University of Manitoba

Running jobs on HPC clusters:

All you should know to submit and monitor your jobs

UofM-Autumn-Workshop 2023

Nov. 6th-8th, 2023

Ali Kerrache

HPC Analyst

Outline

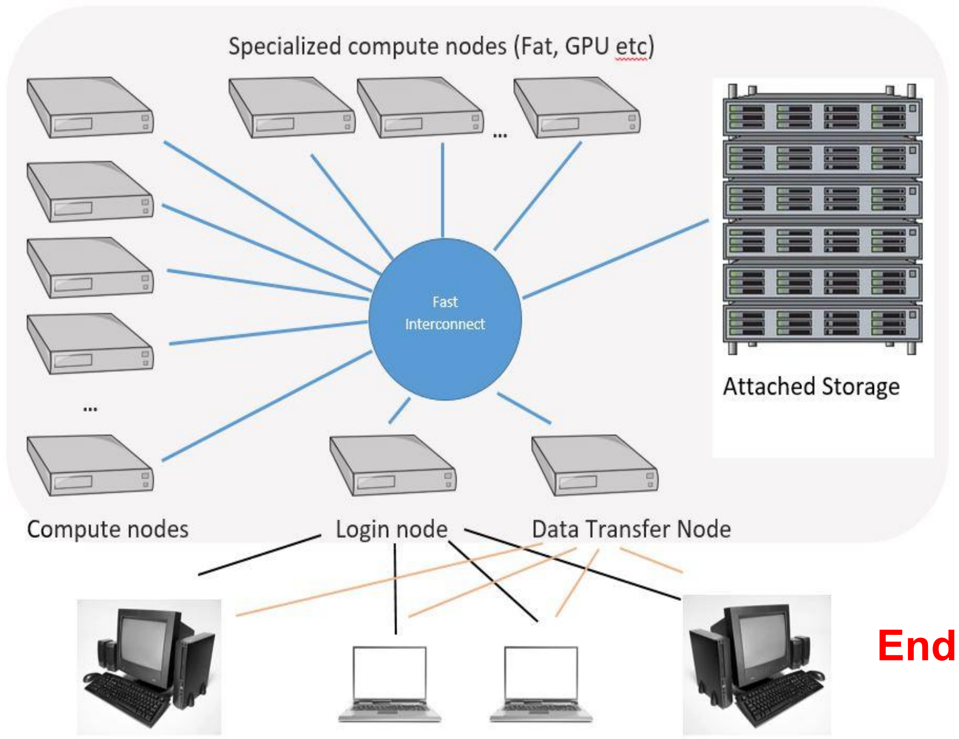
- ➔ **SLURM as a scheduler for HPC:**
  - ◆ Why use a scheduler?
  - ◆ SLURM
- ➔ **Jobs:**
  - ◆ Login nodes and Interactive jobs
  - ◆ Batch jobs
  - ◆ OpenOnDemand on Grex
- ➔ **Type of jobs and script examples:**
  - ◆ Serial
  - ◆ OpenMP
  - ◆ MPI and Hybrid
  - ◆ GPU

HPC workflow

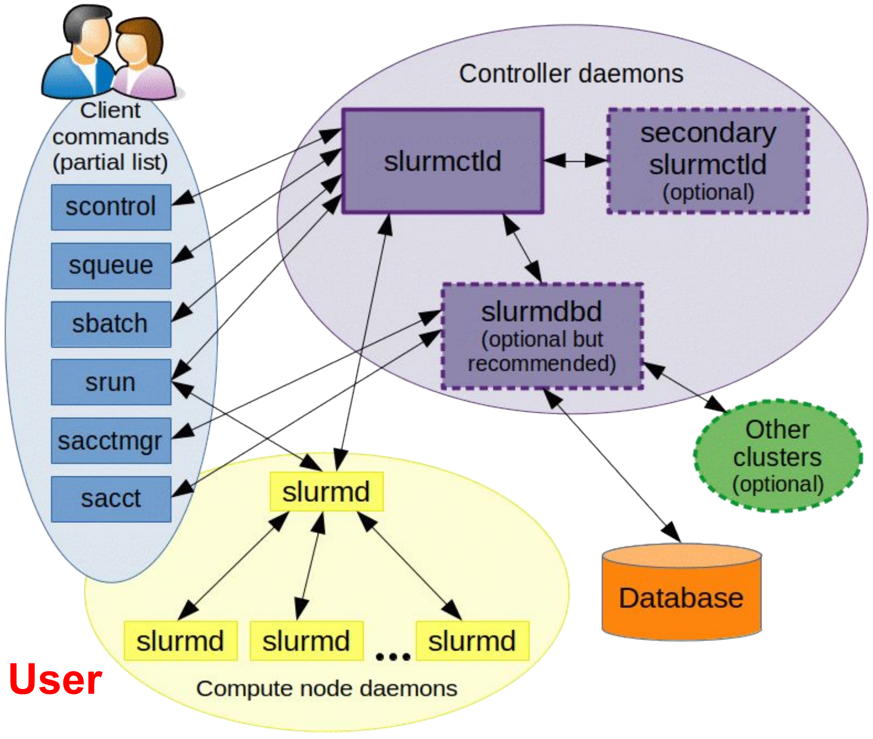
- ★ Connect
- ★ Transfer files
- ★ Compile codes
- ★ Testing jobs
- ★ Running jobs
- ★ Analyze data
- ★ Visualisation

