ABEL USERGUIDE, FAQ, and SPECs

# User Account

## Getting an account on Abel

You get an account on Abel by [applying for access](http://www.uio.no/english/services/it/research/hpc/abel/help/access/).

## Accessing Abel

After you get a valid username and password, you can access the cluster the following way(s).

### From Unix/Linux systems

Use secure shell to connect to Abel:

ssh -Y myusername@abel.uio.no

To copy a file myfile, use the secure copy command:

scp myfile myusername@abel.uio.no:~

### From Windows systems

If you don't have X emulators for Windows, such as cygwin, you have at least to have a program for remote terminal, which supports the ssh protocol (secure shell), such as putty.

It is also better if you have an X11 server installed, so that you can open windows.  Note that X11 forwarding has to be selected.

### From Mac systems

Start the Terminal program of OS X (or any other terminal program, like [iTerm](http://iterm2.com/)). In the terminal window, you can use the command

ssh myusername@abel.uio.no

to log in to abel, and the command

scp myfile myusername@abel.uio.no:~

to copy a file myfile to Abel.

## Home Directory

By default, a user's home directory cannot be accessed by other users. In case a user wants to make data under the home directory readable to others, the user in question must perform the necessary actions for this. (E.g., use chmod on the relevant directories/files.) We do not allow world writable directories outside of scratch areas. The filesystem is periodically scanned for world writable directories, and the permissions are corrected if such a directory is found.

A project may request that a mechanism for access controll to the directories of the users connected to the project is provided for all users in that project. This can only be requested by the project responsible. This allows for sharing data, both read and write, between users in the same project. The group members are themselves responsible for any conflicts resulting from such sharing. This access control is implemented with a unique group id (file-group) for the project. The group name will be the same as the project account, e.g., nn1234k, if possible.

More about [available file-systems on Abel](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html).

## Account validity

For Notur users, the Abel account is valid for as long as the user belongs to an active project (a project with an active allocation). When an allocation period ends, unless the project is granted a quota for the following period, the account will be terminated and any user data removed to free disk space. This operation normally takes place after a short grace period, i.e. one week. Users are responsible for taking any necessary action before this time, i.e. copying their files elsewhere.

For UiO users, their account is valid as long as their affiliation to the university.

# Queue System

This page documents the queue system on the **Abel** (HPC for UiO) and **Colossus** (HPC for TSD) clusters.

Table of Contents

* [Finding your projects](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc1)
  + [Project quota for Notur and TSD projects](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc2)
* [Inspecting Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc3)
* [Inspecting the Job Queue](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc4)
* [General Job Limitations](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc5)
* [Scheduling](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc6)

When you log in to **Abel**, you are logged in to a loginnode shared by all users. The login nodes are meant for logging in, copying files, editing, compiling, running short tests (no more than a couple of minutes), submitting jobs, checking job status, etc.On **Colossus**, there are no shared login nodes; you use your own secure Linux Virtual Machine (VM).

To run a job on the cluster, you submit a [job script](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html) into the job queue, and the job is started when one or more suitable compute nodes ar available. The job queue is managed by a queue system (scheduler and resource manager) called SLURM [(SLURM's documentation page)](http://slurm.schedmd.com/).

Note that it is not allowed to run jobs directly on the login nodes of **Abel**. If you fail to comply with this rule, your access to **Abel** might be suspended.

For **Colossus** users: jobscript name should not contain sensitive information.

[Job scripts](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html) are submitted with the sbatch command:

sbatch YourJobscript

The sbatch command returns a jobid, an id number that identifies the submitted job. The job will be waiting in the job queue until there are free compute resources it can use. A job in that state is said to be pending (PD). When it has started, it is called running (R). Any output (stdout or stderr) of the job script will be written to a file called slurm-jobid.out in the directory where you ran sbatch, unless otherwise specified.

All commands in the job script are performed on the compute-node(s) allocated by the queue system. The script also specifies a number of requirements (memory usage, number of CPUs, run-time, etc.), used by the queue system to find one or more suitable machines for your job.

You can cancel running or pending (waiting) jobs with scancel:

scancel jobid # Cancel job with id jobid (as returned from sbatch)

scancel --user=MyUsername # Cancel all your jobs

scancel --account=MyProject # Cancel all jobs in MyProject

See man scancel for more details.

## Finding your projects

Projects may change names over time, they may expire and you may get access to new projects. To see which projects you currently have access to, you can use the command projects:

$ projects

nn2977k

staff

Note that the queue system calls projects accounts. We will try to use the term project in this documentation.

The projects command does not exist on **Colossus**. Each user only has access to a single project, the name of which (pNN) is the prefix of the user name.

### Project quota for Notur and TSD projects

Notur projects (projects with name nnXXXXk or xaXXXXk) on **Abel** and TSD projects (pNN) on**Colossus** have a CPU (core) hour quota. The cost command is used to inspect the quota. Use

cost -u

to see how much you have used of the quota on your projects (this switch is not available on the TSD VMs, but is available inside jobscripts on **Colossus**).  Use

cost

to see the available, used and total quota of your projects.  See cost --help for details.

[Historic usage on Abel can be found here](http://www.uio.no/english/services/it/research/hpc/abel/historic-cpu-hour-usage.html).

## Inspecting Jobs

To get a quick view of the status of a job, you can use squeue:

squeue -j JobId

where JobId is the job id number that sbatch returns.  To see more details about a job, use

scontrol show job JobId

See man squeue and man scontrol for details about these commands.

## Inspecting the Job Queue

There are several available commands to inspect the job queue:

* squeue: list jobs in the queue
* qsumm: show summary of queue usage

To see the list of running or pending jobs in the queue, use the command squeue. Useful squeueoptions:

[-j jobids] show only the specified jobs

[-w nodes] show only jobs on the specified nodes

[-A projects] show only jobs belonging to the specified projects

[-t states] show only jobs in the specified states (pending, running,

suspended, etc.)

[-u users] show only jobs belonging to the specified users

All specifications can be comma separated lists. See man squeue for details. Examples:

squeue -j 14132,14133 # shows jobs 4132 and 4133

squeue -w c23-11 # shows jobs running on c23-11

squeue -u foo -t PD # shows pending jobs belonging to user 'foo'

squeue -A bar # shows all jobs in the project 'bar'

| Squeue status (ST) | |
| --- | --- |
| **Status** | **Text** |
| PD | Pendind |
| R | Running |
| S | Suspended |
| CG | Completing |
| CD | Completed |
| CF | Configuring |
| CA | Cancelled |
| F | Failed |
| TO | Timeout |
| PR | Preemepted |
| NF | Node failed |

The qsumm command displays a summary of the how many cores ("CPU"s) are used by the different projects on the cluster. This gives an overview of the load of the cluster.

Useful qsumm options:

--nonzero # only show projects with at least one running or pending job

--pe # show processor equivalents (PEs) instead of CPUs

--memory # show memory usage instead of CPUs

--group # do not show the individual Notur and Grid projects

--user=username # only count jobs belonging to username

--help # show all options

See man squeue and qsumm --help for details about these commands.

## General Job Limitations

Default values when nothing is specified:

* 1 core (CPU)

The rest (time, mem per cpu, etc.) must be specified.

Limits

* Each project has its own limit of the number of concurrent used CPUs. See qsumm --group. (Notur projects share the same limit.)
* On **Abel**, the max wall time is 1 week, except for jobs in the hugemem or long partitions, where it is 4 weeks. ([definition of wall time](http://en.wikipedia.org/wiki/Wall_clock_time)) On **Colossus**, the max wall time is 4 weeks, but do ***not*** submit jobs that will run for more than 7 days unless they implement[checkpointing](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Checkpointing): **None of the nodes on Colossus have dual power, and we reserve the right to shutdown any node for maintenance at any time with 7 days notice!**
* Max 400 submitted jobs per user at any time.

## Scheduling

Jobs are started by priority. Pending jobs are prioritized according to

* Queue time
* Processor Equivalents (how many CPUs and how much memory the job asks for)
* Fairshare usage
* QoS (normal or lowpri)

The queue system uses backfilling, so small, short jobs can start earlier than jobs with higher priority, as long as they do not delay the higher priority jobs.

Lowpri jobs can be preempted by normal jobs, based on priority. Priority on running jobs is based on

* Processor Equivalents (how many CPUs and how much memory the job asks for)
* QoS (normal or lowpri)

On **Colossus**, there is no lowpri partition.

# Job Scripts

This page documents how to write job scripts to submit jobs on the **Abel** (HPC for UiO) and **Colossus**(HPC for TSD) clusters.

