



Introduction to LRZ HPC Systems

2022-06-27 | Gerald Mathias

Leibniz-Rechenzentrum (LRZ)/ Leibniz Supercomputing Centre



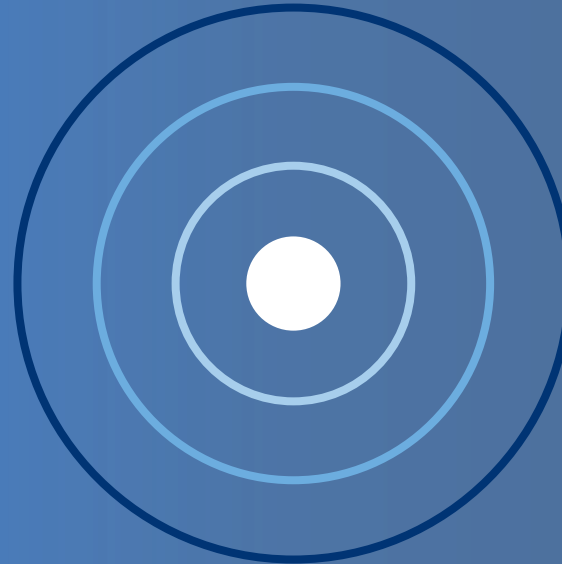
of the Bayerischen Akademie der Wissenschaften/
Bavarian Academy of Sciences and Humanities



250
employees
approx.



56
years of
IT support



Computer Centre
for all Munich Universities

Regional Computer Centre
for all Bavarian Universities

National Supercomputing Centre
(GCS)

European Supercomputing Centre
(PRACE)



Partnership for Advanced Computing in Europe (PRACE) | 25 Countries

Federated, pan-European Tier-0
supercomputing infrastructure

Hosting Members:

- GCS (Germany: LRZ, HLRS, JSC)
- BSC (Spain)
- CSCS (Switzerland)
- CINECA (Italy)
- GENCI (France)

PRACE 2: 2017 – 2020



A long row of black server racks in a data center. The racks are filled with server hardware, and the front panels are illuminated with a blue light. The racks are arranged in a long line, and the floor is made of light-colored tiles. The ceiling is visible, showing various pipes and lighting fixtures. The overall atmosphere is clean and professional.

SuperMUC-NG

SuperMUC-NG: Intel / Lenovo

	thin	fat
Processor type (205/240 W TDP) Xeon Platinum 8174, 24 cores	Intel Skylake	Intel Skylake
Number of cores per node	48 (2x24)	48 (2x24)
Memory per node	96 GB	768 GB
Nominal frequency	2.7 GHz	2.7 GHz
AVX-512 frequency (all cores active), current default frequency	2.3 GHz	2.3 GHz
Floating point operations per clock (Fused MulAdd = 2)	32	32
Total number of nodes of this type	6336	144
Number of islands with this node type (792 nodes / 144 Nodes)	8	1
Fat Tree with island, pruning 1:3.8 between islands		
Total Cores	311,040	
Total Nodes	6480	
Total Memory	719 TByte	
„Peak Performance“	26.9 PF	
Linpack (Nov 18: rank 8, Jun 19 and Nov 19: rank 9)	19.5 PF	
<i>Cf. Vector Triad RINF $vd=va*vb+vc$ (Average/Large+OpenMP)</i>	1.0 / 0.2 PF	



IBM Spectrum Scale (GPFS) Parallel File System

- SCRATCH/WORK
 - 50 PByte capacity
 - 500 GByte/s I/O bandwidth
- LRZ DSS: Data Science Storage for Long Term Data Storage
 - 20 PByte capacity and
 - 70 GByte/s I/O bandwidth
- HOME
 - 256 TB + 256 TB Replika
 - 28 Gbytes/s SSD Tier, 7 Gbyte/s HDD Tier, 40000 IOPS



OpenStack Compute Cloud (100 GigE)

- 32 nodes with 2x Intel Xeon 6148 processors, 192 Gbyte
- 32 nodes with 2x Intel Xeon 6148 processors, 2x Nvidia Volta 100 GPUs, 768 GByte memory
- 1 huge memory node with 8x Intel Xeon 8160 processors, 6144 GByte memory

