Introduction to Abel/Colossus and the queuing system

November 02, 2017

Sabry Razick

The Research Computing Services Group, USIT
Topics

• Abel/Colossus details, Getting an account & Logging in
• Understanding resources
• Queuing system
• Running a simple job
The Research Computing Services Seksjon for IT i Forskning

• The RCS group provides access to IT resources and high performance computing to researchers at UiO and to NOTUR users

• http://www.uio.no/english/services/it/research/

• Part of USIT

• Contact:
  • Abel : hpc-drift@usit.uio.no
  • TSD : tsd-drift@usit.uio.no
The Research Computing Services

• Operation of high performance computer clusters
• User support
• Data storage
• Secure data analysis and storage - TSD
• Portals
  • Lifeportal (https://lifeportal.uio.no/)
  • Lap Language (https://lap.clarino.uio.no/)
  • Geo (https://geoportal-dev.hpc.uio.no)
Abel/Colossus

• Computer cluster
  – Similar Computers connected by a local area network (LAN). Different than a Cloud or a Grid.

• Enables parallel computing

• Science presents multiple problems of parallel nature
  – Sequence database searches
  – Genome assembly and annotation
  – Simulations
Bunch of computers -> Cluster

• Hardware
  – Powerful computers (nodes)
  – High-speed connection between node
  – Access to a common file system

• Software
  – Operating system 64 bit Centos 6.8 (Rocks Cluster Distribution based)
  – Identical mass installations.
  – Queuing system enables timely execution of many concurrent processes
Numbers

- Nodes - 700+ (Abel - 703), (Colossus - 68)
- Cores - 10000+ (Abel - 11,392), (Colossus - 1392)
- Total memory - 50 TiB+ (Abel - 50), (Colossus 5)
- Total storage - 400 TiB using BeeGFS
- 96th most powerful in 2012, now 444th (June 2015)
Getting access

• If you are working or studying at UiO, you can have an Abel account directly from us.

• If you are Norwegian scientist (or need large resources), you can apply through NOTUR –
  • [https://www.sigma2.no/](https://www.sigma2.no/)

• Write to us for information:
  • [hpc-drift@usit.uio.no / tsd-drift@usit.uio.no](mailto:hpc-drift@usit.uio.no, tsd-drift@usit.uio.no)

• Read about getting access:
  • [http://www.uio.no/hpc/abel/help/access](http://www.uio.no/hpc/abel/help/access)
  • [https://www.uio.no/english/services/it/research/storage/sensitive-data/access/](https://www.uio.no/english/services/it/research/storage/sensitive-data/access/)
Connecting to Abel

- Linux
  - Redhat - RHEL
  - Ubuntu
- Windows - using Putty, Gitbash, WinSCP
  - https://git-for-windows.github.io/
- Mac OS
Connecting to Colossus

- Must come through TSD VM and needs to request HPC to use Colossus.
- Linux and Windows VMs (virtual machines)
- Thinlinc
  - https://login.tl.tsd.usit.no/main/
- Vmware horizon
  - https://view.tsd.usit.no/
Available software

• Available:
  
  http://www.uio.no/hpc/abel/help/software

• Software organized as modules.
  – List all software (and version) organized in modules:
    • module avail
  
  – Load software from a module:
    • module load module_name
    • (e.g module load python/2.9.10)

• Install your own software
  – Separate lecture tommrrrow 12:15
Using Abel

- When you log into Abel you are in one of the login nodes login0 - login3.
- Please **DO NOT** execute programs (jobs) directly on the login nodes.
- Jobs are submitted to Abel via the queuing system.
- The login nodes are just for logging in, copying files, editing, compiling, running short tests (no more than a couple of minutes), submitting jobs, checking job status, etc.
- For interactive execution use `qlogin`.
We do NOT run jobs here.
Using Colossus

- You may run jobs in your VM if you wish
- For heavy jobs use Colossus (procedure same as Abel)
- There is no qlogin on TSD
Queue management - SLURM

- Simple Linux Utility for Resource Management (workload manager)
- Allocates exclusive and/or non-exclusive access to resources (computer nodes) to users for some duration of time
- Provides a framework for starting, executing, and monitoring work on a set of allocated nodes.
- Managing a queue of pending work.
Fair resource allocation

• When you request resources, SLURM will consider number of things before granting it
  • Does your project has enough CPU hours to pay for this. It will consider total allocated and reserved (running jobs) when doing this.
  • Is your account using more than the allowed resources.
  • Can/should the cluster provide you with resources (resource combination)
  • Depending on the current load, how long others need to wait if your job starts.
SLURM
Running a job on a laptop compared to submitting to a queue

- Double click on an icon, give parameters or upload data - wait
- Terminal
  - ./myprg input1.txt out_put.file
- Inspect results
Computing on Abel

• Submit a job to the queuing system
  – Software that executes jobs on available resources on the cluster (and much more)

• Communicate with the queuing system using a shell (or job) script

• Retrieve results (or errors) when the job is done

• Read tutorial: http://www.uio.no/hpc/abel/help/user-guide
Interactive login (Qlogin)
Abel only
Running a job on the cluster -1

- Login to Abel from you laptop
- Request to occupy some resources from SLURM
- Wait until SLURM grant you the resources
- Execute the job as it was in your laptop
qlogin