Table of Contents

* [A Simple Serial Job](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#A_Simple_Serial_Job)
* [Work Directory](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Work_Directory)
  + [/work/users/username](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#toc1)
* [Splitting a Job into Tasks (Array Jobs)](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Splitting_a_Job_into_Tasks)
* [Parallel Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Parallel_Jobs)
  + [OpenMP or Threading](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#OpenMP_or_Threading)
  + [MPI](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#MPI)
  + [Combining MPI with OpenMP or threading](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#toc2)
* [Debugging tools](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Debugging_tools)
* [Large Memory Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Large_Memory_Jobs)
  + [Accounting of Large Memory Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Accounting_of_Large_Memory_Jobs)
* [GPU Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#GPU_Jobs)
* [The Lowpri QoS](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#The_Lowpri_QoS)
* [Checkpointing](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Checkpointing)
* [Useful sbatch/qlogin/srun parametres](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Useful_sbatch_parametres)
  + [sbatch policy settings](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#sbatch_policy_settings)

To run a job on the cluster involves creating a shell script called a job script. The job script is a plain-text file containing any number of commands, including your main computational task, i.e., it may copy or rename files, cd into the proper directory, etc., all before doing the "real" job. The lines in the script file are the commands to be executed, in the given order. Lines starting with a "#" are ignored as comments, except lines that start with a "#SBATCH" which are not executed, but contain special instructions to the [queue system](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html).

If you are not familiar with shell scripts, they are simply a set of commands that you could have typed at the command line. You can find more information about shell scripts here: [Introduction to Bash shell scripts](http://www.linuxconfig.org/Bash_scripting_Tutorial).

A job script consists of a couple of parts:

* Instructions to the queue system
* Commands to set up the execution environment
* The actual commands you want to be run

Instruction parameters to the queue system may be specified on the sbatch command line and/or in #SBATCH lines in the job script. There can be as many #SBATCH lines as you need, and you can combine several parameters on the same line. If a parameter is specified both on the command line and in the jobscript, the parameter specified on the command line takes precedence. The#SBATCH lines should precede any commands in the script. A couple of parameters are compulsory. If they are not present, the job will not run:

* **--account**: Specifies the project the job will run in.
* **--time**: Specifies the maximal wall clock time of the job. Do not specify the limit too low, because the job will be killed if it has not finished within this time. On the other hand, shorter jobs will be started sooner, so do not specify longer than you need. The default maximum allowed **--time** specification is 1 week; [see details here](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/queue-system.html#toc5).
* **--mem-per-cpu**: Specifies how much RAM each task (default is 1 task; see [Parallell Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html" \l "Parallel_Jobs)) of the job needs.

The commands to set up the environment should include

source /cluster/bin/jobsetup

and will most likely include one or more

module load SomeProgram/SomeVersion

jobsetup will set up needed environment variables and shell functions, and must be the first command in the script. module will set up environment variables to get access to the specified program.  It is recommended to specify the explicit version in the module load command.  We also encourage you to use

module purge

prior to the module load statements, to avoid inheriting unknown environment variable settings from the shell you use to submit the job.  We also advice using

set -o errexit

early in the script.  This makes the script exit immediately if any command below it fails, instead of simply going on to the next command.  This makes it much easier to find out whether anything went wrong in the job, and if so, where it happened.

## A Simple Serial Job

This is a script for running a simple, serial job

#!/bin/bash

# Job name:

#SBATCH --job-name=YourJobname

#

# Project:

#SBATCH --account=YourProject

#

# Wall clock limit:

#SBATCH --time=hh:mm:ss

#

# Max memory usage:

#SBATCH --mem-per-cpu=Size

## Set up job environment:

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

## Copy input files to the work directory:

cp MyInputFile $SCRATCH

## Make sure the results are copied back to the submit directory (see Work Directory below):

chkfile MyResultFile

## Do some work:

cd $SCRATCH

YourCommands

Substitute real values for YourJobname, YourProject, hh:mm:ss, Size (with 'M' for megabytes or 'G' for gigabytes; e.g. 200M or 1G), MyInputFile, MyResultFile and YourCommands.

## Work Directory

Each job has access to a separate scratch space directory on the shared file system /work. The name of the directory is stored in the environment variable $SCRATCH. As a general rule, all jobs should use the scratch directory ($SCRATCH) as its work directory. This is especially important if the job uses a lot of files, or does much random access on the files (repeatedly read and write to different places of the files).

There are several reasons for using $SCRATCH:

* $SCRATCH is on a faster file system than user home directories.
* There is less risk of interfering with running jobs by accidentally modifying or deleting the jobs' input or output files.
* Temporary files are automatically cleaned up, because the scratch directory is removed when the job finishes.
* It avoids taking unneeded backups of temporary and partial files, because $SCRATCH is not backed up.

If you need to access the $SCRATCH area from outside a job (for instance for monitoring the job), the directory is (currently) /work/jobs/jobid.d, where jobid is the job id.

The directory where you ran sbatch is stored in the environment variable $SUBMITDIR. If you want automatic copying of files or directories back to $SUBMITDIR when the job is terminated, mark them with the command chkfile in the job script:

chkfile OneFile AnotherFile SomeDirectory

Note that the chkfile command should be placed early in the script, before the main computational commands: The files will be copied back even if the script crashes (even when you use set -o errexit), but not if it is terminated before it got to the chkfile command.  Also note that if you want to use shell-metacharacters, they should be quoted.  I.e., use chkfile "Results\*"instead of chkfile Results\*.  Finally, do not use absolute paths or things like ".\*" or .. in thechkfile command; use paths relative to $SCRATCH.

We recommend using chkfile (or cleanup; see below) instead of explicitly copying the result files with cp.

For instance:

#!/bin/bash

#SBATCH --job-name=YourJobname --account=YourProject

#SBATCH --time=hh:mm:ss --mem-per-cpu=Size

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

## Copy files to work directory:

cp $SUBMITDIR/YourDatafile $SCRATCH

## Mark outfiles for automatic copying to $SUBMITDIR:

chkfile YourOutputfile

## Run command

cd $SCRATCH

YourProgram YourDatafile > YourOutputfile

The $SCRATCH directory is removed upon job exit (after copying back chkfiled files).

If you want more flexibility than what chkfile gives, you can use the command cleanup instead. It is used to specify commands to run when your job exits (before the $SCRATCH directory is removed). Just like chkfile, the cleanup commands are run even if your script crashes (even when you use set -o errexit), but not if it crashes before reaching the cleanup command, so place the command early in the script.

For instance:

cleanup "cp $SCRATCH/outputfile /some/other/directory/newName"

Note: do not use single quotes (') if the commands contain variables like $SCRATCH and $SUBMITDIR.

### /work/users/username

If you need to share files between jobs, or use a file several times, it can be copied to/work/users/username (where username is your user name). Files in this directory are automatically deleted after a certain time. Currently, they are deleted after 45 days, but that can change in the future. There is no backup of files in*/work/users/username*. Note: /work/users/username does not exist on **Colossus**.

## Splitting a Job into Tasks (Array Jobs)

To run many instances of the same job, use the arrayrun command. This is useful if you have a lot of data-sets which you want to process in the same way with the same job-script. arrayrunworks very similarly to mpirun:

arrayrun [-r] from-to [sbatch switches] YourCommand

Typically, YourCommand is a compiled program or a job script. If it is a compiled program, thearrayrun command line must include the sbatch switches neccessary to submit the program. If it is a script, it can contain #SBATCH lines in addition to or instead of switches on the arrayruncommand line. from and to are the first and last task number. Each instance of YourCommandcan use the environment variable $TASK\_ID for selecting which data set to use, etc. For instance:

arrayrun 1-100 MyScript

will run 100 copies of MyScript, setting the environment variable $TASK\_ID to 1, 2, ..., 100 in turn.

It is possible to specify the TASK\_IDs in other ways than from-to: it can be a single number, a range (from-to), a range with a step size (from-to:step), or a comma separated list of these. A couple of examples:

Specification Resulting TASK\_IDs

1,4,42 # 1, 4, 42

1-5 # 1, 2, 3, 4, 5

0-10:2 # 0, 2, 4, 6, 8, 10

32,56,100-200 # 32, 56, 100, 101, 102, ..., 200

Note: spaces, decimal numbers or negative numbers are not allowed.

The instances of an array job are independent, they have their own $SCRATCH and are treated like separate jobs. You may also specify a parallel environment in array-jobs, so that each instance gets e.g. 8 processors. If the job script running arrayrun is cancelled, all instances of the array job will also be cancelled.

