University of Manitoba

Digital Research Alliance of Canada

Running jobs on HPC cluster using SLURM: batch and interactive jobs

Ali Kerrache

HPC Specialist

UManitoba Autumn 2025 HPC and Cloud computing workshop - October 14–17, 2025

University of Manitoba

Outline

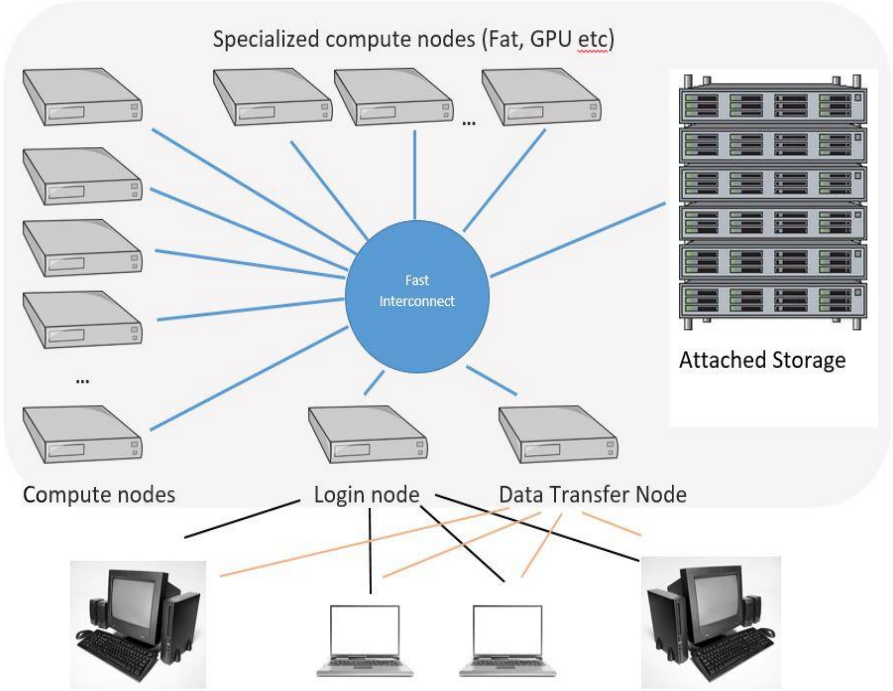
- Interactive work on the login nodes
- SLURM: a scheduler to run jobs on HPC cluster.
  - Interactive jobs via salloc
  - Batch jobs jobs via sbatch
  - What do you need to write and set a job script?
- Exercises:
  - Sleep test job; Serial Job, OpenMP job,
  - MPI Job, GPU job.
- Remarks and conclusions.

Interactive work on the login node

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 nor cpu intensive tests}
- Performance can suffer greatly from over-subscription

Recommendation: use the scheduler



University of Manitoba

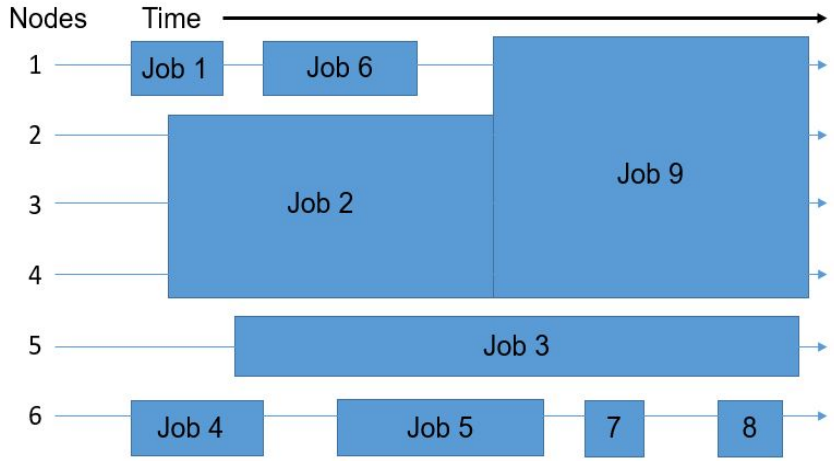
SLURM as a scheduler for HPC clusters

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 - sinfo - smail - sprio - squeue - sreport - srun - sshare - sstat - strigger - sview



