CHEOPS

Cologne High Efficient Operating Platform for Science

Brief Instructions

(Version: 19.07.2022)

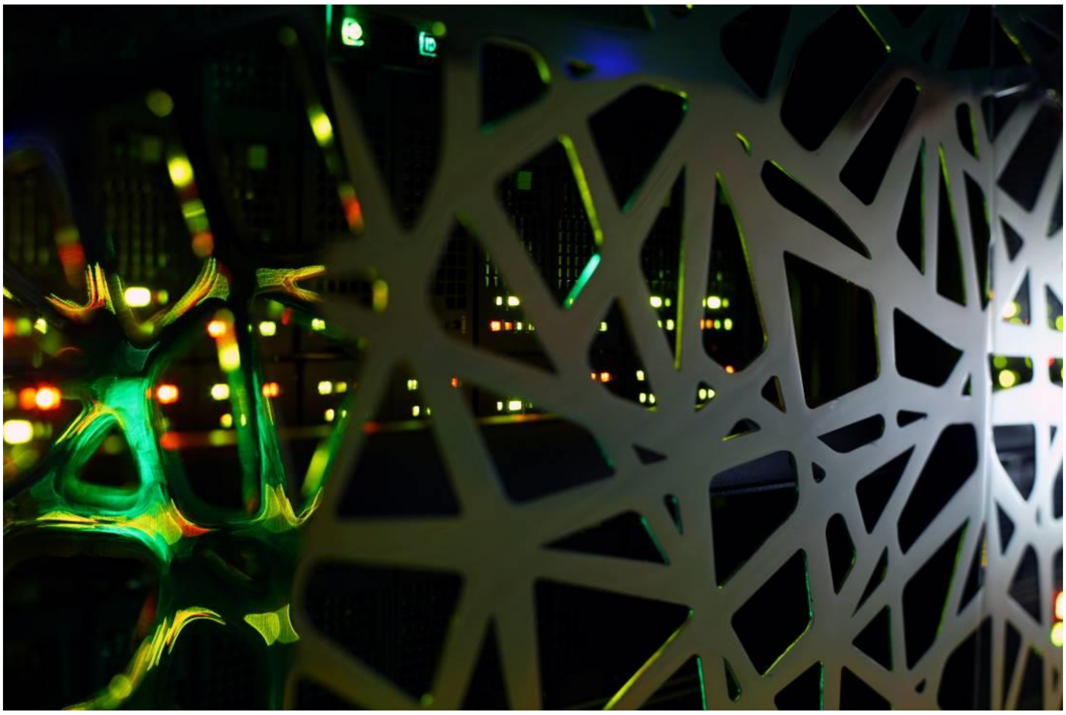

Foto: Thomas Josek/JosekDesign

Viktor Achter
Dr. Stefan Borowski
Lech Nieroda
Dr. Lars Packschies
Volker Winkelmann

HPC team: hpc-mgr@uni-koeln.de
Scientific support: wiss-anwendung@uni-koeln.de

Contents

1 Scope of this document

This document introduces to the usage of the CHEOPS cluster, which includes the overview of the development software environment. For information on the application software, please see the document “CHEOPS Application Software”.

2 Contacts for support

HPC team, hpc-mgr@uni-koeln.de:
Operation of HPC systems, parallel computing, batch system, development software

Scientific support, wiss-anwendung@uni-koeln.de:
Scientific applications, scientific computing, access to HPC systems

If you send a support request to one of these contacts, please provide all relevant information to describe the issue you encountered. First, error messages are crucial for analysis and should be provided with the request. Such messages are usually printed to standard error (`stderr`) or standard output (`stdout`) streams. Depending on what you are doing, either you will see these messages on the screen or the streams are redirected into files. Moreover, accompanying information is very helpful to shorten analysis. For instance, if the batch system fails to run a job, you should always provide the job identifier with your report. If building of an application fails with an error, you should provide name and version of compiler and libraries used.

