C A U

Kiel University
Christian-Albrechts-Universität zu Kiel

# Welcome to the virtual HPC introduction course

University Computing Centre (RZ)
13 May 2020

# Preliminaries

- Weblink to HPC related contents:
  - https://www.hiperf.rz.uni-kiel.de
- Slides will be available online after the meeting:
  - https://www.rz.uni-kiel.de/en/our-portfolio/hiperf/hpc-courses

## Email support (RZ ticket system)

- For further questions, queries, problem reports etc.
  - **hpcsupport@rz.uni-kiel.de**

# HPC support team

- **User consulting**
  - Dr. Simone Knief (chemistry*)
  - Dr. Michael Kisiela (bioinformatics*)
  - Dr. Karsten Balzer (physics*)
- **System administration**
  - Dr. Holger Naundorf (physics*)
  - Dr. Cebel Kücükkaraca (mathematics*)

*Educational/academic backgound

# Outline

## Introduction

- High performance computing (HPC)
  - What does HPC mean? How does HPC work?

## HPC systems @ RZ

- Overview
  - Linux cluster ("caucluster")
  - NEC HPC system
- How to get access?
- Login and file systems
- Software and batch processing

## HPC systems @ HLRN

- North-German Supercomputing Alliance
  - HLRN IV: Lise in Berlin, Emmy in Göttingen
- How to get access?

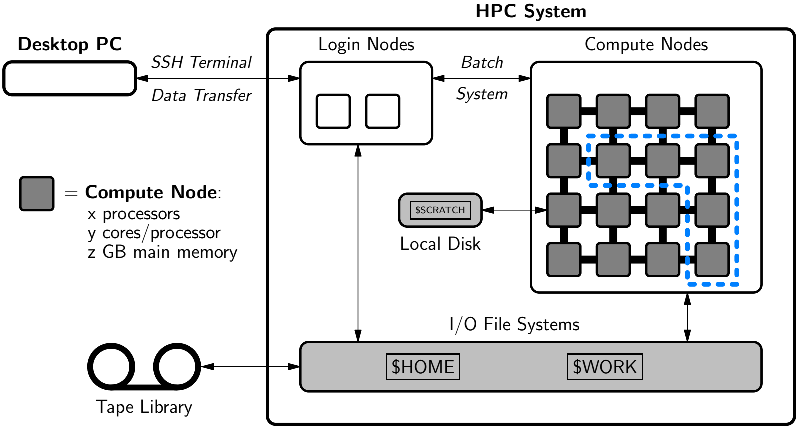
# Introduction

# What does HPC mean?

- **High performance computing**
  - = computer aided computation
  - Ability to process data and perform complex calculations at “high speeds”
  - Nowadays mainly synonymous with “parallel computing”
  - Due to the fact that typical HPC hardware allows for parallel computations
  - Even desktop PCs are nowadays multi-core platforms
- 3 typical ingredients for a **HPC machine**
  - *Compute:* Many nodes with multi-core processors
  - *Network:* Fast communication network between nodes
  - *Storage:* Powerful I/O file systems

# How does HPC work?

- **Typical machine layout**

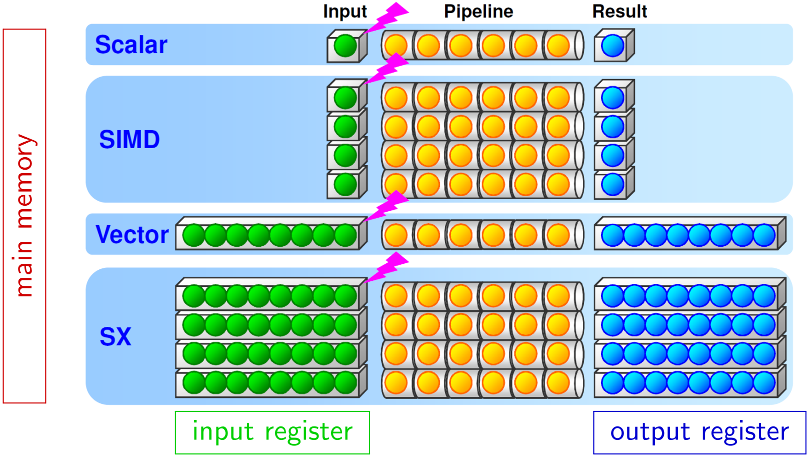


# Processor architectures

- **Scalar CPUs**
  - Linear execution of instructions, one instruction at a time
  - Each instruction operates on a single data item
  - Extensions for processing multiple instructions: e.g., SIMD or AVX
  - *Examples:* x86-based Intel/AMD processors

- **Vector CPUs**
  - Each instruction operates on a one-dimensional array (vector) instead of on a single data item
  - Thus a vector processor can work on an entire vector at a time
  - Realization: On-chip parallel processing
  - Suitable for vectorizable code, such as matrix-vector operations
  - *Examples:* graphics cards (GPUs: Nvidia Tesla, AMD Radeon), manycore processors (Intel Xeon Phi), special vector processors (NEC SX-processors)

