University of Manitoba

compute canada | calcul canada

Scheduling jobs on HPC clusters: How to optimize your jobs and get more from the resources available?

UofM-Autumn-Workshop 2021

Nov 1st-2nd, 2021

University of Manitoba

Ali Kerrache

University of Manitoba

Outline

- ★ Scheduler: SLURM
- ★ SLURM directives.
- ★ SLURM environment variables.
- ★ Submitting jobs: partitions on Grex
- ★ SLURM examples:
  - ○ Serial, OpenMP, MPI, GPU
  - ○ Job arrays, GLOST
- ★ Optimize jobs and get more from HPC resources.
- ★ Monitoring and controlling jobs.

University of Manitoba

Working on HPC cluster

- ★ Get an account; connect to a cluster: ssh, Putty, MobaXterm, X2Go.
- ★ Prepare your input files: use an editor to change your files.
- ★ Upload or Transfer your files (if prepared elsewhere):
  - scp; stfp; WinScp; FileZilla; Globus; …
- ★ Compile your own code (ask for support if needed):
  - Intel or GNU compilers; Libraries; Tools; …
- ★ Use existing modules and programs;
- ★ Or ask support team to install new ones if needed
- ★ Read about the scheduler used and its directives: SLURM
- ★ Prepare and Test your scripts and programs: salloc, seff, sacct

University of Manitoba

Why do we need a scheduler?

★ What happened if there is no scheduler?

- Over usage for some resources.
- Abuse from users: others will never get a chance to run a job.
- Over-subscribing the resources.
- Failure of compute nodes.
- Low efficiency: some nodes can be over-loaded while the rest stay idle.

★ High Performance Computing:

- Multiple users and different allocations: RAC; Default, …
- Multiple applications.
- Multiple jobs: each job has a particular set of resources.

University of Manitoba

What is a scheduler used for?

- ★ Put the jobs on the queue: one or more queues.
- ★ Assign a priority to each job:
  - ○ Resources asked for: mem, ntasks, nodes, wall time, …
  - ○ RAC or Default allocation.
  - ○ Dynamically changed priorities based on recent usage.
- ★ Assign a status to a job: Q [queue], R [running], H [hold], ...
- ★ Run the job when the resources are available.
- ★ Report some stats about the jobs: mem, run time, ...
- ★ Remove the job from the queue when it is done
  - ○ or exceeded the resources: wall time, memory.

University of Manitoba

Running batch jobs on a cluster

- When you connect you get interactive session on a login node:
  - Resources there are limited: used for basic operations
    - editing files, compiling codes, download or transfer data, submit and monitor jobs, run short tests {no memory intensive test}
  - Performance can suffer greatly from oversubscription
- 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 <some options> myscript.sh
- For interactive work, submit interactive jobs: salloc
  - SLURM uses salloc for interactive jobs
  - The jobs will run on dedicated compute nodes

University of Manitoba

Job requirements

What do you need to know {or do} before submitting jobs?

- Is the program available? If not, install it or ask support for help.
- What type of program are you using?
  - 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 login node if the test is not memory nor CPU intensive.
- Prepare a script "my-script.sh" with the all requirements:
  - Memory, Number of cores, Nodes, Wall time, modules, **partition**, **accounting group**, command line to run the code, … etc.
- Submit the job and monitor it: sbatch, squeue, sacct, seff, … [SLURM]

