Exercises: Abel and SLURM

Nov 14, 2018

Sabry Razick

Research Infrastructure Services Group, USIT
Topics

• Connect to Abel/Colossus
• Running a simple job
• Running a simple job -- qlogin
• Job script
• Customize batch script
• Parallel jobs
Log into Abel

• If on Windows, download
  – Git BASH/Putty for connecting
  – WinSCP for copying files

• On Unix systems, open a terminal and type:
  
  ssh <username>@abel.uio.no

• Use your UiO/Nortur username and password.
PuTTY Configuration

Category:
- Session
  - Logging
- Terminal
  - Keyboard
  - Bell
  - Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address)  Port
abel.uio.no  22

Connection type:
- Raw
- Telnet
- Rlogin
- SSH
- Serial

Load, save or delete a stored session

Saved Sessions

Default Settings

User@abel.uio.no

Load
Gitbash

windows_pc: ssh user_@abel.uio.no
What are the available software?

- On Abel issue the following command to list modules/installed software
  - module avail
  - module avail python

- Load a module
  - module load python2/2.7.10

- Currently loaded modules
  - module list

- Clear modules
  - module purge

- http://www.uio.no/english/services/it/research/hpc/abel/help/software/
Examples - file location

/cluster/teaching/abel_tutorial/NOV2018
Interactive login (Qlogin)
Running a job on the cluster -1

- Login to Abel from your laptop
- Request to occupy some resources from workload manager
- Wait until workload manager grant you the resources
- Execute the job as it was in your laptop
> qlogin --account=xxx --time=00:10:00
salloc: Pending job allocation 12989428
salloc: job 12989428 queued and waiting for resources
salloc: job 12989428 has been allocated resources
salloc: Granted job allocation 12989428
srun: Job step created
>source /cluster/bin/jobsetup
>hostname
Job script
● Login to Abel from your laptop
● Create a job script, with parameters and include the program to run
● Hand it over to the workload manager
● The workload manager will handle the job queue, monitor the progress and let you know the outcome.
● Inspect results
#!/bin/bash

#SBATCH --job-name=RCS2018_hello
#SBATCH --account=ln0002k
#SBATCH --time=00:02:00
#SBATCH --mem-per-cpu=256M
#SBATCH --ntasks=1

source /cluster/bin/bin/jobsetup
set -o errexit

echo "Hello from Abel"
hostname
sleep 90
Running a simple job

- Create a directory as follows
  - `mkdir $HOME/RCS_tutorial`
- Find the project name
  - `projects`
- Create job script
  - Sample script from following location
  - `cd $HOME/RCS_tutorial`
  - `cp /cluster/teaching/abel_tutorial/NOV2018/hello.slurm $HOME/RCS_tutorial`
- Set permission, so you can edit it
  - `chmod +w hello.slurm`
Running a simple job

• Correct the project name in the hello.slurm file
• Submit the job to Abel
  • `sbatch hello.slurm`
Join the queue

> sbatch hello.slurm

• Investigate while the job is running

> squeue -u $USER
> scontrol show job ####

• If you want to cancel the job (don't do this now)

> scancel ######
>scontrol show job 12989353
     JobId=12989353 Name=RCS1115_hello
     UserId=sabryr(243460) GroupId=users(100)
     Priority=22501 Nice=0 Account=staff
     QOS=staff
         JobState=COMPLETED  Reason=None
...
     RunTime=00:00:02 TimeLimit=00:01:00 Ti
     Command=../RCS_tutorial/hello.slurm
     WorkDir=../RCS_tutorial
     StdErr=../RCS_tutorial/slurm-12989552.out
     StdOut=../RCS_tutorial/slurm-12989552.out
Input/Output (IO)
I/O - Files

• If you are accessing a file multiple time during a job
  • User scratch directory
  • Use /work/users/<USERNAME>

• Choose where you start the job from
  • On Abel jobs accessing files and/or writing out large outputs will run faster if /work is used compared to running from $HOME
#!/bin/bash

#SBATCH --job-name=RCS2018_usescratch
#SBATCH --account=xxx
#SBATCH --time=00:00:15
#SBATCH --mem-per-cpu=1G
#SBATCH --ntasks=1

source /cluster/bin/jobsetup
set -o errexit

echo $SUBMITDIR
echo $SCRATCH
cp $SUBMITDIR/input* $SCRATCH
callfile output.txt
cat input* > output.txt
Scratch job

• Create job script
  • Sample script from following location
  • `cd $HOME/RCS_tutorial`
  • `cp /cluster/teaching/abel_tutorial/NOV2018/scratch* $HOME/RCS_tutorial`

• Set permission, so you can edit it
  • `chmod +w scratch.slurm`

• Submit job
  • `sbatch scratch.slurm`
I/O - Files

• When handling very large number of files try to use
  • Archives - just in time extract or write directly to archive
  • Cleanup unwanted files
  • Copy back only output files needed when using $SCRATCH, e.g., do not copy back input data

• Get advice on using /tmp directory on compute nodes when thousands of files.
I/O - Files

- Create an archive (no compression)
  - tar -cvf <ARCHIVE_NM> <FILES>
  - tar -cvf file.tar *txt

