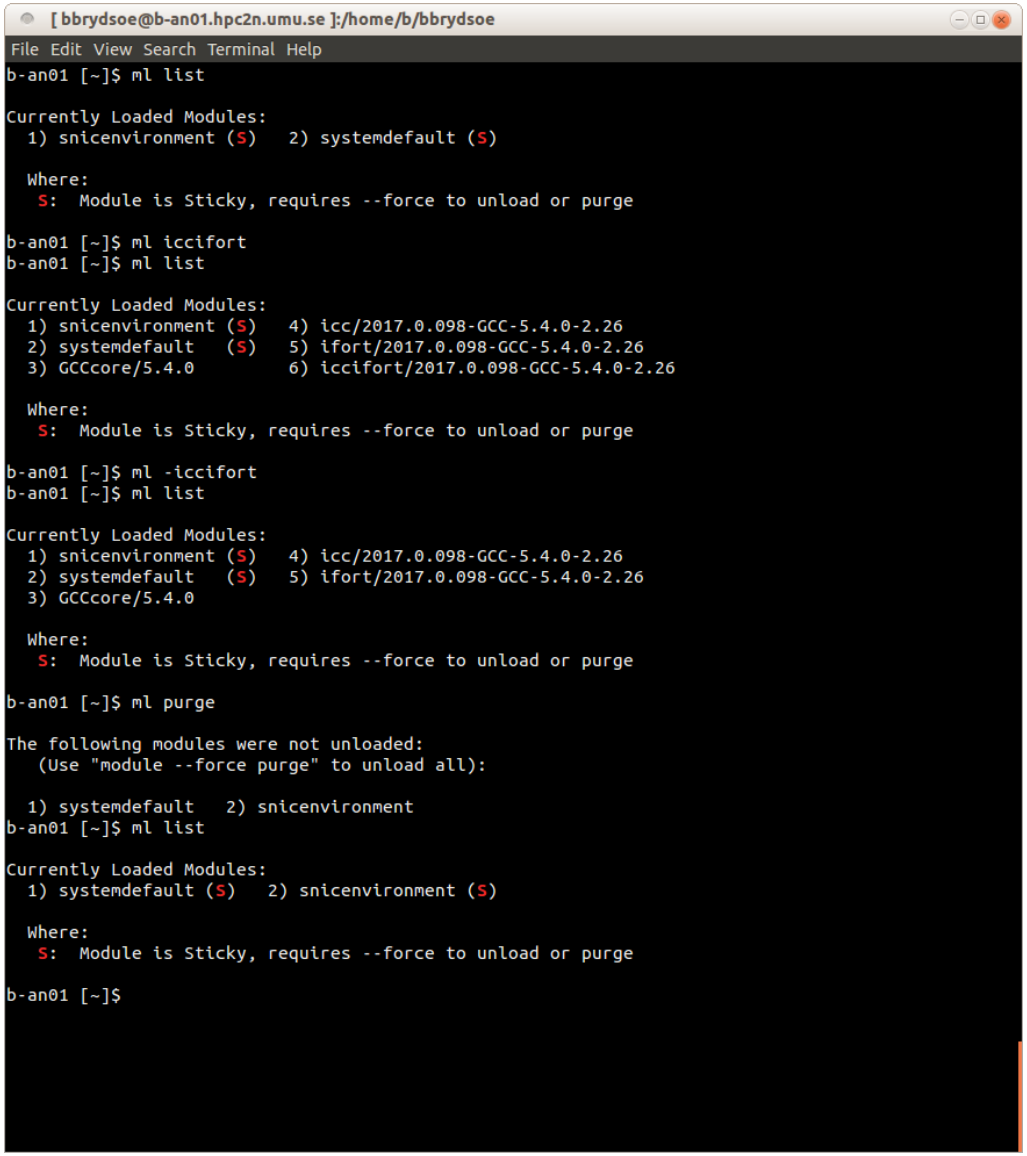

[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
File Edit View Search Terminal Help
b-an01 [~]$ ml spider