# Scalar versus vector processors



# HPC systems @ RZ

# Systems

- **Linux cluster ("caucluster")**
  - *Available for all CAU/GEOMAR members*

- **Linux cluster ("medcluster")**
  - *Currently, reserved for IKMB users only*
- **Hybrid NEC HPC system**
  - **NEC x86-based Linux cluster**
  - **NEC SX-Aurora TSUBASA vector system**
  - *Available for all CAU/GEOMAR members*

# Linux cluster (“caucluster”), I

- **Hardware**
  - 2 *login nodes*: 16 cores, 128 GB main memory (*caucluster.rz.uni-kiel.de*)
  - ≈90 *compute nodes* with in total ≈1900 cores, different features:
    - 4× INTEL Sandy Bridge, 16 cores, 256 GB main memory
    - 36× INTEL Sandy Bridge, 16 cores, 128 GB main memory
    - 12× INTEL Sandy Bridge, 16 cores, 128 GB main memory
    - 4× INTEL Ivy Bridge, 16 cores, 64 GB main memory
    - 2× INTEL Ivy Bridge, 32 cores, 1024 GB main memory
    - 7× INTEL Haswell, 40 cores, 256 GB main memory
    - 4× INTEL Haswell, 32 cores, 256 GB main memory
    - 3× INTEL Haswell, 16 cores, 128 GB main memory
    - 5× INTEL Haswell, 16 cores, 256 GB main memory
    - 1× INTEL Broadwell, 24 cores, 1024 GB main memory
    - 1× INTEL Broadwell, 24 cores, 1024 GB main memory
    - 4× AMD Epyc (Naples), 32 cores, 256 GB main memory
    - 3× AMD Epyc (Rome), 64 cores, 512 GB main memory

# Linux cluster (“caucluster”), II

- **Operating system**
  - CentOS 7.x; *bash* as supported default Unix shell
- **File systems**
  - *Home file system*: 22 TB, user quota
  - *Work file system*: 350 TB BeeGFS storage, /work_beegfs, user quota
  - Access to magnetic tape data storage (*tape library*)
- **Batch system**
  - *SLURM* with fair-share scheduling
  - Fair-share ≠ first-in first-out
  - *Tracked resources*: core and memory usage

# NEC HPC system, I

- **Hybrid HPC system**, consisting of
  - A Scalar x86 Linux cluster
  - B SX-Aurora TSUBASA vector system
- **Uniform system environment** for A *and* B
  - 4 *login nodes*: 32 cores, 768 GB main memory (*nesh-fe.rz.uni-kiel.de*)
  - File systems
    - *Home file system*: 64 TB NEC GxFS, 2 partitions a 32 TB
    - *Work file system*: 5 PB = 5000 TB NEC ScaTeFS, 2 partitions a 2/3 PB
    - Access to magnetic tape data storage (*tape library*)
  - Operating system: Red Hat Enterprise Linux 7.x; *bash* as default Unix shell
  - Batch system: NQSV

# NEC HPC system, II

- **Compute hardware**

A Scalar x86 Linux cluster

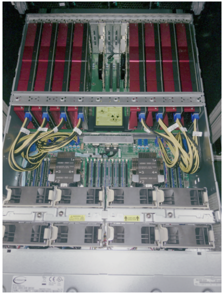
- 172 *compute nodes*: Intel Skylake, 32 cores each, 192 GB main memory
- 8 *compute nodes*: Intel Skylake, 32 cores each, 384 GB main memory
- 18 *compute nodes*: Intel Haswell, 24 cores each, 128 GB main memory
- Fast *EDR infiniband network* between nodes

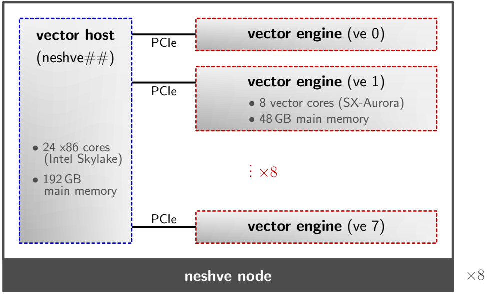
B SX-Aurora TSUBASA vector system

- 8 *compute nodes*, also called “vector hosts” (VH) by NEC
- Each compute node (A300-8, type 10B) allows for access to 8 *SX vector engines* (VE), which are attached via PCIe busses
- Infiniband network between nodes (i.e., between vector hosts)

# NEC HPC system, III

- SX-Aurora TSUBASA vector system
  - *Vector engine features*
    - 8 SX vector cores, 48 GB main memory
    - Clock freuency: 1.4 GHz
    - Vector length: 256×64 Bits (64 logical vector registers per core)
    - Memory bandwidth: 1.2 TB/s
  - Hybrid use of VE and VH possible (e.g., via offloading)