3 Description of the system

CHEOPS is a high performance computing cluster for science, which is provided by the Regional Computing Centre of Cologne (RRZK) together with the HPC expert Bull.

Foto: Thomas Josek/JosekDesign

The complete cluster is in production since the end of 2010. With more than 100 TFlop/s peak and 85.9 TFlop/s Linpack performance, it was ranked to be the 89th fastest supercomputer worldwide in the Top500 list from November 2010 (www.top500.org).

CHEOPS is an InfiniBand coupled HPC cluster with dual socket INCA compute nodes:

- 215 x 2 Nehalem EP quad-core processors (Xeon X5550, 2.66 GHz), 24 GB RAM
- 432 x 2 Westmere hexa-core processors (Xeon X5650, 2.66 GHz), 24 GB RAM
- 170 x 2 Westmere hexa-core processors (Xeon X5650, 2.66 GHz), 48 GB RAM

and quad socket MESCA compute nodes:

- 24 x 4 Nehalem EX octo-core processors (Xeon X7560, 2.27 GHz), 512 GB RAM

The node interconnect is based on InfiniBand QDR, which is a broadband bus technology with exceptionally small latency. It was specifically developed for HPC clusters and delivers a gross bandwidth of 40 Gb/s and latencies of less than 5 µs. The INCA compute nodes provide two Intel Nehalem EP quad-core processors (Xeon X5550) and Intel Westmere hexa-core processors (Xeon X5650), respectively. The MESCA nodes are equipped with four Intel Nehalem EX octo-core processors (Xeon X7560). Intel's Nehalem (and Westmere) architecture stands for high memory bandwidth and a fast interconnect between cores as well as processors, which perfectly matches the demands of HPC.
CHEOPS uses the General Parallel File System (GPFS) as parallel file systems. GPFS is based on a DDN GRIDScaler GS7K appliance providing a gross capacity of 2.8 PB (2.1 PB net) and a Dell EMC storage delivering a gross capacity of 484 TB (363 TB net).

3.1 Who may use this service?

Scientists from NRW may use CHEOPS for scientific, non-commercial purposes. This service is free of charge. We kindly ask all users to acknowledge the usage of CHEOPS in their publications by a statement: We furthermore thank the Regional Computing Center of the University of Cologne (RRZK) for providing computing time on the DFG-funded (Funding number: INST 216/512/1FUGG) High Performance Computing (HPC) system CHEOPS as well as support.

3.2 How do I gain access?

To gain access to CHEOPS, you need an existing user account of the University of Cologne and HPC authorization. The user account enables you to use common services offered by the RRZK. The HPC authorization additionally enables access to the HPC systems. If you need help with completing the forms, you may ask for support at the RRZK-Helpdesk or the scientific support.

3.2.1 How to get a user account

Students and employees of the University of Cologne already have a student (smail) and employee account, respectively. Members from other universities of the federal state Nordrhein-Westfalen may apply for a guest account using the guest account form.
With your user account there comes an email account username@(smail.)uni-koeln.de. Please retrieve emails from this account to receive important messages (e.g. maintenance of the cluster, expiration of your account). If you are using a different email account, you should set an according forward on the mail portal.

3.2.2 How to get HPC authorization

If you have your user account, you need to send the completed HPC authorization form to the scientific support. With HPC authorization granted, you also get an account on CHEOPS.
If you only need a small amount of computing time, e.g. your project is in an early phase and you only want to test your application, you may apply for a test account. It will be limited to a certain amount of computing time (e.g. 1000 CPU hours). For a test account, there is no need for a project description.
If you need more computing time, you may apply for a full account, which requires a detailed project description. Of course, you can apply for a full account right away, if you are already set to go and have a clear description of your project.
In both cases, please use the HPC authorization form.

