Hessisches Kompetenzzentrum für Hochleistungsrechnen

LOEWE-CSC Cluster | quick reference | Frankfurt

Center for Scientific Computing Frankfurt

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

1 Cluster Usage

Access — LOEWE Cluster Frankfurt

```
ssh <username>@loewe-csc.hhlr-gu.de
```

Go to CSC-Website/Access to get an account at the LOEWE Cluster. The project manager has to send a request to Prof. Lüdde to get CPU-Time for research projects. Please download the file and use a regular PDF viewer to open the forms.

How-To — execute myBatchScript.sh

1. first you have to login to one of the login nodes
2. prepare a batch script with your requirements
3. execute the batch script to run your application

Hardware Resources — LOEWE Cluster Frankfurt

#nodes	CPU	GHz	# CPUs Cores	RAM	GPU
600	AMD Opteron 6172	2.10	2/24	64GB	1xATI Radeon HD5870 1GB
198	Intel Xeon E5- 2670v2	2.50	2/20	128GB	
52	Intel Xeon E5- 2630v2	2.60	2/16	128GB	2xAMD FirePro S10000 12GB

File Systems — storage systems

mountpoint	/home	/scratch	/local	/data0[1|2]
size	10GB PU	764 TB	1.4 T	500TB each
access time	slow	fast	fast	slow
system	NFS	FhGFS	ext3	NFS
network	Ethernet	InfiniBand		Ethernet

Public Administration Meeting

Every first Wednesday of the month at 10:00am there is a administration meeting of the CSC, where you can attend. It takes place in the physics building, room 1.101.

Module — setting program environments

Syntax:	`module <command> <modulename>`
`avail`	display all available modules
`list`	display all loaded modules
`load | add <module>`	load a module
`load unstable`	load a deprecated or unstable module
`unload | rm <module>`	unload a module
`switch | swap <old-module> <new-module>`	first unloads an old module then loads a new module
`purge`	unload all currently loaded modules

You can load your own modules:	currently loaded module:
`load use.own`	`slurm/2.6.3`
`load ~/privatemodules`	

`load use.own` creates a directory `$HOME/privatemodules` & changes your `MODULEPATH` environment variable to ensure that the module command looks for the modules in your own area.

How-To — prepare custom modules

1. write module files & put them in directory `$HOME/privatemodules`
2. put module `load use.own` in your .bashrc
3. use the module commands on your local modules

Resource Manager — LOEWE Cluster Frankfurt

On our systems, compute jobs are managed by SLURM. At the LOEWE Cluster, the node allocation is exclusive. You can find more examples on our CSC-Website/ClusterUsage/LOEWE. In SlurmCommands, there is a detailed summary of the different options.

Contact — HPC Frankfurt

If you have any HPC-questions about SLURM and want help by debugging & optimizing your program, please write to hpc-support@csc.uni-frankfurt.de.
Else, you can contact the system administrators if you need software to be installed: support@csc.uni-....
Detailed documentation on using the LOEWE Cluster can be found at CSC-Website.

Partitions — LOEWE Cluster Frankfurt

partition	run time	Max Nodes	Max NodesPU	Max JobsPU	Max SubmitPU	Max ArraySize
`parallel`	30d	750	150	30	50	1001
`gpu`	30d	50	50	10	30	1001
`test`	2h	2-12		10	10	1001

To view such informations on the cluster, use the command:

```
sacctmgr list QOS partition format=maxnodes,maxnodesperuser
    ,maxjobsperuser,maxsubmitjobsperuser
scontrol show partition
sinfo -p partition
squeue -p partition
```

Per-User Resource Limits — LOEWE Cluster Frankfurt

limit	description
MaxNodes	max No of nodes
MaxNodesPU	max No of nodes to use at the same time
MaxJobsPU	max No of jobs to run simultaneously
MaxSubmitPU	max No of jobs in running or pending state
MaxArraySize	max job array size

Architecture of Partitions — LOEWE Cluster Frankfurt

partition	description
`parallel`	A mix of AMD Magny-Cours nodes and Intel Xeon Ivy Bridge E5-2670v2 nodes. Both partitions will contain 99 Intel nodes.
`gpu`	dual-socket Intel Xeon Ivy Bridge E5-2650v2 CPU/GPU nodes, each with two AMD FirePro S10000 dual-GPU cards

The architecture will be selectable via the `'--constraint'` option,
`dual` = dual-socket AMD Magny-Cours CPU/GPU nodes,
`intel20` = dual-socket Intel Ivy Bridge CPU nodes.