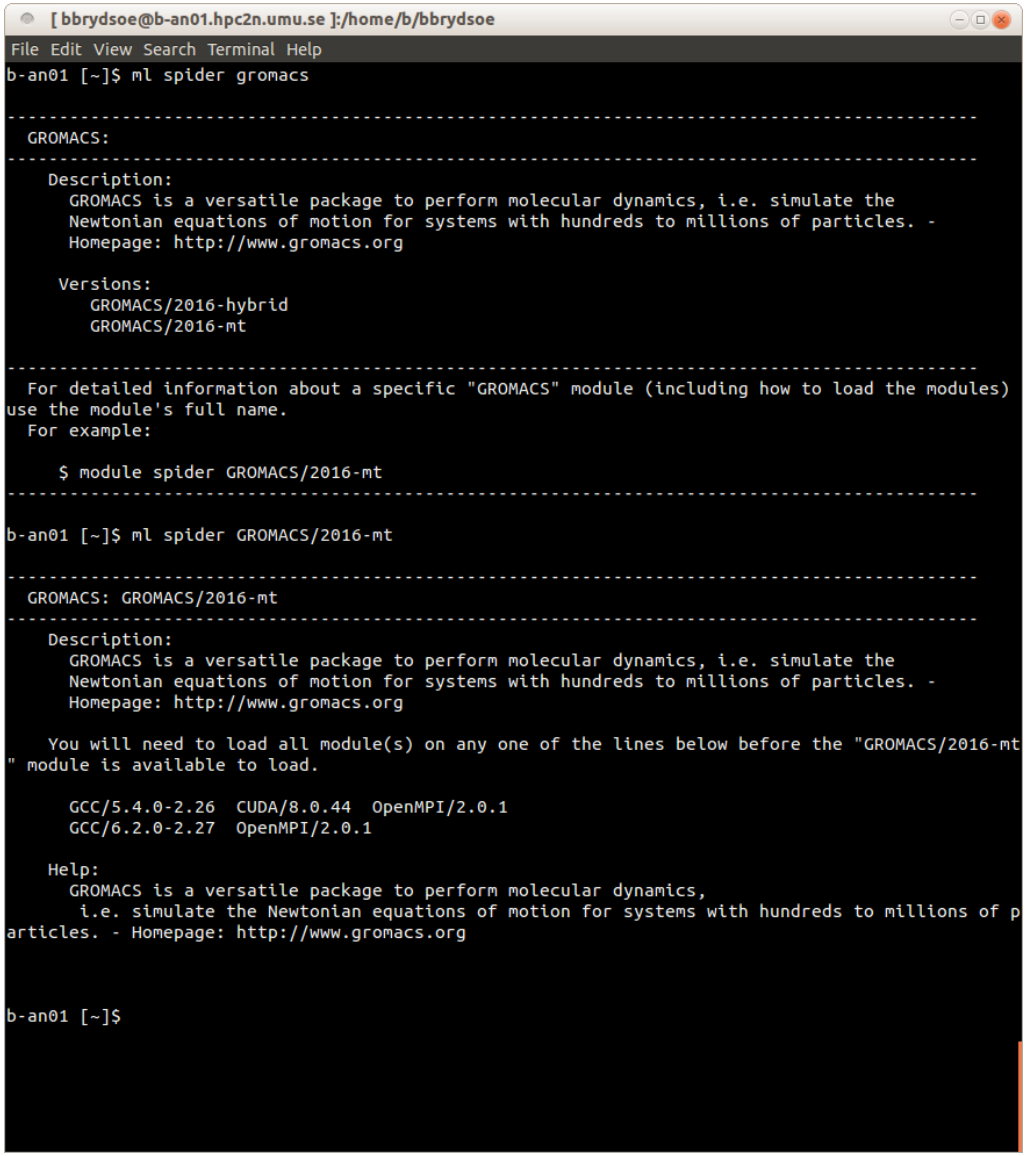
The following is a list of the modules currently available:

Autoconf: Autoconf/2.69
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/
Automake: Automake/1.15
Automake: GNU Standards-compliant Makefile generator - Homepage: http://www.gnu.org/software/automake/automake.html
Autotools: Autotools/20150215
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: 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
GC3Pie: GC3Pie/2.4.2
GC3Pie is a Python package for running large job campaigns on diverse batch-oriented execution environments. - Homepage: https://gc3pie.readthedocs.org
GCC: GCC/4.9.3-binutils-2.25, GCC/5.4.0-2.26, GCC/6.2.0-2.27