Links:
http://rrzk.uni-koeln.de/autorisierung-hpc.html?&L=1
https://mailportal.uni-koeln.de

3.3 Server addresses

NOTE: For the time being, we have two operating system installations on CHEOPS, CentOS7 (the new one) accessible via **cheops1.rrz.uni-koeln.de** and RHEL6 (the old one) accessible via **cheops0.rrz.uni-koeln.de**. The default **cheops.rrz.uni-koeln.de** refers to **cheops1.rrz.uni-koeln.de**. Please, start your workflow on the new CentOS7 side of the cluster.

Function	Address	Protocols	Status
Login Node	cheops.rrz.uni-koeln.de	ssh	OK
Monitoring/Ganglia	https://cheops-ohpc-ganglia.rrz.uni-koeln.de/	http	OK

All Nodes are accessible within the university network (UKLAN). If you are outside this network, you may login through the server **dialog.rrz.uni-koeln.de** and from there access CHEOPS via Secure Shell (SSH) with trusted X11 forwarding:

```
ssh -Y userid@dialog.rrz.uni-koeln.de
ssh -Y userid@cheops.rrz.uni-koeln.de
```

Alternatively, you may connect to UKLAN via a Virtual Private Network (VPN) client. To do this, follow the instructions for VPN access on the RRZK webpage. You will then be virtually integrated into the UKLAN and will be able to directly connect to all services.

Links:
http://rrzk.uni-koeln.de/vpn.html?&L=1

3.4 Structure of the file system

As our users are used to, the parallel storage based on GPFS is organized into three file systems with differing bandwidth. The following table gives an overview:

File system /location	Storage	Capacity	Quota per user	Speed	Backup/archive

/home	GPFS	22 TiB (24 TB)	100 GB, 100,000 files	Low	Daily
/projects	GPFS	1.9 PiB (2.1 PB)	As requested	Medium - low	Archive must be requested (elsewise none)
/scratch	GPFS	330 TiB (363 TB)	1,000,000 files	High	None (data will be deleted after 30 days)

NOTE: Parallel file systems are not generally known to be the most stable files systems. Please make sure that important data in **/projects** is archived (see Section 3.6).

As can be seen in the table, the number of files in home and scratch directories is limited by separate quotas. The reason for this is that the amount of files has severe impact on the performance of a parallel file system as well as on backup and cleanup. If you are getting in trouble because of this, please contact the RRZK scientific support or HPC team.

3.4.1 Quotas

On GPFS, there are quotas. Usage or block quota limits the amount of data (e.g. 100GB). File or inode quota limits the number of files (e.g. 100,000). To see the quota status of GPFS file systems **/home**, **/projects** as well as **/scratch** use the shell script command

```
lsquota
```

we have provided for summarizing all quotas.

3.5 What you should do on first login

If you access CHEOPS for the first time via SSH, a key pair is generated for the access to the compute nodes. Please do **not** provide a password for the keys. Accessing the compute nodes has to work without a password. After this, you are set to go!

3.6 How to archive data

To archive data in your project directory, you first need a TSM registration for archiving. With the registration, you are provided a **nodename** referring to your archive space. Then, you can access this archive space with the TSM client on the login node **cheops.rrz.uni-koeln.de**. Before archiving of data you should create a personal configuration file **dsm.opt** with the following content:

```
servername      adsm4n
virtualnode     nodename
subdir          yes
```

The server for archiving is named **adsm4n**. As virtual node, you should provide your personal **nodename**. For simple data management, subdirectories are to be included. The TSM client can be launched with the command line user interface **dsmc** or the graphical user interface **dsmj**. For archiving, you should use the command line client:

```
dsmc archive /projects/project/tobearchived/ \
-des=archivename -optfile=/path_to_dsm.opt/dsm.opt
```

The trailing slash on the directory is crucial for **dsmc** to recognize: it is a directory not a file. The archive name **archivename** will help you finding data to be retrieved from a specific archive. For retrieving data, you should use the graphical client

```
dsmj -optfile=/path_to_dsm.opt/dsm.opt
```

Within the graphical user interface, you can browse the content of your archives for data to be retrieved. For both command and graphical user interface, an absolute pathname of **dsm.opt** is needed (with slash in front).

