Introduction to Abisko

Jerry Eriksson, Mikael Rännar and P. Ojeda-May

HPC2N,
Umeå University,
901 87, Sweden.

April 23, 2015
Table of contents

1 Why?
2 How to apply
   - Projects
   - Accounts
3 Support
4 Using Abisko
   - Connecting
   - Modules
   - File Systems
   - Batch System
5 Hands-on
Why?

Computations take too long time
- Use an already parallelized software
- Doing it yourself
  - Doing it yourself
    - Run each instance on a separate core
    - If inside a loop, maybe suitable for simple parallelization
  - More complex problems requires more explicit parallel programming

Use your computer for other things

Requires a lot of memory (up to 256 GB main memory)

Solve larger (more interesting/realistic) problems
Get a login at SUPR (https://supr.snic.se/)

Create a proposal for a project
(at least: Title, Abstract, Resource Usage, Classification, length of project (1-12 months), which resource(s) you wish to use, and the amount of core hours/month you need.)

Get an account at HPC2N
Small level requests

- Max 5000 core hours/month/resource (Abisko)
- Can be submitted at any time (short abstract)
- Applications handled locally at the SNIC centers
- For small projects and new groups that want to gain experience in using HPC systems
- The PI must be employed at a Swedish university (e.g., PhD student or higher)
Medium level requests

- Up to 160000 core hours/month/resource (Abisko)
- Can be submitted at any time
- Applications handled locally at the SNIC centers and assesses the feasibility of using the requested resources
- Evaluated once a month
- The PI must be a senior scientist in Swedish academia
Large level requests

- Above the medium level (160000 on Abisko)
- Calls for proposals for large level allocations are issued twice a year by SNAC
- Applications are evaluated by SNAC and they also decides on the allocations, based on scientific merit, need for the resources, efficient use of the resources, and impact
- The PI must be a senior scientist in Swedish academia
User Account

- Submit an account application using our online form
  - Name and affiliation
  - Which project you are taking part in (if any)
  - Choose a user name

- Print the final form twice
  - Sign and send one to us together with a copy of your passport (do NOT send by email)
  - Keep one (your initial password is on it)
User Support

- www.hpc2n.umu.se
  - Quick-start guides
  - How to access, compile, and submit
  - Installed software
    - Descriptions and how to use them at HPC2N
- support@hpc2n.umu.se
  - Problems
  - Requests
User Support

HPC2N - High Performance Computing Center North

About News Events Resources System status Support

Contacting Support/Help
Quick-start guide
Accounts »
Access »
FAQ
Environment »
File system »
Software »
Compiling »
Batch systems »

Search

- Apply for project HPC resources
- Apply for a user account

Events
- Course: Introduction to Distributed Memory Programming and MPI, 23 April
- Course: Introduction to Linux and Abisko, 22 April
- SeSE: High Performance Computing II, April 13-17

News
- SNIC: Call for Large allocations open. Deadline 27 April 2015.

HPC2N flyer

High Performance Computing Center North (HPC2N) is a national center for Scientific and Parallel Computing.

We are a collaboration between universities and research institutes who form a competence network for high performance and parallel computing, grid and cloud computing, scientific visualization and virtual reality (VR), as well as effective mass-storage solutions, in Northern Sweden. The primary objective of the center is to raise the national level of competence in HPC and to transfer HPC knowledge and technology to new users in academia and industry.

Today, the use of HPC include compute-intensive as well as data- and communication-intensive applications. HPC2N is one of six national centers funded by the Swedish National Infrastructure for Computing (SNIC), a metacentre under The Swedish Research Council.

HPC2N has a Board of Directors consisting of a Chairman and six members, representing the HPC2N partners and the industry.

Contact information can be found on the 'Contact Us' page.

The partners of HPC2N are:
User Support

- Meetings with individuals or groups
  - To see how can HPC2N be of help
  - Help to get started
  - Help to parallelize
- HPC2N Think Tank - Open house
- Courses (0.5 - 3 days)
  - Introduction on how to use our system
  - Parallel programming (MPI, OpenMP)
  - ...

Jerry Eriksson, Mikael Rännar and P. Ojeda-May

Introduction to Abisko
Why? How to apply Support Using Abisko Hands-on

Abisko

332 nodes/15936 cores
10 fat nodes (512 GB RAM), 318 thin nodes (128 GB RAM)
CPUs: (thin) 4 × AMD Opteron 6238 (Interlagos) 12 core (2.6 GHz)
CPUs: (fat) 4 × AMD Opteron 6344 (Abu Dhabi) 12 core (2.6 GHz)
Interconnect: Infiniband QDR, 40Gb/s, Mellanox
Installed 2011
Application software: Abinit, Ansys, DDT, Espresso, Gamess, Gaussian, Gromacs, HDF5, Matlab, NetCDF, NWChem, Octave, PETSc, R, Siesta, VASP, WRF, ...

Num. and Comm. libraries: BLACS, FFTW, BLAS, LAPACK, ScaLAPACK, ACML, Intel MKL, ParMETIS, RECSY, SLICOT, ...

