



Kevin Lüdemann, Azat Khuziyakhmetov

Slurm

Using a cluster for remote computation

Single node vs. MPI

- MPI jobs are a lot of independent tasks that (usually) use one core each
 - ▶ started with `srun` or `mpirun`
 - ▶ Slurm calls these tasks
- Single node jobs are usually just one task with many cores
- Both can be combined into hybrid jobs: multiple tasks using multiple cores each

Resource selection: CPU

s run options for parallel (SMP or MPI) jobs.

- | | |
|------------------------|--|
| -N <min>-<max> , | Minimum and maximum node count. You can also |
| --nodes=<min>-<max> | specify the exact number. |
| -n,--ntasks=<n> | Number of tasks (not equally distributed!) |
| --tasks-per-node=<n> | Tasks per node. If used with -n it denotes the maximum number of tasks per node. |
| -c,--cpus-per-task=<n> | CPUs per tasks. |

Rule of thumb

- -c for single node jobs
- -n for MPI jobs

Resource Selection: Memory

srun options

`--mem <size[K|M|G|T]>` Memory per node.

`--mem-per-cpu <size[K|M|G|T]>` Memory per core.

■ without options:

- ▶ each partition has a `DefMemPerCPU` option
- ▶ can be retrieved via `scontrol show partition <name>`

How to avoid waiting

Reservation: pchpc-2024 and pchpc-2024-2

Usage

Either: `--reservation=pchpc-2024` for each job

Or: `export SBATCH_RESERVATION=pchpc-2024`

The latter has to be unset, if you want to submit to a partition besides medium.

Exercises

- Try these job configurations:
 - 1 10 tasks
 - 2 10 tasks distributed over 3 nodes
 - 3 3 nodes with 3 tasks each
 - 4 1 task with 5 cores
 - 5 2 tasks per node on 2 nodes with 4 cores per task
- Play with the combination of number of cores or tasks, nodes and their effect on your available memory:
 - 1 1 core and `--mem 4G`
 - 2 3 tasks and 2 nodes, see effect of `--mem` and `--mem-per-cpu`
 - 3 20 tasks, see distribution of memory over hosts.
- use `slurm_resources` script to see the resources of your job

Time: 20 Minutes

Non interactive Jobs

Problem

- if you have big jobs, your queue time will be long
- srun needs you to stay logged in
- jobs can run for days

Non interactive Jobs

Problem

- if you have big jobs, your queue time will be long
- srun needs you to stay logged in
- jobs can run for days

Solution

Batch Jobs!

A job script is a shell script with a special comment section.
The #SBATCH lines have to come first!

`sbatch`: Basic job script example

```
#!/bin/bash
#SBATCH -p scc-cpu
#SBATCH -t 10:00
#SBATCH -o job-%J.out
```

```
slurm_resources
```

Submit with:

```
sbatch <script name>
```

Jobscripts

- A job script is essentially a normal script
- usually bash/shell, but can be any scripting language (R, python, perl)
- #SBATCH lines need to be at the top!
- you can copy files, load modules, and do any scripting you want
- for MPI, use `srun` or `mpirun` to start your program

SBATCH: Using Job Scripts

More Options

```
sbatch <slurm options> jobscript
```

- `--mail-type=<TYPE>` get mail notifications (type: BEGIN, END, etc.)
- `--mail-user=<address>` Default: `${USER}@gwdg.de`
- `-o/-e <file>` Store job output in file (slurm-<jobid>.out by default). `%J` in the filename stands for the jobid.

Slurm Commands

sinfo Info about the system and partitions.

-p <partition>, -t <state>

squeue Show the job queue.

-p <partition>, --me

scancel Cancel Job

scancel <JobID>

scancel -p <partition>|-u \$USER

Exercises

Exercise

Write your own Job script.

- Use `echo`, `hostname`, and `sleep X` (sleep for X seconds) to generate output or have it running for a longer time.
- Have the job send you an email. Advanced: Take a look at the different mail-type options. What do they do?
- Write the output to a different file. Redirect output and error into different files. Advanced: Take a look at the filename pattern options. Include node and job name in the output file.

Time: 20 Minutes

Task Distribution

Distributing tasks in the medium partition

```
#SBATCH -p scc-cpu
#SBATCH -n 240
#SBATCH -o job-%J.out

module purge
module load intel-oneapi-compilers intel-oneapi-mkl intel-oneapi-mpi namd

srun namd2 +setcpuaffinity apoal.namd
```

This will spread tasks among many nodes.

Task Distribution fixed

Distributing tasks in the medium partition

```
#SBATCH -p scc-cpu
#SBATCH -N 10
#SBATCH --ntasks-per-node 24
#SBATCH -o job-%J.out

module purge
module load intel-oneapi-compilers intel-oneapi-mkl intel-oneapi-mpi namd

srun namd2 +setcpuaffinity apoal.namd
```

Memory is faster than network!

Try to spread your tasks to as little nodes as possible.

Job Disk Space Usage Options

- \$TMP_LOCAL** Local hard disk of the node. SSD based and therefore a very fast option for storing temporary data. Automatic file deletion. A temporary directory is created on all nodes.
- \$WORK** Shared scratch space, available on most nodes, but there are two instances . Very fast, no automatic file deletion, but also no backup! Files may have to be deleted manually when we run out of space.
- \$HOME** Available everywhere, permanent, with backup. Personal disk space can be increased. Comparably slow.

Recipe: Using /scratch

```
#!/bin/bash
#SBATCH -p scc-cpu
#SBATCH -n 24
#SBATCH -N 1
#SBATCH -t 1-00:00:00

export g09root="/usr/product/gaussian/g09/d01"
source $g09root/g09/bsd/g09.profile

if [ ${WORK} -a -d ${WORK} ]; then
    export GAUSS_SCRDIR=${WORK}
else
    export GAUSS_SCRDIR=$TMP_LOCAL
fi

g09 myjob.com myjob.log
```

Exercises

Exercise

Write a job script, where you

- create a scratch directory
- copy data from your home file system to the scratch directory
- run a job with the data
- copy the results back
- delete the scratch directory

If you do not have a program/data to try this on, there is a small python program in `/scratch/projects/scc-course/` and a bit of input data.

Recipe: Combine shared memory and MPI

Running hybrid jobs

```
#SBATCH -p scc-cpu
#SBATCH -N 5
#SBATCH --ntasks-per-node=4
#SBATCH --cpus-per-task=6
#SBATCH -o job-%J.out

module purge
module load openmpi

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

srun hybrid_job
```