- List content
  - tar -tvf <ARCHIVE_NM>

- Append a file
  - tar --append --file <ARCHIVE_NM> <NEW_FILES>
  - tar --append --file files.tar.gz 1.txt

- Extract all
  - tar -xvzf <ARCHIVE_NM>

- Extract one
  - tar -xvf <ARCHIVE_NM> <FILES>
sbatch - memory

- `#SBATCH --mem-per-cpu=<#G/M>`
  - Memory required per allocated core (format: 2G or 2048M)
  - How much memory should one specify? The maximum usage of RAM by your program (plus some). Exaggerated values might delay the job start.
- `#SBATCH --partition=hugemem`
  - If you need more than 61GB of RAM on a single node (up to 1 TiB).
sbatch - time

- `#SBATCH --time=hh:mm:ss`
  - Wall clock time limit on the job
  - Some prior testing is necessary. One might, for example, test on smaller data sets and extrapolate. As with the memory, unnecessarily large values might delay the job start.
  - This costs you (from allocated operation resources)
  - Until a job is finished this will be reserved.
- `#SBATCH --begin=hh:mm:ss`
  - Start the job at a given time (or later)
- `#SBATCH --partition=long`
  - Maximum time for a job is 1 week (168 hours). If more needed, use hugemem or long partitions
sbatch – CPUs and nodes

- Does your program support more than one CPU?
- If so, do they have to be on a single node?
- How many CPUs will the program run efficiently on?
- `#SBATCH --nodes=Nodes`
  - Number of nodes to allocate
- `#SBATCH --ntasks-per-node=Cores`
  - Number of cores to allocate within each allocated node
- `#SBATCH --ntasks=Cores`
  - Number of cores to allocate
- `#SBATCH --cpus-per-task=Cores`
  - (Threads) on one node
sbatch – CPUs and nodes

- `#SBATCH --ntasks=17`
  - If you just need some cpus, no matter where:
- `#SBATCH --nodes=8 --ntasks-per-node=4`
  - If you need a specific number of cpus on each node
- `#SBATCH --nodes=1 --ntasks-per-node=8`
  - If you need the cpu's on a single node
- `#SBATCH --nodes=1 --exclusive`
  - A node only for you
sbatch - constraint

- #SBATCH --constraint=feature
  - Run job on nodes with a certain feature - *ib*, *rackN*.
- #SBATCH --constraint=ib
  - Run job on nodes with infiniband (Gigabit Ethernet)
  - All nodes on Abel are equipped with InfiniBand (56 Gbits/s)
  - Select if you run MPI jobs
- #SBATCH --constraint=ib&rack21
  - If you need more than one constraint
  - *in case of multiple specifications, the later overrides the earlier*
sbatch - files

- `#SBATCH --output=file`
  - Send 'stdout' (and stderr) to the specified file (instead of slurm-xxx.out)
- `#SBATCH --error=file`
  - Send 'stderr' to the specified file
- `#SBATCH --input=file`
  - Read 'stdin' from the specified file
sbatch – low priority

- `#SBATCH --qos=lowpri`
  - Run a job in the lowpri queue
  - Even if all of your project's cpus are busy, you may utilize other cpus
  - Such a job may be terminated and put back into the queue at any time.
  - If possible, your job should ensure its state is saved regularly, and should be prepared to pick up on where it left off.
- Note: Notur projects cannot access lowpri.
#!/bin/bash

#SBATCH --job-name=RCS1115_hello
#SBATCH --account=staff
#SBATCH --time=00:01:05
#SBATCH --mem-per-cpu=512M
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1

source /cluster/bin/jobsetup

echo "SLURM_JOBID=" $SLURM_JOBID
echo "SCRATCH=" $SCRATCH
echo "SLURM_NPROCS=" $SLURM_NPROCS
echo "SLURM_CPUS_ON_NODE=" $SLURM_CPUS_ON_NODE
echo "SUBMITDIR=" $SUBMITDIR
echo "TASK_ID=" $TASK_ID
Don't

- Make sure that the same requested is not made again. Last instruction overrides previous (maybe).
  - #SBATCH --nodes=4
  - #SBATCH --nodes=2
- Request far more than you actually need (identify the bottlenecks).
- Try to make an inefficient program go faster by pumping more resources.
- Include any other instruction before all the SBATCH instructions are given.
Parreralizing example
Parreralizing example

Task:
1. Read the input file
3. Write this to a new file

Serial way of doing this:
Read to 2 process at the same time
Could be nearly twice as fast
Requires less resources per process
#!/bin/bash

#SBATCH --array=0-3
#SBATCH --job-name<ArrayTest
#SBATCH --time=00:10:00
#SBATCH --account=ln0002k
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=1G

source /cluster/bin/jobsetup
set -o errexit # exit on error

./serial.sh $SLURM_ARRAY_TASK_ID
Scratch job

• Create job script
  • Sample script from following location
  • `cd $HOME/RCS_tutorial`
  • `cp /cluster/teaching/abel_tutorial/NOV2018/array-test* $HOME/RCS_tutorial`

• Set permission, so you can edit it
  • `chmod +w array-test.slurm`
  • Edit the project
  • Note usage of array-test.slurm

• Submit job
  • `sbatch array-test.slurm`
Thank you.

http://www.uio.no/english/services/it/research/hpc/abel/

hpc-drift@usit.uio.no