**vector host** (neshve##)

- 24 x86 cores (Intel Skylake)
- 192 GB main memory

PCIe

**vector engine** (ve 0)

PCIe

**vector engine** (ve 1)

- 8 vector cores (SX-Aurora)
- 48 GB main memory

⋮ ×8

PCIe

**vector engine** (ve 7)

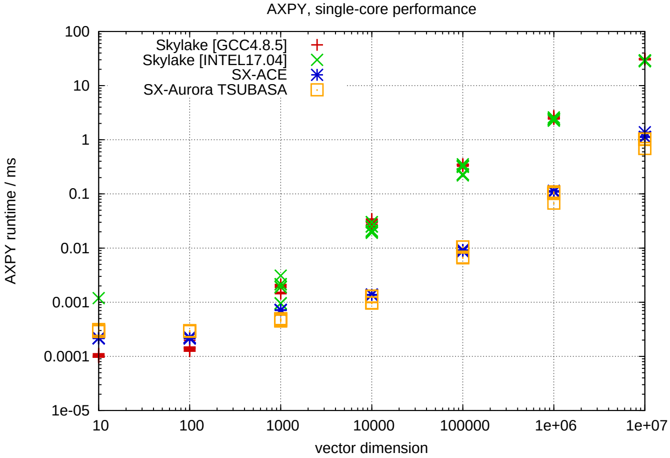
**neshve node**

×8

# NEC HPC system, IV

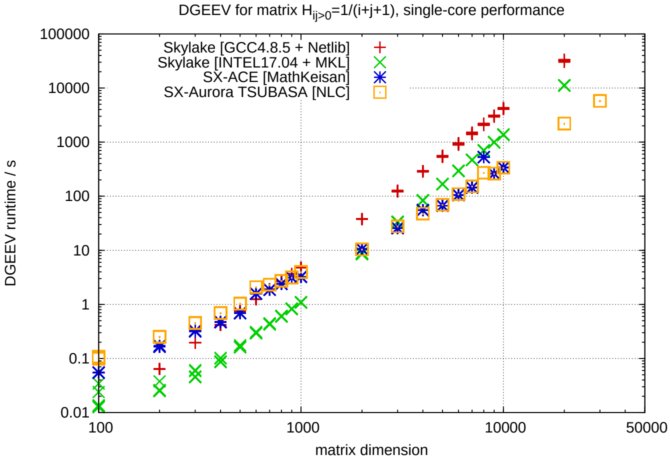
- *Performance*: Linux cluster versus SX-Aurora TSUBASA

  A Simple vector operation: $\mathbf{z} = a \cdot \mathbf{x} + \mathbf{y}$ with vectors $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$



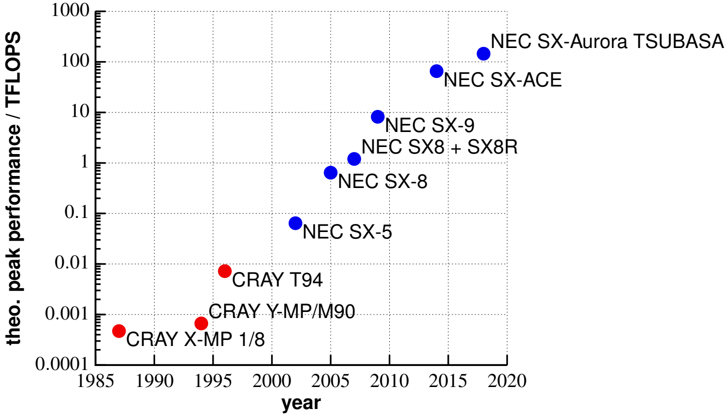
# NEC HPC system, V

- *Performance*: Linux cluster versus SX-Aurora TSUBASA
  - B Simple matrix diagonalization with a Lapack library routine

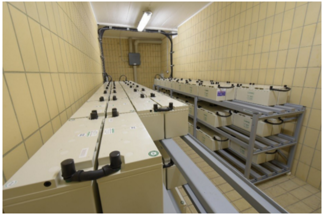


# NEC HPC system, VI

- *Vector performance @ RZ*: SX-Aurora TSUBASA and predecessor systems
  - Measured in TFLOPS = $10^{12}$ floating point operations per second



# Some impressions ...



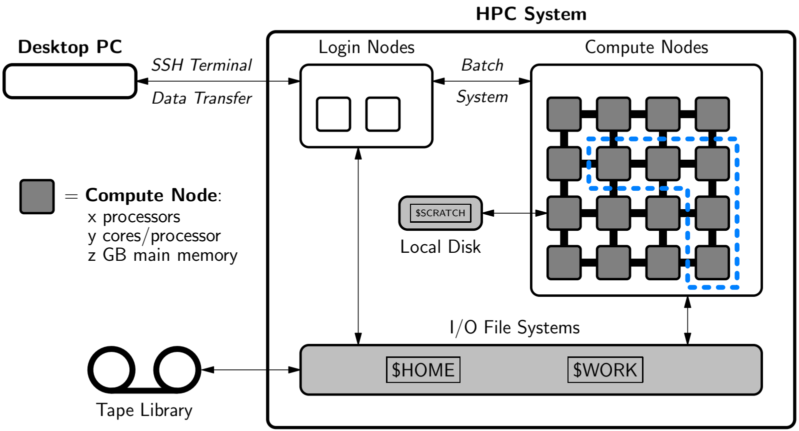
# How to get access?

- **Account application**
  - You can request access at any time
  - Use form no. 3*: "Request for use of a High-Performance Computer"
  - Submission procedure:
    - *Always:* Return completed form to RZ user administration (original form!)
    - *Home office:* An electronic version can be submitted by your project leader and/or directory to *anmeldung@rz.uni-kiel.de* with a remark such as "Yes, I agree with the application." Please, also state "In advance per e-mail" on the original form.
  - After successful registration, you will receive the login details in written form (NEC HPC system) or you will be able to set a password for the service "caucluster" in the CIM portal (caucluster)
  - User account comprises subscription to HPC mailing lists:
    - hpc_user@uni-kiel.de:
      General information: courses, events, new installations, ...
    - nesh_user@lists.uni-kiel.de resp. caucluster_user@lists.uni-kiel.de:
      HPC system specific information: downtimes, problems, new software, ...