University of Manitoba

SLURM, a scheduler for HPC

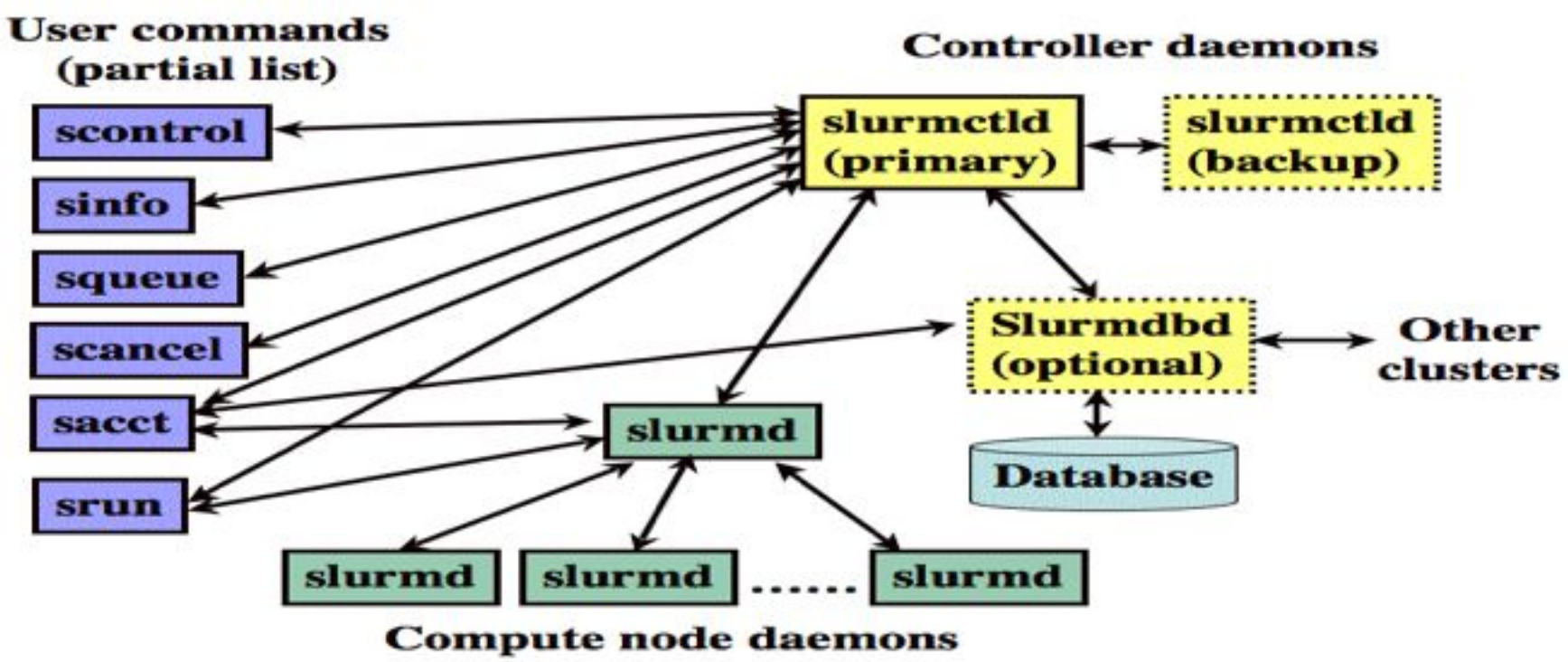
SLURM: Simple Linux Utility for Resource Management

- ★ Free and open-source job scheduler for Linux and Unix-like kernels, used by many of the world's super-computers and computer clusters.
- ★ https://slurm.schedmd.com/overview.html

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

University of Manitoba

SLURM, a scheduler for HPC



University of Manitoba

Main SLURM commands

- ★ salloc: submit interactive jobs.
- ★ sbatch: submit jobs to compute nodes.
- ★ scontrol: list and/or change parameters for jobs.
- ★ 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

Main SLURM directives

SLURM option	Description
#SBATCH --account=def-someuser	Use the accounting group for jobs.
#SBATCH --time=0-3:05:00	Wall time in the format: DD-HH:MM:00
#SBATCH --ntasks=4	Request 4 tasks for MPI job
#SBATCH --ntasks-per-node=4	Request 4 tasks per-node for MPI job
#SBATCH --nodes=2	Request 2 nodes
#SBATCH --mem=4000M	Memory of 4GB for the job
#SBATCH --mem-per-cpu=2000M	Memory of 2GB per CPU
#SBATCH --cpus-per-task=4	Number of threads (OpenMP)
#SBATCH --job-name="JOB_NAME"	Job name.
#SBATCH --output=job_output.txt	Standard output (default: slurm-JOBID.out).
#SBATCH --partition=compute	Partition Name: compute, skylake, largemem
#SBATCH --array=0-120%10	Job array

