

1 Cluster Hardware

Access

```
ssh <username>@goethe.hhlr-gu.de
```

GOETHE-HLR

Go to CSC-Website/Access/Goethe-HLR to get an account. The project manager has to send a request to submission@csc.uni-frankfurt.de to get CPU-Time for research projects. Further information at our website.

Architecture

#Nodes --nodes	CPU	#Cores --ntasks	RAM --mem
400	Intel Xeon Skylake Gold 6148	40	192GB
72	Intel Xeon Skylake Gold 6148	40	768GB
198	Intel Xeon E5-2670 Ivy Bridge	20	128GB
139	Intel Xeon E5-2640 Broadwell	20	128GB
110	AMD EPYC 7452 8xAMD Radeon Instinct MI50 16GB	64	512GB

File Systems

storage systems

mountpoint	/home	/scratch	/local	/arc[1 2]
size	10 GB PU	5 TB PG	1.4 TB PU	10 TB PG
access time	slow	fast	fast	slow
system	NFS	FhGFS	ext3	NFS
network	Ethernet	InfiniBand		Ethernet

- Use the **/scratch**-directory instead of **/home** to write out the standard output and error
- /scratch** is a parallel filesystem & is available for huge data set during the computation
- /local** is only available during computation. Files will be deleted when the job is finished.
- /arc0[1|2]** is persistent & must be requested. They are only mounted on the login node, afterwards you can work on the node with rsync

Partitions & Constraints

p:	--partition	time	Nodes	NodesPU	JobsPU	SubmitPU
c:	--constraint					
p:	general1 skylake	21d	475	150	40	50
p:	general2 broadwell	21d	337	150	40	50
c:	ivy					
p:	gpu	21d	110	40	20	30
p:	test skylake	1h	2-12	3	3	4

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

Per-User Resource Limits

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 1001

Hyperthreading

Nodes	Sockets	cores	Threads	--extra- node-info	Cores	on off
skylake	2	20	1	2:20:1	40	off
	2	20	2	2:20:2	80	on
ivybride	2	10	1	2:10:1	20	off
	2	10	2	2:10:2	40	on
gpu	2	32	1	2:32:1	64	off
	2	32	2	2:32:2	128	on

2 Cluster Usage

Getting Help

You will find further information about usable commands on the clusters with `man <command>`.

You will find further information about usable commands of spack with `spack --help`.

Documentation

<https://csc.uni-frankfurt.de/>
→ Cluster Usage

<https://spack.io/>
<https://spack.readthedocs.io>

Support

General questions, Hardware, Storage, unreachable nodes: support@csc.uni-frankfurt.de

Softwaresupport(Spack, Python, Code optimization, Debugging): hpc-support@csc.uni-frankfurt.de

<https://github.com/spack/spack>
Submit GitHub issues and talk to the Spack developers

How-To-Compile

- C1 install spack
- C2 source spack
- C3 compile software
- C4 prepare module file

How-To-Run

- R1 load module file
- R2 write bash script
- R3 submit job with slurm

Spack Workflow

- W1 building packages
- W2 running binaries
- W3 developing software

Managing Modules

- basic packages by CSC
- modules by SPACK
- modules by Lmod
- your own modules

Configuration

change tmp folder

change `/spack/etc/spack/defaults/config.yaml` otherwise you may block users on the cluster

```
build_stage:
# - $tempdir
# - $spack/var/spack/stage
- /scratch/<your-project-name>/<your-name>/spack/tmp
```

3 Software Handling

Module	setting program environments	R1
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>	switch modules	
purge	unload all currently loaded modules	

How-To	use custom modules
1	writing a module file in tcl to set environment variables
2	module load use.own enables you to load your own modules
3	module load ~/privatemodules/modulename
4	use facilities provided by module

Installation Spack itself	C1
1	git clone https://github.com/spack/spack.git
2	cd spack

Basic Spack Usage	C2
. share/spack/setup-env.sh	Has to be made after each login
echo ". spack/share/spack/setup-env.sh" >> ~/.bash_profile	If you want this to be permanent
- /scratch/<your-project-name>/<your-user-name>/spack/tmp	add to build_stage in config.yaml

Lmod	C4
spack install lmod	handle installed packages
nano /.spack/modules.yaml	create file
spack module tcl refresh	apply settings
--delete-tree -y	
modules.yaml	show only directly installed modules

How-To	
spack install mpileaks	1
spack install mpileaks@3.3	2
spack install mpileaks@3.3 %gcc@4.7.3	3
spack install mpileaks@3.3 %gcc@4.7.3 +thread	4
spack install mpileaks@3.3 cflags="-O3 -g3"	5
spack install mpileaks@3.3 ^mpich@3.2 %gcc@4.9.3	6

Basic Commands	use spack commands
Syntax:	spack <command> [<arg> <package>]
list	listing available packages
list <package>	search for package
info <package>	info about package
versions <package>	search for specific version of package
extensions <package>	package live inside prefix of another package
find	show installed packages with versions
find -dlvf	show dependence tree
find --path <package>	show path of package
location --install-dir <package>	show path of package
spec -I <package>	-I = is the package already installed

