Submitting jobs

The computing clusters are equipped with SLURM queuing system.

In practice, all jobs are submitted to a queue and SLURM starts the execution of the job when the hardware resources required by the job are available.

jsub

All jobs must be submitted using the command *jsub*:

```
jsub [jsub-params] [qs-params] <program ID> <jobfile/jobname> [program-params]
```

For example:

```
jsub cry diamond.d12
```

All *<program IDs>* and general instructions for using *jsub* can be printed out by executing

```
jsub --help
```

Executing the command

```
jsub <program id>
```

prints out more detailed instructions for a particular program. For example:

```
jsub cry
```

jsub examples for all available programs

The table below lists short *jsub* examples for all available programs. For some programs, more information can be found by clicking the program name.

Program	<program ID>	jsub example
CRYSTAL	cry	`jsub cry diamond.d12`
TURBOMOLE	tm	`jsub tm <jobname> <TURBOMOLE-command>` `jsub tm c6h6_dft_opt jobex -ri`
ORCA	orca	`jsub orca benzene.inp`
Quantum ESPRESSO	qe	`jsub qe sio2 pw.x`
NWChem	nw	`jsub nw h2o.nw`
CASTEP	cas	`jsub cas SiO2_NMR`
Dalton	dal	`jsub dal benzene -dal benzene.dal -mol benzene.mol`
CFOUR	cf	`jsub cf acetone_ccsdt_opt`
CP2K	cp2k	`jsub cp2k h2o.cp2k`
Gromacs	gmx	`jsub -name my_gmx_job gmx mdrun -v`
GPAW	gpaw	`jsub gpaw pd100.py`
GULP	gulp	`jsub gulp quartz.gin`
Phonopy	-	`jsub cmd ...`
Phono3py	-	`jsub cmd ...`
ShengBTE	-	`jsub cmd ...`

XTB	-	`jsub cmd runxtb [options] <geometry> [options]` `(see runxtb --help for options)`
USPEX guide	-	`(submits jobs internally)`

Gaussian
 	
(only on Taito)
	gau	`jsub gau <inputfile>`
		
		`jsub gau methane.gjf`

General command	cmd	`jsub cmd uptime`

batch-log file

jsub generates a file called *<jobname>.batch-log* for all running jobs. This file includes general information about the job. Furthermore, error messages (stderr) will be directed to this file, so it always makes sense to check batch-log in case of an error.

Checking available resources with *jnodes*

Before submitting a job (especially on Taito), you can have a look at the available resources by executing the command *jnodes*. The view is unfortunately bit messy on Taito, but more useful on CMAT clusters.

Parallel jobs

For parallel jobs (MPI or OpenMP), the number of CPU cores needs to be allocated with *-np* parameter:

```
jsub -np 12 cry diamond.d12
```

If you want to restrict a parallel job to one node (a must for OpenMP parallel jobs), use the *-smp* parameter:

```
jsub -np 12 -smp orca benzene.inp
```

If you want to reserve a certain number of nodes (and you know what you are doing), use the *--nodes=<minnodes-maxnodes>* SLURM parameter:

```
jsub -np 12 -mem 6G --nodes=2-2 cry bigjob.d12
```

Important:

- While most programs listed above can be run in parallel, the parallelization efficiency is highly dependent on the system size and the methods used.
- When you start using a new program, you should run some benchmarks or discuss with a colleague to find out reasonable parallelization settings.
- Running all your jobs with lots of CPUs without testing first is a very good way to waste computing resources.
- A general rule of thumb for parallel jobs: **If you don't know what you are doing, then don't do it.**

Memory allocation

By default, jsub reserves 1GB of memory per CPU core. For some programs, it tries to make an automatic estimate of the memory usage based on input keywords (for example, Gaussian %mem parameter), but it is better to specify the memory usage manually. If the job exceeds its memory allocation, SLURM will kill it.

You can change the amount of memory with the *-mem <memory>* parameter:

```
jsub -mem 2G -np 12 cry silicon.d12
```

```
jsub -mem 2000M -np 12 cry silicon.d12
```

Both examples reserve 2GB of memory per CPU core. So, please use units **G** (gigabytes) or **M** (megabytes) when specifying the memory. Please try to make a realistic estimate. Typical DFT jobs normally do not consume more than 1 or 2 GB of memory per CPU core. Exceptions are huge systems and *ab initio* calculations with correlated methods such as CCSD(T).

Never allocate more memory than the cluster has per CPU core, unless you REALLY know what you are doing. Doing so will allocate "extra" CPUs and will waste resources, unless you really need a huge amount of memory. On Taito, CSC charges extra computing time for memory allocations larger than 4GB per CPU. Please discuss with Antti, if you are unsure about your memory allocation.

Choosing the queue

On CMAT clusters, there is only one queue called CLUSTER (*partition* in SLURM terminology). You don't need to specify the queue when submitting a job.

At CSC supercomputers, there are several queues (partitions). Please specify the partition when you are submitting a job. You can choose the queue on Taito using the *-q* parameter:

```
jsub -np 12 -q small cry diamond.d12
```

Setting the runtime on CSC supercomputers

On the CMAT computing cluters, the runtime of a job is unlimited. On CSC supercomputers you **need** to specify the runtime of the job with parameter *-t*. The maximum runtimes depend on the queue, but in normal queues the maximum runtime is typically three days. The default runtime is just few minutes. Here are some examples on setting the runtime.

Two days:

```
jsub -np 4 -q small -t 2-0 cry diamond.d12
```

12 hours (0 minutes, 0 seconds):

```
jsub -np 4 -q small -t 12:0:0
```

360 minutes (6 hours):

```
jsub -np 4 -q small -t 360 cry diamond.d12
```

Typically, the shorter your runtime, the easier it is to get a job allocation. However, too short time may result in a job that is killed before it can finish.

If you just want to quickly test a job, use the test queue (max. 15 minutes runtime on Puhti):

```
jsub -np 6 -q test -t 2 cry quickjob.d12
```

If the queue is very long, it's certainly worth testing a job before submitting it to the real queue. It's pretty annoying if your job crashes in 5 seconds after one day of queuing.

For more information about the -t parameter, type *man sbatch* in the console and find the parameter -t from the documentation.