University of Manitoba

SLURM environment variables

Memory, number of CPU cores, threads often set in Input. It should match the resource request from SLURM:

- **srun** sets correct cpu cores and nodelists automatically for MPI jobs. It also takes care of the proper process placement.
- for SMP jobs, use $SLURM_CPUS_PER_TASK variable to set number of threads.
  - export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
- for jobs using MKL or other threaded BLAS/LAPACK, set MKL_NUM_THREADS either to 1 or to $SLURM_CPUS_PER_TASK, depending on what you are doing.
- It is a good practice to set memory request 10-15% smaller in the input than in the SLURM request, to allow for file buffers, etc.

University of Manitoba

SLURM environment variables

Environment variables	Description
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
SLURM_MEM_PER_CPU	Memory allocated per CPU
SLURM_JOB_NODELIST	List of nodes allocated for a Job
SLURM_JOB_CPUS_PER_NODE	Number of CPUs allocated per Node
SLURM_JOB_PARTITION	List of Partition(s) that the job is in.
SLURM_JOB_ACCOUNT	Account under which this job is run.

University of Manitoba

Job scheduling policy

- There is a policing engine, the Scheduler, in SLURM that will enforce priorities, allocations, and limits.
- 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
- 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 limit).

University of Manitoba

Submitting jobs with SLURM

Submit Interactive jobs:

```
salloc {options}
salloc --ntasks=1 --mem=4000M --time=1:00:00 --account=def-someuser
salloc --ntasks-per-node=4 --mem-per-cpu=4000M --time=1:00:00 --x11
salloc --ntasks=1 --cpus-per-task=4 --mem-per-cpu=4000M
```

Submit batch job to compute nodes:

```
sbatch my-slurm-scritpt.sh
sbatch {some options} my-slurm-scritpt.sh
sbatch --array=0-100%5 --partition=compute my-slurm-array-scritpt.sh
```

SLURM directives:

- ★ Introduced in the script via #SBATCH --{directive}={value}
- ★ Or in the command line via --{directive}={value}

University of Manitoba

Interactive job: salloc

```
[kerrache@tatanka ~]$  salloc --account=def-kerrache --nodes=1 --ntasks-per-node=8
--mem-per-cpu=1000M --time=1:00:00 --partition=compute --x11
salloc: using account: def-kerrache
salloc: partition selected:compute
salloc: Granted job allocation 4783413
salloc: Waiting for resource configuration
salloc: Nodes c318 are ready for job
[kerrache@c318 ~]$  sq
             JOBID     PARTITION     PRIORITY       USER     ACCOUNT       NAME  ST  TIME_LEFT
NODES CPUS MIN_MEM NODELIST [REASON]
             4783413   compute 0.0002922439 kerrache def-kerrache       sh   R       56:00       1      8
1000M c318 [None]
[kerrache@c318 ~]$  exit
exit
salloc: Relinquishing job allocation 4783413
```

University of Manitoba

Job script template

```
#!/bin/bash
#SBATCH --account=def-someuser

# Add the resources and other options

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

# Load appropriate modules.
# command line to run your program.

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

File name: my-script.sh

Parameters to adjust:

- Wall time
- Number of tasks, cpus
- Memory, … etc.

Load modules, set a path
Command to run your code

Submit/Monitor the job using:

```
sbatch <+options> my-script.sh
sq
squeue -u $USER
```

University of Manitoba

Accounting groups

