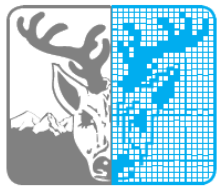
LUND UNIVERSITY



HPC2N, UmU

Compiling and running OpenMP codes on Kebnekaise at HPC2N

Pedro Ojeda & Joachim Hein

High Performance Computing Center North &
Lund University

Overview

- HPC2N's Kebnekaise system
- The module environment
- Building OpenMP executables on HPC2N systems
- Executing OpenMP jobs on HPC2N system

Kebnekaise@ HPC2N

- 432 compute nodes
- 52 compute-skylake nodes
- 20 large-memory nodes
- 32 2xK80 GPU nodes
- 4 4xK80 GPU nodes
- 10 2xV100 GPU nodes
- 36 KNL nodes

- Compute nodes (Broadwell)
  - Intel Xeon E5-2690 v4 processors
  - 14 cores per processor (2 sockets)
  - 28 cores per node
  - 128 GB memory per node

Connecting to Kebnekaise: Thinlinc client

- Download and install the client from: http://www.cendio.com/downloads/clients/
- Launch the client
- Enter kebnekaise-tl.hpc2n.umu.se in the server field.
- Just for the first time, go to “Options” button in the client dialog box, then “Screen” and uncheck “Full screen mode”
- Enter your user-id & password
- Click [Connect]

Detailed descriptions:

https://www.hpc2n.umu.se/documentation/guides/thinlinc

Connecting to Kebnekaise: Command prompt

- Kebnekaise is a Linux based machine
- Connect to it via: `ssh`
- Address: `kebnekaise.hpc2n.umu.se`
- Linux/Mac OSX: run `ssh -Y username` in terminal window
- Windows: install putty to run `ssh -Y username`
- Correct password

- Detailed description:
- https://www.hpc2n.umu.se/documentation/guides/mac-connection (Mac)
- https://www.hpc2n.umu.se/documentation/guides/windows-connection (Windows)
- https://www.hpc2n.umu.se/documentation/guides/linux-connection (LINUX)

Building OpenMP executables

- Use standard compilers (e.g. Intel, Gnu, …) to compile the source
- Enable OpenMP via compiler flag
- HPC2N systems use modules to make these steps convenient

Hands on: compiling OpenMP code

- Load the compiler

Compiler	Kebnekaise example
GNU 10.2.0	module load foss/2020b
intel/2020a	module load intel/2020a

- Examples for compiling code

Case	Command
C, GCC compiler	gcc -O3 -march=native -fopenmp -o prog prog.f90
F90, GNU compiler	gfortran -O3 -march=native -fopenmp -o prog prog.f90
C, Intel compiler	icc –qopenmp -O3 -xHost -o prog prog.c
F90, Intel compiler	ifort -qopenmp -O3 -xHost -o prog prog.f90

Rem: specify other flags as usual, e.g. optimisation

OpenMP and CMake

- Good OpenMP support for C, C++ and Fortran in CMake

- The cmake module `FindOpenMP` sets the variables
  - `OpenMP_C_FLAGS` - flags to add to the C compiler
  - `OpenMP_CXX_FLAGS` - flags to add to the CXX compiler
  - `OPENMP_FOUND` - true if openmp is detected

- Old version do not have Fortran flags!

