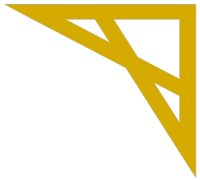
University of Manitoba



Digital Research Alliance of Canada

Running jobs efficiently on HPC clusters:

All you should know to get more from the scheduler and the available resources on the clusters

UofM-Spring-Workshop 2023

May 17th-19th, 2023

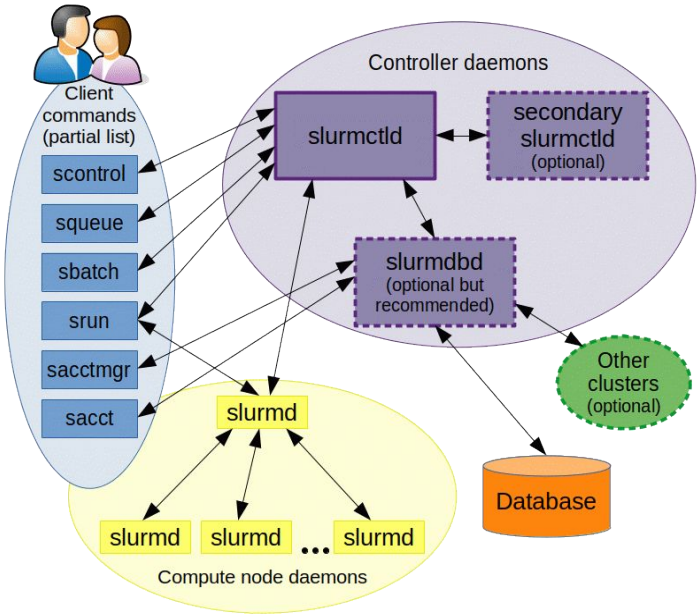
Ali Kerrache

HPC Analyst

University of Manitoba

Outline

- Most used commands in SLURM:
  - salloc, sbatch, seff, ….
- Available resources:
  - CPUs; GPUs; Memory, …
  - Partitions on Grex
- Optimization of the resources:
  - Memory efficiency
  - CPU efficiency
  - Scalability: OpenMP and MPI jobs
- Bundle many jobs into one:
  - Job Arrays; GLOST; GNU parallel



University of Manitoba

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

Information about the cluster

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

```
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 about nodes}
sinfo --p largemem      {shows more detailed information about the partition}
```

- 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

Available resources: Grex

Partition	Nodes [CPUs/GPUs]	Cores	Total	Memory	Wall Time
compute[1]	312	12	3456	46 GB	21 days
largemem	12	40	480	376 GB	14 days
skylake	42	52	2184	96 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

University of Manitoba

List of partitions

```
[~@bison ~]$ partition-list
[~@bison ~]$ sinfo -s --format=" %10P %10A %.10l %.12L %.6a %.18C %.8m"
```

PARTITION	NODES(A/I)	TIMELIMIT	DEFAULTTIME	AVAIL	CPUS(A/I/O/T)	MEMORY
compute*	206/0	21-00:00:0	3:00:00	up	1715/757/1328/3800	31000+
gpu	0/1	3-00:00:00	3:00:00	up	0/32/32/64	191000
largemem	12/0	14-00:00:0	3:00:00	up	400/80/0/480	381500
skylake	34/7	21-00:00:0	3:00:00	up	1460/672/104/2236	87000
test	0/1	12:00:00	3:00:00	up	0/18/0/18	509000
stamps	0/3	21-00:00:0	3:00:00	up	0/96/0/96	191000
stamps-b	0/3	7-00:00:00	3:00:00	up	0/96/0/96	191000
livi	0/1	21-00:00:0	3:00:00	up	0/48/0/48	1501000
livi-b	0/1	7-00:00:00	3:00:00	up	0/48/0/48	1501000
agro	0/2	21-00:00:0	3:00:00	up	0/48/0/48	248000
agro-b	0/2	7-00:00:00	3:00:00	up	0/48/0/48	248000

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

Job requirements

What do you need to optimize your job script?

- 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 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 the job and monitor it: sbatch, squeue, sacct, seff … etc

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

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
- 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

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

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).

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

Program with steps: 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 500 mb

Steps: 10000 x 10
Wall time: {1:30 to 2:00} x 10
Memory: 300 mb to 500 mb

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}.

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.

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
```

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.

University of Manitoba

Bundle jobs with job arrays

```
#!/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

University of Manitoba

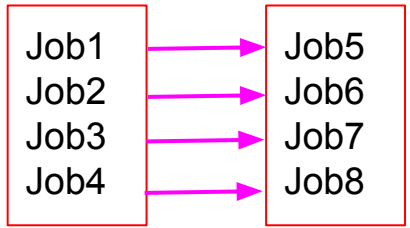
Bundle jobs with GLOST

```
#!/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



Total time divided by number of commands in the list

How to get most of the scheduler?

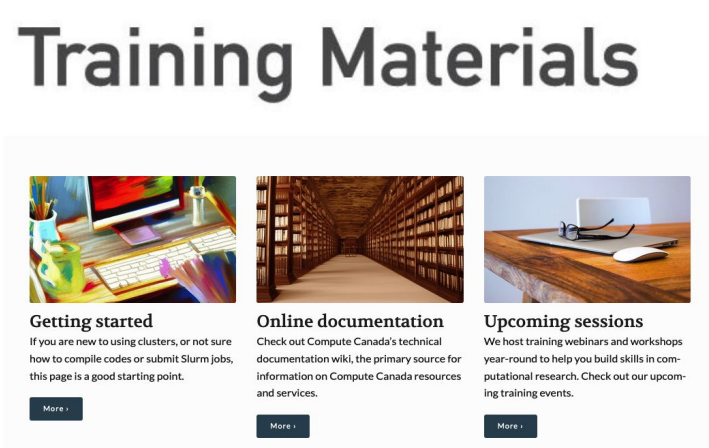
The key is to know what resources are available on a given HPC machine, and to adjust your requests accordingly.

- ★ 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

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/
- X2Go: https://wiki.x2go.org/doku.php
- Grex: https://um-grex.github.io/grex-docs/



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

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

University of Manitoba

Thank you for your attention

Any question?

University of Manitoba

Slurm

Examples

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:

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

University of Manitoba

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
```

```
#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

SLURM: simple 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 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 <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: MPI jobs

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

SLURM script: OpenMP+MPI jobs