[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
File Edit View Search Terminal Help
b-an01 [~]$ ml av icc
/hpc2n/eb/modules/all/Core
icc/2015.3.187-GNU-4.9.3-2.25
icc/2016.1.150-GCC-4.9.3-2.25
icc/2016.3.210-GCC-5.4.0-2.26
icc/2017.0.098-GCC-5.4.0-2.26 (D)
iccifort/2015.3.187-GNU-4.9.3-2.25
iccifort/2016.1.150-GCC-4.9.3-2.25
iccifort/2016.3.210-GCC-5.4.0-2.26
iccifort/2017.0.098-GCC-5.4.0-2.26 (D)
iccifortcuda/2016.10.0
Where:
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 [~]$ ml spider icc
icc:
Description:
C and C++ compiler from Intel - Homepage:
http://software.intel.com/en-us/intel-compilers/
Versions:
icc/2015.3.187-GNU-4.9.3-2.25
icc/2016.1.150-GCC-4.9.3-2.25
icc/2016.3.210-GCC-5.4.0-2.26
icc/2017.0.098-GCC-5.4.0-2.26
Other possible modules matches:
iccifort iccifortcuda
To find other possible module matches do:
module -r spider '.*icc.*'
For detailed information about a specific "icc" module (including how to load the modules) use the module's full name.
For example:
$ module spider icc/2017.0.098-GCC-5.4.0-2.26
b-an01 [~]$


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


[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
File Edit View Search Terminal Help
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 [~]$ ml iccifort
b-an01 [~]$ ml list
Currently Loaded Modules:
1) snicenvironment (S) 4) icc/2017.0.098-GCC-5.4.0-2.26
2) systemdefault (S) 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:
S: Module is Sticky, requires --force to unload or purge
b-an01 [~]$ ml -iccifort
b-an01 [~]$ ml list
Currently Loaded Modules:
1) snicenvironment (S) 4) icc/2017.0.098-GCC-5.4.0-2.26
2) systemdefault (S) 5) ifort/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) systemdefault 2) snicenvironment
b-an01 [~]$ ml list
Currently Loaded Modules:
1) systemdefault (S) 2) snicenvironment (S)
Where:
S: Module is Sticky, requires --force to unload or purge
b-an01 [~]$


[bbrydsoe@b-an01.hpc2n.umu.se]:/home/b/bbrydsoe
File Edit View Search Terminal Help
b-an01 [~]$ ml spider gromacs
GROMACS:
Description:
GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the
Newtonian equations of motion for systems with hundreds to millions of particles. -
Homepage: http://www.gromacs.org
Versions:
GROMACS/2016-hybrid
GROMACS/2016-mt
For detailed information about a specific "GROMACS" module (including how to load the modules)
use the module's full name.
For example:
$ module spider GROMACS/2016-mt
b-an01 [~]$ ml spider GROMACS/2016-mt
GROMACS: GROMACS/2016-mt
Description:
GROMACS is a versatile package to perform molecular dynamics, i.e. simulate the
Newtonian equations of motion for systems with hundreds to millions of particles. -
Homepage: http://www.gromacs.org
You will need to load all module(s) on any one of the lines below before the "GROMACS/2016-mt
" module is available to load.
GCC/5.4.0-2.26 CUDA/8.0.44 OpenMPI/2.0.1
GCC/6.2.0-2.27 OpenMPI/2.0.1
Help:
GROMACS is a versatile package to perform molecular dynamics,
i.e. simulate the Newtonian equations of motion for systems with hundreds to millions of p
articles. - Homepage: http://www.gromacs.org
b-an01 [~]$