Running jobs on HPC clusters

HPC cluster components



End User


Client commands (partial list)
scontrol
squeue
sbatch
srun
sacctmgr
sacct
Controller daemons
slurmctld
secondary slurmctld (optional)
slurmdbd (optional but recommended)
Other clusters (optional)
Database
slurmd
slurmd
slurmd
...
slurmd
Compute node daemons

Slurm components

University of Manitoba

Resources on Grex: partitions

Partition	Nodes [CPUs/GPUs]	Cores	Total	Memory	Wall Time
compute[1]	312	12	3456	48 GB	21 days
largemem	12	40	480	376 GB	14 days
skylake	42	52	2184	188 GB	21 days
gpu	2 [4 V100 - 32 GB]	32	64	187 GB	3 days
stamps; -b	3 [4 V100 - 16 GB]	32	96	187 GB	21 days / 7 days
livi; -b	[16 V100 - 32 GB]	48	48	1.5 TB	21 days / 7 days
agro; -b	2 AMD [A30]	24	48	250 GB	21 days / 7 days
test	-	18	18	500 GB	12 hours

[1] to be decommissioned in the near future.

https://um-grex.github.io/grex-docs/

Running jobs on HPC clusters

- ★ Job requirements: CPUs, Memory, Time, … etc.
- ★ SLURM template: structure of a job script
- ★ Interactive jobs via salloc
- ★ Example of SLURM script: Gaussian
- ★ SLURM directives
- ★ SLURM environment variables
- ★ Examples: Serial, OpenMP, MPI, GPU
- ★ *Bundle multiple jobs: job arrays and GLOST*
- ★ Monitor and control your jobs: seff, scancel, sacct, …
- ★ *Estimating resources: CPUs, MEM, TIME*
- ★ *How to pick a partition on Grex?*

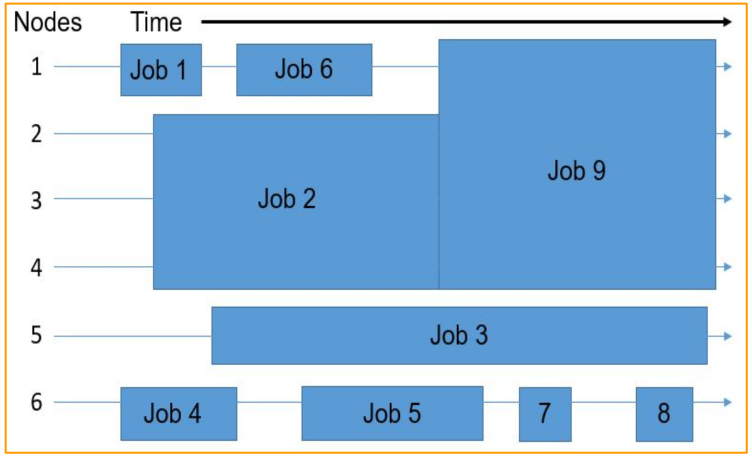
Scheduler: SLURM

SLURM: Simple Linux Utility for Resource Management

- ➔ free and open-source job scheduler for Linux and Unix-like kernels
- ➔ used by many of the world's supercomputers and computer clusters.