CMake example for C

```
...

if(OpenMP_Build)
 find_package(OpenMP)

 if (OPENMP_FOUND)
  set (CMAKE_C_FLAGS "${CMAKE_C_FLAGS} ${OpenMP_C_FLAGS}")
 endif (OPENMP_FOUND)
endif(OpenMP_Build)

...
```

CMake example for Fortran

```
...

if(OpenMP_Build)
 find_package(OpenMP)

 if (OPENMP_FOUND)
  set (CMAKE_Fortran_FLAGS "${CMAKE_Fortran_FLAGS} ${OpenMP_Fortran_FLAGS}")
 endif (OPENMP_FOUND)
endif(OpenMP_Build)

...
```

Running OpenMP programs

- Make sure the compiler module used for building the executable is loaded (shared libraries!)
- Parallel jobs need to run inside the batch submission system
- Batch submission systems are vital to get consistent runtimes
- Expect a similar set-up on any well managed service

Standard OpenMP job
Run executable: `processor_omp`

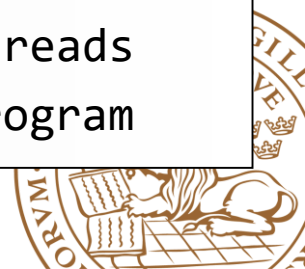
```
#!/bin/bash
#SBATCH -c 5                          # number of cpus per task
#SBATCH -t 00:05:00                   # job-time – here 5 min
#SBATCH -J data_process               # name of job
#SBATCH -o process_omp_%j.out         # output file
#SBATCH -e process_omp_%j.err         # error messages

cat $0
ml purge  > /dev/null 2>&1
module load <compiler/version>                  # replace as needed
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK # set nr. threads
./processor_omp                                 # run the program
```

OpenMP job in course reservation
Run executable: processor_omp

```
#!/bin/bash
#SBATCH -c 5                        # number of cpus per task
#SBATCH -t 00:05:00                 # job-time – here 5 min
#SBATCH -J data_process             # name of job
#SBATCH –A openmpkurs               # course project
#SBATCH --reservation=openmpkurs # access reserved nodes
#SBATCH -o process_omp_%j.out    # output file
#SBATCH -e process_omp_%j.err    # error messages

cat $0
ml purge  > /dev/null 2>&1
module load <compiler/version>                  # replace as needed
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK # set nr. threads
./processor_omp                                 # run the program
```



Running different number of threads than cores specified

- Can be useful for benchmarking
  - No disturbance from other jobs
  - **Remark:** You pay more cores than you use

- E.g.: to specify to run four threads specify

```
export OMP_NUM_THREADS=4
```

- Specify before starting your executable

- Make sure you asked for **more processors** than specified

Submission, queue monitoring and modification

INTERACTING WITH SLURM

Submission with `sbatch`

- Use sbatch to submit your job script to the job-queue

- Example:

```
[fred@alarik Timetest]$ sbatch runjob.sh
Submitted batch job 7197
```

- Submit script "`runjob.sh`"
- Successful submission returns a job-id number (7197)

Monitoring the queue with squeue

- Use **squeue** to monitor the job queue

```
JOBID  PARTITION    NAME    USER  ST    TIME  NODES NODELIST(REASON)
  7303    snic   hybrid_n   fred  PD    0:00     32 (Priority)
  7302    snic   hybrid_n   fred  PD    0:00     32 (Priority)
  7301    snic   hybrid_n   fred  PD    0:00     32 (Resources)
  7304    snic   preproce   karl  PD    0:00      6 (Priority)
  7300    snic   hybrid_n   fred   R    0:24     32 an[001-032]
  7305    snic   preproce   karl   R    0:37      6 an[081-086]
  7306    snic   hybrid_n   fred   R    0:37      6 an[081-086]
  7307    snic   testsimu   sven   R    0:07      1 an081
```

- Typically lots of output – use options of squeue to filter

Options of `squeue`

- Showing jobs for a specific user

```
squeue -u fred
```

will show the jobs of user “`fred`” only

- Option `--start` gives the estimated job start time
  - This can shift in either direction

Deleting jobs with `scancel`

- You can cancel a queued or running job
- Determine job-id, e.g. with `squeue`
- Use scancel

```
scancel 7103
```

  - terminates job 7103, if running
  - removes from the queue

Summary

- Kebnekaise's HPC2N hardware
- Building an OpenMP executable on Kebnekaise
- Running the executable using SLURM