* https://www.rz.uni-kiel.de/en/about-us/terms-and-policies/forms/sign-up

# Login, I

- **Typical machine layout**



# Login, II

- **Access to the login nodes**
  - Connect via SSH from within the networks of CAU and GEOMAR
  - *From outside:* Establish first a VPN connection to above networks (with RZ account!). GEOMAR users need to use their own VPN

- **How to establish an SSH connection?**
  - You need an SSH client
  - Linux and Mac:
    - Available through the standard distribution
  - Windows:
    - Putty (simple, no x11 window forwarding)

      https://www.putty.org
    - MobaXterm (preferred, versatile ssh client incl. x11 and sftp)

      https://mobaxterm.mobatek.net
    - X-Win 32 (alternative solution, campus license)

      https://www.rz.uni-kiel.de/de/angebote/software/x-win32

# Login, III

- **NEC HPC system** (**Linux cluster** and **SX-Aurora TSUBASA**)

```
ssh -X username@nesh-fe.rz.uni-kiel.de
```

  - -X option enables X11 forwarding
  - Command for password change: `passwd`

- **Linux cluster (“caucluster”)**

```
ssh -X username@caucluster.rz.uni-kiel.de
```

  - User access/password managed via CIM; go to: https://cim.rz.uni-kiel.de/cimportal

# Data transfer

- **Data transfer to/from the HPC system**
  - Linux and Mac:
    - Use shell commands like scp or rsync
  - Windows:
    - MobaXterm, sftp

      https://mobaxterm.mobatek.net
    - WinSCP, which is a gui-based scp client

      https://winscp.net
    - *Remark*: Windows-based editors generally put an extra "carriage return" character (control-M/^M) at the end of each line of text. This will cause problems for most Linux-based applications. Appearance of errors such as

      /bin/bash^M: bad interpreter: no such file or directory

      For correction, execute the following built-in utility on problematic files:

      ```
      dos2unix filename
      ```

# Available file systems, I

A *Home file system:* Contains the user's **HOME directory**

B *Work file system:* Contains the user's **WORK directory**

C *Magnetic tape storage:* Extra storage automatic file relocation onto magnetic tapes

D *Local disk space:* Temporary disk space on a compute node

# Available file systems, II

A *Home file system:* The user's **HOME directory**

- The directory after SSH login
- Accessible via the environment variable $HOME
- Total disk space (for all users)
  - **NEC HPC system**: 60 TB, no user quota
  - **Linux cluster ("caucluster")**: 22 TB, disk space user quota (defaults: 100 GB soft and 150 GB hard)
- Mounted on all compute nodes
- Regular backup (daily)
- Slow access times

- Suitable for important data which need a backup, e.g., software, programs, code, scripts, small amount of results
- **NOT** for the execution of production runs (batch jobs)!

# Available file systems, III

B *Work file system:* The user's **WORK directory**

- Directory on a high-performance, global parallel file system, **no backup!**
- Accessible via the environment variable $WORK
- Total disk space (for all users)
  - **NEC HPC system**: 5000 TB = 5 PB (ScaTeFS),
    disk space and inode user quota, defaults:
    - disk space limits
      soft: 1.8 TB (CAU), 4.5 TB (GEOMAR)
      hard: 2.0 TB (CAU), 5.0 TB (GEOMAR)
    - inodes limits: 225000 soft and 250000 hard
  - **Linux cluster ("caucluster")**: 350 TB (BeeGFS),
    disk space and chunk files user quota, defaults:
    - disk space limits: 1 TB
    - chunk files limits: 1000000
- Must be used for the execution of production runs (batch jobs)!
- Display quota usage/settings with command workquota

# Available file systems, IV

C *Magnetic tape storage:* Extra storage on tapes

- Looks like additional disk space, but files are automatically relocated onto magnetic tape in the background
- Accessibility:
  - **NEC HPC system**: Environment variable $TAPE_CACHE
  - **Linux cluster ("caucluster")**: Environment variable $TAPE
- Tape cache directory is only mounted on the login nodes (NEC HPC system: also available via batch queue "feque")
- Slow access times

- Use the tape library to store non-active data
- Transfer and store only archived data, e.g., tar-files (max. 1 TB, recommended: 3-50 GB)
- Do not store many small files on the tape library
- Never work interactively within the tape cache directory!
- Deleted data cannot be recovered (single copy, no archive system)

# Available file systems, V

D *Local disk space:* Temporary disk space for batch calculations

- Local disks directly on the compute nodes ($\approx$ 500 GB)
- Only temporarily available within a batch calculation
- Access via environment variables:
  - **NEC HPC system**: $TMPDIR after including the line

    ```
    export $TMPDIR="/scratch/"‘echo $PBS_JOBID | cut -f2 -d\:‘
    ```

    in the batch script
  - **Linux cluster (“caucluster”)**: $TMPDIR
- Advantage:
  - Fast access times
  - Typically faster I/O for read- and write-intensive computations

# Data protection and integrity

- **Some remarks**:
    - *HOME directory*: Regular backup (daily)
        - Backup history comprises the last 8 weeks
        - For last 2 week, files can be recovered on a daily basis, prior to that on a weekly basis
    - *WORK directory* and *tape library*: **No backup!**
    - User data is only accessible from an active account!
    - The computing center does currently not provide a long-term data archiving service!
    - Check your data stock from time to time!