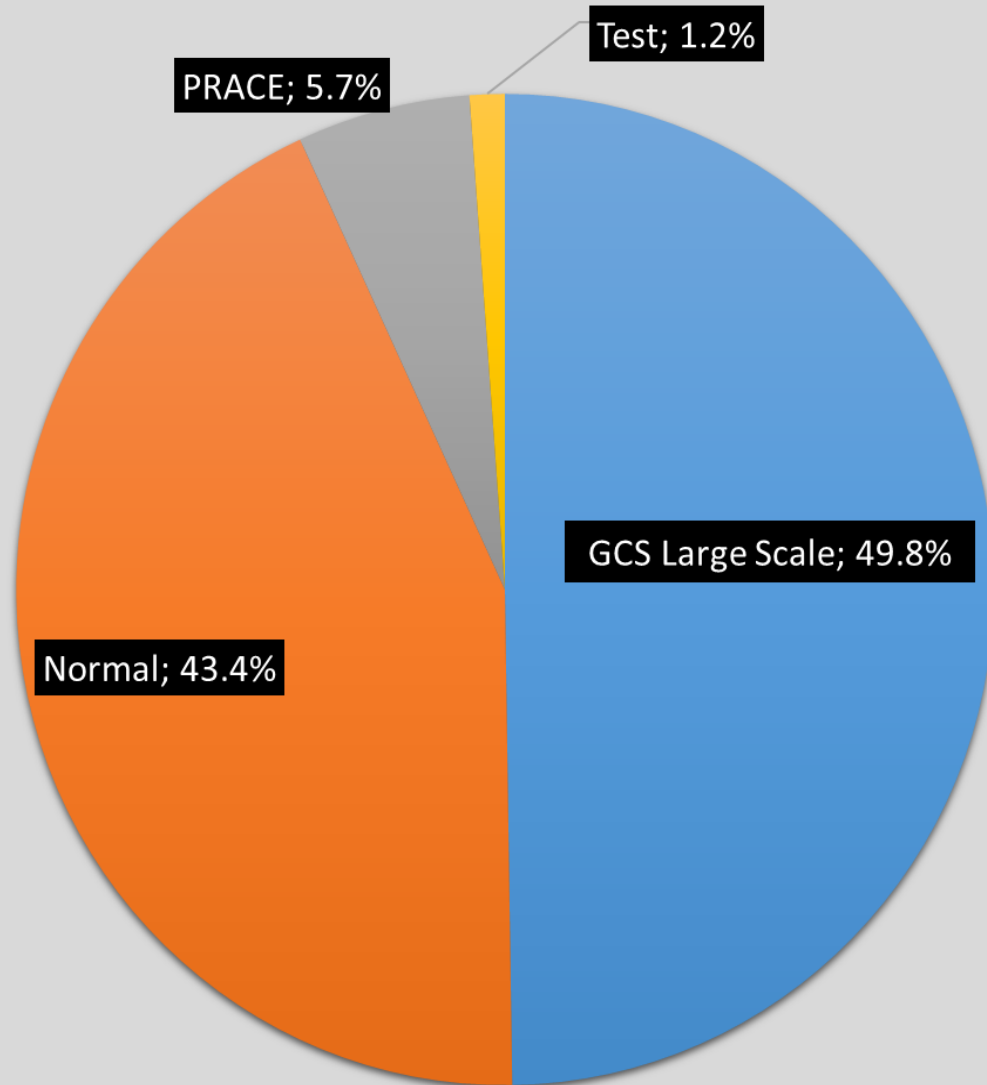


Coming 2023: SuperMUC-NG Phase 2



- ~240 Compute nodes
 - 2 Intel SapphireRapids CPUs
 - 4 Intel PonteVecchio accelerators
- *Distributed Asynchronous Object Storage* (DAOS) for AI and ML workloads (1PB)
- Programming: Intel OneAPI software stack
- Course series planned for fall 2022

Usage 2019/2020



up to 2.6 G core-h / year

main scientific areas

- 28% CFD / engeneering
- 20% elementary particle physics
- 18% astrophysics
- 8% condensed matter

System Access

Test Accounts and Compute Projects



Scientists in Europe are eligible, proposals for computing time are reviewed.

- Scientists with affiliation in Germany
<https://doku.lrz.de/display/PUBLIC/Access+and+Login+to+SuperMUC-NG>
- ~~European scientists apply via~~
<https://prace-ri.eu/hpc-access/preparatory-access/>
<https://prace-ri.eu/hpc-access/project-access/>

System used today: CoolMUC2

- Nodes were part of the SuperMUC Phase2 system installed in 2015
- Dual Intel Haswell nodes (2x14 cores)
- TIER-2 system.

Login onto the system via secure shell from a terminal window (Linux/macOS)

On Windows systems, install putty <https://www.putty.org/>

```
:~> ssh -Y lxlogin[1-4].lrz.de -l xxyyyzz
```

Please use login nodes for compiling but not for compute!