```
#SBATCH --account=def-someuser
```

```
sbatch --account=def-someuser script.sh
```

```
[kerrache@bison ~]$  sshare -U --user kerrache
             Account       User  RawShares  NormShares    RawUsage  EffectvUsage  FairShare
-------------------- ---------- ---------- ----------- ----------- ------------- ----------
def-kerrache         kerrache           10    1.000000    36689594      1.000000   0.251012
def-kerrache-ab      kerrache            1    1.000000       46910      1.000000   0.374494
fmq-002-aa           kerrache            1    1.000000           0      0.000000   0.201417
fmq-002-ab           kerrache            1    1.000000           0      0.000000   0.201417
```

```
[kerrache@tatanka ~]$  sshare -U --user $USER --format=Account
             Account
--------------------
def-kerrache
def-kerrache-ab
fmq-002-aa
fmq-002-ab
```

Only def-* accounts are used on Grex

A user may have more than one def-* accounts:

- ★ Working with 2 sponsors for example
- ★ choose which one to use

University of Manitoba

Accounting groups

```
[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-kerrache-ab, def-kerrache, def-training-wa,
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:

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

How to use partitions on Grex?

Grex is a very heterogeneous system now:

- ★ Old compute nodes [12 cores, 48 GB RAM] **--partition=compute**
- ★ New compute with more memory [40 cores, 384 GB RAM] **--partition=largemem**
- ★ Newest compute with less memory [52 cores, 96GB RAM] **--partition=skylake**
- ★ New GPU partition [32 cores, 192GB RAM, 4x V100 GPUs] **--partition=gpu**
- ★ Contributed GPUs: **--partition=stamps-b** **or** **--partition=livi-b**

Unlike ComputeCanada, partition selection is manual, except that by default, short jobs would go to **compute** and longer to **skylake**.

Use "scontrol show partition <partition name>" for more information.

University of Manitoba

How to use partitions on Grex?

Partition	Nodes [CPUs/GPUs]	Cores per node	Cores	Memory	Memory per core	Max Wall Time
compute	316	12	3456	46 GB	3900 M	21 days
largemem	12	40	480	380 GB	8000 M	14 days
skylake	42	52	2184	87 GB	1500 M	21 days
gpu	2	32	64	187 GB	6000 M	3 days
stamps; -b	3	32	96	187 GB	*	21 days / 7 days
livi, -b	1	48	48	1.5 TB	*	21 days / 7 days

```
#SBATCH --partition=compute
```

```
sbatch --partition=compute my-script.sh
```

University of Manitoba

Some examples

- Serial jobs
- OpenMP {or threaded} jobs
- MPI jobs
- GPU jobs
- Gaussian
- ORCA
- ANSYS
- Run multiple jobs:
  - Job arrays
  - Glost

University of Manitoba

Job script for serial programs

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --mem=2500M
#SBATCH --time=1-00:30
#SBATCH --partition=compute

echo "Starting run at: `date`"

module load python

python my-python-program.py

echo “Program finished at: `date`"
```

```
#SBATCH --mem=256M
#SBATCH --time=3-00:00
#SBATCH --partition=compute
```

```
#SBATCH --mem=2500M
#SBATCH --time=1-00:30
#SBATCH --partition=compute
```

```
#SBATCH --mem=1200M
#SBATCH --time=7-00:00
#SBATCH --partition=skylake
```

```
#SBATCH --mem=800M
#SBATCH --time=3-00:00
#SBATCH --partition=largemem
```

Job script for OpenMP programs

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1500M
#SBATCH --time=3-00:00
#SBATCH --partition=compute

echo "Starting run at: `date`"

export OMP_NUM_THREADS=
$SLURM_CPUS_PER_TASK

echo “Program finished at: `date`"
```

```
--cpus-per-task=N --mem-per-cpu=X
--cpus-per-task=N --mem=Y
```

```
#SBATCH --cpus-per-task=4
#SBATCH --mem-per-cpu=1500M
#SBATCH --time=3-00:00
#SBATCH --partition=compute
```

```
#SBATCH --cpus-per-task=12
#SBATCH --mem=0
#SBATCH --partition=compute
```

```
#SBATCH --cpus-per-task=52
#SBATCH --mem=0
#SBATCH --partition=skylake
```