If you specify the -r switch (before the TASK\_IDs), arrayrun will restart a task that has failed. To avoid endless loops, a task is only restarted once, and a maximum of 5 tasks will be restarted.

See arrayrun --help for details.

An extended example:

$ cat workerScript

#!/bin/bash

#SBATCH --account=YourProject

#SBATCH --time=1:0:0

#SBATCH --mem-per-cpu=1G

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

DATASET=dataset.$TASK\_ID

OUTFILE=result.$TASK\_ID

cp $DATASET $SCRATCH

cd $SCRATCH

chkfile $OUTFILE

YourProgram $DATASET > $OUTFILE

$ cat submitScript

#!/bin/sh

#SBATCH --account=YourProject

#SBATCH --time=50:0:0

#SBATCH --mem-per-cpu=200M

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

arrayrun 1-200 workerScript

$ sbatch submitScript

This job will process the datasets dataset.1, dataset.2, ... dataset.200 and leave the results in result.1, result.2, ... result.200.

Note that the arrayrun command should be executed in the directory that contains theworkerScript, so the submitScript should not cd to $SCRATCH. (The workerScript should, though.)

On **Abel**, the worker scripts of array jobs are good candidates for running in the [lowpri QoS](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html" \l "The_Lowpri_QoS). But do not run the submit script itself in the lowpri QoS: if the submit script is restarted, all worker scripts will be rerun.  (There is no lowpri on **Colossus**.)

Note: the arrayrun command is a locally developed command, and it is quite possible that it will be further developed and changed. Suggestions can be sent to hpc(at)usit.uio.no.

## Parallel Jobs

Since the clusters consist of multi-cpu and multi-core compute nodes, there are several ways of parallelising code and running jobs in parallel. One can use MPI, OpenMP and threading. (For SIMD jobs and other jobs with very independent parallel tasks, [array jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Splitting_a_Job_into_Tasks) is a good alternative.) In this section we will explain how to run in parallel in either of these ways.

Note: A parallel job will get one, shared scratch directory ($SCRATCH), not separate directories for each node! This means that if more than one process or thread write output to disk, they must use different file names, or put the files in different subdirectories.

### OpenMP or Threading

In order to run in parallel on one node using either threading or OpenMP, the only thing you need to remember is to tell the queue system that your task needs more than one core, all on the same node. This is done with the --cpus-per-task switch.  It will reserve the needed cores on the node and set the environment variable $OMP\_NUM\_THREADS. For example:

#!/bin/bash

# Job name:

#SBATCH --job-name=YourJobname

#

# Project:

#SBATCH --account=YourProject

#

# Wall clock limit:

#SBATCH --time=hh:mm:ss

#

# Max memory usage per core (MB):

#SBATCH --mem-per-cpu=MegaBytes

#

# Number of cores:

#SBATCH --cpus-per-task=NumCores

## Set up job environment:

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

## Copy files to work directory:

cp $SUBMITDIR/YourDatafile $SCRATCH

## Mark outfiles for automatic copying to $SUBMITDIR:

chkfile YourOutputfile

## Run command

cd $SCRATCH

## (For non-OpenMP-programs, you must control the number of threads manually, using $OMP\_NUM\_THREADS.)

YourCommand YourDatafile > YourOutputfile

The --cpus-per-task will ensure all cores are allocated on a single node. If you ask for more cores than is available on any node, you will not be able to submiut the job. Most nodes on **Abel** have 16 cores, and on **Colossus** 20 cores.

### MPI

To run MPI jobs, you must specify how many tasks to run (i.e., cores to use), and set up the desired MPI environment. **Abel** supports OpenMPI, which can be used by setting up the environment with modules:

module load openmpi.compiler/version

where compiler is the compiler you used when compiling your program. See

module avail openmpi

for available versions. The symbol generation for different fortran compilers differ, hence versions of the MPI fortran interface for GNU, Intel, Portland and Open64. All OpenMPI libraries are built using gcc.

You need to use module load both in order to compile your code and to run it. You also should compile and run in the same MPI-environment. Although some MPI-versions may be compatible, they usually are not.

If you need to compile C MPI code with icc please see OpenMPI's documentation about environment variables to set in order to force mpicc to use icc.

The simplest way to specify the number of tasks (cores) to use in an MPI job, is to use the sbatchswitch --ntasks. For instance

#SBATCH --ntasks 20

would give you 20 tasks. The queue system allocates the tasks to nodes depending on available cores and memory, etc. A simple MPI jobscript can then be like:

#!/bin/bash

# Job name:

#SBATCH --job-name=YourJobname

#

# Project:

#SBATCH --account=YourProject

#

# Wall clock limit:

#SBATCH --time=hh:mm:ss

#

# Max memory usage per task:

#SBATCH --mem-per-cpu=Size

#

# Number of tasks (cores):

#SBATCH --ntasks=NumTasks

## Set up job environment:

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

module load openmpi.intel

## Set up input and output files:

cp InputFile $SCRATCH

chkfile OutputFile

cd $SCRATCH

mpirun YourCommand

#### Queue System Options

Using the queue option --ntasks in the previous example, we have assumed that it doesn't matter how your tasks are allocated to nodes. The queue system will run your tasks on nodes as it sees fit (however, it will try to allocate as many tasks as possible on each node). Usually that is OK. Sometimes, however, you might need more control. Then you can use the switches --nodes and --ntasks-per-node instead of --ntasks:

* --nodes: How many nodes to use
* --ntasks-per-node: How many tasks to run on each node.

For instance, to get 1 task on each of 16 nodes, you can use

#SBATCH --nodes=16 --ntasks-per-node=1

Or, to use 8 task on each of 2 nodes:

#SBATCH --nodes=2 --ntasks-per-node=8

There are more advanced options for selecting cores and nodes, as well. See man sbatch for the gory details.

#### TCP/IP over InfiniBand for MPI

If you have MPI jobs hard linked to use TCP/IP we have some tricks to use InfiniBand even for these. It is possible to run the TCP/IP over InfiniBand with far better performance than over Ethernet. However this only apply to communications between the compute nodes. Please contact us if you have such an application or want to use TCP/IP over InfiniBand. All nodes have two IP numbers one for the Ethernetnet and one for InfiniBand.

#### Useful Commands in MPI Scripts

If you need to execute a command once on each node of a job, you can use

srun --ntasks=$SLURM\_JOB\_NUM\_NODES command

### Combining MPI with OpenMP or threading

You can combine MPI with OpenMP or threading, such that MPI is used for launching multi-threaded processes on each node. The best way to do this is:

#!/bin/bash

# Job name:

#SBATCH --job-name=YourJobname

#

# Project:

#SBATCH --account=YourProject

#

# Wall clock limit:

#SBATCH --time=hh:mm:ss

#

# Max memory usage per task:

#SBATCH --mem-per-cpu=Size

#

# Number of tasks (MPI ranks):

#SBATCH --ntasks=NumTasks

#

# Number of threads per task:

#SBATCH --cpus-per-task=NumThreads

## Set up job environment:

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

module load openmpi.intel

## Set up input/output files:

cp InputFile $SCRATCH

chkfile OutputFile

## Run command

## (If YourCommand is not OpenMP, use $OMP\_NUM\_THREADS to control the number of threads manually.)

mpirun YourCommand

This makes mpirun start NumTasks ranks (processes), each of which having NumThreadsthreads. If you are not using OpenMP, your program must make sure not to start more thanNumThreads threads.

Just as with single-threaded MPI jobs, you can get more than one rank (MPI process) on each node. If you need more control over how many ranks are started on each node, use --ntasks-per-node and --nodes as above. For instance:

#SBATCH --nodes=3 --ntasks-per-node=2 --cpus-per-task=4

will start 2 MPI ranks on each of 3 machines, and each process is allowed to use 4 threads.

## Debugging tools

TotalView is installed on **Abel**. The corresponding module is totalview. For more information please visit the totalview web site at<http://www.roguewave.com/products/totalview.aspx> (TotalView is not available on **Colossus**.)

## Large Memory Jobs

Most nodes are equipped with 16 cores (**Abel**) or 20 cores (**Colossus**) and 64 GiB of RAM, of which 61.5 GiB can be used for jobs. (The rest is used by the operating system.) There are also a couple of special hugemem nodes with 32 cores and 1 TiB memory, of which about 1006 GiB can be used for jobs. To see how many nodes there currently are in the cluster, together with their number of cores and MB RAM, use

sinfo -e -o '%D %c %m'

You specify how much memory per core your job should be allowed to use by setting the --mem-per-cpu parameter. Technically, this limit specifies the amount of resident memory + swap the job can use. If the job tries to use more than this setting, it will be killed. For instance

#SBATCH --mem-per-cpu=2000M

If the job tries to use more than 2000 MiB resident memory, it will be killed. Note that --mem-per-cpu is specified per requested core.