NOTE: The purpose of an archive is different from that of a backup. A backup saves frequently changing data. However, an archive saves data that does not change anymore (e.g. results). Therefore, you should not archive your whole project directory but finished subdirectories. You can retrieve this data on any computer with a TSM client (e.g. workstation or laptop) for further processing.

Links:
http://rrzk.uni-koeln.de/tsm.html
https://www.ibm.com/docs/en/spectrum-protect/8.1.14?topic=clients-archive-retrieve-your-data-unix-linux

4 Environment modules

CHEOPS provides a wide range of development software (compilers, libraries, debuggers, profilers, etc.) as well as specific scientific applications. Many of the available programs require certain environment variables to be set or changed, e.g. **PATH**, **LD_LIBRARY_PATH** and **MANPATH**. The environment modules package is employed to access or switch between various sometimes conflicting versions of software. It provides the means to change the environment dynamically by loading, switching or unloading specific software modules. The module installation is customized to automatically resolve dependencies between modules. Furthermore, a quick usage primer is displayed upon loading.

4.1 Overview of important module commands

Command	Function
`module avail`	Shows all available modules
`module whatis [module]`	Shows a short description of all or a specific module
`module display|show module`	Shows the environment changes the module would have applied if loaded
`module add|load module`	Loads the module ***module***, i.e. sets all necessary variables for the corresponding software and loads required modules
`module list`	Shows all currently loaded modules
`module rm|unload module`	Removes the module ***module***, i.e. removes all envi-

	ronment settings, which were introduced by loading the module, and removes dependent modules
`module purge`	Removes all loaded modules

4.2 *Usage notes*

A module can either be identified by its name only or its name together with a specific version, for example **`intel`** or **`intel/19.0`**, respectively. When only the name is specified, the default version as shown by **`module avail`** is loaded.

- Loading a module changes the environment of the shell in which it is loaded
- If other already loaded modules conflict with the module to load, an according error message is displayed and no further changes are made
- If other modules are required by the module to load, these are loaded recursively and an according message is displayed
- Removing a module reverts the changes in the environment of the shell
- If other modules depend on the module to remove, these are removed recursively and an according message is displayed

Links:
http://modules.sourceforge.net

5 Batch system (SLURM)

CHEOPS uses the Simple Linux Utility for Resource Management (SLURM) to control the usage of the compute nodes and to schedule and execute jobs.
Interaction with the system is carried out by SLURM commands, e.g. **`sbatch`**, **`squeue`**, **`sinfo`**, **`scancel`**. Resource requests can be added directly as command line options or through appropriate **`#SBATCH`** directives in the job scripts. Users can submit jobs from the login node **`cheops`**. The total number of submitted jobs or job instances per user (see section 5.5 concerning job arrays) is limited to 1536.

5.1 *Overview of important SLURM commands*

Command	**Function**
`sbatch` ***`script.sh`***	Submits batch job running script ***`script.sh`***
`salloc` ***`[options]`*** **`srun`** ***`[options] command`***	Submits interactive job with ***`command`***
`salloc srun --pty bash`	Submits interactive job shell for 1 core, 1GB RAM,6 hours
`squeue`	Shows all current jobs
`squeue -j` ***`jobid`***	Shows detailed status information on job ***`jobid`***
`sstat` ***`jobid`***	Shows detailed runtime information on job ***`jobid`***
`scontrol show job` ***`jobid`***	Shows detailed resource information on job ***`jobid`***
`scancel` ***`jobid`***	Cancels the job ***`jobid`***
`squeue --start -j` ***`jobid`***	Shows an approximate time slot for job ***`jobid`***. This estimation might change significantly until the job starts.