SLURM Batch System



Submit a job:

```
sbatch --reservation=hcow1s22 job.sh
```

List own jobs:

```
squeue -M cm2
```

Cancel jobs:

```
scancel -M cm2 jobid
```

Interactive Access:

```
module load salloc_conf/cm2_tiny
salloc --partition=cm2_tiny --time=00:30:00 --reservation=hcow1s22
  export OMP_NUM_THREADS=28
  srun --reservation=hcow1s22 ./myprog.exe
  exit
or: srun --reservation=hcow1s22 --pty bash
```

Reservation is only valid during the workshop, for general usage on our Linux Cluster remove the "--reservation=hcow1w21"

Details: <https://doku.lrz.de/display/PUBLIC/Running+parallel+jobs+on+the+Linux-Cluster>

Examples: <https://doku.lrz.de/display/PUBLIC/Example+parallel+job+scripts+on+the+Linux-Cluster>

Resource limits: <https://doku.lrz.de/display/PUBLIC/Resource+limits+for+parallel+jobs+on+Linux+Cluster>

Example OpenMP Batch File



```
#!/bin/bash
#SBATCH -o /dss/dsshome1/0D/hpckurs99/test.%j.%N.out
#SBATCH -D/dss/dsshome1/0D/hpckurs99
#SBATCH -J test
#SBATCH --clusters=cm2
#SBATCH --partition=cm2_std
#SBATCH --nodes=1
#SBATCH --qos=unlimitnodes
#SBATCH --cpus-per-task=28
#SBATCH --get-user-env
#SBATCH --reservation=hcowls22
#SBATCH --time=02:00:00
module load slurm_setup
export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
echo hello, world
```

- Used by most centres
- provide standard software environment by the computing centre
- modular installation of software
- provide multiple versions of a package/compiler/library
- central software installation

Which modules are loaded by default?

```
:~> module list
Currently Loaded Modulefiles:
 1) admin/1.0           5) intel-oneapi-compilers/2021.4.0
 2) tempdir/1.0        6) intel-mkl/2020
 3) lrz/1.0            7) intel-mpi/2019-intel
 4) spack/22.2.1
```

Good practise

Use `module list` in batch scripts and compile scripts to document run/compile conditions.

 helps debugging problems

Which modules are provided?

```
# complete list
:~> module available
/lrz/sys/spack/.../linux-sles15-skylake -
abinit/8.10.3-intel19-impi
adios/1.13.1-gcc8-impi
adios2/2.5.0-intel19-impi
bigmpi/0.1-intel19-impi
blitz/1.0.1-gcc8
...

# selected package
:~> module av likwid
-/lrz/sys/spack/.../linux-sles15-skylake -
likwid/4.3.3-gcc8-msr      likwid/5.0.1-gcc8-msr
likwid/4.3.3-intel19-msr  likwid/5.0.1-intel19-msr

----- /lrz/sys/.../files_sles15/tools -----
likwid/4.2  likwid/4.3  likwid/4.3-perf
```

Find module by keyword

```
:~> module search compiler
- /lrz/sys/spack/.../linux-sles15-x86_64 -
    gcc/8.4.0: Compilers:GNU compiler collection:GCC
    gcc/9.3.0: Compilers:GNU compiler collection:GCC
    gcc/9.3.0-nv: Compilers:GNU compiler collection:GCC
    intel/19.0.5: Compilers:HPC languages:Intel Fortran/C/C++
    intel/19.1.1: Compilers:HPC languages:Intel Fortran/C/C++
    llvm/8.0.0: compiler:clang:llvm
    llvm/9.0.0: compiler:clang:llvm
matlab-mcr/R2019a-generic: applications:scientific programming
    framework:MATLAB compiler runtime
matlab-mcr/R2019b-generic: applications:scientific programming
    framework:MATLAB compiler runtime>
perl/5.30.0: Compilers:tools:perl interpreter:scripting
```

What does a module do?

```
~> module show elpa
```

```
-----  
/lrz/sys/spack/release/21.1.1/modules/haswell/linux-sles15-haswell/elpa/2020.05.001-intel19-impi-openmp:  
  
setenv          ELPA_SPEC /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/.spack/spec.yaml  
conflict        elpa  
prepend-path    LD_LIBRARY_PATH /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/lib  
prepend-path    LIBRARY_PATH /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/lib  
prepend-path    C_INCLUDE_PATH /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/include  
prepend-path    INCLUDE /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/include  
prepend-path    PKG_CONFIG_PATH /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/lib/pkgconfig  
setenv          ELPA_LIB /.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/lib/libelpa_openmp.a  
setenv          ELPA_SHLIB {-L/.../21.1.1/opt/haswell/elpa/2020.05.001-intel-4pvo3pf/lib -lelpa_openmp}
```

prepend-path: extend path variable

setenv: provide environment variable

Hint: Use module-provided variables in makefiles, the command line and in shell scripts!