University of Manitoba

Interactive and batch jobs

- For interactive test, submit interactive jobs: salloc [+options]
  - SLURM uses salloc to ask for interactive jobs [compute nodes] to test and debug jobs.
  - The jobs will run on compute nodes [CPUs, GPUs]
- Submitting batch jobs for production work: 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
- Interactive jobs:
  - salloc [+options]
  - Load modules, run tests, debug, … etc.
  - Run "exit" to exit from the node.
- Batch jobs:
  - Add resources, commands in your script
  - sbatch myscript.sh
  - monitor jobs

University of Manitoba

Slurm directives

Interactive jobs:

- salloc [+options]
- Load modules, run tests, debug, …

- - - account=def-sponsor
- - - nodes=1
- - - ntasks=1
- - - cpus-per-task=4
- - - mem=1000M
- - - mem-per-cpu=1000M
- - - time=1-12:30:00
- - - partition=genoa,skylake

Batch jobs:

- Add resources, commands, …
- sbatch [+options] myscript.sh

- #SBATCH - - account=def-sponsor
- #SBATCH - - nodes=1
- #SBATCH - - ntasks=1
- #SBATCH - - cpus-per-task=4
- #SBATCH - - mem=1000M
- #SBATCH - - mem-per-cpu=1000M
- #SBATCH - - time=1-12:30:00
- #SBATCH - - partition=genoa,skylake

University of Manitoba

Interactive jobs

```
[~@bison ]$  salloc --cpus-per-task=4 --mem-per-cpu=1000M --time=1:00:00
salloc: using account: def-sponsor
salloc: No partition specified? It is recommended to set one! Will guess
salloc: selected partitions: skylake
salloc: Pending job allocation 6170818
salloc: job 6170818 queued and waiting for resources
salloc: job 6170818 has been allocated resources
salloc: Granted job allocation 6170818
salloc: Nodes n373 are ready for job
[~@n373 ~]$ {load modules, run tests, … etc}
[~@n373 ~]$  exit
exit
salloc: Relinquishing job allocation 6170818
```

Equivalent SLURM: my-script.sh

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

load modules …
Run the code …
```

University of Manitoba

Job requirements

What do you need to set your job script?

- What type of program are you going to run?
  - Serial, Threaded [OpenMP], MPI based, Hybrid [MPI+OPenMP], GPU, …
- Prepare your input files: locally or transfer from your computer.
- Test your program:
  - Interactive job via salloc: access to a compute node
  - On login node if the test is not memory nor CPU intensive.
- Prepare a script “my-job-script.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 your jobs: sbatch, squeue, sacct, seff … etc

University of Manitoba

Scheduling jobs: exercises

Directory: jobs

- interactive: directory for interactive test via salloc.
- sleep-job: directory for a sleep job with defaults parameters.
- serial-job: directory for a serial job: 1 CPU.
- openmp-job: directory for a job with OpenMP support.
- mpi-job: directory for a job with MPI support.
- gpu-job: directory for a GPU job

- oom-kill: directory for a job killed with oom-kill event.
- time-out: directory for a job that times out.

- job-array-job: directory for an example with arrays.
- glost: directory for job farming with glost.

University of Manitoba

Concluding remarks

The key is to know what resources are available on a given HPC machine, and how to use them efficiently [adjust resources where needed]:

- ➔ It is up to the users to go through the documentation and run tests, …
- ➔ Monitor the jobs for efficiency [seff, sacct, reportseff, …]
- ➔ Know what partitions are there, and what are their limits: sinfo, …
- ➔ Know about the hardware (how many CPUs per node, how much memory per CPU available, ….) documentation for each cluster.
- ➔ Know if your code is efficient for a given set of resources: benchmarks
- ➔ Know time limits and estimate runtime of your jobs
  - ◆ comes after some trials and errors [with experience].
- ➔ Make sure your application obeys the SLURM resource limits

University of Manitoba

More readings

Grex:

- https://um-grex.github.io/grex-docs/start-guide/
- https://um-grex.github.io/grex-docs/running-jobs/
- https://um-grex.github.io/grex-docs/running-jobs/slurm-partitions/

Alliance clusters:

- https://docs.alliancecan.ca/wiki/Running_jobs
- https://docs.alliancecan.ca/wiki/Advanced_MPI_scheduling/en
- https://docs.alliancecan.ca/wiki/Using_GPUs_with_Slurm/en
- https://docs.alliancecan.ca/wiki/Using_node-local_storage

Thank you for your attention

Any question?

University of Manitoba

Digital Research Alliance of Canada

Additional Slides

Digital Research
Alliance of Canada

Examples of job scripts

University of Manitoba

Slurm most used commands

- ★ salloc: submit interactive jobs.
- ★ sbatch: submit batch jobs to compute nodes.
- ★ scontrol: list and/or change parameters for jobs, partitions, ...
- ★ scancel: cancel submitted jobs
- ★ sacct: reports about jobs
- ★ seff: reports resources used about a given job.
- ★ sinfo: check the nodes (idle, drain, down), …
- ★ sprio: check the priority
- ★ squeue: list the jobs on the queue
- ★ srun: used to run MPI jobs (mpiexec, mpirun still work)
- ★ sshare: check the recent usage and LevelFS

University of Manitoba

Slurm 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 the program}


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

Script: test-job.sh

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

Default parameters:

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

University of Manitoba

Slurm accounting groups