# Software, I

- There is a whole bunch of user software available:
  - **Standard user software**
    - *Examples:* Python, Perl, R, Matlab, Octave, Gaussian, Turbomole, Quantum Espresso, Gnuplot, Xmgrace, ...
  - **Compilers**
    - GNU compilers
    - INTEL compilers
    - NEC cross compilers for using the SX-Aurora TSUBASA vector system
  - **MPI environment**
    - Intel MPI (all Linux clusters)
    - Special SX MPI (NEC SX-Aurora vector system)
  - **Libraries**
    - *Examples:* NetCDF, HDF5, MKL, PETSc, GSL, Boost, Eigen, ...
  - **Editors**
    - *Examples:* nedit, emacs, nano, vi, mc

# Software, II

- **Software deployment**
  - Most user software is provided via so called module files, which means that software is **not** in the standard path
  - Instead, software is activated on a user request by loading a specific module file
  - This allows for an easy deployment of software and, in particular, for the installation of multiple versions of one and the same software
  - *Important module file commands:*

| Command | Explanation |
|---|---|
| module avail | Shows all available modules |
| module load *name* | Loads the module *name* and performs all required settings |
| module list | Lists all modules which are currently loaded |
| module unload *name* | Removes the module *name*, i.e., resets all corresponding settings |
| module purge | Removes all currently loaded modules (module list becomes empty) |
| module show *name* | Displays the settings which are performed by the module |

# Software, III

- **Matlab**
  - *For CAU users:*

```
module load matlab2018b
matlab
matlab -nodisplay
```

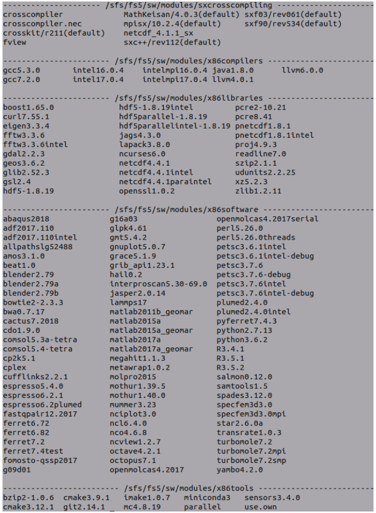
  - *For GEOMAR users (special [toolbox] licenses):*

```
module load matlab2018b_geomar
```

- **Intel compilers**

```
module load intel17.0.4
ifort ...
icc ...
icpc ...
```