Also, if you need more than 61.5 GiB RAM on a single node, you must specify --partition=hugemem to get access to the nodes with more RAM. For instance

#SBATCH --ntasks-per-node=8

#SBATCH --mem-per-cpu=10G --partition=hugemem

Note: There is no need to specify how much RAM a node must have; the queue system will not allocate jobs on nodes with too little free RAM. Therefore, one should not use the --memspecification.

### Accounting of Large Memory Jobs

To ensure maximal utilisation of the cluster, memory usage is accounted as well as cpu usage. Memory specifications are converted to "Processor Equivalents" (PE) using a conversion factor ofapproximately 4 GiB / core(\*). If a job specifies more than 4 GiB RAM per task, i.e., --mem-per-cpu=M, where M > 4G, each task will count as M / 4G cores instead of 1 core. For instance, a job with --ntasks=2 --mem-per-cpu=8G will be counted as using 4 cores instead of 2.

The reason for this is that large memory jobs make the "unused" cores inaccessible to other jobs. For instance, a job on a 64 GiB node using --ntasks=1 --mem-per-cpu=61G will in practice use all cores on the node, and should be accounted as such.

Note that only jobs which specify more than 4 GiB per core will be affected by this; all other jobs will be accounted with the number of tasks specified.

(\*) The exact value of the factor depends on the total amount of RAM per core in the cluster, and is currently about 4.5 GiB / core on **Abel** and 4.3 GiB / core on **Colossus**.

## GPU Jobs

**Abel** has a few [nodes with GPU accelerators](http://www.uio.no/english/services/it/research/hpc/abel/more/).  (There are no GPUs on **Colossus**.) Each node has two GPU cards. To get access to these nodes, specify

#SBATCH --partition=accel --gres=gpu:1

to use one card, or

#SBATCH --partition=accel --gres=gpu:2

to use both.

In addition, you must specify how many CPUs you need with --ntasks, --ntasks-per-node and/or --cpus-per-task as usual, as well as the amount of RAM the job needs.

In the job script, the environment variable CUDA\_VISIBLE\_DEVICES will show which GPU device(s) to use. It will have values '0', '1' or '0,1' corresponding to /dev/nvidia0, /dev/nvidia1 or both, respectively.

Here is an example script, running MrBayes on 1 CPU (the default) and 1 GPU:

#!/bin/bash

#

#SBATCH --job-name=YourJobName --account=YourProject

#SBATCH --time=TimeLimit

#

## Ask for 1 GPU

#SBATCH --partition=accel --gres=gpu:1

#

## and 3000 MiB RAM

#SBATCH --mem-per-cpu=3000

## Set up job environtment:

source /cluster/bin/jobsetup

module purge # clear any inherited modules

set -o errexit # exit on errors

## Get access to the GPU enabled MrBayes:

module load mrbayes\_gpu

mb YourInputFile

## The Lowpri QoS

On **Abel**, it is possible to run your program om somebody else's cores if they are not using them. To do this you specify that you want to use the **lowpri** qos (Quality of Service). This is done by specifying:

#SBATCH --qos=lowpri

in your script.

If the owner of the cores you are running on wants to use them again before your job is finished, your job will be stopped and put back on the queue. Therefore, if your job is longer than a couple of hours, you should make sure it checkpoints, i.e. saves intermediate results so that it can start from where it left if it is restarted.

Note that it is not possible to run hugemem jobs, GPU jobs or Notur jobs in the lowpri QoS. There is no lowpri QoS on **Colossus**.

## Checkpointing

Checkpointing a job means that the job can be stopped and started somewhere else, and continues where it left off.

Long-running jobs should implement some form of checkpointing, by saving intermediate results at intervals and being able to start with the latest intermediate results if restarted. This is especially important for lowpri-jobs which can be stopped and requeued, but also for regular jobs in order to guard against node failures etc.

Automatic checkpointing is very hard to implement and the current trend is leave this to the application. Using a parallel file system and parallel IO the complete data structure can be saved to disk within a reasonable amount of time. The MTBF for very large clusters (larger than Abel, PRACE tier-0 size clusters) are low and for very large jobs checkpoint / restart need to be in place.

## Useful sbatch/qlogin/srun parametres

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| --account=project | Specify the project to run under. This parameter is required. |
| --begin=time | Start the job at a given time |
| --constraint=feature | Request nodes with a certain feature. Currently supported features include intel, ib, rackN. If you need more than one feature, they must be combined with & in the same --constraint specification, e.g. --constraint=ib&rack21. Note: If you try to use more than one --constraint specification, the last one will override the earlier. |
| --cpus-per-task=cores | Specify the number of cpus (actually: cores) to allocate for each task in the job.  See --ntasks and --ntasks-per-node, or man sbatch. |
| --dependency=dependency list | Defer the start of this job until the specified dependencies have been satisfied. See man sbatch for details. |
| --error=file | Send 'stderr' to the specified file. Default is to send it to the same file as 'stdout'. (Note: $HOME or ~ cannot be used. Use absolute or relative paths instead.) |
| --input=file | Read 'stdin' from the specified file. (Note: $HOME or ~ cannot be used. Use absolute or relative paths instead.) |
| --job-name=jobname | Specify job name |
| --mem-per-cpu=size | Specify the memory required per allocated core. This is the normal way of specifying memory requirements (see [Large Memory Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Large_Memory_Jobs)).size should be an integer followed by 'M' or 'G'. |
| --partition=hugemem | Run on a hugemem node (see [Large Memory Jobs](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html#Large_Memory_Jobs)). |
| --nodes=nodes | Specify the number of nodes to allocate. nodes can be an integer or a range (min-max). This is often combined with --ntasks-per-node. |
| --ntasks=tasks | Specify the number of tasks (usually cores, but see --cpus-per-task) to allocate. This is the usual way to specify cores in MPI jobs. |
| --ntasks-per-node=tasks | Specify the number of tasks (usually cores, but see --cpus-per-task) to allocate within each allocated node.Often combined with --nodes. |
| --output=file | Send 'stdout' (and 'stderr' if not redirected with --error) to the specified file instead of slurm-%j.out.  Note:   1. $HOME or ~ cannot be used. Use absolute or relative paths instead. 2. The jobid should always be included in the filename in order to produce a separate file for each job (i.e. --output=custom-name%j.out). This is necessary for error tracing. |
| --qos=lowpri | Run a job in the [lowpri qos](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html" \l "The_Lowpri_QoS) (not available on **Colossus**) |
| --time=time | Specify a (wall clock) time limit for the job. time can be hh:mm:ss ordd-hh:mm:ss. This parameter is required. |
| --mail-type=event | Notify user by email when certain event types occur. Valid events are BEGIN, END, FAIL, REQUEUE, and ALL. (Mail notifications are not available on **Colossus**.) |

### sbatch policy settings

A set of extra tests and policy settings has been added to the sbatch command, to make sure jobs will start and run properly. The additional tests ensure:

* Jobs that will run on only one node are not allowed to specify --constraint=ib
* All account names are lower cased
* That all job comments and constraints are passed to SLURM (per default, only the last specification will be passed on when multiple specifications exists)

Interactive logins

If you need to test something interactively, you can log in via the queue system with the command qlogin --account=*YourProject*. You can specify almost the same arguments as sbatch. E.g., you may use the command

qlogin --account=*YourProject* --ntasks=2

or use

qlogin --account=*YourProject* --ntasks-per-node=16

to get a complete node.

*Note: It is not advisable to use --qos=lowpri with qlogin.*

If you want to use several constraints (for instance intel and ib) you must use the following syntax:

qlogin *other options* --constraint='intel&ib'

An example

qlogin --account=uio --ntasks-per-node=8 --constraint='intel&ib'

This will allocate a node with Intel processors and Infiniband. The single quotes are important, otherwise the shell will respond to the & and background the job.

When you have gotten an allocation, you must run

source /cluster/bin/jobsetup

just like in batch jobs. Then you can run your program interactively, e.g.,

module load openmpi.gnu

mpirun *YourCommand*

It might be that you'll need to supply the full path of mpirun (which mpirun gives info) and also supply the LD\_LIBRARY\_PATH to it (mpirun -h give info).