```
salloc --ntasks=1 --mem=4000M --account=def-prof1
```

➔ Submit Interactive job:

```
[~yak ]$  salloc --ntasks=1 --mem=4000M
salloc: error: ----------------------------------------
salloc: error: You have more than one account:
salloc: error:  - account # 1: use sbatch/salloc --account=def-prof1
salloc: error:  - account # 2: use sbatch/salloc --account=def-prof2
salloc: error: Please pick one 'sbatch/salloc --account=_VALUE_' option from the list above ^^^
salloc: error: Or set 'export SBATCH_ACCOUNT=_VALUE_' (or 'export
SALLOC_ACCOUNT=_VALUE_') to one of the above accounts
salloc: error: Job submit/allocate failed: Invalid account or account/partition combination specified
```

➔ 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

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=genoa	**GREX:** Partition name: skylake, largemem, genoa, 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 job

```
#!/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=genoa

# 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 [+options] myscript.sh
- squeue -u $USER; sq; sacct -j JOB_ID

More information:

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

University of Manitoba

Slurm script: OpenMP

```
#!/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:

- genoa: N up to 192
- skylake: N up to 52
- largemem: N up to 40

University of Manitoba

Slurm script: MPI job

```
#!/bin/bash
#SBATCH --account=def-someprof
#SBATCH --nodes=2-4
#SBATCH --ntasks=96
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=2-00:00:00
#SBATCH --partition=skylake
# Load appropriate modules:
module load arch/avx512  gcc/13.2.0  openmpi/4.1.6
lammps/2024-08-29p1
echo "Starting run at: `date`"
srun lmp -in  in.lammps
echo "Program finished with exit code $? at: `date`"
```

```
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=192
#SBATCH --mem=0
#SBATCH --partition=genoa
```

```
#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: MPI+OpenMP job

```
#!/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=skylake

# Load appropriate modules:
module load <dependencies> <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=genoa
```

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

Slurm script: GPU job

```
#!/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 [+options] myscript.sh
- squeue -u $USER

Partition:

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

University of Manitoba

Slurm script: 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=genoa
# 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=0-999** 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

University of Manitoba

Bundle many jobs: Job Arrays

- Files: in.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 < 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 ${SLURM_ARRAY_TASK_ID}
lmp < in.melt > log_lammps_array-${SLURM_ARRAY_TASK_ID}.txt
```

https://docs.alliancecan.ca/wiki/Job_arrays/en
https://um-grex.github.io/grex-docs/running-jobs/batch-jobs/#job-arrays

University of Manitoba

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=genoa

# Load appropriate modules + glost:
module load arch/avx512  gcc/13.2.0
openmpi/4.1.6 glost/0.3.1
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
–
–
job199
job200
```

University of Manitoba

Digital Research Alliance of Canada

Estimating resources

University of Manitoba

What resources to ask for?

- CPUs:
  - How many CPUs are required to run a particular job?
- Memory
  - How much memory is needed to run a particular job?
- Time
  - How to estimate the run time for run a particular job?
- GPUs
  - Could I run my job on a GPU?

University of Manitoba

Jobs optimization

Estimate the resources for your jobs:

- Number of CPUs: serial, OpenMP, MPI
- Memory: total memory or memory per core
- Run time

➔ Use interactive jobs

➔ Submit test jobs:

  ❖ Run a benchmark if needed {OpenMP; MPI jobs, Hybrid}

➔ Collect the stats about memory usage, wall time, .. etc.

➔ Adjust your scripts for similar jobs

University of Manitoba

Estimating the resources: CPU

- 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=X]
  - MPI jobs: can run across the nodes [--nodes=2 --ntasks-per-node=52 --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.

University of Manitoba

Estimating the 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

Estimating the resources: Wall Time