University of Manitoba

Job script for MPI programs

```
#!/bin/bash
#SBATCH --ntasks=8
#SBATCH --mem-per-cpu=1500M
#SBATCH --time=3-00:00
#SBATCH --partition=compute

echo "Starting run at: `date`"

ml intel/2019.5  ompi/3.1.4 lammps/29Sep21

srun lmp_grex < in.lammps > lammps-output.txt

echo “Program finished at: `date`"
```

```
#SBATCH --ntasks-per-node=8
#SBATCH --mem-per-cpu=1500M
#SBATCH --partition=compute
```

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

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

```
#SBATCH --ntasks=160
#SBATCH --mem-per-cpu=1200M
#SBATCH --partition=skylake
```

Job script for GPU programs

```
#!/bin/bash
#SBATCH --gpus=1
#SBATCH --partition=stamps-b
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=6
#SBATCH --mem-per-cpu=6000M
#SBATCH --time=0-12:00

module load gcc/4.8 cuda/10.2
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK


guppy_basecaller -x auto --gpu_runners_per_device
6 -i Fast5 -s GuppyFast5 -c
dna_r9.4.1_450bps_hac.cfg
```

```
#SBATCH --gpus=1
#SBATCH --partition=stamps-b
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=6
#SBATCH --mem-per-cpu=6000M
```

GPU partitions:

- ★ gpu: 2 nodes; 4 V100-32 GB; 32 cores; 187 GB; 3 days
- ★ spamps; -b: 3 nodes; 4 V100-16 GB, 32 cores; 187 GB, 7 days.
- ★ livi; -b: 1 nodes, 16 V100-32 GB; 48 cores; 7 days.

University of Manitoba

Example for Gaussian

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --mem=12000M
#SBATCH --time=3-00:00:00
#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`"
```

Gaussian:

- Can not run across the nodes
- Do not take automatically the resources from the script.

Adjust the input file:

%nprocshared=8

%mem=10000M

Submit and monitor the job:

- sbatch myscript.sh
- squeue -u $USER

University of Manitoba

Example for ORCA

```
#!/bin/bash
#SBATCH --account=def-somegroup
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=8
#SBATCH --mem-per-cpu=1200M
#SBATCH --time=3-00:00:00
#SBATCH --partition=compute

# Load appropriate modules:
module load gcc/5.2  ompi/3.1.4 orca/4.2.1
echo "Starting run at: `date`"
`which orca` CoPcPyrFreq.inp
echo "Program finished with exit code $? at: `date`"
```

ORCA:

- Can run across the nodes
- Do not take automatically all the resources from the script.

Adjust the input file:

```
%maxcore 1000
%pal nprocs 32 end
```

```
#SBATCH --ntasks=32
#SBATCH --mem-per-cpu=1200M
```

University of Manitoba

SLURM: machinefile

Some programs require a machinefile or list of nodes:

ANSYS-CFX; ANSYS-FLUENT; HP-MPI; PDSH; GAUSSIAN; CHARM; STAR-CCM+; GNU-Parallel

Script: [~]$ which slurm_hl2hl.py

/opt/westgrid/bin/slurm_hl2hl.py

Usage :

/opt/westgrid/bin/slurm_hl2hl.py --format (ANSYS-CFX | ANSYS-FLUENT | HP-MPI | PDSH | GAUSSIAN | CHARM | STAR-CCM+ | MPIHOSTLIST | GNU-Parallel)

Example for ANSYS: slurm_hl2hl.py --format ANSYS-FLUENT> machinefile

University of Manitoba

Example for ANSYS