# Managing Data on Abel

Table of Contents

* [Available file-systems on Abel](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc1)
  + [High performance file-systems](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc2)
  + [Project file-systems](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc3)
* [Disk quotas and disk usage accounting](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc4)
* [Compression of data](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc5)
* [Endianness](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc6)
* [Transferring data to/from the system](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc7)
  + [Transferring of large files](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc8)
* [Management of many small files](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc9)
* [Back-up of data](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc10)
* [Archiving of data](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc11)
* [BeeGFS df](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc12)
* [BeeGFS dusage](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/data.html#toc13)

## Available file-systems on Abel

| **File system** | **Quota / Limitation** | **Recommended Use / Links** |
| --- | --- | --- |
| $HOME  /cluster/home/USER.. | 500 GB | software, job configurations, few files only  DO NOT use $HOME for processing data in a job |
| $SCRATCH  /work/jobs/JOBID.d | automatically deleted at the end of the job | processing data in a job, [chkfile](https://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html" \l "Work_Directory) to retain output data |
| $USERWORK  /work/users/USER.. | anything older than 45\* days is deleted | use for sharing data among different jobs |
| /projects | depends on the project | use for sharing data within a project |

(\*) Policy will be adjusted when necessary.

### High performance file-systems

Abel has an enterprise class disk system of over 400 TiB where the main shared file-systems reside. The communication with this storage-system is over ultra-fast 56 Gbps Infiniband.

The main file-system is the /cluster file-system. All user's $HOME (home directory) reside on this file-system. Scratch, or work-disk, is available under /work. The /cluster and /work file-systems are available on all nodes, concurrently, using the FhGFS global parallel file-system.

Serving and keeping track of a parallel cluster file-system for over 600 nodes is a complicated task. While the hardware is high-end, once in a while the FhGFS software will get in trouble. When this happens users might experience delay when doing simple commands as "ls" or even hangs. Usually these problems lasts for a very short time, but if there is a serious problem there will be an announcement on the Operational Events Log and on the Abel Operations email list.

#### Home directory

Each user has a home directory ($HOME) on the /cluster file system.  By default, the disk quota for the home directory is 500 GiB (see below).

The home directory is backed up regularly (see below), but anything inside directories namesnobackup is skipped.  Backup is slow and expensive, so please put temporary files, files that can be downloaded again, installed software and other files that can easily be recreated or do not need to be backed up inside a nobackup directory.

Also note that one is not supposed to use the home directory as scratch area for jobs, especially not I/O intensive jobs.  Use the scratch area for that (see below).

#### Scratch disk space

While a job runs, it has access to a temporary scratch directory on /work. The directory is individual for each job, is automatically created, and is deleted when the job finishes (or gets requeued). There is no backup of this directory. The name of the directory is stored in the environment variable $SCRATCH, which is set within the job script.

In general, jobs should copy their work files to $SCRATCH and run there. This is especially important for I/O intensive jobs.  The scratch disk is faster than the home directory disk, and running I/O intensive jobs in $HOME slows down not only the job, but also interactive work for other users.

If you need to access the scratch directory from outside the job (for instance for monitoring a running job), the directory is /work/jobs/jobid.d, where jobid is the job id of the job in question.

All users also have access to a directory /work/users/username, where username is the user's user name. The purpose of the directory is to stage files that are needed by more than one job. Files in this directory are automatically deleted after a certain time. Currently, they are deleted after 45 days, but that can change in the future. There is no backup of files in*/work/users/*. Please note that this area is not meant for storage of data.  If attempts to avoid the automatic deletion are detected, counter measures will be applied, which could include removing the user's area alltogether.

### Project file-systems

The /projects file-system is exported to all nodes on Abel using NFS. As such the performance when working with this file-system will generally be good for sequential access, but interactive (small) usage might experience intermittent variation in response time when accessing this file-system from the command-line (e.g. ls, cd).

We therefore do not recommend using this file-system for anything but data storage. Specifically, one should not run jobs that do several small operations directly on this file-system, but instead do calculations on the native file-system (/work) by only reading input/output or copying from the necessary files from the /projects file-system.

## Disk quotas and disk usage accounting

All Notur users will get a **500 GB disk space quota** for their home directory when receiving the account. Unfortunately we can not provide larger quotas to individual users, simply because Notur have none for us to give away.

However, currently in some ad-hoc cases we can do some exceptions from the rule on temporary basis under strict requirements.

In some cases projects may rent extra project disk-space from USIT. Also projects may apply for extra disk space from [NorStore](https://www.norstore.no/" \o "NorStore).

To check your current quota and usage you can use the dusage command (see below).

For questions and inquiries about disk quotas please contact hpc-drift->at<-usit.uio.no

## Compression of data

The standard unix/linux utility today is gzip. Read the man pages for more information.

gzip file.dta

This will produce a file file.dta.gz, hopefully a much smaller file. Not all types of data compress equally well. Text compresses well, jpg pictures not well at all. For files that are to be unpacked at Windows machines and the GUN software gzip (it's free) is not installed the zip utility can be used. There are some limitation to the zip, neither the input files or the resulting archive can be larger than 4 GB. For files larger that 4 GB use gzip. Giving it a numeric argument like -9 forces higher compression at the expense of longer compression time.

A more effcient alternative is bzip2.

To unpack the file :

gunzip file.dta.gz

This will result in the original file in it's uncompressed form.

You might want to try the hpczip module which provide Intel architecture optimized versions of bzip2 and gzip. If bunzip2 or gunzip complains, use bzip2 -d or gzip -d whichs forces them to decompress.

## Endianness

Not all files have the same byte ordering, little endian and big endian. For byte oriented files this is done by swapping bytes at input and many compilers can to this while reading input files. For the Intel fortran compiler there is an environment variable called F\_UFMTENDIAN which sets this.

MODE = big | little

EXCEPTION = big:ULIST | little:ULIST | ULIST

ULIST = U | ULIST,U

U = decimal | decimal -decimal

The Portland compiler can also do this by a flag called "byteswapio".

pgf90 -byteswapio

The GNU fortran compiler gfortran have the following syntax:

gfortran -fconvert=conversion

where conversion is one of native or swap. See man page for more information.

For more special convert formats where the solution with byte swapping is not enough one might convert the file as ascci if oen have access to both systems.

program read

open(2,"file.dta",unformatted)

while IOstat=0

read(2) a

write(\*,\*) a

endwhile

program write

open(2,"file.dta",unformatted)

while IOstat=0

read(\*,\*) a

write(2) a

endwhile

Then start the read executable where file reside and receive the data at the system where you want the data copied.

./read.x | (ssh remote.system; ./write.x)

After some time the data will reside in a binary file with the correct layout on the receiving system.

## Transferring data to/from the system

Access to Abel is provided with ssh only. Use sftp or scp to upload or download files at address

abel.uio.no

### Transferring of large files

Secure copy with scp is the common way of transferring files. However, if the communication fails half way through the scp will not pick up where it left. In addition the transfer is not compressed, which is an advantage for bandwidth limited transfers. The utility rsync is the suggested way of moving large files.

rsync [OPTION]... SRC [SRC]... [USER@]HOST:DEST

Please see rsync man page for more information about rsync usage. A simple example may look like, assume you want to copy files to abel :

rsync -z file.tar user@abel.uio.no:mywork/

This will copy the file using ssh secure protocol in the user's directory named mywork. If the transfer crashes and this command is reissued it will pick up from where it left. This is vital for very large files.

## Management of many small files

FhGFS is not very happy with a huge number of small files (being a multimedia file system all files less than 1 MB are small). Millions of small files pose a challenge for FhGFS and is to be avoided. If possible pack the small files in archives, ar or tar. Making operations on them easy. However, in the cases where you actually need to work on a large set of small files try to minimize the number if possible. One way is to copy the tar file to $SCRATCH unpacking them there and work on a locally mounted file three.

## Back-up of data

Data in home partition (/cluster/home) and project partition (/projects) is incrementally backed-up once a day on tapes. To restore deleted or corrupted files send a request to hpc-drift@usit.uio.no. Note that the following backup rules apply to files:

* A backup copy of a file is kept on tape for 90 days after the file has been deleted (see<http://www.uio.no/english/services/it/news/backup.html>)
* Changes to the file are also retained for 90 days. This means that you can ask for the file as it was up to 90 days ago. Note that not all changes to the file are recorded - backup runs once a day.
* Directories with name "nobackup" or "no\_backup" (and all the sub-tree) will be excluded from the daily backup (to be used as much as possible to help reducing the task of the daily backup !). This is also the case for directories named "\*.moved" or for files named (core, core.[0-9][0-9][0-9])".

**Important:** note that /work is not backed up

## Archiving of data

There are no services for long-term archiving of data in connection with Abel. Users with such needs may try to contact [NorStore](https://www.norstore.no/" \o "NorStore), although at present it is not clear if NorStore will offer long-term archival of data.

