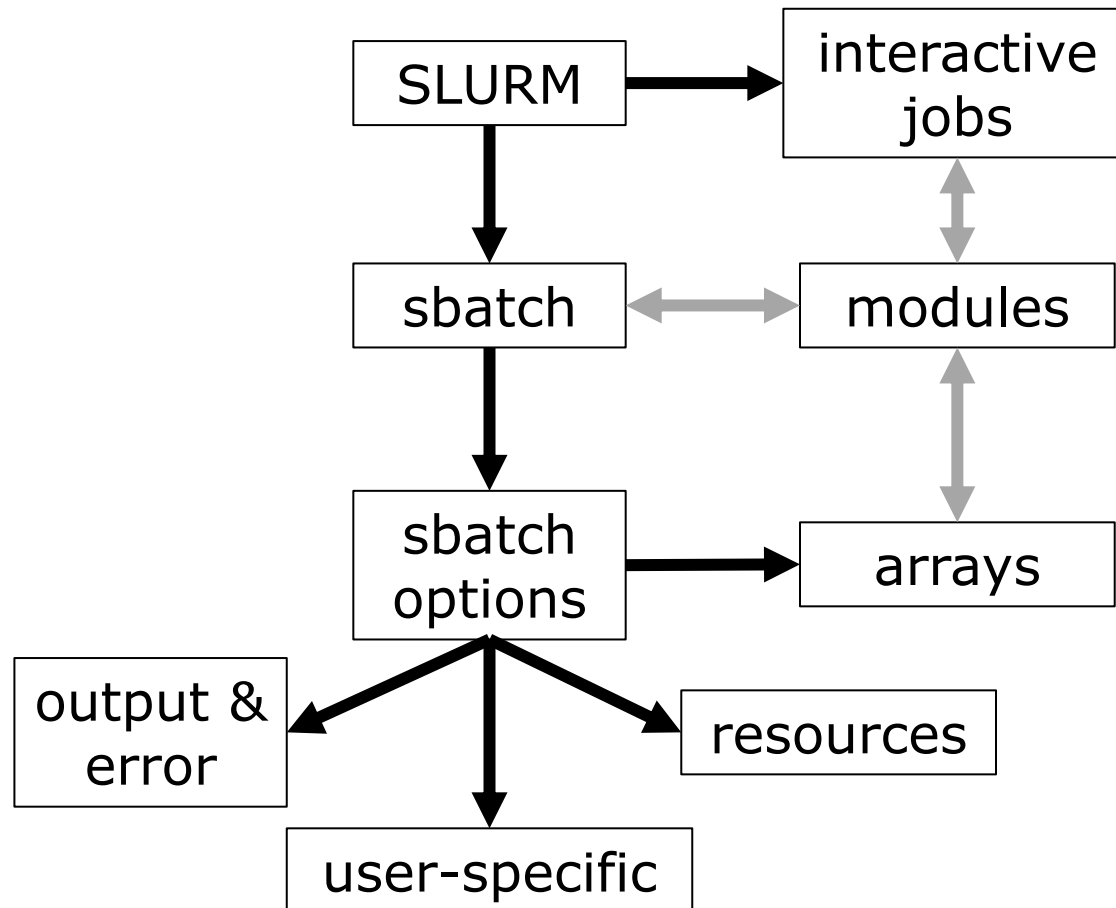


Introduction to SLURM and modules

Geert van Geest
Interfaculty Bioinformatics Unit
UniBe

Outline



1. SLURM: introduction

Limited resources

- Cluster has many users wanting to run jobs, which limits:
 - CPU
 - Working memory
 - Time
- How to assign which resources to which job?

Job scheduling

Job (computing)

From Wikipedia, the free encyclopedia

In [computing](#), a **job** is a unit of work or unit of execution (that performs said work).