```
#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>

University of Manitoba

Job array

```
#!/bin/bash
#SBATCH --account=def-kerrache
#SBATCH --ntasks=1
#SBATCH --mem-per-cpu=4000M
#SBATCH --time=0-8:00
#SBATCH --partition=compute
#SBATCH --array=0-9%2

echo "Starting run at: `date`"
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21

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

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

- Files: n.melt-0.txt, ….
  In.melt-9.txt
- Job 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

Job array: another example

- 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
```

University of Manitoba

MD

Performance

University of Manitoba

Demonstration: MD simulation

- Serial job
- MPI job:
  - 4; 8; 16; 32; 48; 64; 96
- Job array: parameter sweep
  - data in one directory
  - data in multiple directories
- Estimation of wall wall time
- Memory efficiency
- CPU efficiency

```
module load intel/2019.5  ompi/3.1.4 lammps/29Sep21
srun lmp_grex < in.melt > log_lammps_output.txt
```

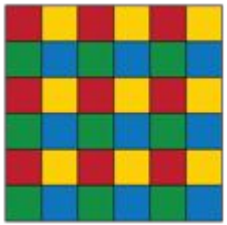
in.melt

```
# 3d Lennard-Jones melt
units           lj
atom_style      atomic
lattice         fcc 0.8442
region          box block 0 50 0 50 0 50
create_box      1 box
create_atoms    1 box
mass            1 1.0
velocity        all create 3.0 87287
pair_style      lj/cut 2.5
pair_coeff      1 1 1.0 1.0 2.5
neighbor        0.3 bin
neigh_modify    every 20 delay 0 check no
fix             1 all nve
thermo          250
run             10000
write_data  config.end_melt
```

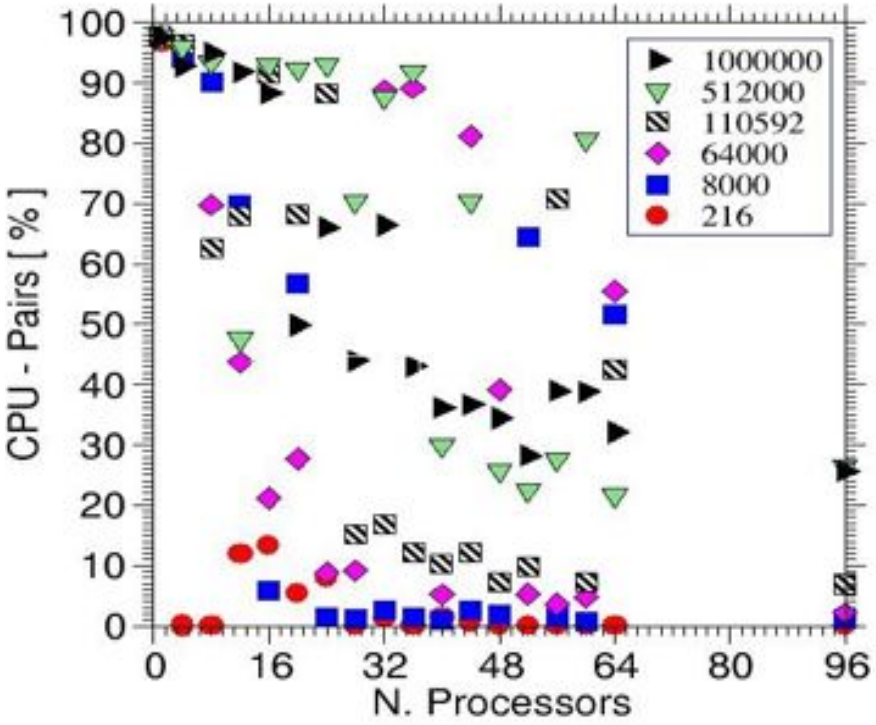
University of Manitoba

Performance tests: MD

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.



University of Manitoba

Time breakdown

Loop time of 5316.35 on 1 procs for 10000 steps with 500000 atoms

Performance: 812.587 tau/day, 1.881 timesteps/s
99.8% CPU use with 1 MPI tasks x no OpenMP threads

MPI task timing breakdown:

Section	min time	avg time	max time	%varavg	%total
Pair	4617.5	4617.5	4617.5	0.0	86.86
Neigh	517.75	517.75	517.75	0.0	9.74
Comm	35.556	35.556	35.556	0.0	0.67
Output	0.11397	0.11397	0.11397	0.0	0.00
Modify	119.55	119.55	119.55	0.0	2.25
Other		25.83			0.49

University of Manitoba

MPI job

CPUs	CPU Efficiency	CPU time	Run time [s]	Performance tau/day	Pair Interactions	Communication time [%]
1	99.87%	5317	5317	812.587	86.86	-
4	99.76%	6200	1550	2787	84.50	3.13
8	99.09%	6312	789	5479	78.13	10.77
16	99.21%	7360	460	9388	67.74	21.35
32	98.32%	5984	187	23136	76.97	10.32
48	97.56%	5904	123	35220	74.85	12.63
64	95.17%	5888	92	47175	73.62	14.52
96	95.03%	5760	60	71874	73.24	15.16