```
[~@bison ]$  seff 5080534
Job ID: 5080534
Cluster: grex
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Cores: 1
CPU Utilized: 01:28:33
CPU Efficiency: 99.87% of 01:28:40
core-walltime
Job Wall-clock time: 01:28:40
Memory Utilized: 274.48 MB
Memory Efficiency: 3.43% of 7.81 GB
```

- ➔ Job completed
- ➔ CPU Efficiency: 99.87%
- ➔ Wall time: 01:28:40
- ➔ Memory Utilized: 274.48 MB

Steps: 10000 (iterations)
Wall time: 1:30 to 2:00
Memory: 300 mb to 400 mb

Steps: 10000 x 10
Wall time: {1:30 to 2:00} x 10
Memory: 300 mb to 400 mb

University of Manitoba

Optimizing jobs: Memory 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 depending on the limits sets on the cluster: partition, GrpTRESRunMins, …

University of Manitoba

Digital Research Alliance of Canada

Monitor and get more information about jobs, partitions, … etc

University of Manitoba

Scheduling policies

- There is a policing engine, the Scheduler, in SLURM that will enforce priorities, allocations, and limits {per user, group, partitions, …}.
- Limits can be per User or Group, in the form of Maximum Walltime, Max number of jobs per User, … etc.

- Limits can be per Partition {for example max WallTime}.
- Limits can be per QoS policy (not used on Grex or CC)

- Priorities are set based on allocation values and recent usage and waiting time in the queue. Priorities is what determines which job can run first (if not blocked by a given limit).

University of Manitoba

Monitor and control jobs

```
squeue -u $USER [-t RUNNING] [-t PENDING]                    # list all current jobs.
squeue -p PartitionName [genoa, skylake, largemem]           # list all jobs in a partition.
sinfo --format="%P"                                          # 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

Memory and CPU efficiencies: seff

Output from seff command for a job {OpenMP} that asked for 24 CPUs and 187 GB of memory on cedar:

Job ID: 123456789
Cluster: cedar
User/Group: someuser/someuser
State: COMPLETED (exit code 0)
Nodes: 1
Cores per node: 24
CPU Utilized: 38-14:26:22
CPU Efficiency: 38.46% of 100-08:45:36 core-walltime
Job Wall-clock time: 4-04:21:54
Memory Utilized: 26.86 GB
Memory Efficiency: 14.37% of 187.00 GB

Successful job

Low CPU efficiency: 40 %
Better performance with 8 CPU

Used less memory: 15 %

billing=46,cpu=24,mem=187G,node=1

Optimization:
Better performance with 8 CPU
Memory: 4000 M per core [32 GB]

```
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=4000M
```

University of Manitoba

Memory and CPU efficiencies: reportseff

```
[~@yak ~]$  reportseff -u someuser -s COMPLETED
   JobID    State                Elapsed  TimeEff   CPUEff   MemEff
 6865022  COMPLETED             00:01:26     0.0%     9.9%     0.9%
 6866046  COMPLETED           1-17:11:47     8.2%     8.8%     3.4%
 6866067  COMPLETED             22:12:02     4.4%    11.0%     3.8%
 6866070  COMPLETED           1-18:06:54     8.4%    10.9%     3.8%
 6866074  COMPLETED           2-16:58:29    12.9%    10.9%     3.8%
 6866078  COMPLETED           3-14:49:23    17.2%    10.9%     2.7%
 6866082  COMPLETED           2-17:31:19    13.0%    10.9%     2.7%
 6868344  COMPLETED           1-14:49:19     7.7%     8.7%     4.3%
```

```
[~@yak ~]$ reportseff -u someuser -s COMPLETED
[~@yak ~]$ reportseff -u someuser --partition skylake -s COMPLETED
[~@yak ~]$ reportseff <JOB_ID>
[~@yak ~]$ reportseff --help
```

https://github.com/troycomi/reportseff

University of Manitoba

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 the low priority due to a recent usage.
- 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
  - [~@yak ~]$ scontrol show config | grep MaxArraySize
    MaxArraySize = 2000
- ReqNodeNotAvail, UnavailableNodes: n410: node not available

University of Manitoba

SLURM messages: Reason

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 the 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