Job scheduler

From Wikipedia, the free encyclopedia

A **job scheduler** is a computer application for controlling unattended background program execution of [jobs](#).^[1]

SLURM

- **S**imple **L**inux **U**tility for **R**esource **M**anagement
- Job scheduler on:
 - UBELIX
 - IBU cluster
 - and many more



Resource allocation commands

- sbatch
- srun
- salloc

script.sh

```
#!/usr/bin/env bash
```

```
my_program \  
--cpu 32 \  
--memory 128G
```

sbatch [options] script

```
$ sbatch --cpus-per-task=32 --mem-per-cpu=4G ./script.sh
```


script.sh

```
#!/usr/bin/env bash

#SBATCH --cpus-per-task=32
#SBATCH --mem-per-cpu=4G

my_program \
--cpu 32 \
--memory 128G
```

\$ sbatch ./script.sh

```
$ sbatch ./script.sh
```

```
Submitted batch job 6245994
```

```
$ squeue --job 6245994
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
6245995	pall	script.sh	gvangees	R	0:07	1	binfservas01

```
$ squeue -A gvangeest
```

JOBID	PARTITION	NAME	USER	ST	TIME	NODES	NODELIST(REASON)
6245995	pall	script.sh	gvangees	R	0:07	1	binfservas01

2. Frequently used sbatch options

2.1 Required resources

- CPU
 - `--cpus-per-task=2`
- Working memory
 - `--mem-per-cpu=4G`
- Time (days-hours:minutes:seconds)
 - `--time=1-05:00:00`

Low values could cause your job
to start earlier

But: job will fail if resources are
overrequested!

2.2 user specific

- Job name:
 - `--job-name=my_job_name`
- e-mail
 - `--mail-user=user@students.unibe.ch`
 - `--mail-type=begin,end,fail`

2.3 output & error

- `--output=existing/path/output_%j.o`
- `--error=existing/path/error_%j.e`

Path should exist!
Job will fail otherwise
(without error message)

3. Interactive jobs

Why submit interactive job?

- **Interactive job:** allocated resources that are approachable with shell
- Head (login) node is not for computation
- Debugging and testing can be much more convenient if interactive

srun

- Versatile command
- Used for job steps within sbatch (not treated in this course)
- Also for allocation of interactive job with pty (pseudo-terminal mode)

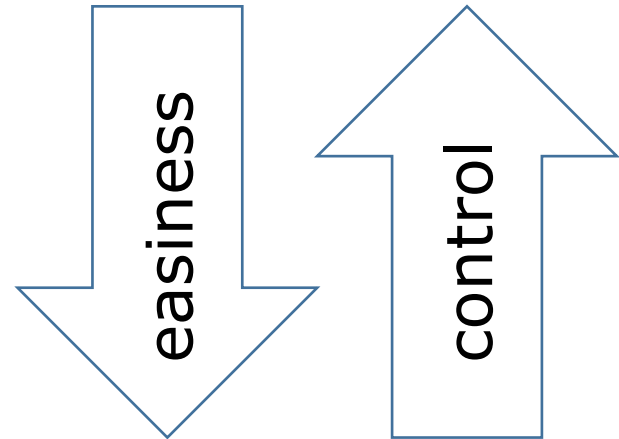
```
$ srun --cpus-per-task=1 --mem-per-cpu=4000 \  
> --time=00:05:00 --pty bash
```

Exit the interactive job
with `exit`

4. Modules

Software

- Install it yourself (at ~)
- Use a container
- Install with conda
- Use modules



Modules

- Check for available modules:
 - `module avail`
- Add a module to environment:
 - `module add`
- Unload a module:
 - `module rm`
- Available modules:
 - <https://www.vital-it.ch/services>

5. Job arrays

5.1 Jobs in parallel

- Run similar command with different parameters: parameter sweep
- E.g. alignment (e.g. with minimap2) on several files.

script.sh

```
#!/usr/bin/env bash
```

```
#SBATCH --array=0-3
```

```
#SBATCH --cpus-per-task=32
```

```
#SBATCH --mem-per-cpu=4G
```

```
my_program \
```

```
$SLURM_ARRAY_TASK_ID
```

--array=0-3

job_0

SLURM_ARRAY_TASK_ID=0

job_1

SLURM_ARRAY_TASK_ID=1

job_2

SLURM_ARRAY_TASK_ID=2

job_3

SLURM_ARRAY_TASK_ID=3

5.2 Using UNIX arrays

```
$ ls
file1.txt file2.txt file3.txt
$ FILES=(./*)
$ echo ${FILES[0]}
file1.txt
$ echo ${FILES[1]}
file2.txt
$ echo ${FILES[2]}
file3.txt
```

UNIX uses zero-based indexing

script.sh

```
#!/bin/bash
```

```
#SBATCH --cpus-per-task=32
```

```
#SBATCH --mem-per-cpu=4G
```

```
#SBATCH --array=0-7
```

```
FILES=(/path/to/input_data/*)
```

```
my_program ${FILES[$SLURM_ARRAY_TASK_ID]}
```