

# Job Templates

Here you can find example job script templates for a variety of job types

1. [Single-threaded tasks](#)
2. [Array jobs](#)
3. [Multi-threaded tasks](#)
4. [MPI tasks](#)
5. [Hybrid MPI/OpenMP tasks](#)
6. [GPU tasks](#)
7. [MPI+GPU tasks](#)

You can copy and paste the examples to use as a base - don't forget to edit the account and e-mail address as well as which software you want to use!

For all the possible things you can ask for see the [official documentation](#)

## Single threaded tasks

Here we want to use a tool that cannot make use of more than one CPU at a time.

The important things to know are:

- How long do I expect the job to run for?
- How much memory do I think I need?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

```
#!/bin/bash

#SBATCH --cpus-per-task 1
#SBATCH --partition cpu
#SBATCH --mem 8G
#SBATCH --time 12:00:00

#SBATCH --account ulambda_gruyere
#SBATCH --mail-type END,FAIL
#SBATCH --mail-user ursula.lambda@unil.ch
```

```
# Load the required software: e.g.  
# module purge  
# module load gcc
```

# Array jobs

Here we want to run an array job where there are N almost identical jobs that differ only in the input parameters.

In this example we use 1 CPU per task but you can obviously use more (see the multi-threaded task example)

See our introductory course for more details

The important things to know are:

- How long do I expect each individual job to run for?
- How much memory do I think I need per individual job?
- How many array elements do I have?
- How am I going to prepare my inputs for the elements?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

```
#!/bin/bash  
  
#SBATCH --cpus-per-task 1  
#SBATCH --mem 8G  
#SBATCH --partition cpu  
#SBATCH --time 12:00:00  
#SBATCH --array=1-100  
  
#SBATCH --account ulambda_gruyere  
#SBATCH --mail-type END,FAIL  
#SBATCH --mail-user ursula.lambda@unil.ch  
  
# Extract the parameters from a file (one line per job array element)  
  
INPUT=$(sed -n ${SLURM_ARRAY_TASK_ID}p in.list)  
  
# Load the required software: e.g.
```

```
# module purge
# module load gcc
```

# Multi-threaded tasks

Here we want to use a tool that makes use of more than one CPU at a time.

The important things to know are:

- How long do I expect the job to run for?
- How much memory do I think I need?
- How many cores can the task use efficiently?
- How do I tell the code how many cores/threads it should use?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

Note that on the DCSR clusters the variable `OMP_NUM_THREADS` is set to the same value as `cpus-per-task` but here we set it explicitly as an example

```
#!/bin/bash

#SBATCH --cpus-per-task 8
#SBATCH --mem 64G
#SBATCH --partition cpu
#SBATCH --time 12:00:00

#SBATCH --account ulambda_gruyere
#SBATCH --mail-type END,FAIL
#SBATCH --mail-user ursula.lambda@unil.ch

# Set the number of threads for OpenMP codes

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Load the required software: e.g.
# module purge
# module load gcc
```

# MPI tasks

Here we want to use code that uses MPI to allow for distributed memory parallel calculations.

The important things to know are:

- How many ranks (MPI tasks) do I want to run?
- How does my code performance scale as I increase the number of ranks?
- How much memory do I think I need per rank?
- How long do I expect the job to run for?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

Here we give the example of a code that we know runs efficiently with ~100 ranks so we choose 96 as this completely fills two compute nodes.

With MPI tasks always choose a number of tasks that entirely fills nodes: 48 / 96 / 144 / 192 etc - this is where the `--ntasks-per-node` directive is useful.