Loading and unloading

```
:~> module av gcc
--- /apps/modules/data/development -----
gcc/4.9.4 gcc/5.4.0 gcc/6.2.0 gcc/7.3.0 gcc/8.1.0

# load new module
:~> module load gcc
:~> module list
Currently Loaded Modulefiles:
  1) gcc/8.1.0
# change current module (unload & load)
:~> module switch gcc gcc/7.3.0
:~> module list
Currently Loaded Modulefiles:
  1) gcc/7.3.0
# unload the module
:~> module unload gcc
:~> module list
No Modulefiles Currently Loaded.
```

Aliases and defaults

```
:~> module alias
--- Aliases -----
intel-mkl/2019 -> intel-mkl/2019.5.281
intel-mkl/2019-gcc8 -> intel-mkl/2019.5.281
intel-mkl/2019-seq -> intel-mkl/2019.5.281
intel-mpi/2019-gcc -> intel-mpi/2019.7.217
intel-mpi/2019-intel -> intel-mpi/2019.7.217
intel/19.1 -> intel/19.1.1

----- Versions -----
admin/default -> admin/1.0
allinea-reports/default -> allinea-reports/18.3
amber/default -> amber/18
amira/default -> amira/2019.3
```

depends on the local module-file setup

How was the module build: `#{PACKAGE}_SPEC`



```
:~> module load elpa
:~> less $ELPA_SPEC
spec:
- elpa:
  version: 2020.05.001
  arch:
    platform: linux
    platform_os: sles15
    target:
      name: haswell
      vendor: GenuineIntel
  compiler:
    name: intel
    version: 19.0.5.281
  namespace: fixes015x
  parameters:
    openmp: true
    optflags: true
    cflags: []
    cppflags: []
    ...
```

```
dependencies:
intel-mkl:
  hash:
    3zuek252xqyrotkmaqbbzq7e3q6tyjv4h
  type:
    - build
    - link
intel-mpi:
  hash:
    xi6c5vue36rwmzhw4zqexs2yfymaho3z
  type:
    - build
    - link
  hash:
    4pvo3pf7eqkymzf7agava74m7wm5ptij
- intel-mkl:
  version: 2020.1.217
  arch:
    platform: linux
    platform_os: sles15
    target:
      name: haswell
```

the `.spec` file contains all relevant information for the dependency tree.

Environment Modules: Intel OneAPI Software Stack



```
:~> module av intel-oneapi
----- /lrz/sys/spack/release/22.2.1/modules/x86_64/linux-sles15-x86_64 -----
intel-oneapi-advisor/2021.4.0      intel-oneapi-inspector/2021.4.0  intel-oneapi-mkl/2022-gcc11
intel-oneapi-ccl/2021.4.0         intel-oneapi-ipp/2021.4.0        intel-oneapi-mkl/2022-seq
intel-oneapi-clck/2021.5.1        intel-oneapi-ippcp/2021.4.0      intel-oneapi-mkl/2022.0.2
intel-oneapi-compilers/2021.4.0   intel-oneapi-itac/2021.5.0      intel-oneapi-mkl/2022.1.0
intel-oneapi-compilers/2022.0.1   intel-oneapi-mkl/2021            intel-oneapi-mpi/2021-gcc
intel-oneapi-compilers/2022.1.0   intel-oneapi-mkl/2021-gcc11     intel-oneapi-mpi/2021-intel
intel-oneapi-dal/2021.4.0         intel-oneapi-mkl/2021-seq       intel-oneapi-tbb/2021.4.0
intel-oneapi-dnn/2021.4.0         intel-oneapi-mkl/2021.4.0       intel-oneapi-vpl/2021.6.0
intel-oneapi-dpcpp-ct/2022.0.0    intel-oneapi-mkl/2022           intel-oneapi-vtune/2021.7.1

----- /lrz/sys/share/modules/files_sles15/environment -----
intel-oneapi/2021.1  intel-oneapi/2021.2  intel-oneapi/2021.4
```

- Individual components for recent stacks (2022)
- Complete package for 2021 OneAPI stacks [not needed]

Tasks for the hands-on



Modules

1. Log on to the `lxlogin[1-4].lrz.de`
2. Which modules are automatically loaded?
3. Identify available compilers.
4. Why are module files for intel compilers in different directories?
5. Find the module of the latest Intel MKL version.

SLURM:

7. Write a bash "Hello World" script
8. Write a batch script to run it with SLURM.
9. Run in in interactive mode on a node.