https://slurm.schedmd.com/overview.html

sacct - sacctmgr - salloc - sattach - sbatch - sbcast - scancel - scontrol - sdiag - seff - sh5util - sinfo - smail - smap - sprio - squeue - sreport - srun - sshare - sstat - strigger - sview



Interactive and batch jobs

- ★ When you connect you get interactive session on a login node:
  - ○ Limited resources: to be used with care for basic operations
    - ■ editing files, compiling codes, download or transfer data, submit and monitor jobs, run short tests {no memory intensive tests}
  - ○ Performance can suffer greatly from over-subscription
- ★ For interactive work, submit interactive jobs: salloc [+options]
  - ○ SLURM uses salloc for interactive jobs [compute nodes]
  - ○ The jobs will run on dedicated compute nodes [CPUs, GPUs]
- ★ Submitting batch jobs for production work is mandatory: sbatch
  - ○ Wrap commands and resource requests in a “job script”: myscript.sh
  - ○ SLURM uses sbatch; submit a job using: sbatch myscript.sh

    sbatch [+options] myscript.sh

Job requirements

What do you need to know before submitting a job?

- ◆ Is the program available? If not, install it or ask support for help.
- ◆ What type of program are you going to run?
  - ● Serial, Threaded [OpenMP], MPI based, GPU, …
- ◆ Prepare your input files: locally or transfer from your computer.
- ◆ Test your program:
  - ● Interactive job via salloc: access to a compute node
  - ● On the login node if the test is not memory nor CPU intensive.
- ◆ Prepare a script "myscript.sh" with the all requirements:
  - ● **Memory**, Number of cores, Nodes, Wall time, modules, **partition**, **accounting group**, command line to run the code.
- ➔ Submit and monitor the jobs: sbatch, squeue, sacct, seff … etc

Accounting groups

★ Submit Interactive job:

```
[~cedar5 scratch]$  salloc --ntasks=1 --mem=4000M
salloc: error: ----------------------------------------
salloc: error: You are associated with multiple _cpu allocations...
salloc: error: Please specify one of the following accounts to submit this job:
salloc: error:   RAS default accounts: def-prof1, def-prof2
salloc: error:           RAC accounts:
salloc: error: Compute-Burst accounts:
salloc: error:         Other accounts: cc-debug,
salloc: error: Use the parameter --account=desired_account when submitting your job
salloc: error: ----------------------------------------
salloc: error: Job submit/allocate failed: Unspecified error
```

★ Accounting groups: sshare -U --user <username>

- if one accounting group, SLURM will take it by default.
- If more than one, it should be specified via: --account={your accounting group}

University of Manitoba

Interactive jobs via salloc

```
[someuser@bison ]$  salloc --cpus-per-task=4 --mem-per-cpu=1000M --time=1:00:00
salloc: using account: def-someprof
salloc: No partition specified? It is recommended to set one! Will guess
salloc: Pending job allocation 5081294
salloc: job 5081294 queued and waiting for resources
salloc: job 5081294 has been allocated resources
salloc: Granted job allocation 5081294
salloc: Waiting for resource configuration
salloc: Nodes n063 are ready for job
        Load modules + run tests
[someuser@n063 ]$  exit
exit
salloc: Relinquishing job allocation 5081294
```

Equivalent SLURM script:

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --time=1:00:00
#SBATCH --account=def-someprof
```

Interactive jobs via salloc

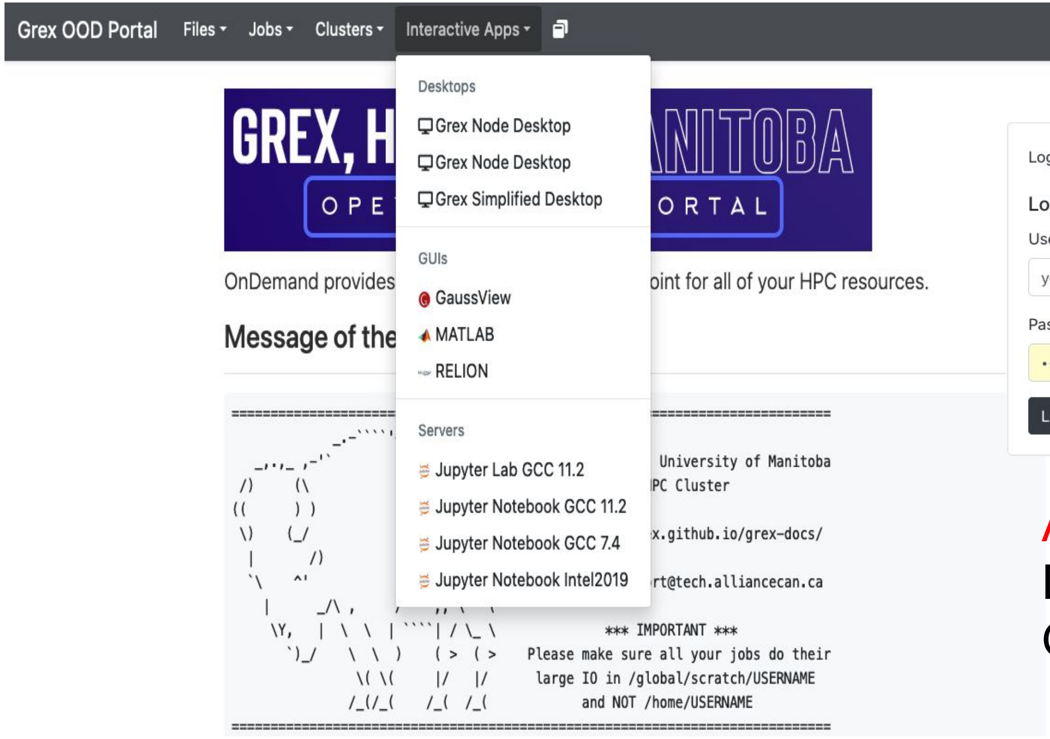
```
[someuser@bison ]$  salloc --ntasks=1 --cpus-per-task=4 --mem-per-cpu=1000M
--account=def-someprof --partition=skylake --x11

salloc: using account: def-someprof
salloc: partition selected:skylake
salloc: Granted job allocation 5081297
salloc: Waiting for resource configuration
salloc: Nodes n376 are ready for job
      Load modules + run tests
[someuser@n376 ]$  exit
exit
salloc: Relinquishing job allocation 5081297
```

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --mem=4000M
#SBATCH --time=3:00:00
#SBATCH --account=def-someprof
#SBATCH --partition=skylake
```

University of Manitoba

OOD web portal



Grex OOD Portal · Files · Jobs · Clusters · Interactive Apps

GREX, H… …NITOBA

OPE… …ORTAL

Desktops
- Grex Node Desktop
- Grex Node Desktop
- Grex Simplified Desktop

GUIs
- GaussView
- MATLAB
- RELION

Servers
- Jupyter Lab GCC 11.2
- Jupyter Notebook GCC 11.2
- Jupyter Notebook GCC 7.4
- Jupyter Notebook Intel2019

OnDemand provides … oint for all of your HPC resources.

Message of the …

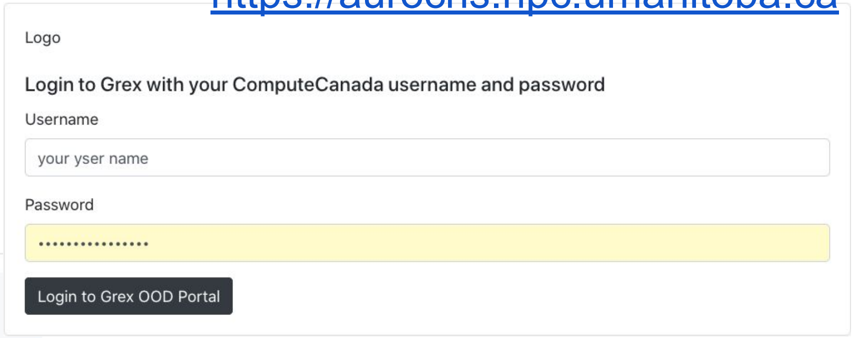
```
============                    ========================
                                 University of Manitoba
                                 PC Cluster

                                 x.github.io/grex-docs/

                                 rt@tech.alliancecan.ca

                                       *** IMPORTANT ***
                               Please make sure all your jobs do their
                                large IO in /global/scratch/USERNAME
                                       and NOT /home/USERNAME
=============================================================
```

https://aurochs.hpc.umanitoba.ca



Logo

Login to Grex with your ComputeCanada username and password

Username: your yser name

Password: ••••••••••••••••

Login to Grex OOD Portal

Access to:

Desktops, Jupyter, Terminal, Jos, Queue view, Running jobs, …

Hostname change: aurochs to zebu

https://um-grex.github.io/grex-docs/ood/

University of Manitoba

SLURM: basic template

```
#!/bin/bash

#SBATCH --account=def-somegroup
{Add the resources and some options}

echo "Current working directory is `pwd`"
echo "Starting run at: `date`”

{Load appropriate modules if needed}
{Command line to run your program}

echo "Program finished with exit code $? at: `date`"
```

Script: test-job.sh

Parameters to adjust for each type of job to submit: serial, MPI, GPU

Default parameters:

- CPUs: 1
- Time: 0-3:00
- Memory: 256mb

University of Manitoba

SLURM: most used directives

#SBATCH --account=def-someprof	Use the accounting group def-someprof for jobs.
#SBATCH --ntasks=8	Request 8 tasks for MPI job; 1 for serial or OpenMP
#SBATCH --cpus-per-task=4	Number of threads (OpenMP); Threaded application
#SBATCH --ntasks-per-node=4	Request 4 tasks per-node for MPI job
#SBATCH --nodes=2	–nodes=<Min>-<Max> Request 2 nodes
#SBATCH --mem=1500M	Memory of 1500M for the job
#SBATCH --mem-per-cpu=2000M	Memory of 2000M per CPU
#SBATCH --partition=compute	**GREX:** Partition name: compute, skylake, largemem, gpu, test
#SBATCH --time=3-00:00:00	Wall time in the format: DD-HH:MM:SS

University of Manitoba

SLURM: environment variables

SLURM_JOB_NAME	User specified job name
SLURM_JOB_ID	Unique slurm job id
SLURM_NNODES	Number of nodes allocated to the job
SLURM_NTASKS	Number of tasks allocated to the job
SLURM_ARRAY_TASK_ID	Array index for this job
SLURM_ARRAY_TASK_MAX	Total number of array indexes for this job: --array=0-999%10
SLURM_CPUS_PER_TASK	Number of threads {OpenMP: OMP_NUM_THREADS}
SLURM_JOB_NODELIST	List of nodes on which resources are allocated to a Job
SLURM_JOB_ACCOUNT	Accounting group under which this job is running.
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.

University of Manitoba

SLURM script: serial jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=1-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load <dep> <software>/<version>
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, tasks = 1, memory per core, wall time, **partition**, …
- Other: E-mail-notification, … etc.

Submit and monitor the job:

- sbatch myscript.sh
- squeue -u $USER; sq; sacct -j JOB_ID

More information:

- partition-list; sinfo --format="%20P"
- Sinfo -s; sinfo -p compute,skylake
- squeue -p compute,skylake -t R {PD}

University of Manitoba

SLURM script: OpenMP jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000M
#SBATCH --time=1-00:00:00
#SBATCH --partition=skylake
# Load appropriate modules:
module load <software>/<version>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
echo "Starting run at: `date`”
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2000M
#SBATCH --time=1-00:00:00
#SBATCH --partition=skylake
```

```
#SBATCH --cpus-per-task=N
#SBATCH --mem=<MEM>
```

Partitions:

- compute: N up to 12
- skylake: N up to 52
- largemem: N up to 40

University of Manitoba

SLURM script: Gaussian

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=7-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load gaussian
echo "Starting run at: `date`"
g16 < my-input.com > my-output.out
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, number of tasks, memory per core, wall time, **partition**, …
- Other: Email notification, … etc.

Submit and monitor the job:

- sbatch [some options] myscript.sh
- squeue -u $USER; sq

Partition:

- partition-list; sinfo --format="%20P"
- sinfo -s; sinfo -p <partition name>

University of Manitoba

SLURM script: MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --nodes=2-4
#SBATCH --ntasks=96
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=2-00:00:00
#SBATCH --partition=skylake
# Load appropriate modules:
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21
echo "Starting run at: `date`"
srun lmp_grex < in.lammps
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=12
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=52
#SBATCH --mem=0
#SBATCH --partition=skylake
```

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=40
#SBATCH --mem=0
#SBATCH --partition=largemem
```

University of Manitoba

SLURM script: OpenMP+MPI jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=6
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=3-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load <software>/<version>
export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}
echo "Starting run at: `date`”
srun program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=6
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=compute
```