UiO-based researcher can contact us on hpc-drift@usit.uio.no for inquiries about data archiving, and we will try to help find a solution.

## BeeGFS df

On Abel, the UNIX df utility is for:

* Disk partitions: The standard UNIX file system container

The df on Abel follows the following behavior:

* df (with no options) list all partitions

## BeeGFS dusage

To check the disk usage of your home directory (as well as the number of files it contains) issue the command

dusage -u username (or simply dusage)

To check the disk usage of a project (in /projects) use:

dusage -p projectname

**note**: currently disk usage of a project in /work/projects is not available with this command. Instead use du -hs /work/projects/projectname.

Modules

This page documents how to use Modulefiles on the Abel (HPC for UiO) and Colossus (HPC for TSD) clusters.

To simplify setting up environments for compilers, MPI-versions and some installed applications, the clusters have the [Environment Modules package](http://modules.sourceforge.net/) installed. This provides means for dynamic modification of a user's environment via *modulefiles*.

Each modulefile contains the information needed to configure the shell for an environment. Typically, a modulefile modifies environment variables like PATH, MANPATH, LD\_LIBRARY\_PATH, CPATHand PKG\_CONFIG\_PATH to give access to an application or library.

Modulefiles can be used from most shells, as well as Perl and Python. Modulefiles can be loaded/unloaded on the command line or within your scripts by typing:

module load *modulefile*

module unload *modulefile*

(On Colossus, the module commands can only be used in job scripts.)  You can also ask for a specific version of a modulefile:

module load *modulefile*/*version*

For the purpose of reproducability, we encourage users to load specific versions of modulefiles.  Also, we recommend using

module purge

prior to any module load commands in job scripts, to prevent inheriting environment variables set by module commands in the shell used when submitting the job.

The available modulefiles can be listed by module avail. Example output:

$ module avail

------------------------ /usr/share/Modules/modulefiles -------------------------

dot module-cvs module-info modules null use.own

--------------------------- /cluster/etc/modulefiles ----------------------------

454apps/2.8(default) mcmcphase/2.0(default)

R/2.15.1(default) microbiomeutil/20110519(default)

R/2.15.1.shlib migrate/3.3.2(default)

abyss/1.3.4(default) mothur/1.26.0(default)

adf/2010.02b(default) mrbayes/3.2.1(default)

adf/2012.01b muscle/3.8.31(default)

ampliconnoise/1.25(default) ncl/2.1.18(default)

beagle/1.0(default) newbler/2.8(default)

[...]

You can also specify a name:

$ module avail blast

--------------------------- /cluster/etc/modulefiles ----------------------------

blast/2.2.26(default) blast+/2.2.26(default)

You will get a list of current loaded modulefiles by

$ module list

Currently Loaded Modulefiles:

1) intel/2011.10 2) R/2.15.1

A help text exists for most of the modulefiles; to view it, type

module help *modulefile*

To view exactly what a modulefile is doing to your environment, type

module show *modulefile*

For further documentation, please refer to the [Environment Modules homepage](http://modules.sourceforge.net/)

Personalize default modulefiles

You may add loading of modulefiles to your .bash\_login file to make sure your favorite modulefile is always loaded when logging in. For instance, to always have the intel modulefile load on login, add:

module load intel

to the .bash\_login file.

## General questions

### What has changed since Titan?

[Here is a desription of the most important differences between Abel and Titan.](http://www.uio.no/english/services/it/research/hpc/abel/help/faq/changes-since-titan.html)

### How do I login to Abel?

You simply use

ssh -Y abel.uio.no

or, if you are already logged on a uio machine, it should be sufficient to do

ssh -Y abel

and use your username and password. In case you have a different username, user, on Abel than the one you are logging in from (note that this is never the case for UiO users), then specify

ssh -Y abel.uio.no -luser

### Setup SSH keys for the first time

To login to Abel you need to first set up an SSH key. The first time you log on, you will be asked about this if you have not set up the keys before. The message will look something like this:

It doesn't appear that you have set up your ssh key.

This process will make the files:

/usit/abel/u1/myusername/.ssh/id\_rsa.pub

/usit/abel/u1/myusername/.ssh/id\_rsa

/usit/abel/u1/myusername/.ssh/authorized\_keys

Generating public/private rsa key pair.

Enter file in which to save the key

(/usit/abel/u1/myusername/.ssh/id\_rsa):

You should simply press "enter" in response to these questions. You do not need to give a passphrase if this is requested ("enter" is enough).

### How do I copy files to and from Abel?

For small and few files, use scp

scp files user@abel.uio.no:~/wherever/you/want

This will put the files named files (which can be a list of files) to a place that you specify below your home directory (indicated by ~).

Use rsync if your files are very big and/or you have a bad connection.

rsync -z file.tar user@abel.uio.no:mywork/

This will copy the file file.tar using ssh secure protocol to the user's directory named mywork. If the transfer crashes and this command is reissued it will pick up from where it left. This is vital for very large files. To copy a whole file-tree or directory structure you can do that with

rsync -azx yourlocalworkdir user@abel.uio.no:myworkdir

See the man page ("man rsync") for more options and details.

You can also use your Linux file manager and connect to server using ssh file system.

If you for some reason are running Microsoft Windows you can download and use the application suite called putty which includes a scp utility.

Information about this utility is found here :<http://en.wikipedia.org/wiki/PuTTY> and it can be dowloaded from : <http://www.chiark.greenend.org.uk/~sgtatham/putty/>

Use port 22 which is the Secure Shell (ssh) port and remember to tick of for X11 forwarding if you want to use X11 (require an X11 client on your Windows workstation)

### What are login nodes?

Login nodes are Abel nodes where you can work with your files, compile code and run interactively. You should only use the login nodes for job preparation and debugging purposes, they are not to be used for running large jobs.

Login nodes run an automatic nice daemon so that your actions do not impact other users. Any process consuming more than 30 minutes of CPU time will automatically be killed.

See more information at [Interactive logins](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/interactive-logins.html) at the Abel User's Guide.

### What is SLURM?

SLURM is what we call the queue system, and decides when jobs should run, and on which nodes.

SLURM stands for Simple Linux Utility for Resource Management [(SLURM's documentation page)](https://computing.llnl.gov/linux/slurm/documentation.html).

### Where can I get further information ?

In addition to the [user's guide](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/), the [More about Abel](http://www.uio.no/english/services/it/research/hpc/abel/more/)page and the other information pages in this site, the manpages of the particular commands you use are a good place to learn more, for example:

man sbatch

man module

## Submitting jobs

### How do I submit a job?

If you have a job script called myjobscript, you submit it with the command:

$ sbatch myjobscript

We assume you have prepared it according to the guidelines in the [user guide](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/).

Several examples can be found [here](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html).

### How do I pick up the job id from sbatch?

If you submit a job within a perl script, you could use something like this to capture the job id:

my $output = `sbatch $scriptFile`;

($output =~ /^Submitted batch job (\d+)/)

or die "Couldn't execute sbatch $scriptFile";

my $jobid = $1;

### How can I rename the slurm-NNN.out file?

One can give the stdout/stderr file(s) new names by using

#SBATCH --output=MyStdoutFile

#SBATCH --error=MyStderrFile

(If you do not use --error, the stderr will go to the --output file.) You can use %j and %N in the names; they will be substituted with the job id and the nodename, respectively.

If you want more flexibility, for instance give the stdout file a name based on a variable in the shell, you can do that by renaming the file **in the end of the script**, like this:

[...the script...]

mv slurm-${SLURM\_JOBID}.out $SomeVariable.log

(If you are afraid that the file should not be renamed in case the job crashes, you can rename it earlier in the script, but it must be done after the source /cluster/bin/jobsetup line.)

A drawback with this technique is that, in the event that the job gets restarted, the file gets overwritten, no matter if you have specified --open-mode=append.

### How do I specify when my job should enter the queue ?

Sometimes it might be useful to hold the submission some some time. If you want to submit your job now, but want it to be placed in the at a later time you can use the --time=hh:mm:ss construct.

#SBATCH --begin=23:49:00

This will hold your job and not insert it into the queue before 23:49:00. At this time it will be placed in the queue and wait in line for available resources.

### How do I specify no re-queue/re-run?

If you do not want a job to restart after being thrown out from a node (because the node crashes, the network fails, specify:

# SBATCH --no-requeue

Note: it is not allowed to use this in the lowpri qos!