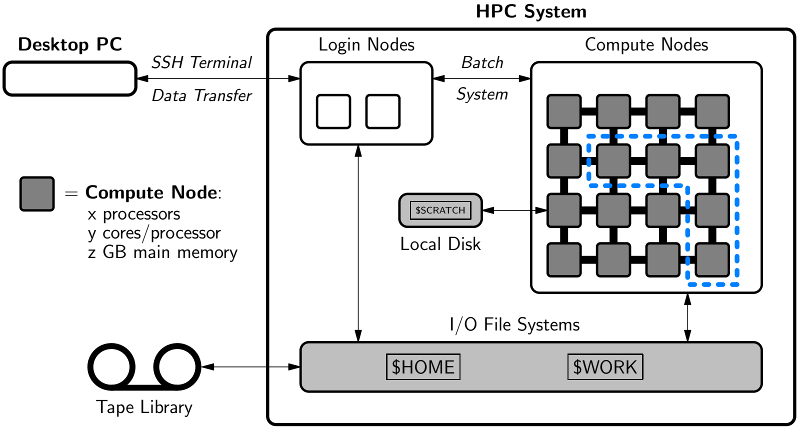
```
module load intel17.0.4 intelmpi17.0.4
mpiifort ...
mpiicc ...
mpiicpc ...
```



```
--------------------- /sfs/fs5/sw/modules/sxcrosscompiling ---------------------
crosscompiler              MathKeisan/4.0.3(default) sxf03/rev061(default)
crosscompiler.nec          mpisx/10.2.4(default)     sxf90/rev534(default)
crosskit/r211(default)     netcdf_4.1.1_sx
fview                      sxc++/rev112(default)

----------------------- /sfs/fs5/sw/modules/x86compilers -----------------------
gcc5.3.0       intel16.0.4    intelmpi16.0.4 java1.8.0      llvm6.0.0
gcc7.2.0       intel17.0.4    intelmpi17.0.4 llvm4.0.1

----------------------- /sfs/fs5/sw/modules/x86libraries -----------------------
boost1.65.0               hdf5-1.8.19intel          pcre2-10.21
curl7.55.1                hdf5parallel-1.8.19       pcre8.41
eigen3.3.4                hdf5parallelintel-1.8.19 pnetcdf1.8.1
fftw3.3.6                 jags4.3.0                 pnetcdf1.8.1intel
fftw3.3.6intel            lapack3.8.0               proj4.9.3
gdal2.2.3                 ncurses6.0                readline7.0
geos3.6.2                 netcdf4.4.1               szip2.1.1
glib2.52.3                netcdf4.4.1intel          udunits2.2.25
gsl2.4                    netcdf4.4.1paraintel      xz5.2.3
hdf5-1.8.19               openssl1.0.2              zlib1.2.11

----------------------- /sfs/fs5/sw/modules/x86software ------------------------
abaqus2018              g16a03                  openmolcas4.2017serial
adf2017.110             glpk4.61                perl5.26.0
adf2017.110intel        gmt5.4.2                perl5.26.0threads
allpathslg52488         gnuplot5.0.7            petsc3.6.1intel
amos3.1.0               grace5.1.9              petsc3.6.1intel-debug
beat1.0                 grib_api1.23.1          petsc3.7.6
blender2.79             hail0.2                 petsc3.7.6-debug
blender2.79a            interproscan5.30-69.0   petsc3.7.6intel
blender2.79b            jasper2.0.14            petsc3.7.6intel-debug
bowtie2-2.3.3           lammps17                plumed2.4.0
bwa0.7.17               matlab2011b_geomar      plumed2.4.0intel
cactus7.2018            matlab2015a             pyferret7.4.3
cdo1.9.0                matlab2015a_geomar      python2.7.13
comsol5.3a-tetra        matlab2017a             python3.6.2
comsol5.4-tetra         matlab2017a_geomar      R3.4.1
cp2k5.1                 megahit1.1.3            R3.5.1
cplex                   metawrap1.0.2           R3.5.2
cufflinks2.2.1          molpro2015              salmon0.12.0
espresso5.4.0           mothur1.39.5            samtools1.5
espresso6.2.1           mothur1.40.0            spades3.12.0
espresso6.2plumed       mummer3.23              specfem3d3.0
fastqpair12.2017        nciplot3.0              specfem3d3.0mpi
ferret6.72              ncl6.4.0                star2.6.0a
ferret6.82              nco4.6.8                transrate1.0.3
ferret7.2               ncview1.2.7             turbomole7.2
ferret7.4test           octave4.2.1             turbomole7.2mpi
fomosto-qssp2017        octopus7.1              turbomole7.2smp
g09d01                  openmolcas4.2017        yambo4.2.0

------------------------- /sfs/fs5/sw/modules/x86tools -------------------------
bzip2-1.0.6  cmake3.9.1   imake1.0.7   miniconda3   sensors3.4.0
cmake3.12.1  git2.14.1 _  mc4.8.19     parallel     use.own
```

# Batch processing, I

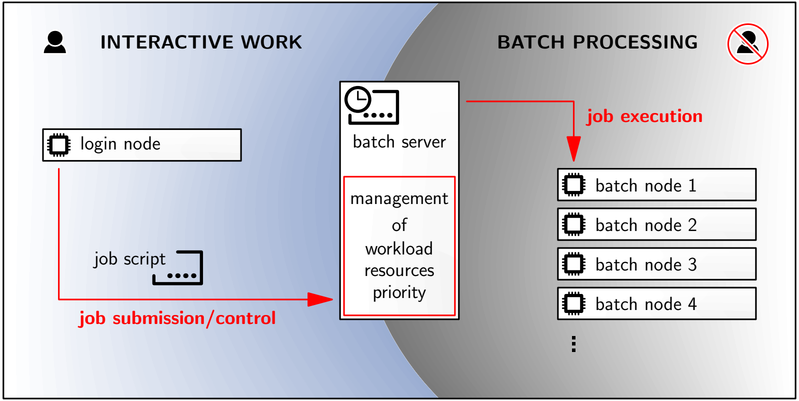
- **Typical machine layout**



# Batch processing, II

- **Batch processing = Running calculations (jobs) on the compute node**
- **How does this work?**
  - One does not start the job from the command line
  - Instead:
    1. Prepare a small ASCII file (called *batch* or *job script*) which contains all necessary information
    2. Submit this script to the *batch system*
  - Typical information provided in a batch script:
    - Required compute resources (# nodes, # cores/node, main memory, walltime, ...)
    - Software environment (module load ..., if required)
    - Program call

# Batch processing, III



- Batch server takes the job script, searches for free, appropriate compute resources and then executes the actual computation or queues the job.

# Batch processing, IV

- **Advantages of batch processing**
  - *Operator perspective*:
    - Allows for a very efficient use of the available compute resources
    - Increases throughput and leads to a good overall utilization
    - ...
  - *User perspective*:
    - Allows for a fair distribution of resources
    - Every user can execute multiple jobs in parallel
    - Presence of requested resources is ensured during a calculation
    - Possibility to set up job dependencies or entire workflows
    - ...