The total memory and CPUs per node should not exceed the available resources on the nodes.

```
#SBATCH --nodes=5
#SBATCH --ntasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --partition=skylake
```

University of Manitoba

Script: by node versus by core

```
#SBATCH --nodes=8
#SBATCH --ntasks-per-node=12
#SBATCH --cpus-per-task=1
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
Job ID: 1234567
Cluster: grex
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Nodes: 8
Cores per node: 12
CPU Utilized: 156-11:07:22
CPU Efficiency: 99.22% of 157-16:44:48 core-walltime
Job Wall-clock time: 1-15:25:28
Memory Utilized: 218.00 GB (estimated maximum)
Memory Efficiency: 59.37% of 367.19 GB (45.90 GB/node)
```

The job used:

- **96 CPUs**
- **about 2400 M per core**

The job may wait longer on the queue to start:
it requires 8 nodes to be available
=> Optimize the resources

```
#SBATCH --ntasks=96
#SBATCH --mem-per-cpu=2400M
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=162
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=skylake
```

University of Manitoba

SLURM script: GPU jobs

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --gpu=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=6
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=0-3:00:00
#SBATCH --partition=gpu
# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`"
program.x [+options and arguments if any]
echo "Program finished with exit code $? at: `date`"
```

SLURM directives:

- Default: 1 core, 256mb, 3 hours
- **account**, number of tasks, memory per core, wall time, **partition**, …
- Other: E-mail-notification, … etc.

Submit and monitor the job:

- sbatch [some options] myscript.sh
- squeue -u $USER

Partition:

- partition-list; sinfo --format="%20P"
- sinfo -p <partition name>

SLURM: messages

- ★ None: the job is running (ST=R)
- ★ PartitionDown: one or more partitions are down (the scheduler is paused)
- ★ Resources: the resources are not available for this job at this time
- ★ Nodes required for job are DOWN, DRAINED or RESERVED for jobs in higher priority partitions: similar to Resources.
- ★ Priority: the job did not start because of its low priority
- ★ Dependency: the job did not start because it depends on another job that is not done yet.
- ★ JobArrayTaskLimit: the user exceeded the maximum size of array jobs

  -
    ```
    [~@tatanka ~]$  scontrol show config | grep MaxArraySize
    MaxArraySize          = 2000
    ```
- ★ ReqNodeNotAvail, UnavailableNodes: n314: node not available

University of Manitoba

Information about the cluster

★ sinfo: check the nodes (idle, drain, down), …

```
sinfo --state=idle       {shows idle nodes on the cluster}
sinfo --R                {shows down, drained and draining nodes and their reason}
sinfo --Node --long      {shows more detailed information}
sinfo --p largemem       {shows more detailed information}
```

★ scontrol: to see reservations and more

```
[~@gra-login1: ~]$ scontrol show res <Outage> --oneliner
ReservationName=Outage StartTime=2022-10-25T08:50:00 EndTime=2022-10-26T10:00:00
Duration=1-01:10:00 Nodes=gra[1-1257,1262-1325,1337-1338,1342] NodeCnt=1324
CoreCnt=44396 Features=(null) PartitionName=(null)
Flags=MAINT,IGNORE_JOBS,SPEC_NODES,ALL_NODES TRES=cpu=44396 Users=root
Groups=(null) Accounts=(null) Licenses=(null) State=INACTIVE BurstBuffer=(null) Watts=n/a
MaxStartDelay=(null)
```

University of Manitoba

Bundle many jobs: job array

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=3-00:00:00
#SBATCH --array=0-999%10
#SBATCH --partition=compute
# Load appropriate modules:
module load <software>/<version>
echo "Starting run at: `date`”
./my_code test${SLURM_ARRAY_TASK_ID}
echo "Program finished with exit code $? at: `date`"
```

- You have regularly named, independent datasets (test0, test1, test2, test3, …, test999) to process with a single software code
- Instead of making and submitting 1000 job scripts, a single script can be used with the **--array=1-999** option to **sbatch**
- Within the job script, $SLURM_ARRAY_TASK_ID can be used to pick an array element to process

```
./my_code test${SLURM_ARRAY_TASK_ID}
```

- When submitted, once, the script will create 1000 jobs with the index added to JobID (12345_1, … , 12345_999)
- You can use usual SLURM commands (scancel, scontrol, squeue) on either entire array or on its individual elements

Bundle many jobs: job array

- Files: n.melt-0.txt, …. In.melt-9.txt; array with 10 elements; Run a maximum of 2 at a time
- All the data in one directory: use appropriate names to avoid data overlapping

```
lmp_grex < in.melt-${SLURM_ARRAY_TASK_ID}.txt > log_lammps_array-${SLURM_ARRAY_TASK_ID}.txt
```

- Directories: 0, …. 9; each directory has a an input file: in.melt
- Job array with 10 elements
- Run a maximum of 2 at a time
- Output in different directories: the data may have the same name.