### How do I run MPI jobs?

You specify how many cpu's you want with #SBATCH --ntasks . If you want 20 cpu's, add the line

#SBATCH --ntasks 20

in your job script. The memory need you have (hopefully) specified with #SBATCH --mem-per-cpu= is per cpu, as stated. Then load the MPI module as before, and execute with mpirun. No options are needed to mpirun just mpirun <exec>. SLURM will launch the job if you use OpenMPI and take care of all the details.

### How do I specify cores=XxY

You specify node layout by using --nodes=Y and --ntasks-per-node=X.

For example, if you need 4 nodes with 8 cores each:

--nodes=4

--ntasks-per-node=8

Note! Other users may share these nodes if resources (memory / cores) are available.

If you need complete nodes **exclusively** (e.g. for **non-MPI** or runs with a single task per node), use:

--nodes=4

--exclusive

If you need complete nodes exclusively for **MPI** or runs with multiple tasks per node, use

--nodes=4

--ntasks-per-node=8

--exclusive

## Inspecting jobs

### How do I inspect my job(s)?

First of all, you can see the status of all your jobs in the queue, by specifying

squeue -u user

replacing user with your username. There you can check if it is already running, or still pending.squeue can print out a lot of information about jobs; see the -o switch in man squeue for details.

Then you can do, for each job (you got the jobid from the squeue command)

scontrol show job jobid

to get further information about where it is running, or why it is pending, etc.

See [Abel user guide](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/) for more information.

## Is it possible to get more time for my job?

In very rare cases, the Abel system operators may augment the wall-time of a particular job if contacted by the owner, and there are good reasons to require it.

But we note that changing the times allocated to jobs is very disruptive of the queue system, and users are responsible to schedule their jobs and budget their time with care.

So often the job gets cancelled and the user has to submit it again specifying a larger walltime.

## Notur projects

### How can I see my project quotas?

Use the cost command to se the CPU (core) hour quota of your Notur projects:

cost # See your usage on all your Notur projects

cost -p # See the available, used and total quota of your Notur projects

cost --help # Show detailed help about cost

## Troubleshooting

Here we have tried to list the most common sources of problems and questions that are posed to us by Abel users.

### I cannot log in to Abel

There may be several reasons:

#### Your account may have been locked

All users have to change passwords at least once a year. Centralized computer management send out an email to ensure that users change their password. If this message is ignored, the user account is locked and you cannot log in.

Send us an email at hpc-drift at usit.uio.no, and we will unlock your account temporarily. Then, you have to change your password. Go to:

<https://www.uio.no/brukerinfo>

After logging in, choose: Change your password

After changing your password, you may want to set a forwarding mail address to a mail address you read frequently:

Select the: E-mail tab, continue by selecting Forwarding. Type in your mail address:firstname.lastname@somesite.domain after the Forward all e-mails to this address: label.

#### RSA host key has changed

When trying ssh to login, you may see messages including

WARNING: POSSIBLE DNS SPOOFING DETECTED!

and

WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!

and

Offending key in /path\_to\_your\_home/.ssh/known\_hosts:N

In most cases, these messages indicate that the entry for the login node(s) in your known\_hostsfile is old (i.e., has been changed on Abel). If you remove the N-th line in your known\_hosts, you should be able to login again.

### I have problems working on Abel

#### Disk quota exceeded

Sometimes users are thrown out of their accounts because they exceeded their disk quota. We may increase it temporarily so that the users can remove or copy to another machine, but every user is granted 500 GB at Abel and is not supposed to exceed it except if s/he buys extra storage. Contact us in that case.

#### Emacs etc. does not work on Abel

If you have problems running emacs etc without it dying, you may have to apply Enables trusted X11 forwarding, this is done by running ssh with the -Y option:

ssh -Y abel.uio.no

### Why does my job not start?

There can be several reasons why a job does not start. Some of the most common are:

#### It is waiting for resources

Sometimes, the job has to wait for resources. This is especially common if scarce resources likehugemem has been specified, if the job needs many cores or exclusive nodes.

#### No complete nodes available

If you have asked for all the CPUs in a node in order to run exclusively on the requested nodes, there might be a longer waiting time compared to a random selection of CPUs around some nodes in the cluster. The benefits of running exclusively must be compared to the shorter wait time when running with a random selection.

#### You have exceeded the allowed number of CPUs

If you have submitted more jobs in a project than the project is allowed to run concurrently, jobs will have to wait until others have finished. To see if this is the case, you can see if

squeue -j JobId

says AssociationResourceLimit or QOSResourceLimit.

You can also look at the line for the project in the output of qsumm -p. If the usage of running normal jobs (nRun) is equal (or almost equal) to the limit, your job must wait.

Note that large memory jobs count as more "cores" than the number of tasks specified. See [here](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/job-scripts.html" \l "Accounting_of_Large_Memory_Jobs" \o "Titan User Guide)for details.

Also note that all Notur projects (nnXXXXk) share the same CPU quota. To see the total Notur usage, look at the Sum notur line in the output of qsumm -p.

### Why did my job fail?

#### Memory exceeded limit, killed

An error message like:

slurmd[compute-7-28]: Job 3524647 exceeded 5120 KB memory limit, being killed

or

slurmstepd-c2-1: Job 22 exceeded memory limit (4196668 > 4195328), being killed

slurmstepd-c2-1: Exceeded job memory limit

slurmstepd-c2-1: \*\*\* JOB 22 CANCELLED AT 2014-09-11T11:41:12 \*\*\*

in the log file means that SLURM has detected that the job tried to use more memory than you requested, and killed it.  Sometimes the Out of Memory Killer (OOM-killer) in the Linux kernel will detect that the job tried to use too much memory and kill one of the job's processes (typically the process trying to use more memory).  In this case, you will see a message like

slurmstepd-c2-1: Exceeded step memory limit at some point. Step may have been partially swapped out to disk.

in the log file (also, sometimes, you will see both sets of messages).  In very rare cases, this message can appear without any process actually being killed.  To see if any of your job's processes was killed by the OOM-killer due to memory over-usage, you can do

ssh node dmesg | grep jobid

where node is the node the job ran on, and jobid is the job id.  If you get a line like "Task in /slurm/... killed as a result of limit of /slurm/...", the job was killed by the OOM-killer.  For instance:

$ ssh c7-10 dmesg | grep 5265461

Task in /slurm/uid\_185644/job\_5265461/step\_4294967294 killed as a result of limit of /slurm/uid\_185644/job\_5265461/step\_4294967294

#### Cannot find the module command

If you get the message

module: command not found

when using module, e.g. module load R, the reason is usually that you have forgotten to use

source /cluster/bin/jobsetup

This must be used first in jobscripts, and also in qlogin's.

#### xterm and missing module command

If you launch a new xterm after logged in to 'abel' the new xterm will not be a login shell, rather just a new shell and this will not have the same environment as a login shell. The practical consequence is that the module command is unavailable. Launch xterm with -ls (login shell) and it will work.

#### The application does not work

Many applications require that you load a module:

module load svn

The application may require environment variables that is set by the Modules system to find the correct libraries etc.

Look at the [Abel user guide](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/) for more information about Modules.

#### The compiler is not there

You have to load a compiler module and perhaps a MPI-module too, i.e.:

module load openmpi

module load intel

Replace openmpi with the MPI-version you wish to use and replace intel with the wanted compiler.

You can list all available modules like this:

module avail

#### Cannot allocate memory messages

If you get error messages like

id: error while loading shared libraries: libsepol.so.1: failed to map segment from shared

object: Cannot allocate memory

/bin/hostname: error while loading shared libraries: libsepol.so.1: failed to map

segment from shared object: Cannot allocate memory

/usr/bin/id: error while loading shared libraries: libsepol.so.1: failed to map

segment from shared object: Cannot allocate memory

/etc/profile.d/krb5-devel.sh: line 5: [: =: unary operator expected

/usr/bin/id: error while loading shared libraries: libsepol.so.1: failed to map

segment from shared object: Cannot allocate memory

/etc/profile.d/krb5-workstation.sh: line 5: [: =: unary operator expected

/site/Modules/3.2.6/bin/modulecmd: error while loading shared libraries: libX11.

so.6: failed to map segment from shared object: Cannot allocate memory

srun: fatal: Invalid user for SlurmUser slurm, ignored

/usr/bin/perl: error while loading shared libraries: libm.so.6: failed to map segment

from shared object: Cannot allocate memory

in the slurm-jobid.out file, it usually means that you have specified a too low memory limit in the script. You should get a warning like

-- Warning: Ulimit will be set to NM. The job may have problems starting

due to too small memory limit.

from sbatch if you specify --mem-per-cpu less than 100M.

