SuperMUC-NG Batch Processing with SLURM

- General
- List of relevant commands
- sbatch and its options
- Batch queues (SLURM partitions) and their limits
- Examples
- Guidelines for resource selection
- SLURM Documentation

General

The batch system on SuperMUC-NG is the open-source workload manager SLURM (Simple Linux Utility for Resource management). For detailed information about the SLURM batch system, see [Slurm Workload Manager](#).

Submit hosts are usually login nodes that permit to submit and manage batch jobs.

Intel processors on SuperMUC-NG support the hyperthreading mode which might increase the performance of your application. With hyperthreading, you have to increase the number of MPI tasks per node from 48 to 96 in your job script. Please be aware that with 96 MPI tasks per node each process gets only half of the memory by default. If you need more memory, you have to specify it in your job script and use the fat nodes (see example batch scripts).

List of relevant commands

The most important commands are:

<table>
<thead>
<tr>
<th>Command’s name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo</td>
<td>provide overview of cluster status</td>
</tr>
<tr>
<td>squeue</td>
<td>print table of submitted jobs and their state. <strong>Note:</strong> non-privileged users can only see their own jobs.</td>
</tr>
<tr>
<td>sbatch</td>
<td>submit a job script</td>
</tr>
<tr>
<td>scancel</td>
<td>delete or terminate a queued or running job</td>
</tr>
<tr>
<td>scontrol</td>
<td>query and modify SLURM state</td>
</tr>
<tr>
<td>salloc</td>
<td>create an interactive SLURM shell</td>
</tr>
<tr>
<td>srun</td>
<td>execute argument command on the resources assigned to a job. <strong>Note:</strong> must be executed inside an active job (script or interactive environment). mpiexec is an alternative in batch job scripts</td>
</tr>
<tr>
<td>sview</td>
<td>GUI for viewing and managing resources and jobs (see below)</td>
</tr>
</tbody>
</table>

sbatch and its options

- [SLURM Workload Manager](#)
- Command/option Summary (two pages)

Batch queues (SLURM partitions) and their limits

- Batch queues are called partitions in SLURM.
- The allocation granularity is multiples of one node (only complete nodes are allocated and accounted for).
- Scheduling and prioritization is based on a multifactor scheme including wait time, job size, partition, and required quality of service.

The following partitions are available. Check with sinfo for more details, and special partitions.

<table>
<thead>
<tr>
<th>partition</th>
<th>min-max nodes per job</th>
<th>max usable memory</th>
<th>cores per node</th>
<th>max run time (hours)</th>
<th>max running jobs per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>interactive</td>
<td>1</td>
<td>90 GB</td>
<td>48</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>test</td>
<td>1</td>
<td>90 GB</td>
<td>48</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>micro</td>
<td>1</td>
<td>90 GB</td>
<td>48</td>
<td>48</td>
<td>20</td>
</tr>
<tr>
<td>general</td>
<td>1</td>
<td>90 GB</td>
<td>48</td>
<td>48</td>
<td>20</td>
</tr>
<tr>
<td>large</td>
<td>1-6</td>
<td>793-3168 (half of system)</td>
<td>760 GB</td>
<td>48</td>
<td>12 to be increased</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>--------------------------</td>
<td>--------</td>
<td>----</td>
<td>-------------------</td>
</tr>
<tr>
<td>fat</td>
<td>1</td>
<td>1-128</td>
<td>760 GB</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>mixed</td>
<td>1-6</td>
<td>64-3168</td>
<td>90 BG and 760 GB</td>
<td>48</td>
<td>12 to be increased</td>
</tr>
</tbody>
</table>

**Examples**

**Pure MPI without hyperthreading**

```
#!/bin/bash
# Job Name:
#SBATCH --J test_slurm
# Standard output and error:
#SBATCH --o ./job.%j.out
#SBATCH --e ./job.err.%j
# Initial working directory:
#SBATCH --D ./
# Notification
#SBATCH --mail-type=none
#SBATCH --mail-user=<youremail>
# Wall clock limit:
#SBATCH --time=24:00:00

# Do not export the environment of the submitting shell into the job
#SBATCH --export=NONE

# Queue and resources
#SBATCH --partition=general
# Number of nodes and MPI tasks per node:
#SBATCH --nodes=128
#SBATCH --ntasks-per-node=28

#initialize module system source
/etc/profile.d/modules.sh

# Run the program
mpiexec ./myprog
```