```
cd ${SLURM_ARRAY_TASK_ID}
lmp_grex < in.melt > log_lammps_array-${SLURM_ARRAY_TASK_ID}.txt
```

Bundle many jobs: GLOST

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --ntasks=4
#SBATCH --mem-per-cpu=2500M
#SBATCH --time=1-00:00:00
#SBATCH --partition=compute

# Load appropriate modules + glost:
module load intel/15.0.5.223  ompi glost

echo "Starting run at: `date`"
srun glost_launch list_glost_tasks.txt
echo "Program finished with exit code $? at: `date`"
```

- You have many short independent jobs (job1, job2, job3, …) to process with a single software code.
- Instead of submitting and running many jobs, a single script can be used to run these jobs as MPI job.

- List of tasks: list_glost_tasks.txt

```
job1
job2
job3
job4
job5
–
job199
job200
```

University of Manitoba

Monitor and control your jobs

```
squeue -u $USER [-t RUNNING] [-t PENDING]                    # list all current jobs.
squeue -p PartitionName [compute, skylake, largemem]    # list all jobs in a partition.
sinfo                                              # view information about Slurm partitions.
sacct -j jobID --format=JobID,MaxRSS,Elapsed     # resources used by completed job.
sacct -u $USER --format=JobID,JobName,AveCPU,MaxRSS,MaxVMSize,Elapsed
seff  -d jobID                        # produce a detailed usage/efficiency report for the job.
sprio [-j jobID1,jobID2] [-u $USER]                           # list job priority information.
sshare -U --user $USER                                          # show usage info for user.
sinfo --state=idle; -s; -p <partition>             # show idle nodes; more about partitions.
scancel [-t PENDING] [-u $USER] [jobID]                                  # kill/cancel jobs.
scontrol show job -dd jobID                           #show more information about the job.
```

University of Manitoba

Summary

- Account and active role:
  - CCDB
- Have a look to the documentation:
  - Hardware, available tools, …
  - policies?
  - login nodes
  - storage, …
- Tools to connect and transfer files
- Access to storage: home, scratch, project
- Access to a program to use:
  - Install the program or ask for it.
  - Use the existing modules

- Test jobs:
  - Login node
  - Interactive job via salloc
- Write a job script:
  - Slurm directives
  - Modules
  - Command line to run the code
- Monitor jobs:
  - Sacct; seff, optimize jobs
- Analyze data:
  - Post processing
  - Visualization

University of Manitoba

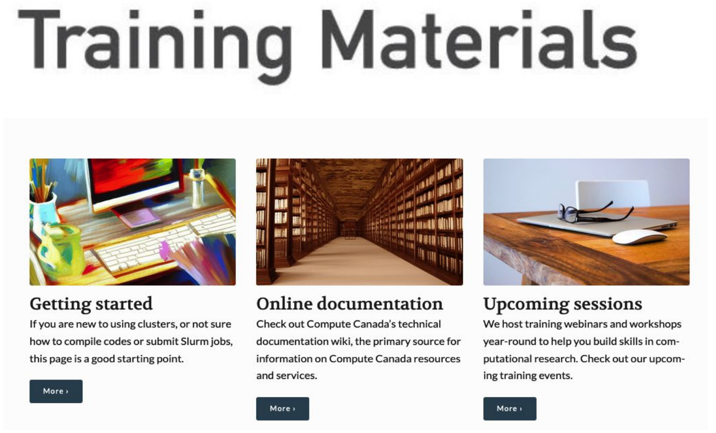
More readings

- The Alliance [Compute Canada]: https://docs.alliancecan.ca/wiki/Main_Page
- CCDB: https://ccdb.computecanada.ca/security/login
- CC Software: https://docs.alliancecan.ca/wiki/Available_software
- Running Jobs: https://docs.alliancecan.ca/wiki/Running_jobs
- SLURM: https://slurm.schedmd.com/
- PuTTy: http://www.putty.org/
- MobaXterm: https://mobaxterm.mobatek.net/