Example - getting started

Connecting from Linux or macOS

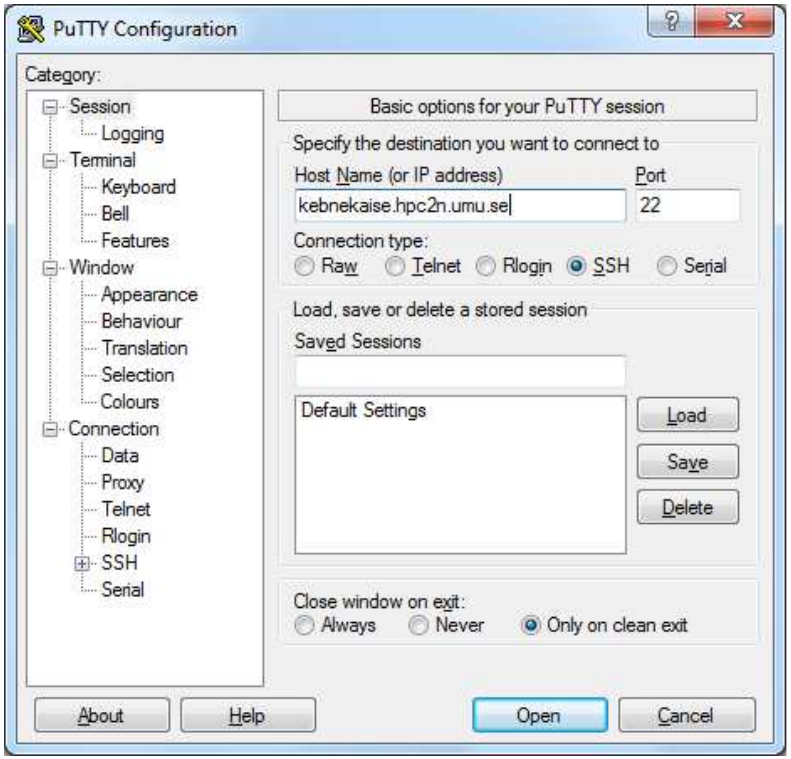
- Open a terminal window
- Type (change `username` to the right value)

  `ssh -X username@kebnekaise.hpc2n.umu.se`
- Enter your password.
- If you need to reset the password, go here:

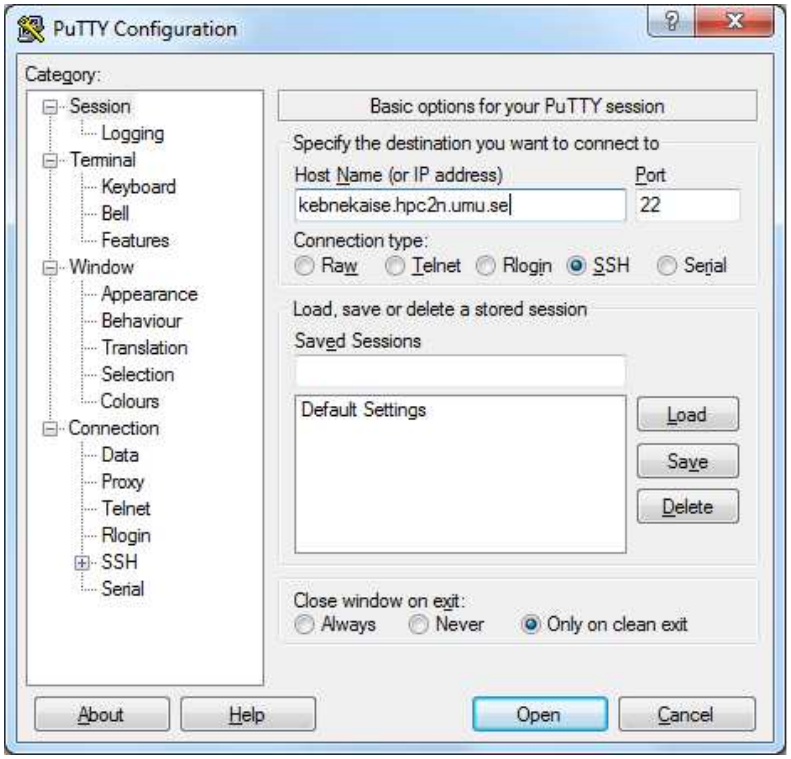
  https://www.hpc2n.umu.se/forms/user/suprauth?action=pwreset

Connecting from a Windows System - PuTTY

- Go to `http://www.chiark.greenend.org.uk/~sgtatham/putty/`.
- Follow the link there to Download PuTTY.
- Get the Zip file with both PuTTY, PSCP, and PSFTP. Unzip, run putty.exe

