Job Processing on the Linux-Cluster

Submit hosts

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

<table>
<thead>
<tr>
<th>Cluster segment</th>
<th>Submit hosts</th>
</tr>
</thead>
<tbody>
<tr>
<td>CooLMUC-2</td>
<td>lxlogin5, lxlogin6, lxlogin7</td>
</tr>
<tr>
<td>CooLMUC-3</td>
<td>lxlogin8, lxlogin9</td>
</tr>
<tr>
<td>IvyMUC</td>
<td>lxlogin10</td>
</tr>
</tbody>
</table>

However, note that cross-submission of jobs to other cluster segments is also possible. The only thing you need to take care of is that different cluster segments support different instruction sets, so you need to make sure that your software build produces the appropriate binary that can execute on the targeted cluster segment.

Guidelines for resource selection

Processing Mode

- Jobs that can only use one or at most a few hardware cores should perform **serial** processing. The particular segment used depends on how much main memory is needed.
- Jobs that can use one or more (but not many) hardware cores and require a single shared memory node, should perform **parallel** processing with the HP DL580 system (interactive queue teramem_inter).
- Programs that use MPI (or PGAS) for parallelization typically require **distributed memory parallel** processing. These should use the **parallel** processing facilities available in the CooLMUC-2 segment. Specific job configuration depends on
  - how many tasks are started per node
  - how much memory is needed by a task (see Memory Requirements below)
  - how many computational nodes are needed in total (there exists an upper limit for this number).

Workflows that alternate between serial and parallel processing, where serial processing requires a significant fraction of the execution wall time, should consider setting up separate serial and parallel SLURM job scripts with appropriately defined dependencies between them. For technical info, search the S LURM Workload Manager Document for the term “dependency”.

Run Time

Please note that all job classes impose a maximum run time limit. The precise values depend on the cluster segment used; it can be adjusted downward for any individual job. If your job cannot complete within the limit, there exist following (non-exclusive) options to enable processing on the Linux Cluster:

- Enable checkpointing to disk: This permits to subdivide a simulation into multiple jobs that are subsequently executed. The program must be capable of writing its state near the end of the job, and re-reading it at the beginning of the next job. Also, sufficient disk space must be available to store your checkpoint data.
- Increase the amount of parallelism of your program. This can be done by requesting more computational resources (if possible), or by improving the parallel algorithms used to achieve better performance with the same amount of computational resources.
- Perform code optimizations of your program: This may involve changes to the used algorithms (e.g. complexity reduction), or - more simply - adding vectorization (SIMD) directives to the code, using suitable switches for compiling, performing restructuring of hot loops and data structures to improve the temporal and/or spatial locality of your code, etc.

Memory Requirements

The technical documentation provides information on the available memory per core as well as per node for each segment.
For applications that are **serially executed**, the user may need to specify a separate memory requirement in the batch script if it is larger than the available per-core memory. This will cause the scheduler to raise the memory limit and will avoid scheduling of other jobs on the used node that would overstrain the node's memory resources.

For **parallel** applications that run on **distributed memory** systems two considerations must be met:

- 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.

Note that 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. The **MPI** startup mechanism (which is implementation dependent - please therefore consult the documentation for the variant you intend to use) usually offers a method for control of the startup procedure.

**Disk and I/O Requirements**

The disk and I/O requirements are not controlled by the batch scheduling system, but rely on either the availability of local disk space (which can be used as scratch space only), or of a shared file system (NFS/NAS based), or of a parallel shared file system (GPFS). The latter two 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. This may result in job failure if job time limits are exceeded due to slowed I/O. At present, LRZ can not make any Quality of Service assurance for I/O bandwidth.

Please consult the **file system page** 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. Please be aware of following aspects:

- if the license server for the software is not located at LRZ, jobs run on the LRZ cluster will need to be able to access it. This may involve opening firewall ports for the cluster networks on a site that is not under LRZ's administrative purview.
- 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.

**Alternative Resources**

If your job profile cannot be matched to one of the Cluster segments, please consider moving to a different LRZ system or service that does fit your requirements:

- For scaling out to very large core counts, due to needing the memory or wanting to reduce the computational time, consider applying for a **SuperMUC** project. The same may also apply if you need to do very large-scale I/O processing. However, except for initial testing, SuperMUC projects undergo a refereeing procedure. So the onus is on you to demonstrate the scientific value of using this expensive resource.
- For workflows that use only moderate computational resources but need long run times, the **LRZ cloud services** will be more appropriate. For permanently needed services you may consider acquiring a virtual machine. Both these options are also relevant if you wish to deploy your own OS images (including a tested application stack). Please note that deploying a virtual machine incurs costs on your part - see the entry "Dienstleistungskatalog" on the **services page** for details.