### Different compilers and gcc

The modules for the different compilers set the include path flags CPATH (for C) and FPATH (for fortran) so that the coresponding include files are searched for in the corresponding paths. This may cause problems when the CPATH points to include files for another compiler and you want to use gcc. The modules have been set up to use the compilers you ask for. If you want to mix different compier like ifort and gcc you need to set the paths in correspondence. Simply unsetting (unset CPATH) will in most cases cause gcc to search for it's built in paths.

The same goes for the flags, FC, F77, F90, F95, CC, CXX etc. For openMPI there is also flags that sets the corresponding compilers : OMPI\_CC, OMP\_CXX, OMPI\_FC, OMPI\_F77. This is typically the case when you want to use ifort for fortran and gcc for c. Setting OMPI\_CC to gcc will cause mpicc to use gcc.

### Intel compiler issues

#### Which flags to use ?

By default the Intel compiler sets a fairly good set of compiler flags for optimization. Our suggestion is the following :

-O3

-xAVX

-mavx

-fomit-frame-pointer (using this prohibits debugging information, do not use during development/debugging)

-fno-alias

In some cases for memory intensive code, -O2 yields better performance -O3, try it out. The easy help : ifort -help gived an overview of switches.

#### MPI linking problems, missing symbols

MPI implementations are compiled and built for each vendor's Fortran compiler. Each vendor has a different scheme for naming symbols, which is usually done by a number of underscores. Some append one or more underscores while some do not. As the MPIs are written in C, each function must have the right name for the linker to find and resolve the symbols. This why an MPI like OpenMPI comes in several different builds.

#### Warning about feupdateenv is not implemented

This problem arises because you are linking against the Intel math library only. The Intel math library (libimf) has optimized replacements for many of the routines in the system math library (libm). However, the Intel math library does not provide replacements for all functions. The solution is simple: Always link against both math libraries: libimf libm

However, we are using optimized versions of libm, so just -lm does not help. The only possibility to remove this is to force linking of a library with this function. The function below shows that one library does not contain the warning symbol (0000000000000000 n \_\_evoke\_link\_warning\_feupdateenv) causing the error message to be printed.

nm /lib/libm.so.6 | grep feupdateenv

nm /lib/i686/nosegneg/libm.so.6 | grep feupdateenv

#### Warning about no version information available

This warning arises when there are versioning issues between Intel compilers and some GNU libraries.

This is normally not a problem, but it is possible to find out what kind of binaries or libraries that are involved by using the command ldd.

ldd -v <your executable>

will list libraries including version information. The "Not found" is an indication of potential problems.

### Where do I find BLAS, LAPACK, BLACS and SCALAPACK?

#### Linear algebra, BLAS/ATLAS/Goto etc

The linear algebra packages are now all part of the Intel Math kernel Library, MKL. BLAS and LAPACK functions are part of the core mkl library while the more special MPI based BLACS and SCALAPACk are available through MPI specific interfaces. See the Intel compiler section under software on Abel.

#### MPI Linear algebra, BLACS/SCALAPACK

For the MPI based linear algebra packages there are interfaces with the Intel compiler. See the Intel compiler section under software on Abel.

### Working and accessing files on /projects is slow

The /projects file-system on Abel is a clustered NFS export of the /projects area from the old Titan disk. At times with heavy access to this file-system users could experience significant latencies on file-system operations.

A procurement is under way for replacing this disk with a new one that will have better performance. The installation will start in January, and we expect to have the new disk in production and all project-areas moved by February-March time-frame.

If possible you should always do interactive work or run jobs from the new Abel filesystems /cluster and /work. If you use /work please remember that there is no backup on this filesystem.

### Why doesn't chkfile copy my files back?

There are two probable causes for this:

1. The job script was interrupted before the chkfile command was executed.  To avoid this, place the chkfile command early in the script, before the files you want copied back are created.
2. The arguments to chkfile contained unquoted shell meta-characters like \* and ?.  They must be quoted for chkfile to function properly.  I.e., use chkfile "MyResults\*" instead ofchkfile MyResults\*.

### Mac users - problem with 'locale' environment variables

In case of error messages concerning 'locale' environment variables (variables with names starting with LC\_) run at commandline 'locale' to check which variables are making problems and unset them in the .bashrc file (for example: unset LC\_CTYPE).

## How do I acknowledge help from you or use of Abel

The Norwegian name for our department is Seksjon for IT i forskningat USIT, aka ITF, but the English name is Research Computing  (RC), at USIT, University of Oslo, Norway. If you have used our services through the **Lifeportal** we appreciate if you include this in your acknowledgment.

An example of a short acknowledgment for having benefited from our help is

We want to acknowledge the help received from the [Department for Research Computing at USIT](http://www.usit.uio.no/english/about/organisation/bps/rc/),

the University of Oslo IT-department.

An example of an acknowledgment of having used Abel is

This work was performed on the Abel Cluster, owned by the University of Oslo and the Norwegian metacenter

for High Performance Computing (NOTUR), and operated by the Department for Research Computing at USIT,

the University of Oslo IT-department. <http://www.hpc.uio.no/>

You should also acknowledge the Research Council of Norway if your work on Abel was carried out with a grant from NOTUR.

If you publish a paper where we, or rather one or more of us, are co-authors, please contact us for the correct author information.

More about Abel

Abel is the high performance computing facility at [UiO](http://www.uio.no/english/research) hosted by [USIT](http://www.usit.uio.no/) by the [RIS](http://www.usit.uio.no/english/about/organisation/bps/rc/ris/)(Research Infrastructure Services) group.

Abel is a powerful computing cluster boasting over 650 computers and having over 10000 cores (CPUs). Abel compute nodes typically have 64GiB memory and are all connected to a large common scratch disk space. All nodes in the Abel cluster have FDR InfiniBand providing low latency and high bandwidth connection between all nodes. All nodes run the Linux Operating system (64 bit CentOS 6).

To get access to Abel, see [getting access to Abel](http://www.uio.no/english/services/it/research/hpc/abel/help/access/). We also maintain a detailed [user guide](http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/) and a[FAQ](http://www.uio.no/english/services/it/research/hpc/abel/help/faq/).

Key numbers

|  |  |
| --- | --- |
| **Number of Cores** | **10000+** |
| **Number of nodes** | **650+** |
| **Max Floating point performance, double** | **258 Teraflops/s** |
| **Total memory** | **40 TebiBytes** |
| **Total local storage** | **400 TebiBytes using [FhGFS](http://www.fhgfs.com/)** |

 Hardware

Abel consist of an array of compute nodes and systems around to support them, login, admin and storage nodes.

Compute nodes

The 650+ Supermicro X9DRT compute nodes are all dual Intel E5-2670 (Sandy Bridge) based running at 2.6 GHz, yielding 16 physical compute cores. Each node have 64 GiBytes of Samsung DDR3 memory operating at 1600 MHz, giving 4 GiB memory per physical core at about 58 GiB/s aggregated bandwidth using all physical cores.

Storage

The storage is provied as two equal size partitions /cluster and /work each capable of a performance of about 6-8 GiB/s when doing sequential IO. The file system is Fraunhofer Global Parallel File System (FhGFS). The storage elements are 2 TB SAS disk RAIDs at RAID level 6 (8+2 conf)(LSI RAID controller 9265-8i) and XFS for data and SAS/SSD RAIDs at RAID level 10 and EXT4 for the metadata. All IO is transported by the Infiniband fabric using RDMA, with failover to IPoIB and IP/GbE. There are 10 IO servers and 2 redundant (active/passive) servers for the metadata.

Hugemem compute nodes

There are also a few nodes with more memory, 1 TiB, and more cores (32). They are Intel E-4620 based running at 2.20 GHz.

Accelerated compute nodes

There is also a set of accelerated nodes with NVIDIA Kepler II cards installed.

Interconnects

Abel network equipment:

* FDR (56 Gbits/s eq 6.78 Gbytes/s) InfiniBand between all nodes
* IP over IB on all nodes (enabling fast tcp communication)
* Gigabit Ethernet on all nodes
* Abel is connected to the other compute facilities in Norway by 10GbE links
* Abel has a dedicated 10GbE link to [CERN](http://public.web.cern.ch/public/) in connection to [Tier-1](http://public.web.cern.ch/public/en/LHC/Computing-en.html) responsibilities

Software

Nodes in Abel run Linux, 64 bit Centos 6.

Please see [Abel software](http://www.uio.no/english/services/it/research/hpc/abel/help/software) for installed software.

Queue system

A queue system ensures effective utilization of the Abel infrastructure. Abel uses the [Slurm](http://www.schedmd.com/slurmdocs/)queue system.