# Batch processing, V

- **How to set up a batch script?**
  - Different syntax and commands for different batch systems
    - **NEC HPC system**: NQSV
    - **Linux cluster (“caucluster”)**: SLURM
  - Different forms for different types of calculations
    - Serial calculation
    - Using multiple cores on a single node (e.g., via OpenMP threads)
    - Using multiple nodes (e.g., via MPI)
    - Hybrid schemes (e.g., MPI+OpenMP)
  - *Remark for NEC HPC system:* For multinode MPI calculations, batch nodes need to communicate without password
    - To this end, generate a key pair once (without setting a passphrase!):

```
ssh-keygen -t rsa # answer all queries just with ENTER
cp $HOME/.ssh/id rsa.pub $HOME/.ssh/authorized_keys
```

# Batch script examples, I

- NEC HPC Linux cluster, NQSV: *A serial calculation*

```
#!/bin/bash
#PBS -b 1
#PBS -l cpunum_job=1
#PBS -l elapstim_req=01:00:00
#PBS -l memsz_job=20gb
#PBS -N test
#PBS -o test.out
#PBS -j o
#PBS -q clmedium

# Change into qsub directory
cd $PBS_O_WORKDIR

# Start of the computation
module load intel17.0.4
time ./program.x

# Output of used resources (computation time, main memory) after the job
qstat -f ${PBS_JOBID/0:}
```

# Batch script examples, II

- NEC HPC Linux cluster, NQSV: *A parallel, multinode MPI calculation*

```
#!/bin/bash
#PBS -T intmpi
#PBS -b 4
#PBS -l cpunum_job=32
#PBS -l elapstim_req=10:00:00
#PBS -l memsz_job=256gb
#PBS -N test
#PBS -o test.out
#PBS -j o
#PBS -q clbigmem

# Change into qsub directory
cd $PBS_O_WORKDIR

# Start of the computation
module load intel17.0.4 intelmpi17.0.4
time mpirun $NQSII_MPIOPTS -np 128 ./program.x

# Output of used resources (computation time, main memory) after the job
qstat -f ${PBS_JOBID/0:}
```

# Batch script examples, III

- NEC SX-Aurora TSUBASA, NQSV: *A single-host MPI calculation*

```
#!/bin/bash
#PBS -T necmpi
#PBS -b 1
#PBS --venum_lhost=4
#PBS -l elapstim_req=01:00:00
#PBS -N test
#PBS -o test.out
#PBS -j o
#PBS -q vequeue

# Change into qsub directory
cd $PBS_O_WORKDIR

# Display performance summary (compile with -proginf flag)
export VE_PROGINF=DETAIL

# Start of the computation
mpirun -nn 1 -nnp 32 -ve 0-3 ./program.sx-aurora
```

# Batch script examples, IV

- Linux cluster ("caucluster"), SLURM: *An OpenMP parallel calculation*

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=4
#SBATCH --mem=1000
#SBATCH --time=01:00:00
#SBATCH --job-name=test
#SBATCH --output=test.out
#SBATCH --error=test.err
#SBATCH --partition=all

export OMP_NUM_THREADS=4

module load intel/18.0.4
time ./program.x
```

# Batch script examples, V

- Linux cluster ("caucluster"), SLURM: *A multinode MPI+OpenMP calculation*

```
#!/bin/bash
#SBATCH --nodes=2
#SBATCH --tasks-per-node=8
#SBATCH --cpus-per-task=4
#SBATCH --mem=1000
#SBATCH --time=01:00:00
#SBATCH --job-name=test
#SBATCH --output=test.out
#SBATCH --error=test.err
#SBATCH --partition=all