[~@fir]$ cat /etc/slurm/slurm.conf | grep MaxArraySize
MaxArraySize=10000 (not the same on other clusters)
ReqNodeNotAvail, UnavailableNodes:fc931: 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

```
[~@narval2: ~]$ scontrol show res
ReservationName=MAINT20251021 StartTime=2025-10-21T08:00:00 EndTime=2025-10-28T08:00:00
Duration=7-00:00:00

Nodes=nc[10101-10156,31401-31433],ng[10101-10104,,20401-20404,20501-20504,20601-20604,30801-30811,,31
301-31305,31401-31402],nl[1001-11003,31001-31002,31101-31103,31201] NodeCnt=1442 CoreCnt=89760
Features=(null) PartitionName=(null) Flags=MAINT,IGNORE_JOBS,SPEC_NODES,ALL_NODES
   TRES=cpu=89760
   Users=root Groups=(null) Accounts=(null) Licenses=(null) State=INACTIVE BurstBuffer=(null) Watts=n/a
   MaxStartDelay=(null)
```

University of Manitoba

Information about a partition

```
[~@bison ~]$  sinfo -p largemem
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
largemem       up 14-00:00:0  5      mix n[328-331,333]
largemem       up 14-00:00:0  6  alloc n[326-327,334-337]
largemem       up 14-00:00:0  1   idle n332
```

```
[~@bison ~]$  scontrol show partition largemem --oneliner
PartitionName=largemem AllowGroups=ALL AllowAccounts=ALL AllowQos=normal,high
AllocNodes=aurochs,tatanka,bison,wisent,yak,n[001-316],g32[1-5],g338,g383,n[326-337],n[33
9-381] Default=NO QoS=N/A DefaultTime=03:00:00 DisableRootJobs=NO ExclusiveUser=NO
GraceTime=0 Hidden=NO MaxNodes=UNLIMITED MaxTime=14-00:00:00 MinNodes=0
LLN=NO MaxCPUsPerNode=UNLIMITED Nodes=n[326-337] PriorityJobFactor=0
PriorityTier=1 RootOnly=NO ReqResv=NO OverSubscribe=NO OverTimeLimit=NONE
PreemptMode=OFF State=UP TotalCPUs=480 TotalNodes=12 SelectTypeParameters=NONE
JobDefaults=(null) DefMemPerCPU=7000 MaxMemPerNode=UNLIMITED
TRESBillingWeights=CPU=2.0,Mem=0
```

University of Manitoba

Information about queued jobs

```
[~@bison ~]$ squeue

[~@bison ~]$ squeue -u $USER

[~@bison ~]$ sq

[~@bison ~]$ squeue -u <someuser>

[~@bison ~]$ squeue -t R

[~@bison ~]$ squeue -t PD

[~@bison ~]$ squeue -p genoa,skylake -t R

[~@bison ~]$ squeue -j <jobid>
```

Monitor queued jobs:

- Per user
- Job ID
- Per partition
- Running jobs
- Pending job
- Combine two or more from the above.
- .. etc.

University of Manitoba

Information about queued jobs: scontrol

```
[~@bison ~]$ scontrol show job 1234567 --oneliner
JobId=1234567 JobName=run-lmp-serial.sh UserId=someuser(3333333)
GroupId=someuser(3333333) MCS_label=N/A Priority=491351 Nice=0 Account=def-someprof
QOS=normal JobState=RUNNING Reason=None Dependency=(null) Requeue=0 Restarts=0
BatchFlag=1 Reboot=0 ExitCode=0:0 RunTime=01:23:18 TimeLimit=12:00:00 TimeMin=N/A
SubmitTime=2023-11-03T09:26:35 EligibleTime=2023-11-03T09:26:35
AccrueTime=2023-11-03T09:26:35 StartTime=2023-11-03T09:26:51 EndTime=2023-11-03T21:26:51
Deadline=N/A SuspendTime=None SecsPreSuspend=0 LastSchedEval=2023-11-03T09:26:51
Scheduler=Backfill Partition=compute AllocNode:Sid=yak:174565 ReqNodeList=(null)
ExcNodeList=(null) NodeList=n204 BatchHost=n204 NumNodes=1 NumCPUs=1 NumTasks=1
CPUs/Task=1 ReqB:S:C:T=0:0:*:* TRES=cpu=1,mem=4000M,node=1 Socks/Node=*
NtasksPerN:B:S:C=0:0:*:* CoreSpec=* MinCPUsNode=1 MinMemoryCPU=4000M
MinTmpDiskNode=0 Features=(null) DelayBoot=00:00:00 OverSubscribe=OK Contiguous=0
Licenses=(null) Network=(null) Command=/home/someuser/Workshop/Serial_Job/run-lmp-serial.sh
WorkDir=/home/someuser/Serial_Job StdErr=/home/someuser/Serial_Job/slurm-1234567.out
StdIn=/dev/null StdOut=/home/someuser/Serial_Job/slurm-1234567.out Power=
```