```
#!/bin/bash
#SBATCH --account=def-someuser
#SBATCH --time=0-06:00:00
#SBATCH --nodes=2
#SBATCH --cpus-per-task=12
#SBATCH --ntasks-per-node=1
#SBATCH --mem=0
module load uofm/cfx/21.1
slurm_hl2hl.py --format ANSYS-FLUENT > machinefile
NCORE=$((SLURM_NTASKS * SLURM_CPUS_PER_TASK))
fluent 3d -t $NCORE -cnf=machinefile -mpi=intel -affinity=0 -g -i fluent_3.jou
```

University of Manitoba

Job arrays example

SLURM has support for automatically running a job script on multiple data sets.

- You have regularly named, independent datasets (test1, test2, test3, …, test999) to process with a single software code
- Instead of making and submitting 999 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 999 jobs with the index added to JobID (12345_1, 12345_2, … , 12345_999)
- You can use usual SLURM commands (scancel, scontrol, squeue) on either entire array or on its individual elements

Job array: multiple serial jobs

```
#!/bin/bash
#SBATCH --ntasks=1
#SBATCH --mem=2500M
#SBATCH --time=1-00:30
#SBATCH --partition=compute

echo "Starting run at: `date`"

 # Load modules here:

./my_code test1

echo “Program finished at: `date`"
```

```
#SBATCH --ntasks=1
#SBATCH --mem=2500M
#SBATCH --time=1-00:30
#SBATCH --partition=compute
#SBATCH --array=1-999
```

```
#SBATCH --array=1-999%20
```

```
sbatch my-script.sh
```

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

```
sbatch --array=1-999 my-script.sh
```

University of Manitoba

Glost

```
#!/bin/bash
#SBATCH --ntasks=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --time=2-00:00
#SBATCH --partition=compute

echo "Starting run at: `date`"

 # Load modules here + glost

module load intel ompi glost
srun glost_launch list-tasks.txt

echo “Program finished at: `date`"
```

GLOST can be used in the following situations:

- large number of serial jobs with comparative run time,
- large number of short serial jobs,
- serial jobs with different parameters (parameter sweep).

- The number of jobs {list-tasks.txt} should be a multiple of ntasks to not waste resources.
- It uses a cyclic distribution of jobs

```
./my-program input-1
./my-program input-2
--
./my-program input-19
./my-program input-20
```

https://docs.computecanada.ca/wiki/GLOST

University of Manitoba

Job dependencies

SLURM provides job dependencies to build pipelines when subsequent jobs run depending on the result of the previous jobs

- A job or jobs is submitted first, their JobIDs are known
- Then dependent jobs can be submitted after as follows:

--dependency=afterok:jobid1

--dependency=afterallok:jobid1:jobid2:jobid3:jobid4

University of Manitoba

Monitor your jobs

```
squeue -u $USER [-t RUNNING] [-t PENDING]                    # list all current jobs.
squeue -p PartitionName                                      # 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 --states=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

SLURM message: squeue

SQUEUE_FORMAT=%.15i %.8u %.12a %.14j %.3t %.10L %.5D %.4C %.10b %.7m %N (%r)

JOBID: Job ID
USER: User name
ACCOUNT: Accounting group
NAME: Name of the script
ST: R (running), PD (pending), H (hold), ST (stopped), CG (completing)
TIME_LEFT: time left till the end of the wall time asked for.
NODES: How many nodes are uses?
CPUS: How many cores used?
GRES: Valid only when using GPUs.
MIN_MEM: Memory asked for.
NODELIST: assigned nodes (only for running jobs)
(REASON): None or other reasons

University of Manitoba

SLURM message: 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 excedded the maximum size of array jobs
[~@cedar]$ cat /etc/slurm/slurm.conf | grep MaxArraySize
MaxArraySize=10000 (not the same on other clusters)
ReqNodeNotAvail, UnavailableNodes:cdr931: node not available

University of Manitoba

SLURM: FAQ

- ★ *sbatch: error: Batch job submission failed: Socket timed out on send/recv operation*
- ★ Why are my jobs taking so long to start?
- ★ Why do my jobs show "Nodes required for job are DOWN, DRAINED or RESERVED for jobs in higher priority partitions"?
- ★ How accurate is START_TIME in squeue output?

https://docs.computecanada.ca/wiki/Frequently_Asked_Questions#sbatch:_error:_Batch_job_submission_failed:_Socket_timed_out_on_send.2Frecv_operation
https://slurm.schedmd.com/faq.html

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}

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

➔ Adjust your scripts for similar jobs

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.

