Example parallel job scripts on the Linux-Cluster

- Introductory remarks
- Shared Memory jobs
- MPI jobs
  - On the CooLMUC-2 cluster
  - On the CooLMUC-3 cluster
- Special job configurations
  - Job Farming (starting multiple serial jobs on a shared memory system)

Introductory remarks

The job scripts for SLURM partitions are provided as templates which you can adapt for your own settings. In particular, you should account for the following points:

- Some entries are **placeholders**, which you **must replace with correct, user-specific settings**. In particular, path specifications and e-Mail addresses must be adapted. Always specify the appropriate directories instead of the names with the three periods in the following examples!
- For recommendations on how to do large-scale I/O please refer to the description of the file systems available on the cluster. It is recommended to keep executables within your HOME file system, in particular for parallel jobs. The example jobs reflect this, assuming that files are opened with relative path names from within the executed program.
- Because you usually have to work with the environment modules package in your batch script, sourcing the file `/etc/profile.d/modules.sh` is included in the example scripts.

Shared Memory jobs

This job type uses a single shared memory node of the designated SLURM partition. Parallelization can be achieved either via (POSIX) thread programming or directive-based OpenMP programming.

In the following, example scripts for starting an OpenMP program are provided. Please note that these scripts are usually **not useful for MPI applications**; scripts for such programs are given in subsequent sections.

<table>
<thead>
<tr>
<th>CooLMUC-2 cluster</th>
<th>CooLMUC-3 cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>#!/bin/bash</code></td>
<td><code>#!/bin/bash</code></td>
</tr>
<tr>
<td><code>#SBATCH --J job_name</code></td>
<td><code>#SBATCH --J job_name</code></td>
</tr>
<tr>
<td><code>#SBATCH --o ./%x.%j.%N.out</code></td>
<td><code>#SBATCH --o ./%x.%j.%N.out</code></td>
</tr>
<tr>
<td><code>#SBATCH --D ./</code></td>
<td><code>#SBATCH --D ./</code></td>
</tr>
<tr>
<td><code>#SBATCH --get-user-env</code></td>
<td><code>#SBATCH --get-user-env</code></td>
</tr>
<tr>
<td><code>#SBATCH --clusters=mpp2</code></td>
<td><code>#SBATCH --clusters=mpp3</code></td>
</tr>
<tr>
<td><code>#SBATCH --nodes=1-1</code></td>
<td><code>#SBATCH --nodes=1-1</code></td>
</tr>
<tr>
<td><code>#SBATCH --cpus-per-task=28</code></td>
<td><code>#SBATCH --cpus-per-task=64</code></td>
</tr>
<tr>
<td><code>#SBATCH --mail-type=end</code></td>
<td><code>#SBATCH --mail-type=end</code></td>
</tr>
<tr>
<td><code>#SBATCH --mail-user=insert_your_email_here</code></td>
<td><code>#SBATCH --mail-user=insert_your_email_here</code></td>
</tr>
<tr>
<td><code>#SBATCH --export=NONE</code></td>
<td><code>#SBATCH --export=NONE</code></td>
</tr>
<tr>
<td><code>#SBATCH --time=08:00:00</code></td>
<td><code>#SBATCH --time=08:00:00</code></td>
</tr>
<tr>
<td>module load slurm_setup</td>
<td>module load slurm_setup</td>
</tr>
<tr>
<td>export OMP_NUM_THREADS=28</td>
<td>export OMP_NUM_THREADS=64</td>
</tr>
<tr>
<td><code>#!/my_openmp_program.exe</code></td>
<td><code>#!/my_openmp_program.exe</code></td>
</tr>
</tbody>
</table>

MPI jobs

For MPI documentation please consult the MPI page on the LRZ web server. On current cluster systems, Intel MPI is used as the default environment. The following examples illustrate jobs that use four computational nodes.

MPI jobs may be jobs that use MPI only for parallelization ("MPP-style"), or jobs that combine usage of MPI and OpenMP ("hybrid")

**On the CooLMUC-2 cluster**

```bash
#!/bin/bash
#SBATCH --J job_name
#SBATCH --o ./%x.%j.%N.out
#SBATCH --D ./
#SBATCH --get-user-env
#SBATCH --clusters=mpp2
#SBATCH --nodes=1-1
#SBATCH --cpus-per-task=28
#SBATCH --mail-type=end
#SBATCH --mail-user=insert_your_email_here
#SBATCH --export=NONE
#SBATCH --time=08:00:00
module load slurm_setup
export OMP_NUM_THREADS=28
#!/my_openmp_program.exe
```
A setup as for the hybrid job can also serve to provide more memory per MPI task without using OpenMP (e.g., by setting `OMP_NUM_THREADS=1`). Note that this will leave cores unused!

### On the CooLMUC-3 cluster

**Notes:**
- starting more than 64 MPI tasks per KNL node is likely to cause startup failures.
- the `--constraint` option supplied in some of the scripts below is a suggestion. See the [KNL features documentation](https://knl.mpg.de) for more details.

---

The example will start 224 MPI tasks distributed over 8 nodes.

---

The example will start 32 MPI tasks with 7 threads each. Each node has 28 cores, so 4 tasks must be started per host.

---

The example will start 512 MPI tasks.

---

The example will start 128 MPI tasks with 16 threads each; 4 hyperthreads per core are used.

---

The example will start 128 MPI tasks with 4 threads each.

### Please note:

For some software packages, it is also possible to use SLURM's own `srun` command; this will however not work well in all situations for programs compiled against Intel MPI.

It is also possible to use the `--ntasks` keyword in combination with `--cpus-per-task` to configure parallel jobs.
Special job configurations

Job Farming (starting multiple serial jobs on a shared memory system)

Please use this with care! If the serial jobs are imbalanced with respect to run time, this usage pattern can waste CPU resources. At LRZ’s discretion, unbalanced jobs may be removed forcibly. The example job script illustrates how to start up multiple serial MATLAB jobs within a shared memory parallel SLURM script. Note that the various subdirectories subdir_1, ..., subdir_28 must exist and contain the needed input data.

```
#!/bin/bash
#SBATCH --job-name job_name
#SBATCH --output ./%x.%j.%N.out
#SBATCH --partition=general
#SBATCH --nodes=1
#SBATCH --mail-type=end
#SBATCH --mail-user=insert_your_email_here
#SBATCH --export=NONE
#SBATCH --time=08:00:00
module load slurm_setup
module load matlab

# Start as many background serial jobs as there are cores available on the node
for i in $(seq 1 28) ; do
    cd subdir_${i}
    matlab -nodesktop input.m > output.res &
    cd..
done
wait # for completion of background tasks
```