University of Manitoba

Jobs and nodes by partition

```
[~@bison ~]$  squeue -p skylake
[~@bison ~]$  squeue -p skylake -t PD
[~@bison ~]$  squeue -p skylake -t R

[~@biison ~]$  sinfo -p skylake
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
skylake    up 21-00:00:0  1  inval n349
skylake    up 21-00:00:0  3  down* n[352,359-360]
skylake    up 21-00:00:0  1  drain n375
skylake    up 21-00:00:0  26    mix n[339-342,346-347,350-351,356-358,366-374,376-381]
skylake    up 21-00:00:0  12  alloc n[343-345,348,353-355,361-365

[~@bison ~]$  sinfo -p skylake --state=down
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
skylake    up 21-00:00:0  3  down* n[352,359-360]
```

University of Manitoba

How to get most of the scheduler?

The key is to know what resources are available on a given HPC machine, and to adjust your requests accordingly.

- ➔ It is up to the users to go through the documentation and run tests, …
- ➔ Know what partitions are there, and what are their limits: sinfo, …
- ➔ Know about the hardware (how many CPUs per node, how much memory per CPU available, …. documentation for each cluster
- ➔ Know if your code is efficient for a given set of resources: benchmarks
- ➔ Know time limits and estimate runtime of your jobs
  - ◆ comes after some trials and errors, with experience
- ➔ Make sure your application obeys the SLURM resource limits

University of Manitoba

Custom script: partition-list

sinfo -se --format=" %10P %.10A %.12l %.6a %.17C %.8m [%.18G]"

```
PARTITION  NODES(A/I)    TIMELIMIT  AVAIL     CPUS(A/I/O/T)   MEMORY [              GRES]
skylake*       22/20  21-00:00:00     up  650/1534/52/2236   186000 [            (null)]
chrim            4/0  21-00:00:00     up      80/688/0/768   750000 [            (null)]
chrimlm          1/0  21-00:00:00     up      144/48/0/192  1500000 [            (null)]
genlm            3/0  21-00:00:00     up      19/557/0/576  1500000 [            (null)]
genoa            9/18 21-00:00:00     up   326/4858/0/5184   750000 [            (null)]
genoacpu-b       4/0   7-00:00:00     up      80/688/0/768   750000 [            (null)]
genoacpu-b       1/0   7-00:00:00     up      144/48/0/192  1500000 [            (null)]
genoacpu-b       1/4   7-00:00:00     up     160/680/0/840  1500000 [            (null)]
largemem        11/0   21-00:00:00    up    147/293/40/480   381500 [            (null)]
mcordcpu         1/4   21-00:00:00    up     160/680/0/840  1500000 [            (null)]
agro             0/2   21-00:00:00    up         0/48/0/48   248000 [    gpu:a30:2(S:0)]
agro-b           0/2   7-00:00:00     up         0/48/0/48   248000 [    gpu:a30:2(S:0)]
gpu              1/1   7-00:00:00     up        32/32/0/64   191000 [ gpu:v100:4(S:0-1)]
lgpu             0/1   3-00:00:00     up         0/64/0/64   381500 [ gpu:l40s:2(S:0-1)]
livi             0/1   21-00:00:00    up         0/48/0/48  1500000 [gpu:v100:16(S:0-1)]
livi-b           0/1   7-00:00:00     up         0/48/0/48  1500000 [gpu:v100:16(S:0-1)]
mcordgpu         0/2   21-00:00:00    up         0/64/0/64   495000 [    gpu:a30:4(S:0)]
mcordgpu-b       0/2   7-00:00:00     up         0/64/0/64   495000 [    gpu:a30:4(S:0)]
stamps           1/2   21-00:00:00    up         8/88/0/96   191000 [ gpu:v100:4(S:0-1)]
stamps-b         1/2   7-00:00:00     up         8/88/0/96   191000 [ gpu:v100:4(S:0-1)]
test             0/1     23:00:00     up         0/18/0/18   509000 [            (null)]
```