

# Slurm

Azat Khuziyakhmetov

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen

Burckhardtweg 4, 37077 Göttingen

Phone: +49 551 39-30000 [gwdg@gwdg.de](mailto:gwdg@gwdg.de) [www.gwdg.de](http://www.gwdg.de)

03.04.2024

## Section 1

### Slurm

#### Getting started with Slurm

# How to use the cluster



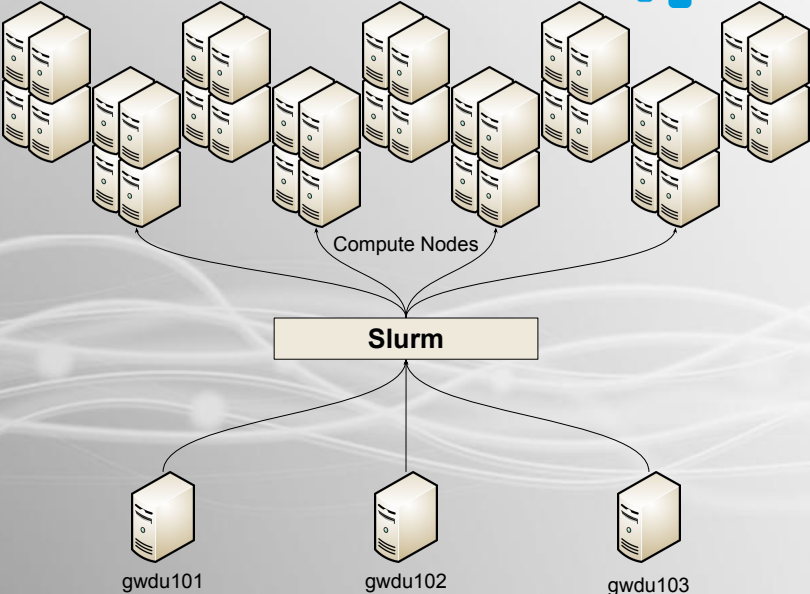
- Cluster divided into frontends and compute nodes
- Compute nodes are for all calculations
- You cannot connect directly to the compute nodes
- You cannot run heavy calculations on the frontends
- So how do you use the compute nodes?

Use our scheduler: Slurm

- Cluster divided into frontends and compute nodes
- Compute nodes are for all calculations
- You cannot connect directly to the compute nodes
- You cannot run heavy calculations on the frontends
- So how do you use the compute nodes?

Use our scheduler: **Slurm!**

# How to use the Cluster



A job is a set of instructions for Slurm, including

- one or multiple programs to execute
- estimated runtime
- required resources (CPUs, GPUs, Memory)
- and more...

Use `srun` to submit a job to Slurm

```
srun <program>
```

Example:

```
gwdu101:27 12:53:50 ~ > hostname
```

```
gwdu101
```

```
gwdu101:27 12:53:53 ~ > srun hostname
```

```
amp078
```

```
gwdu101:27 12:53:56 ~ > srun hostname -f
```

```
amp078.global.gwdg.cluster
```

- `srun` submits information on your job to Slurm
  - What is to be done? (path to your program and required parameters)
  - What are its requirements? (e.g. which nodes, number of tasks, maximum runtime)
- Slurm matches the jobs requirements against the capabilities of our nodes
- When suitable free resources are found the job is started
- Slurm prioritizes the jobs based on a number of factors.



- Different compute nodes have different features
- Slurm differentiates using **Partitions**

General purpose partitions:

**medium** General purpose partition, well suited for most jobs. Up to 1024 cores per job.

**fat** Up to 512 GB in one host.

**fat+** For extreme memory requirements. Up to 2048GB per host.

Special purpose partitions:

**gpu** For jobs using GPU acceleration.

**int** For interactive jobs, i.e. jobs which require a shell or a GUI.

**Cluster** A collection of networked computers intended to provide compute capabilities.

**Node** One of these computers, also called host or server.

**Frontend** Special node provided to interact with the cluster. `login-mdc.hpc.gwdg.de` in our case.

**Job** Program consisting of one or several parallel tasks.

**Partition** A pool of nodes suitable for the job

**Batch System** Management system distributing job tasks across job slots. Slurm.

```
srun <parameters> <program>
```

## common parameters

- p <partition> partition.
- t <hh:mm:ss> Maximum runtime. If this is exceeded the job is killed.
- A <all> Specify account 'all'. Only necessary if you are part of a working group with access to special partitions.

## `srun`: Interactive jobs

- `--x11` Adds X11 (GUI) forwarding. This requires that you connect to the frontend with `ssh -Y` and your local machine supports X-Windows.
- `-p int` Use the interactive partition. In `int` the nodes have no slot limit. They will take jobs until their load crosses a specified threshold, so jobs start immediately.
- `--pty` interactive mode

## Running Matlab

```
> ssh -Y login-mdc.gwdg.de  
> module load matlab  
> srun --x11 -p medium matlab
```

- The job will be dispatched and as soon as an available node is found and the Matlab interface will start.
- If you have your own license for Matlab then you need to place your `license.lic` file in `$HOME/.matlab/R2015a_licenses` directory (depending on the version you are using).