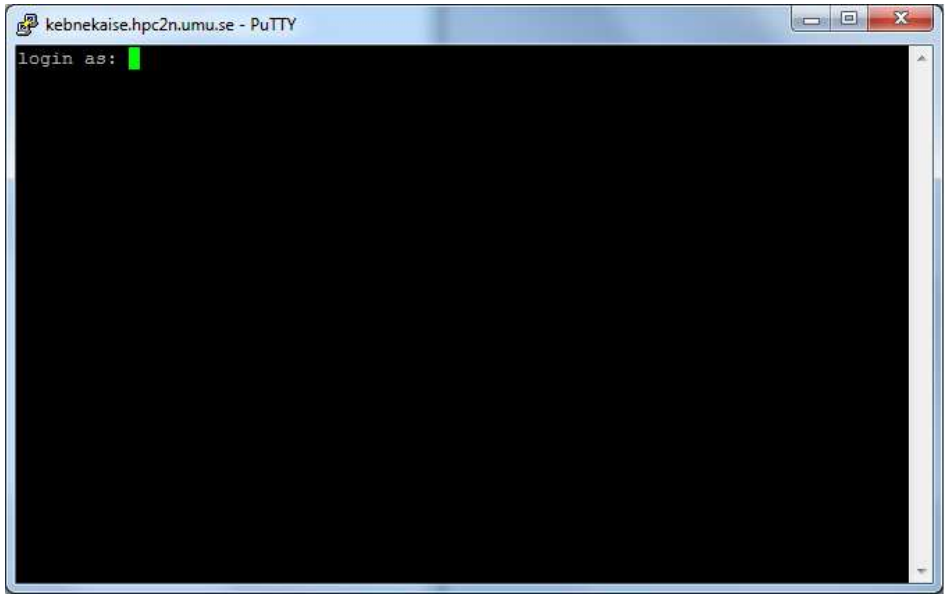


Connecting from a Windows System - PuTTY



Connecting from a Windows System - PuTTY

- Logging on:



Getting a job to run - simple example

- `cd /pfs/nobackup/$HOME`
- `nano hello-mpi.c`

```
#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();

}
```

- To save and exit: `Ctrl-x`

Getting a job to run - simple example

- Load the needed modules. Here just the GCC compilers
- and MPI libraries (using OpenMPI for this):
- `ml gompi`
- Let us see that we have what we expect:
- `ml`
- Compile the C/MPI program:
- `mpicc mpi_hello.c -o mpi_hello`

Getting a job to run - simple example. Job script

```
#!/bin/bash
# Change to your own project id!
#SBATCH -A SNIC2016-1-450
#SBATCH -n 14
#SBATCH --time=00:10:00

module purge
module add gompi

mpirun ./mpi_hello
```

- Submitting: sbatch $< jobscript >$

Job status

- Check the job's status:
  - ◇ `scontrol show job <job id>`
- Show a list of your jobs:
  - ◇ `squeue -u <username>`
- Delete job:
  - ◇ `scancel <job id>`

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)
  - ◇ `#SBATCH -p bigmem` (Abisko)

Job output - example

```
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!
```

Requesting GPU nodes

- Currently there is no separate queue for the GPU nodes
- You request them 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

Compiling and linking with libraries

Compiling and linking with libraries

- MPI C program
  - ◇ Intel, Intel MPI:

    ```
    ml iimpi
    mpicc <program> -o <outfile.name>
    ```

  - ◇ GCC, OpenMPI:

    ```
    ml gompi
    mpicc <program> -o <outfile.name>
    ```

- OpenMP Fortran program
  - ◇ Intel:

    ```
    ml iccifort
    ifort -qopenmp <program> -o <outfile.name>
    ```

  - ◇ GCC:

    ```
    ml GCC
    gfortran -fopenmp <program> -o <outfile.name>
    ```

Compiling and linking with libraries - continued

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

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

  - ◇ GCC, OpenBLAS/LAPACK:

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

Compiling and linking with libraries - continued

- 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
    ```

Linking with Intel

- Very useful for linking with Intel and Intel MKL:
- `https://software.intel.com/en-us/articles/intel-mkl-link-line-advisor`

Questions?

- http://www.hpc2n.umu.se/
- support@hpc2n.umu.se