MPI: OpenMPI, Intel MPI

Other software on request
Connecting from a Windows System

You need an ssh client to connect.

- PuTTY
- Cygwin

If you want to open graphical displays, you need an X11 server

- Xming
- Cygwin

Transferring files (sftp or scp)

- WinSCP
- FileZilla (only sftp)
- PSCP/PSFTP
Connecting from a UNIX/Linux System

- **Login with ssh:**
  
  `local> ssh username@abisko.hpc2n.umu.se`

- **If you want to open graphical displays, you need to enable X11 Forwarding:**
  
  `local> ssh -X username@abisko.hpc2n.umu.se`

- **Use scp for file transfer:**
  
  `local> scp username@abisko.hpc2n.umu.se:file /tmp`
  
  `local> scp file username@abisko.hpc2n.umu.se:file`
Modules

- Many versions of software packages
- Use a tool called modules
  - Can choose a combination of libraries and compilers that will work together
  - Changes the environment
  - User guide and man-page
Some useful module commands

- `help` list all module commands
- `show` display information on a module
- `add` add a module to the environment
- `rm` remove a module from the environment
- `list` list currently activated modules
- `avail` list all modules that exist on the system

Examples

- `module add pgi`
  will add the latest version of the Portland compilers
- `module add openmpi/pgi`
  will add the latest version of openmpi (suitable for the current machine) that was built with the Portland compilers
There are 2 file systems

**AFS**
- Your home directory
- Backed up regularly
- NOT accessible by the batch system

**PFS**
- Parallel file system
- NO BACKUP
- Accessable by the batch system
PFS

- Offers high performance when accessed from the nodes
- To create soft link from your home directory to your corresponding home on the parallel file system
  
  ```
  ln -s /pfs/nobackup$HOME $HOME/pfs
  ```

- Then if you use
  ```
  cd pfs
  ```
  from your home directory you will end up in your "parallel" home directory
Batch System (SLURM)

- Large/parallel programs, run through the batch system
- Keeps track of available system resources
- Takes care of scheduling jobs of multiple users, running tasks simultaneously
- Enforces local system resource usage and job scheduling policies
- Users submit to a queue (running, idle, blocked)
Job script

```bash
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00

module add openmpi/psc
srun ./parallel_prog args
```

- Submitting:
  `sbatch <jobscript>`
- Show the job queue:
  `squeue [-u username]`
- Delete a job:
  `scancel <jobid>`
Job script

```
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00
module add openmpi/psc
srun ./parallel_prog args
```

Your account (-A)

- The account is your project id
- Low priority if not set
- You can find your project id by running: projinfo
Job script

```
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00

module add openmpi/psc
srun ./parallel_prog args
```

Number of tasks (-n)

- The number of tasks is for the most cases the number of processes you want to start.
- The default value is one
- e.g. number of MPI tasks
- e.g. number of serial programs
Job script

```
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00

module add openmpi/psc
srun ./parallel_prog args
```

Number of cores per task (-c)
- For multi threaded applications (OpenMP/pthreads/...)
- indicates the number of cores each task can use
- The default value is one
- Maximum is 48
Job script

```bash
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00
module add openmpi/psc
srun ./parallel_prog args
```

The run/wallclock time

D-HH:MM:SS

- Runtime (wall clock time) of your job
- Try to estimate correctly
  - Hard limit
  - Shorter jobs are more likely to fit into slots of unused space faster.

Jerry Eriksson, Mikael Rännar and P. Ojeda-May
Introduction to Abisko
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00

module add openmpi/psc
srun ./parallel_prog args

Load modules needed or other things. (This is for your program that is compiled with the PathScale compiler and the OpenMPI library.)
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00

module add openmpi/psc

srun ./parallel.prog args

Run your MPI application using srun

- Starts the required number of processes
- Note! If your program is serial it will start many instances
Run your multi threaded application on 36 cores. Change the marked lines with:

```
#SBATCH -c 32

export OMP_NUM_THREADS=36

./my_OpenMP_program args
```
Job script

```bash
#!/bin/bash
#SBATCH -A SNICYYYY-XX-NN
#SBATCH -n 48
#SBATCH --time=01:00:00

module add openmpi/psc
srun ./parallel prog args
```

Output

- Put stdout into the file `<jobid>.out`
  ```bash
  #SBATCH --output=%J.out
  ```
- Put stderr into the file `<jobid>.err`
  ```bash
  #SBATCH --error=%J.err
  ```
- By default both to slurm-<jobid>.out

Input

- Use file.txt as stdin
  ```bash
  #SBATCH --input=file.txt
  ```
Simple hands-on

1. Log in to Abisko.
2. Go to the parallel file system.
3. Copy executable (threaded program) and submit file:
   ```
   cp ~mr/Public/mandel .
   cp ~mr/Public/submit .
   ```
4. Put the submit file into the batch queue.
5. Look at the queue.
6. Where did the output from the program go?
7. Run on 12 cores instead. Does it run twice as fast?
8. The program creates a picture on a file (mandel.ppm) as output. Try looking at it (e.g. using the command display).