export OMP_NUM_THREADS=4

module load intel/18.0.4 intelmpi/18.0.4
time mpirun -np 16 ./program.x
```

# Batch processing, VI

- **Job submission and control**

| NQSV | SLURM | |
|---|---|---|
| qsub *jobscript* | sbatch *jobscript* | Submission of a new batch job |
| qstatall | squeue | List all jobs currently in the system |
| qstat | squeue -u *username* | List only the own jobs |
| qdel *jobid* | scancel *jobid* | Delete or terminate a batch job |
| qstat -f *jobid* | scontrol show job *jobid* | Show details of a specific job |
| qcl | sinfo -Nl | Get information about queues/partitions |
| qstat -J *jobid* | | Lists on which nodes the job is running |

For more informations, see system documentation web pages

# Batch processing, VII

- **Appropriate use of the batch system**
  - **Do not** request more nodes and more cores than are required by the computation
  - **Adapt** the walltime and main memory to the need of the program
    - Note, that a more accurate resource specification can lead to smaller waiting times and increased throughput
    - But do not plan too restrictive!
  - Try to **save intermediate results**
    - Particularly during longer calculations
    - Check if the program has a restart option
  - Finally, the **stdout** of a batch job should be kept small; maybe redirect it to a file on the local disk or on the work directory

# Interactive work, I

- **Interactive work = Working on the login nodes**
  - This usually covers the following activities:
    - Data transfer (desktop PC ↔ HPC systems ↔ tape library)
    - Software provisioning and installation
    - Program development and compilation
    - Preparation of batch scripts and batch jobs
    - Submission and control of batch jobs
    - Small pre- and postprocessing
  - **Do not** start long and recource demanding calculations on the login node!
  - **Be careful** with test calculations

    ⇒ Monitor CPU and memory consumption, as well as runtime!

    ⇒ Simple, useful command: `top` and `top -n 1 -b | grep` *username*

# Interactive work, II

- **Interactive batch jobs**
  - One can request an **interactive session on a compute node**
  - This may be useful for test calculations (*keeping the load on the login node small*) or for calculations that neccessarily need some user interaction during runtime
  - **How does it work?**
    - Linux cluster ("caucluster"), SLURM:

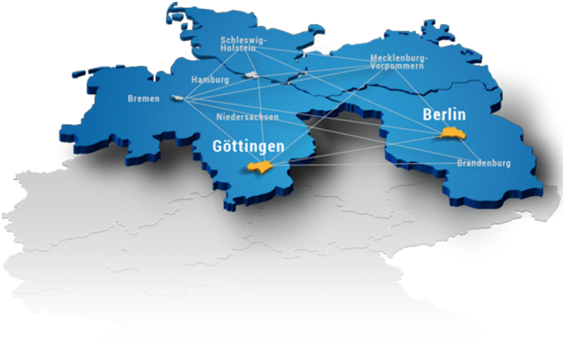
      ```
      srun –x11 –pty –cpus-per-task=4 –time=00:30:00 –partition=all /bin/bash
      ```

    - NEC HPC system, NQSV:

      ```
      qlogin -X -q clinteractive -l cpunum_job=2 -l elapstim_req=01:00:00
      qlogin -q veinteractive -l cpunum_jobs=1 -l elapstim_req=01:00:00
      ```

# HPC systems @ HLRN

# HLRN: North-German Supercomputing Alliance



HLRN

- https://www.hlrn.de

# HLRN: North-German Supercomputing Alliance

- German: Norddeutscher Verbund für Hoch- und Höchstleistungsrechnen (**H**och**l**eistungs**r**echner **N**ord, HLRN)
- Component of the national HPC infrastructure; formed by seven North-German states: Berlin, Brandenburg, Bremen, Hamburg, Mecklenburg-Vorpommern, Niedersachsen and Schleswig-Holstein

HLRN
NORDDEUTSCHER VERBUND FÜR HOCH- UND HÖCHSTLEISTUNGSRECHNEN

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

- In operation since 2001
- Aims:
  - Joint procurement and operation of a Tier-2 HPC centre
  - Establishment and development of a HPC competence network

# HLRN: 2 site locations

Schleswig-Holstein
Mecklenburg-Vorpommern
Hamburg
Bremen
Niedersachsen
Berlin
Göttingen
Brandenburg

- Zuse Institute Berlin (ZIB)
- University of Göttingen, since 09/2018 (former site: University Hannover)

- Current expansion stage:

  *ZIB*: **HLRN IV (Lise)**

  *Göttingen*: **HLRN IV phase 1 (Emmy)**

- Full expansion stage of HLRN IV: 16 PFLOPS, about 244000 cores (Intel-CPUs) over both sites

# HLRN: Currently available resources

- **Lise at ZIB**:
  - The HLRN complex in Berlin at ZIB is named after *Lise Meitner*
  - 1146 compute nodes (with 110,016 compute cores):
    - 1112 nodes with 384 GB main memory (standard node)
    - 32 nodes with 768 GB main memory (large node)
    - 2 nodes with 1.5 TB main memory (huge node)
    - 96 cores per node (Intel Cascade Lake) + Omni-Path

- **Emmy in Göttingen**:
  - The HLRN phase 1 complex at Göttingen University is named after *Emmy Noether*
  - 448 compute nodes
    - 432 nodes with 192 GB main memory
    - 16 nodes with 768 GB main memory
    - 40 cores per node (Intel Skylake) + Omni-Path
  - 1 GPU node
    - 40 cores (Intel Skylake), 192 GB main memory
    - 4× NVIDIA Tesla V100 32 GB main memory

# HLRN: Account application

- Open for all university employees in S.-H., incl. universities of applied sciences
- **Step I. Test account**
  - Apply at any time; duration 3 quarters (= 9 months)
  - Limited computing time: up to 40000 NPL per quarter
    - NPL = "Norddeutsche Parallelrechner Leistungseinheit"
      Depending on node type: 6 - 28 NPL per hour and node
- **Step II. Project proposal (Großprojektantrag)**
  - Submission deadlines: **28.01.**, **28.04.**, **28.07.**, **28.10.**
  - Duration: Total NPL are granted for 1 year and distributed on a quaterly basis
  - Proposals will be assessed by the scientific council of HLRN
  - Prerequisite: a valid HLRN (test) user account
  - Minimum demands for a project proposal:
    - Project abstract
    - project description or status report (for project extension)
    - Justification of requested NPL, other resource demands (main memory, disk space)

# Summary

- **HPC services of the CAU Computing Centre (RZ)**
  - *Provisioning of compute resources of different performance classes:*
    - **Local HPC systems @ RZ**
    - **Regional compute resources @ HLRN**
  - *User consulting:*
    - Choice of appropriate computer architecture (x86/vector, RZ/HLRN)
    - Support on software installation and porting
    - Support on program and code optimization
    - Support on parallelization and vectorization issues
    - Support on HLRN project proposals

# Documentation

## HPC web pages

- https://www.hiperf.rz.uni-kiel.de
- https://www.rz.uni-kiel.de/en/our-portfolio/hiperf
- System-specific documentation:
    - Hard- and software
    - File systems
    - Batch system
- Course materials

## Email support (RZ ticket system)

- For further questions, queries, problem reports etc.
    - **hpcsupport@rz.uni-kiel.de**

# Thank you for your attention and stay healthy!

- Your HPC support team -