5.2 *Partitions*

The batch system offers several partitions, also called queues, which have various resource limits and requirements. Jobs submitted without a partition specified are automatically routed to a partition which corresponds to the given resource request; should this fail due to exceeded limits, the job is immediately rejected at submission time with a corresponding error message. Single-node jobs are routed to the partition **`smp-rh7`** and are assigned to cores of an INCA or MESCA node depending on the size of memory requested. Multi-node jobs are routed to the partitions **`mpi-rh7`** or **`mpi-core-rh7`**. The partition **`mpi-rh7`** is for node-based jobs with memory request per node and assigns to a specific number of cores per INCA node. The partition **`mpi-core-rh7`** is for core-based jobs with memory request per core and assigns to a specific total number of cores gathered from a certain number of INCA nodes needed. The development partition **`devel-rh7`** is for short test jobs – it offers a high response time albeit at the price of strict limits. The partition **`interactive-rh7`** is reserved for interactive jobs only, regular batch jobs submitted with **`sbatch`** are not allowed (see Section 5.5). Specific partitions must be explicitly selected with the option **`--partition`**. Here the current partition based resource limits and requirements for all available partitions:

Partition	Resource	Minimum	Maximum
`mpi-rh7`	Number of nodes	2	128
	Number of cores	2	1536
	Memory **per node**	--	46gb
	Total wall time	--	360 h
`mpi-core-rh7`	Number of nodes	2	--
	Number of cores	2	1536
	Memory **per core**	--	46gb
	Total wall time	--	360 h
`smp-rh7`	Number of nodes	1	1
	Number of cores	1	32
	Memory per node	--	500gb
	Total wall time	--	720 h
`devel-rh7`	Number of nodes	1	2
	Number of cores	1	24
	Memory per node	--	46gb
	Total wall time	--	1 h
`interactive-rh7`	Number of nodes	1	1
	Number of cores	1	32
	Memory per node	--	450gb
	Total wall time	--	336 h

Node based resource limits introduced by features of different node types are listed in the feature table in Section 5.3. Please use those to specify resource requests in detail.

5.3 Resources and environment

In SLURM terminology "tasks" are the resources required to handle separate processes, including MPI processes. They can be requested for the entire job (**`--ntasks`**) or per node (**`--ntasks-per-node`**). Multiple cores per task (**`--cpus-per-task`**) are required for threads, e.g. within hybrid MPI or SMP Jobs. The memory resource can be requested per node (**`--mem`**) or per core (**`--mem-per-cpu`**). In order to use a local HDD or SSD on a compute node, it is necessary to use the special **`--gres=localtmp`** option and access a certain directory (see following table). Please note that most of the INCA nodes have 70 GB local storage available and the remaining INCA nodes provide 10+ GB only. The MESCA nodes have about 390 GB available.

The following table provides a more detailed description of the most common resource specifications:

sbatch / salloc Option	Function
`--partition=`*`partition`*	Defines the destination of the job. Only necessary for partition ***devel-rh7*** and ***interactive-rh7***. Other partitions are automatically addressed according to resource request. Default: automatic routing
`--nodes=`*`nodes`*	Defines the number of nodes for the job. Default: 1
`--ntasks=`*`ntasks`*	Defines the number of tasks for the job. Each task is mapped to a core, unless the option **`--cpus-per-task`** is specified. Default: 1
`--ntasks-per-node=`*`tpn`*	Defines the maximum number of tasks per node. Used mainly with MPI applications. Default: 1
`--cpus-per-task=`*`cpt`*	Defines the number of cores per task. The resource manager will allocate those cores for threads within the according task. Default: 1
`--mem=`*`mem`*	Defines the per-node memory limit of the job. The job will be canceled automatically when this limit is exceeded. The format for ***mem*** is an integer value followed by ***mb*** or ***gb***, e.g. ***10gb***. Default: **--mem=*1000mb***
`--mem-per-cpu=`*`mem`*	Defines the per-core memory limit of the job. See **`--mem`** above for further information. Default: none
`--time=`*`walltime`*	Defines the wall time limit of the job. The job will be canceled automatically when this limit is exceeded. The format for **`walltime`** is ***HH:MM:SS***. Default: **`walltime=`*`01:00:00`***
`--gres=localtmp:`*`size`*	Defines the size (in gigabytes) to use in the **`/local`** partition, the format for ***size*** is an integer followed by ***G***, e.g. ***10G***. The requested space is then accessible through the directory ***/local/${USER}.${SLURM_JOB_ID}*** in the job
`--account=`*`acct_string`*	Defines the account string. You should provide the appropriate project/account. Default: ***UniKoeln***
`--output=`*`filename`* (only in sbatch)	Specifies file ***filename*** to collect **`stdout`** stream. Default: join both **`stdout`** and **`stderr`** into ***slurm-%j.out*** with job ID ***%j***
`--error=`*`filename`* (only in sbatch)	Specifies file ***filename*** to collect **`stderr`** stream. Default: none

`--mail-type=BEGIN,`
`END,FAIL,REQUEUE,ALL`
	Specifies when email is sent. The option argument may consist of a combination of the allowed mail types.

`--mail-user=`
`username@uni-koeln.de`
	Specifies the address to which email is sent. Please note that only addresses of the form ***username@uni-koeln.de*** or ***username@smail.uni-koeln.de*** are allowed.
	Default: none

`--array=`*`array_request`*	Specifies a job array (see Section 5.5).

(only in sbatch)	

The batch system offers two basic procedures to allocate jobs on compute nodes:

- The user can provide all necessary resources and resource limits, e.g. the number of nodes and tasks per node, the maximum memory per node and the maximum wall time. Then, the job will automatically be allocated to nodes that satisfy these requests and waste as few cores and little memory as possible. For example

  ```
  sbatch --nodes=2 --ntasks-per-node=4 --mem=50gb script.sh
  ```

  automatically allocates the job to 2 nodes which have 4 cores and 50 GB available each. Nodes with fewer available cores and less available memory, which meet the requests, are preferred to reserve the remaining resources for larger jobs.
- In addition to given resources and resource limits the user can explicitly request specific groups of nodes by features. The allocation then considers only those nodes. The syntax is as follows

  ```
  sbatch --nodes=nnodes --ntasks-per-node=tpn --constraint
  feature script.sh
  ```

  Multiple features can be specified, together with logical operators like AND, OR, XOR and a number of nodes required with each set of groups, for example:

  ```
  sbatch --nodes=6 --ntasks-per-node=4 --constraint=
  “[inca8*4&inca12*2]” script.sh
  ```

  allocates the job with 6 nodes with 4 tasks each to 4 INCA Nehalem EP nodes and 2 INCA Westmere nodes.

Please refer to the following table for a detailed description of the available features to identify according node types with certain resource limits:

Feature	#nodes	Processor type	#cores	Physical memory	Available memory
inca	747	Nehalem EP/Westmere	8/12	24/48/192 GiB	22/46/188gb
inca8	180	Nehalem EP	8	24 GiB	22gb
inca8_24gb	180	Nehalem EP	8	24 GiB	22gb
inca12	567	Westmere	12	24/48/192 GiB	22/46/188gb
inca12_24gb	337	Westmere	12	24 GiB	22gb
inca12_48gb	225	Westmere	12	48 GiB	46gb
inca12_192gb	5	Westmere	12	192 GB	188gb
inca24gb	517	Nehalem EP/Westmere	8/12	24 GiB	22gb
inca48gb	225	Westmere	12	48 GiB	46gb
inca192gb	5	Westmere	12	192 GB	188gb
mesca	22	Nehalem EX	32	512 GiB	500gb

The common launcher **mpirun** from the MPI libraries has been replaced with the SLURM launcher **srun** which automatically evaluates these resource settings, thus additional options like **–n**,**–np**,**-perhost**, etc. are not necessary (see Section 6.3.3).