As we know that we are using the entire node it makes sense to ask for all the memory even if we don't need it.

```
#!/bin/bash

#SBATCH --nodes 2
#SBATCH --ntasks-per-node 48
#SBATCH --cpus-per-task 1
#SBATCH --mem 500G
#SBATCH --partition cpu
#SBATCH --time 12:00:00

#SBATCH --account ulambda_gruyere
#SBATCH --mail-type END,FAIL
#SBATCH --mail-user ursula.lambda@unil.ch

# Load the required software: e.g.
# module purge
# module load gcc mvapich2

# MPI codes must be launched with srun

srun mycode.x
```

## Hybrid MPI/OpenMP tasks

Here we want to run a hybrid MPI/OpenMP code where each MPI rank uses OpenMP for shared memory parallelisation.

Based on the code and the CPU architecture we know that 12 threads per rank is efficient - always run tests to find the best ratio of threads per rank!

The important things to know are:

- How many ranks (MPI tasks) do I want to run?
- How does my code performance scale as I increase the number of ranks and threads per rank?
- How much memory do I think I need per rank (taking into account OpenMP)?
- How long do I expect the job to run for?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

```
#!/bin/bash

#SBATCH --nodes 2
#SBATCH --ntasks-per-node 4
#SBATCH --cpus-per-task 12
#SBATCH --mem 500G
#SBATCH --partition cpu
#SBATCH --time 12:00:00

#SBATCH --account ulambda_gruyere
#SBATCH --mail-type END,FAIL
#SBATCH --mail-user ursula.lambda@unil.ch

# Load the required software: e.g.
# module purge
# module load gcc mvapich2

# Set the number of threads for the OpenMP tasks (12 in this case)

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# MPI codes must be launched with srun

srun mycode.x
```

## GPU tasks

Here we want to run a code that makes use of one GPU and one CPU core - some codes are able to use multiple GPUs and CPU cores but please check how the performance scales!

The important things to know are:

- How many GPUs do I need (1 or 2)
- How does my code performance scale as I increase the number GPUs?
- How much memory do I think I need for the CPU part of the job.
- How long do I expect the job to run for?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

Note the use of the `--gres-flags enforce-binding` directive to ensure that the CPU part of the code is on the same bus as the GPU used so as to maximise memory bandwidth.

In this example we run 2 tasks per node over 4 nodes for a total of 8 ranks and 8 GPUs.

```
#!/bin/bash

#SBATCH --cpus-per-task 1
#SBATCH --mem 500G
#SBATCH --partition gpu
#SBATCH --time 12:00:00
#SBATCH --gres gpu:1
#SBATCH --gres-flags enforce-binding

#SBATCH --account ulambda_gruyere
#SBATCH --mail-type END,FAIL
#SBATCH --mail-user ursula.lambda@unil.ch

# Load the required software: e.g.
# module purge
# module load gcc cuda
```

## MPI+GPU tasks

Here we have a code that used MPI for distributed memory parallelisation with one GPU per rank for computation.

The important things to know are:

- How many GPUs per rank do I need (probably 1)
- How does my code performance scale as I increase the number ranks?

- How much memory do I think I need for the CPU part of the job.
- How long do I expect the job to run for?
- Do I want e-mail notifications?
- What modules (or other software) do I need to load?

Note the use of the `--gres-flags enforce-binding` directive to ensure that the CPU part of the code is on the same bus as the GPU used so as to maximise memory bandwidth.

In this example we run 2 tasks per node over 4 nodes for a total of 8 ranks and 8 GPUs.

```
#!/bin/bash

#SBATCH --nodes 4
#SBATCH --ntasks-per-node 2
#SBATCH --cpus-per-task 8
#SBATCH --mem 500G
#SBATCH --partition gpu
#SBATCH --time 12:00:00
#SBATCH --gpus-per-task 1
#SBATCH --gres-flags enforce-binding

#SBATCH --account ulambda_gruyere
#SBATCH --mail-type END,FAIL
#SBATCH --mail-user ursula.lambda@unil.ch

# Load the required software: e.g.
# module purge
# module load gcc mvapich2 cuda

# MPI codes must be launched with srun

srun mycode.x
```

---

Révision #24

Créé 23 mai 2023 08:46:24 par Ewan Roche

Mis à jour 27 septembre 2024 08:28:38 par Cristian Ruiz