**Hybrid MPI/Openmp without hyperthreading**

```
#!/bin/bash
# Job Name:
#SBATCH --J test_slurm
# Standard output and error:
#SBATCH --o ./job.%j.out
#SBATCH --e ./job.err.%j
# Initial working directory:
#SBATCH --D ./
# Notification
#SBATCH --mail-type=none
#SBATCH --mail-user=<youremail>
# Wall clock limit:
#SBATCH --time=24:00:00

# Do not export the environment of the submitting shell into the job
#SBATCH --export=NONE

# Queue and resources
#SBATCH --partition=general
# Number of nodes and MPI tasks per node:
#SBATCH --nodes=128
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=3

#initialize module system source
/etc/profile.d/modules.sh

# Run the program:
export OMP_NUM_THREADS=6
for pinning threads correctly:
export OMP_PLACES=threads
mpiexec ./myprog
```

**Pure MPI with hyperthreading**

```
# Queue and resources
#SBATCH --partition=general
# Number of nodes and MPI tasks per node:
#SBATCH --nodes=128
#SBATCH --ntasks-per-node=96

# Enable Hyperthreading:
#SBATCH --ntasks-per-core=2

#initialize module system source
/etc/profile.d/modules.sh

# Run the program
mpiexec ./myprog
```

**Hybrid MPI/Openmp with hyperthreading**

```
# Queue and resources
#SBATCH --partition=general
# Number of nodes and MPI tasks per node:
#SBATCH --nodes=128
#SBATCH --ntasks-per-node=16
#SBATCH --cpus-per-task=3

#initialize module system source
/etc/profile.d/modules.sh

# Run the program:
export OMP_NUM_THREADS=6
for pinning threads correctly:
export OMP_PLACES=threads
mpiexec ./myprog
```
Guidelines for resource selection

Processing Mode

- Jobs that only use one or at most a few hardware cores perform serial processing and are not supported on SuperMUC-NG. Use the SuperMUC-Cloud for such purposes.
- Bunches of multiple independent tasks can be bundled into one job, using one or more nodes.

Run time limits

- Please note that all job classes impose a maximum run time limit. It can be adjusted downward for any individual job. Since the scheduler uses a backfill algorithm, the better you specify a realistic runtime limit, the better throughput of your job may be achieved.

Memory Requirements

- The total memory available in user space for the set of nodes requested by the job must not be exceeded.
- The memory used on each individual node must not be exceeded by all tasks run on that node.
- Applications exist for which the memory usage is unsymmetric. In this case it may become necessary to work with a variable number of tasks per node. One relevant scenario is a master-worker scheme where the master may need an order of magnitude more memory and therefore requires a node of its own, while worker nodes can share a node. LRZ provides the "mixed" partition for using thin and fat nodes concurrently.

Disk and I/O Requirements

- The disk and I/O requirements are not controlled by the batch scheduling system, but rely on parallel shared file systems, which provide system-global services with respect to bandwidth - this means that the total I/O bandwidth is shared between all users. The consequence is that all I/O may be significantly slowed down if heavily used by multiple users at the same time, or even - for large scale parallel jobs - by a single user. At present, LRZ can not make any Quality of Service assurance for I/O bandwidth.
- The appropriate usage of the parallel file systems is essential.
- Please consult File Systems and I/O for SuperMUC-NG for more detailed technical information.

Licences

- Some jobs may make use of licensed software, either from the LRZ software application stack, or of software installed in the user's HOME directory. In many cases, the software needs to access a license server because there exist limits on how many instances of the software may run and who may access it at all.
- There is no connection from SuperMUC-NG to the outside. Check with LRZ if you are in need of such licenses.
- LRZ is currently not able to manage license contingents. The reason is that a significant additional effort is required, not only with suitable configuration of SLURM, but also with how the license servers are managed. The situation implies that a job will fail if the usage limit of a licensed software is exceeded when the job starts.

Conversion of scripts from LoadLeveler and other Workload Managers table

- see: List of the most common command, environment variables, and job specification options used by the major workload management systems

SLURM Documentation

- SLURM Quick Start User Guide at SchedMD
- Documentation at SchedMD
- The manual pages slurm(1), sinfo(1), squeue(1), scontrol(1), scancel(1), sview(1)