Environment variables of the submission shell, e.g. set or changed by loaded modules, are not carried over to a batch job but they are carried over to an interactive job. This default behavior can be changed with the `--export` option. In both cases, the shell limits set in `.bashrc/.cshrc` are propagated to the job shell.

For further options and details concerning the submission of batch jobs, see Section 5.4, for interactive jobs see Section 5.5.

Links:
http://www.schedmd.com/slurmdocs/documentation.html

5.4 Batch job scripts

Job scripts are executable shell scripts, which contain resource request as well as the actual commands to be executed. Please note that modules necessary for successful compilation should be loaded accordingly in the job script during runtime. The runtime environment provides variables to refer to assigned resources: total number of tasks by `SLURM_NTASKS`, number of tasks per node by `SLURM_NTASKS_PER_NODE`, and number of cores per task by `SLURM_CPUS_PER_TASK`. For a quick start, use the following script templates:

SMP Job with multiple threads (e.g. 4 threads)

```
#!/bin/bash -l
#SBATCH --cpus-per-task=4
#SBATCH --mem=1024mb
#SBATCH --time=01:00:00
#SBATCH --account=UniKoeln

module load intel/19.0

workdir=/scratch/${USER}/${SLURM_JOB_ID}
mkdir -p $workdir

cp input.dat $workdir
cd $workdir

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
program input.dat

cd -
cp ${workdir}/results.dat .
```

Node-based MPI Job (e.g. 2 nodes, 8 MPI processes per node)

```
#!/bin/bash -l
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=8
#SBATCH --ntasks=16
#SBATCH --mem=8gb
#SBATCH --time=01:00:00
#SBATCH --account=UniKoeln

module load intel/19.0 intelmpi/2019
```

```
workdir=/scratch/${USER}/${SLURM_JOB_ID}
mkdir -p $workdir

cp input.dat $workdir
cd $workdir

srun -n $SLURM_NTASKS program input.dat

cd -
cp ${workdir}/results.dat .
```

Core-based MPI Job (e.g. 16 MPI processes)

```
#!/bin/bash -l
#SBATCH --ntasks=16
#SBATCH --mem-per-cpu=1gb
#SBATCH --time=01:00:00
#SBATCH --account=UniKoeln

module load intel/19.0 intelmpi/2019

workdir=/scratch/${USER}/${SLURM_JOB_ID}
mkdir -p $workdir

cp input.dat $workdir
cd $workdir

srun -n $SLURM_NTASKS program input.dat

cd -
cp ${workdir}/results.dat .
```

To schedule a node-based MPI job, the batch system has to wait for a specified number of idle cores per node. When scheduled, those jobs run more efficiently because of the regular resources assigned. The batch system can schedule core-based MPI jobs faster because it assigns a total number of idle cores from any set of nodes available. However, jobs distributed over a broader set of nodes run less efficiently because of the fragmentation of resources assigned.

Hybrid MPI Job (e.g. 2 nodes, 2 MPI processes per node, 4 threads per MPI process)

```
#!/bin/bash -l
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=2
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=4
#SBATCH --mem=2gb
#SBATCH --time=01:00:00
#SBATCH --account=UniKoeln

module load intel/19.0 intelmpi/2019

workdir=/scratch/${USER}/${SLURM_JOB_ID}
```

```
mkdir -p $workdir

cp input.dat $workdir
cd $workdir

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK
srun -n $SLURM_NTASKS program input.dat

cd -
cp ${workdir}/results.dat .
```

In Bourne shell like batch jobs the modules functionality is properly initialized by adding the option `-l`, e.g. `#!/bin/sh -l`, in C shell like jobs no addition is needed.

