Running serial jobs on the Linux-Cluster

Serial job processing is based on usage of the SLURM scheduler. This document describes usage, policies and resources available for submission and management of such serial jobs.

Examples, Policies and Commands

The subdocuments linked to in the following table provide further information about usage of SLURM on LRZ's HPC systems:

| Examples | provides example job scripts which cover the most common usage patterns |
| Policies and Limits | provides information about the policies, such as memory limits, run time limits etc; also information about queues with specific properties (housed segments, large memory segment). |
| SLURM Workload Manager | SLURM commands and options for job submission and explains them, making appropriate recommendations where necessary. Provides hints for aborting jobs. |

Detailed Instructions (for Beginners)

Prerequisites

All serial programs in the serial segments of the cluster must be started up using either

- an interactive SLURM shell
- a SLURM batch script

In order to access the SLURM infrastructure described here, please first log in to a login node of the cluster, as described in the introduction.

Please be aware that misuse of the resources described here can result in the invalidation of the violating account. Such misuse would for example be constituted by:

- running production-like runs that take longer than 30 minutes, or chaining of many such runs
- running very many tasks on the same node, or starting more tasks when the load on the system is already very high (use the "uptime" or "top" command to see the current load)
- running tasks that use a lot of memory (> 2-3 GB), especially if the node memory is already fully booked (use the "free" command to find out how much is presently used)

Note that usage like compiling programs or running the tape archiver is to some extent exempted from the above strictures due to technical necessity. Serial job processing is supported on CoolMUC-2 and the shared memory system teramem1, but is not supported on CooLMUC-3 or IvyMUC.

Interactive SLURM shell

An interactive SLURM shells can be generated to execute tasks on the new multi-terabyte HP DL580 system "teramem". The following procedure can be used on the login node of CoolMUC3 (lxlogin8.lrz.de):

```
module load salloc_conf/teramem
salloc --cpus-per-task=32 --mem=2000000
srun ./my_shared_memory_program.exe
```

The above commands execute the binary "my_shared_memory_program.exe" using 32 threads and up to 2 TBytes of memory (the units are MBytes). Additional tuning and resource settings (e.g. OpenMP environment variables) can be explicitly performed before executing the srun command. Please note that the target system currently (still) uses the NAS-based SCRATCH area (as opposed to the GPFS based area available on CoolMUC2). Please note that the DL580 can also be used by script-driven jobs (see the examples document linked below).
If you want to work in an interactive bash shell then please use the following command:

```bash
$ salloc -pteramem_inter -n 1 srun --pty bash -i
```

## Script-driven SLURM jobs

This type of execution method should be used for all production runs. A step-by-step recipe for the simplest type of parallel job is given, illustrating the use of the SLURM commands for users of the bash shell. See the documentation section at the end for pointers to more complex setups.

### Step 1: Edit a job script

The following script is assumed to be stored in the file `myjob.cmd`.

```bash
#!/bin/bash

#SBATCH -J <job_name>  # (Placeholder) name of job (not more than 10 characters please)
#SBATCH -o ./%x.%j.%N.out  # (Placeholder) standard output and error go there. Note that the directory where the output file is placed must exist before the job starts, and the full path name must be specified (no environment variable!). The %x encodes the job name into the output file. The %j encodes the job ID into the output file. The %N encodes the master node of the job and can be added if job IDs from different SLURM clusters might be the same. Here, the specified path is relative to the directory specified in the -D spec.
#SBATCH -D ./  # directory used by script as starting point (working directory). The directory specified must exist. Here, the path is relative to the submission directory.
#SBATCH --clusters=serial  # Configure for serial processing
#SBATCH --partition=serial_std  # see also: Example serial job scripts or Policies and Limits
#SBATCH --get-user-env  # Set user environment properly
#SBATCH --mail-type=end  # Send an e-mail at job completion
#SBATCH --mem=800mb  # Specify maximum memory the job can use. Very large values can cause assignment of additional cores to the job that remain unused, so this feature should be used with care.
#SBATCH --mail-user=<email_address>@<domain>  # (Placeholder) e-mail address (don't forget, and please enter a valid address!)
#SBATCH --export=NONE  # Do not export the environment of the submitting shell into the job; while SLURM allows to also use ALL here, this is strongly discouraged, because the submission environment is very likely to be inconsistent with the environment required for execution of the job.

#SBATCH --time=08:00:00  # maximum run time is 8 hours 0 minutes 0 seconds; this may be increased up to the queue limit

module load slurm_setup  # First executed line: SLURM settings necessary for proper setup of batch environment.

module load ...  # load any required environment modules (usually needed if program is linked against shared libraries, or if paths to applications are needed). The "..." is of course a placeholder.

./my_serial_program.exe  # start executable. Please consult the example jobs or software-specific documentation for specific startup mechanisms
```

This script essentially looks like a bash script. However, there are specially marked comment lines ("control sequences"), which have a special meaning in the SLURM context explained on the right hand of the above table. The entries marked "Placeholder" must be suitably modified to have valid user-specific values.

For this script, the environment of the submitting shell will not be exported to the job's environment. The latter is completely set up via the module system inside the script.
Step 2: Submission procedure

The job script is submitted to the queue via the command

`sbatch myjob.cmd`

At submission time the control sequences are evaluated and stored in the queuing database, and the script is copied into an internal directory for later execution. If the command was executed successfully, the Job ID will be returned as follows:

Submitted batch job 65648.

It is a good idea to note down your Job ID's, for example to provide to LRZ HPC support as information if anything goes wrong. The submission command can also contain control sequences. For example,

`sbatch --time=12:00:00 myjob.cmd`

would override the setting inside the script, forcing it to run 12 instead of 8 hours.

Step 3: Checking the status of a job

Once submitted, the job will be queued for some time, depending on how many jobs are presently submitted. Eventually, more or less after previously submitted jobs have completed, the job will be started on one or more of the systems determined by its resource requirements. The status of the job can be queried with the `squeue --clusters=[all | cluster_name]` command, which will give an output like

```
CLUSTER: mpp1
JOBID PARTITION NAME USER ST TIME NODES NODELIST(REASON) 65646 mpp1_batch job1 xyz1 R 24:19 2 lxa[7-8] 65647 mpp1_batch myjob xza2 R 0:09 1 lxa14 65648 mpp1_batch calc yaz7 PD 0:00 6 (Resources)
```

(assuming mpp1 is specified as the clusters argument) indicating that the job is queued. Once the job is running, the output would indicate the state to be "R" (=running), and would also list the host(s) it was running on. For jobs that have not yet started, the `--start` option, applied to `squeue`, will provide an estimate (!) for the starting time. The `sinfo --clusters=[all | cluster_name]` command prints out an overview of the status of all clusters or a particular clusters in the SLURM configuration.

Inspection and modification of jobs

Queued jobs can be inspected for their characteristics via the command

`scontrol --clusters=<cluster_name> show jobid=<job ID>`

which will print out a list of "Keyword=Value" pairs which characterize the job. As long as a job is waiting in the queue, it is possible to modify at least some of these; for example, the command

`scontrol --clusters=<cluster_name> update jobid=65648 TimeLimit=04:00:00`

would change the run time limit of the above-mentioned example job from 8 hours to 4 hours.

Deleting jobs from the queue

To forcibly remove a job from SLURM, the command

`scontrol --clusters=<cluster_name> <JOB_ID>`

can be used. Please do not forget to specify the cluster! The `scancel (1)` man page provides further information on the use of this command.