Compiler Configuration	C3
Syntax:	spack <command> [<arg> <package>]
compilers	list compilers on cluster
compiler list	list compilers on cluster
compiler add	alias for spack compiler find
compiler find	list location where compilers are installed
compiler info	see specifics on a particular compiler
config edit compilers	configure a compiler manually

modules.yaml	
modules:	
tcl:	
hash_length: 0	
naming_scheme: \${PACK}/\${VERS}-\${COMPNAME}-\${COMPVERS}	
blacklist_implicit: true	

How-To	
1	unconstrained
2	@ custom version
3	% custom compiler
4	x - build option
5	setting compiler flags
6	^ dependency constraints

Uninstall Spack	
Syntax: spack <command> <package> @<version> +<variant> ^<dependency>	
uninstall --dependents <package>	
install package	install a package & its dependencies
uninstall package	uninstall a package & its dependencies
uninstall package /id	to be more specific
uninstall <spec>	
uninstall /<hash>	

Building Packages W1

for just one application
packages.yaml

```
packages:
  python:
    version: [3.5.1]
  modele-utils:
    version: [cmake]

everytrace:
  version: [develop]
netcdf:
  variants: +mpi

all:
  compiler: [gcc@5
             .3.0]
  providers:
    mpi: [openmpi]
    blas: [openblas]
    lapack: [openblas]
```

Building Packages W1

for multiple applications
script of combinatorial sets of
installs

```
#!/bin/bash

compilers=( %gcc
            %intel
            %pgi )

mpis=( openmpi
       intel-mpi )

for compiler in "${compilers[@]}"
do
  # Serial installs
  spack install szip
  spack install hdf
  spack install hdf5
  spack install netcdf
  spack install netcdf-fortran
  spack install ncview

  # Parallel installs
  for mpi in "${mpis[@]}"
  do
    spack install $mpi
    spack install hdf5+mpi
    spack install parallel-
      netcdf
  done
done
```

Building Packages W2

1. `spack create <my_package>`

```
from spack import *

class Mpileaks(Package):
    """FIXME: Description of your package."""

    # FIXME: Url for your packages homepage.
    homepage = "http://www.example.com"
    url       = "https://github.com/hpc/mpileaks/
                releases/download/v1.0/mpileaks-1.0.tar.gz"
    version(1.0, 8838c574b39202a57d7c2d68692718aa)

    # FIXME: Add dependencies if required.
    # depends_on(foo)

    def install(self, spec, prefix):
        # FIXME: Unknown build system
        make()
        make(install)
```
2. `spack edit <my_package>`
3. `spack install <my_package>`

compilers.yaml

```
- compiler:
  flags: | cflags: | cxxflags: | cppflags: | modules: []
  extra_rpaths:
  - /cluster/intel/compilers_and_libraries_2019.5/linux/
    compiler/lib/intel64/
  - /cluster/intel/compilers_and_libraries_2019.5/linux/
    compiler/lib/intel64_lin/
  'operating_system': rhel7
  paths:
    cc: /cluster/intel/compilers_and_libraries_2019.5/
      linux/bin/intel64/gcc
    cxx: /cluster/intel/compilers_and_libraries_2019.5/
      linux/bin/intel64/g++
    f77: /cluster/intel/compilers_and_libraries_2019.5/
      linux/bin/intel64/gfortran
    fc: /cluster/intel/compilers_and_libraries_2019.5/
      linux/bin/intel64/gfortran
  spec: intel@19.5
```

Running Binaries from Packages

W2

- Spack builds binaries with RPATHs
- Find Spack installation directories


```
spack location --install-dir
```

 1. `$ CMAKE = spack location --install-dir cmake/bin/cmake`
 2. `$ find ~/spack/opt/ -name cmake | grep bin`
- Spack-Generated Modules

number of spack load commands into .bashrc
- Generated Load Scripts

pre-compute which modules to load with spack module tcl loads

put source ~/env/spackenv into .bashrc

```
#!/bin/sh
# Generate module load commands in /env/spackenv

cat <<EOF /bin/sh >${HOME}/env/spackenv
FIND=spack module tcl loads --prefix python

\FIND modele-utils
\FIND emacs
\FIND ncview
\FIND nco
\FIND modele-control
EOF
```

Filesystem Views

W2

Syntax: `spack <command> [<arg> <package>]`

`view statlink (status, check) <my_view>`

check status of packages in view

`view symlink <my_name>` create a view via symlink

`view hardlink <my_name>` create a view via hardlink

`view remove <my_name> <package>` remove package from view

4 Job Submission & Execution

Resource Manager Cluster Frankfurt

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

sbatch execute myBatchScript.sh | batch mode R2

```
#!/bin/bash
#SBATCH -p general1      # partition (queue)
#SBATCH -C ivy          # 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
```

sbatch batch mode | salloc interactive | allocation R3

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

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.

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

5 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

6 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

Syntax: `scontrol [options] [command] ENTITY_ID`

- d, --details print show command print more details
- o, --oneliner print information one line per record
- 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
- show job <job_id > print job informations
- show node <name> print node informations
- show partition < name> print partition informations
- show reservation print list of reservations