## Running R interactively

```
> ssh login-mdc.hpc.gwdg.de  
> srun --pty -p medium R
```

## Exercise

- Run a command on a compute node (e.g. `hostname`)
- Get an interactive shell on a compute node
- Try X11 forwarding (you can use `xterm` instead of Matlab)

10 Minutes



- Serial job** Job consisting of one task using one job slot.
- SMP job** Job with shared memory parallelization (often realized with OpenMP), meaning that all processes need access to the memory of the same node. Therefore uses several job slots **on the same node**.
- MPI job** Job with distributed memory parallelization, realized with MPI. Can use several job slots on several nodes and needs to be started with a helper program, e.g., `mpirun` or `srun`.

- 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

`srun` options for parallel (SMP or MPI) jobs.

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

## Rule of thumb

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

## 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>`

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.

## Exercise

- 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

## 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!

## 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 medium
#SBATCH -t 10:00
#SBATCH -o job-%J.out
```

slurm\_resources

Submit with:

```
sbatch <script name>
```

- 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

## 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.

**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`

## 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

## Distributing tasks in the medium partition

```
#SBATCH -p medium
#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 apoa1.namd
```

This will spread tasks among many nodes.

## Distributing tasks in the medium partition

```
#SBATCH -p medium
#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 apoa1.namd
```

Memory is faster than network!

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

**/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 at `$TMP_LOCAL`.

**/scratch** Shared scratch space, available on most nodes, but there are two instances (use `-C scratch` or `-C scratch2`). Very fast, no automatic file deletion, but also no backup! Files may have to be deleted manually when we run out of space.

**/scratch/ssd** special extra fast scratch file system only on `scratch1`. Ideal for temporary data in jobs spanning multiple nodes. Automatic file deletion. A per-job directory is created at `$TMP_SCRATCH`.

**\$HOME** Available everywhere, permanent, with backup. Personal disk space can be increased. Comparably slow.



# Recipe: Using /scratch



```
#!/bin/bash
#SBATCH -p medium
#SBATCH -n 24
#SBATCH -N 1
#SBATCH -C scratch
#SBATCH -t 1-00:00:00

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

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

g09 myjob.com myjob.log
```

## 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.

# The fat+ partition



The fat+ partition contains:

- 5 nodes with 1.5Tb Memory
- 1 node with 2Tb Memory

Usage recommendations:

- Work your way up. Start in fat and only use fat+ if your jobs runs out of memory.
- Use `sacct` or `profit-hpc`, see if your job really is memory bound
- When unsure, ask us!
- `--mem` or `--mem-per-cpu` is mandatory
- You might get angry mails from me, if you waste resources here

## Running hybrid jobs

```
#SBATCH -p medium
#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
```

## The `--qos` parameter

- Default maximum runtime: 2 days
- `--qos= <qos>` can select a QoS
- Two extra QoS available:
  - `short` for shorter jobs (max. 2h), has higher priority, limited job slots
  - `long` longer jobs (max. 5d), limited job slots.

## But my job is even longer

- try parallelizing more
- break it down into smaller steps
- check, if your software supports checkpoints
- check again!
- contact us

`scontrol show [partition|node|job] <x>` where x should be a node name, JobID or partition name.

`sprio` Priority information about pending jobs

`sacct` Get information about a job after it finished

`-j <jobid>`

`--format=JobID,User,JobName,MaxRSS,Elapsed,Timelimit`

## GPU parameters

`-G | --gpus=[type:]<n>` requests `n` GPUs of type

`--gpus-per-task=[type:]<n>` requests `n` GPUs of type per task

`--gpus-per-node=[type:]<n>` requests `n` GPUs of type per node

- CPUs are evenly distributed for every GPU

- Available types are:

- ➔ `rtx5000`

- ➔ `v100`

- ➔ `gtx1080`

- See: `sinfo -p gpu --format=%N,%G`

- take a look at your output files, while the job is running:
  - ➔ `tail -f /path/to/output`
- take a look at the jobs, while it is running
  - ➔ you can `ssh` into every node that currently calculates your job
  - ➔ use `htop` to see the processor and ram usage



## Read the extra job information

```
=====  
JobID = 4383174  
User = mboden, Account = admin  
Partition = gpu, Nodelist = dge[001,006]  
=====
```

[job output]

```
===== Job Information =====  
Submitted: 2020-04-24T17:35:41  
Started: 2020-04-24T17:35:41  
Ended: 2020-04-24T17:45:45  
Elapsed: 10 min, Limit: 60 min, Difference: 50 min  
CPUs: 2, Nodes: 2
```

```
===== Profit-HPC =====  
To generate the Profit-HPC text report, run the following command  
profit-hpc 4383174  
=====
```

Take a look at all the information. Is it as expected?

## Read your errors!

```
slurmstepd: error: Detected 1064 oom-kill event(s) in step XXXXXX.0 cgroup.  
Some of your processes may have been killed by the cgroup out-of-memory handler.  
srun: error: gwda024: task 3: Out Of Memory
```

Might have something to do with memory!

Have a look at your jobs memory with:

```
sacct -j JOBID -o jobid,MaxRSS,MaxRSSNode
```

And for more advanced job statistics, use profit-hpc