- Reserve some resources for a given time.
- Example - Reserve one node (or 16 cores) on Abel for your interactive use for 1 hour:

```
qlogin

--account=your_project
--nodes=1 --ntask=16
--mem-per-cpu=3G
--time=01:00:00
```

http://www.uio.no/english/services/it/research/hpc/abel/help/user-guide/interactive-logins.htm
Interactive use of Abel - qlogin

```
login-0-0.local : qlogin --account=staff --time=00:10:10 --nodes=1 --ntasks=2 --mem-per-cpu=1G
salloc: Pending job allocation 16976652
salloc: job 16976652 queued and waiting for resources
salloc: job 16976652 has been allocated resources
salloc: Granted job allocation 16976652
srun: Job step created
bash-4.1$ export PS1="$(hostname) :
compute-1-34.local :
compute-1-34.local :
compute-1-34.local : source /cluster/bin/jobsetup
Starting job 16976652 ("qlogin") on c1-34 at ma. 03. april 23:57:57 +0200 2017
compute-1-34.local :
compute-1-34.local : echo $SCRATCH
/work/jobs/16976652.d
compute-1-34.local : echo $SLURM_JOB_USER
sabryr
compute-1-34.local : set | grep SLURM
SLURMD_NODENAME=c1-34
SLURM_CHECKPOINT_IMAGE_DIR=/var/slurm/checkpoint
SLURM_CLUSTER_NAME=abel
```
Job script
Abel and Colossus
Running a job on the cluster - 2

- Login to Abel/TSD Linux/TSD Windows+putty
- 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
Resources
- Supermicro X9DRT compute node
- Dual Intel E5-2670 (Sandy Bridge) based running at 2.6 GHz (2 sockets)
- 16 physical compute cores.
- Each node have 64 GiBytes of Samsung DDR3 memory
tasks?

- A piece of work to be done
- The computing resource needed for that
- A normal compute node on abel has two processors which can do 8 things each.
- So a compute node can do 16 things at once
#SBATCH --ntasks=8

OR

OR

-----------------------

1 1 1 1 1 1 1 1 1
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=8
8 X 1 = 8
*All tasks will share memory*

#SBATCH --nodes=2
#SBATCH --ntasks=2
#SBATCH --cpus-per-task=8
8 X 2 = 16

#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
2 X 4 = 8
Calculating CPU hours

● Use one task for one hour = 1 cpu hour
● If you use one entire compute node for one hour
  ○ 16 X 1 = 16 CPU hours
● For more precise value - next slide
Calculating CPU hours

KMEM=4580.2007628294
(/cluster/var/accounting/PE_factor)

PE= NumCPUs

if(MinMemoryCPU>KMEM){
    PE=PE*(MinMemoryCPU/KMEM)
}

PE_hours = $PE * TimeLimit / 3600
#SBATCH --nodes=1
#SBATCH --time=01:00:00
#SBATCH --ntasks-per-node=4
#SBATCH --mem-per-cpu=15G

*only 4/16 cores are used
*but all memory occupied,

\[
\text{KMEM} = 4580.2007628294 \\
\text{PE} = 4 \\
#(15 \times 1024) > \text{KMEM} \text{ so} \\
\text{PE} = 4 \times \left(\frac{15 \times 1024}{\text{KMEM}}\right) = 13.41 \\
\text{PE}_{\text{hours}} = 13.41 \times \left(\frac{1 \times 60 \times 60}{3600}\right) = 13.41
\]

**Use the command **cost** to check account balance.**
Project/Account

• Each user belongs to one or more project on Abel

• Colossus uses use the same value as their TSD project

• Each project has set of resources

• Learn about your project(s):
  – Use: projects
Job script

• Job script - shell script including the command that one needs to execute (order is important)

• EXTRA comments read by the queuing system
  • “#SBATCH --xxxx”

• Compulsory values:
  #SBATCH --account
  #SBATCH --time
  #SBATCH --mem-per-cpu

• Setting up a job environment
  source /cluster/bin/jobsetup
Example job script

#!/bin/bash

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

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

sleep 1m
python hello.py
Submitting a job - `sbatch`

```
-bash-4.1$ sbatch myjob.slurm
Submitted batch job 16973754
-bash-4.1$  
```

Job ID
Checking a job

- squeue -u <USER_NAME>
- scontrol show job <JOB_ID>
- After the job has ended
  - sacct -j <JOB_ID>
Use of the SCRATCH area

```
#!/bin/sh
#SBATCH --job-name=Job_1
#SBATCH --account=P1
#SBATCH --time=00:30:00
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=m256M
source /cluster/bin/jobsetup

## Copy files to work directory:
cp $SUBMITDIR/YourData $SCRATCH

## Mark outfiles for automatic copying to $SUBMITDIR:
chkfile YourOutput

## Run command
cd $SCRATCH
executable YourData > YourOutput
```
Some usefull commands

• **scancel** `<JOBID>` - Cancel a job before it ends
• **dusage** - find out your disk usage
• **squeue** - list all queued jobs and find out the position of your job
  • `STATE` - `PENDING,RUNNING,SUSPENDED,COMPLETED,CANCELLED,FAILED,TIMEOUT,NODE_FAIL,PREEMPTED,BOOT_FAIL,COMPLETING,CONFIGURING,RESIZING,SPECIAL_EXIT`
• **cost** - account balance
Environment variables

• SLURM_JOBID – job-id of the job
• SCRATCH – name of job-specific scratch-area
• SLURM_NPROCS – total number of cpus requested
• SLURM_CPUS_ON_NODE – number of cpus allocated on node
• SUBMITDIR – directory where sbatch were issued
• TASK_ID – task number (for arrayrun-jobs)
Arrayrun

- Parallel jobs - executing many instances of the same executable at the same time.
- Many input datasets
- Simulations with different input parameters.
- Possible to split a large input file into chunks and parallelize your job.
Message Passing Interface

MPI is a language-independent communications protocol used for programming parallel computers.

We support Open MPI

- module load openmpi

Jobs specifying more than one node automatically get

- `#SBATCH --constraint=ib`
Thank you.

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

hpc-drift@usit.uio.no