University of Manitoba

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

Estimating resources: run time

- ★ 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 (Compute Canada clusters).

University of Manitoba

Memory optimization: MPI job

```
#SBATCH --ntasks=48
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=0-3:30:00
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=48
#SBATCH --mem-per-cpu=256M
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=8
#SBATCH --mem-per-cpu=1000M
#SBATCH --partition=compute
```

```
#SBATCH --ntasks=8
#SBATCH --mem-per-cpu=1000M
```

```
[kerrache@bison MPI]$  seff 4783417
Job ID: 4783417
Cluster: grex
User/Group: kerrache/kerrache
State: COMPLETED (exit code 0)
Nodes: 5
Cores per node: 9
CPU Utilized: 11:33:41
CPU Efficiency: 98.53% of 11:44:00 core-walltime
Job Wall-clock time: 00:14:40
Memory Utilized: 5.66 GB (estimated maximum)
Memory Efficiency: 3.02% of 187.50 GB (3.91 GB/core)
```

- ★ Run a test job
- ★ Use “seff” to estimate the memory, time, efficiency.
- ★ Adjust the memory for similar cases.

University of Manitoba

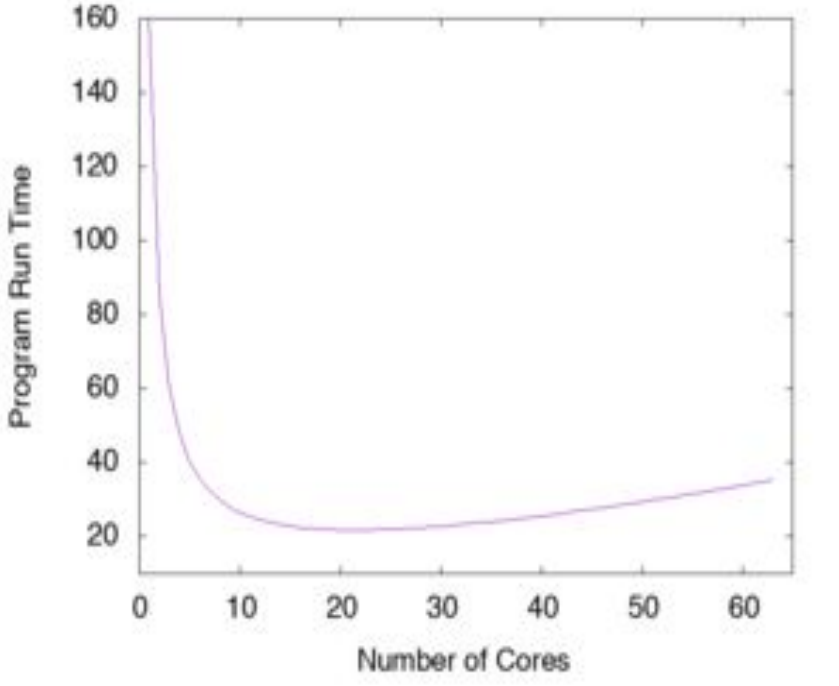
Number of CPUs for OpenMP

OpenMP jobs:

- ★ no more than available cores on a node:
  --cpus-per-task=X --partition=compute [X up to 12]
  --cpus-per-task=X --partition=skylake [X and 52]
- ★ Efficiency: are OpenMP jobs very efficient?
  - Depends on how the code is written
  - Does not scale very well

Benchmark:

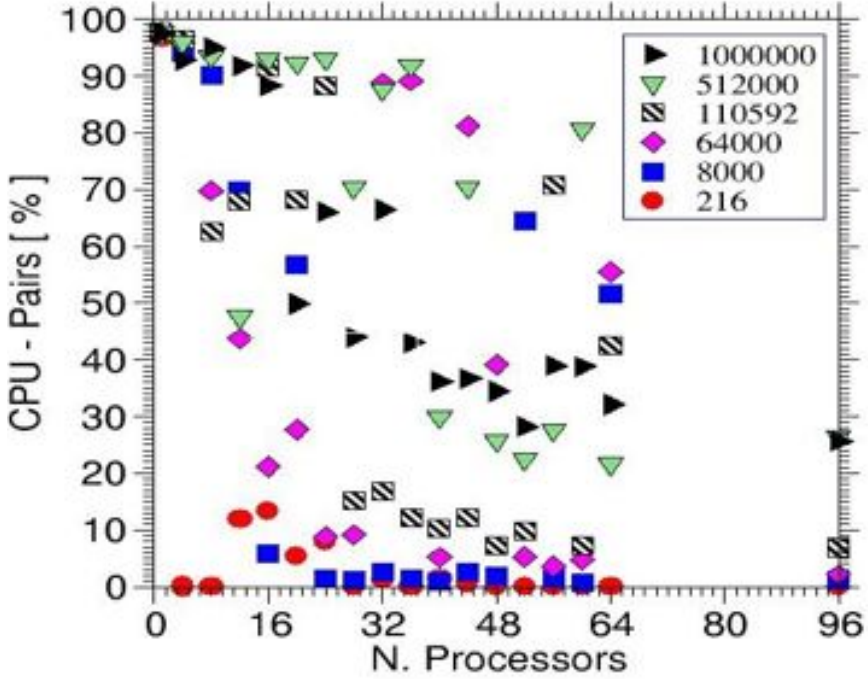
- ★ Run the same job with different number of threads
- ★ Use “seff” command to estimate the efficiency
- ★ Choose a combination that gives a higher efficiency



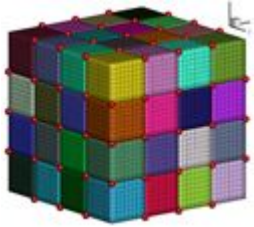
https://docs.computecanada.ca/wiki/Scalability/en

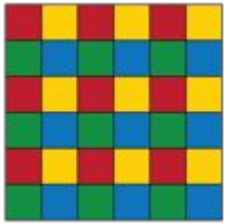
University of Manitoba

Number of CPUs for MPI jobs



Domain decomposition





- ★ Size, shape of the system.
- ★ Number of processors.
- ★ size of the small units.
- ★ correlation between the communications and the number of small units.
- ★ Reduce the number of cells to reduce communications.

MPI jobs by core versus by node

Cedar:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=48
#SBATCH --mem=0
```

```
#SBATCH --ntasks=96
#SBATCH --mem-per-cpu=4000M
```

Graham:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=32
#SBATCH --mem=0
```

```
#SBATCH --ntasks=64
#SBATCH --mem-per-cpu=4000M
```

Beluga:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=40
#SBATCH --mem=0
```

```
#SBATCH --ntasks=80
#SBATCH --mem-per-cpu=4000M
```

by node

by core

MPI jobs by core versus by node

compute:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=12
#SBATCH --mem=0
```

```
#SBATCH --ntasks=24
#SBATCH --mem-per-cpu=4000M
```

skylake:

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

```
#SBATCH --ntasks=104
#SBATCH --mem-per-cpu=1200M
```

largemem:

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=40
#SBATCH --mem=0
```

```
#SBATCH --ntasks=80
#SBATCH --mem-per-cpu=8000M
```

by node

by core

University of Manitoba

Useful links

SLURM website: https://slurm.schedmd.com/documentation.html

ComputeCanada documentation on running jobs:
https://docs.computecanada.ca/wiki/Running_jobs

Advanced MPI scheduling:
https://docs.computecanada.ca/wiki/Advanced_MPI_scheduling#Whole_nodes

Grex documentation on running jobs:
https://monitor.hpc.umanitoba.ca/doc/docs/grex/running/
https://um-grex.github.io/grex-docs/docs/grex/running/

Westgrid training materials:
https://westgrid.github.io/trainingMaterials/

Thank you for your attention

Any question?