`'--constraint=gpu'` will become obsolete, you will have to use `'--partition=gpu'` instead.

Hessisches Kompetenzzentrum für Hochleistungsrechnen

LOEWE-CSC Cluster | quick reference | Frankfurt

Center for Scientific Computing Frankfurt

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

2 Job Submission & Execution

sbatch — allocate resources | batch mode

Syntax:	sbatch myBatchScript.sh
-a, --array=<indexes>	submit a job array
-C, --constraint=<feature>	specify features of a Cluster
-c, --cpus-per-task=<ncpus>	Threads How many threads run on the node? with OpenMP
-J, --job-name=<job-name>	specify a name for the allocation
-m, --distribution=<block|cyclic|arbitrary|plane>	mapping of processes
--mem=<MB>	specify real memory required per node
--mem-per-cpu	min memory required per allocated CPU
--mem_bind=<type>	bind tasks to memory
-N, --nodes=<min[-max]>	Nodes How many nodes will be allocated to this job?
-n, --ntasks=<number>	Tasks How many processes are started? important for OpenMP
-p <partition>	request specific partition for the resource
-t <time>	set limit on total run time of the job
-w, --nodelist=<node_name_list>	request a specific list of node names

process binding

constraints each process to run on specific processors

--cpu_bind	process binding to cores & CPUs	**srun**
--bind-to	|-core|-socket|-none|	**mpirun**
--cpus-per-proc <#perproc>	bind each process to the specified number of cpus	
--report-bindings	report any bindings for launched processes	
--slot-list <id>	list of processor IDs to be used for binding MPI processes	

salloc — allocate ressources | interactive mode

Syntax:	salloc [options] [<command> [command args]]

same features like sbatch

srun — run parallel jobs | interactive mode

After modulefiles are loaded and resources have been allocated, an application on the assigned node can be started with preceding
srun run parallel jobs | mpiexec run mpi program
In this shell window more applications can be started.

sbatch — execute myBatchScript.sh | batch mode

```
#!/bin/bash
#SBATCH -p parallel # partition (queue)
#SBATCH -C dual|intel20 # class of nodes
#SBATCH -N|-n|-c 1 # number of nodes|processes|cores
#SBATCH --mem 100 # memory pool for all cores
#SBATCH -t 0-2:00 # time (D-HH:MM)
srun helloworld.sh # start program
```

3 Accounting

sacct — display accounting data

Syntax:	sacct [options]
-b, --brief	displays jobid, status, exitcode
-e, --helpformat	print a list of available fields
-o, --format	comma separated list of fields

sacctmgr — view Slurm account information

Syntax:	sacctmgr [options] [command]
list | show	display information about the specified entity

4 Job Management

scancel — cancel a job

Syntax:	scancel <jobid>
-u <username>	cancel all the jobs for a user
-t PD -u <username>	cancel all the pending jobs for a user

sinfo — view info about nodes and partitions

Syntax:	sinfo [options]
-i <seconds>	print state on a periodic basis
-l, --long	print more detailed information
-n <nodes>	print info only about the specific node
-p <partition>	print info about the specified partition
-R, --list-reasons	list reasons why nodes are in the down, drained, fail or failing state
-s, --summarize	list only a partition state summary with no node state details

squeue — view job info located in scheduling queue

Syntax:	squeue [options]
-i <seconds>	report requested information
-j <job_id_list>	print list of job IDs
-r	print one job array element per line
--start	report expected start time & resources to be allocated for pending jobs
-t <state_list>	print specified states of jobs
-u <user_list>	print jobs from list of users

scontrol — view state of specified entity | hold & resume

Syntax:	scontrol [options] [command]
-d, --details	print show command print more details
-o, --oneliner	print information one line per record
Syntax:	**scontrol show ENTITY_ID**
job <job_id>	print job informations
node <name>	print node informations
partition <name>	print partition informations
reservation	print list of reservations
Syntax:	**scontrol hold|resume|requeue**
hold <jobid>	pause a particular job
resume <jobid>	resume a particular job
requeue <jobid>	requeue (cancel & rerun) a particular job
suspend <jobid>	suspend a running job