- Grex: https://um-grex.github.io/grex-docs/

→ WG training material: https://training.westdri.ca/

→ Help and support {Grex+Alliance}: support@tech.alliancecan.ca


Training Materials

Getting started
If you are new to using clusters, or not sure how to compile codes or submit Slurm jobs, this page is a good starting point.
More ›

Online documentation
Check out Compute Canada's technical documentation wiki, the primary source for information on Compute Canada resources and services.
More ›

Upcoming sessions
We host training webinars and workshops year-round to help you build skills in computational research. Check out our upcoming training events.
More ›

University of Manitoba

Thank you for your attention

Any question?

University of Manitoba

Estimating resources: CPUs

- ★ How to estimate the CPU resources?
  - ○ No direct answer: it depends on the code
  - ○ Serial code: 1 core [--ntasks=1 --mem=2500M]
  - ○ Threaded and OpenMP: no more than available cores on a node [--cpus-per-task=12]
  - ○ MPI jobs: can run across the nodes [--nodes=2 --ntasks-per-node=12 --mem=0].
- ★ Are threaded jobs very efficient?
  - ○ Depends on how the code is written
  - ○ Does not scale very well
  - ○ Run a benchmark and compare the performance and efficiency.
- ★ Are MPI jobs very efficient?
  - ○ Scale very well with the problem size
  - ○ Limited number of cores for small size: when using domain decomposition
  - ○ Run a benchmark and compare the efficiency.

Estimating resources: memory

- ★ How to estimate the memory for my job?
  - No direct answer: it depends on the code
  - Java applications require more memory in general
  - Hard to estimate the memory when running R, Python, Perl, …
- ★ To estimate the memory, run tests:
  - Interactive job, ssh to the node and run top -u $USER {-H}
  - Start smaller and increase the memory
  - Use whole memory of the node; seff <JOBID>; then adjust for similar jobs
  - MPI jobs can aggregate more memory when increasing the number of cores
- ★ What are the best practices for evaluation the memory:
  - Run tests and see how much memory is used for your jobs {seff; sacct}
  - **Do not oversubscribe the memory** since it will affect the usage and the waiting time: accounting group charged for resources reserved and not used properly.

University of Manitoba

Optimizing jobs: mem and CPU

- ★ How to estimate the run time for my job?
  - No direct answer: it depends on the job and the problem size
  - See if the code can use checkpoints
  - For linear problems: use a small set; then estimate the run time accordingly if you use more steps (extrapolate).
- ★ To estimate the time, run tests:
  - Over-estimate the time for the first tests and adjust for similar jobs and problem size.
- ★ What are the best practices for time used to run jobs?
  - Have a good estimation of the run time after multiple tests.
  - Analyse the time used for previous successful jobs.
  - Add a margin of 15 to 20 % of that time to be sure that the jobs will finish.
  - Do not overestimate the wall time since it will affect the start time: longer jobs have access to smaller partition on the cluster (the Alliance clusters).

How to pick a CPU partition on Grex?

Many jobs are submitted to skylake partition and asing for large memory: by over-subscribing the memory, many CPUs will stay idle [low usage of].

Partition	Nodes	Cores	Total	Memory	MEM/CPU
compute	312	12	3456	46 GB	3.8 GB
largemem	12	40	480	376 GB	9.4 GB
skylake	42	52	2184	96 GB	1.6 GB

Some tips for usage optimization:

- Run tests and check the memory usage {seff}
- Adjust the memory for similar jobs
- Submit with appropriate resources {no more}.

Output from: partition-list

```
PARTITION     CPUS(A/I/O/T)
compute*      2280/300/1280/3860
largemem      480/0/0/480
skylake       781/1455/0/2236
```

Partitions and memory:

compute: many nodes {312} and many CPUs {3456}
serial and MPI jobs with memory per CPU around 4 GB.

skylake: only 42 nodes but many CPUs {2184}
serial and MPI jobs with memory per CPU around 1.6 GB.

largemem: few nodes {12}, 480 CPUs
serial and MPI jobs with memory per CPU around 9 GB.

Skylake partition shows 781 allocated CPUs and 1455 idle CPUs. These CPUs are idle and can not run other job because all the memory was allocated to other jobs.