5.5 Job arrays and interactive jobs

In addition to common jobs, there are two more special job types. The job array is meant for jobs equal in script structure but using different input. It allows the submission of many job instances of a single job script. The set of instances is provided by the argument *`array_request`* following the specifying option `--array`. The format of *`array_request`* is a range or a comma delimited list of ranges, e.g. *`1-5`* or *`1,3-5`*. Each instance is assigned to a new job ID, which is provided by the environment variable `SLURM_JOB_ID`. The task identifier can be referenced in the job script with `SLURM_ARRAY_TASK_ID` while the job array identifier is retained in `SLURM_ARRAY_JOB_ID`. Please refer to the following example script for a job array:

Job Array (e.g. 3 jobs)

```
#!/bin/bash -l
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --mem=1gb
#SBATCH --time=01:00:00
#SBATCH --account=UniKoeln
#SBATCH --array=0-2

module load intel/19.0

work-
dir=/scratch/${USER}/${SLURM_ARRAY_JOB_ID}_${SLURM_ARRAY_TASK_ID}
mkdir -p $workdir

cp input.${SLURM_ARRAY_TASK_ID} $workdir
cd $workdir

program input.${SLURM_ARRAY_TASK_ID}

cd -
cp ${workdir}/results.${SLURM_ARRAY_TASK_ID} .
```

The total number of submitted jobs and job array instances per user is limited to 1536. Please keep in mind that job arrays instances should take a reasonable amount of wall time, i.e. longer than 1 hour. Shorter runs should be merged into one job array instance appropriately. The resource requests in the job script apply to each instance, not the entire array.

The interactive job offers flexibility for tests and experiments, both serial and parallel. It can be submitted with the `salloc` command and will run on most cluster nodes. There is also a dedicated partition `interactive-rh7` as well. This partition can be requested by adding the option `--partition interactive-rh7`. Otherwise, the job is scheduled on a regular cluster node. The partition offers up to 32 cores and 450GB memory per job for a maximum runtime of 2 weeks on a single node.
An interactive submission process can be described as follows: The submitted job is queued and waits for scheduling after invoking `salloc` with various resource requests on the login node. Once the job has been granted an allocation, the owner is given an environment for executing his commands interactively with the `srun` command. He can use this environment until either one of the resource limits is exceeded or the allocation is revoked. The resources, e.g. number of tasks or nodes are taken automatically from the allocation and do not have to be specified again for the `srun` command. The `srun` command can be launched together with `salloc` in a single line or separately, one after the other, as shown in the following examples.

Interactive MPI Jobs must be allocated and launched directly from the login node, e.g.

```
cheops1$ salloc --ntasks-per-node=3 --nodes=2 --time=10:00 --mem 1gb
salloc: Pending job allocation <JOBID>
salloc: job <JOBID> queued and waiting for resources
salloc: job <JOBID> has been allocated resources
salloc: Granted job allocation <JOBID>
cheops1$ srun ./mpi_application
cheops1$ exit
salloc: Relinquishing job allocation <JOBID>
salloc: Job allocation <JOBID> has been revoked.
```

In the above example, the MPI application can use 1GB memory on 2 compute nodes with 3 tasks each for 10 minutes.

Interactive SMP Jobs must be allocated from the login node. The `srun` command is then used to launch a shell on the compute node in pseudo terminal mode and the application can be executed within, e.g.

```
cheops1$ salloc --ntasks=1 --cpus-per-task=8 --time=10:00 --mem 1gb
salloc: Pending job allocation <JOBID>
salloc: job <JOBID> queued and waiting for resources
salloc: job <JOBID> has been allocated resources
salloc: Granted job allocation <JOBID>
cheops1$ srun --pty bash
cheops10101$ ./smp_application
```

In the above example the application runs on a compute node, it can use up to 8 available cores and 1GB memory for 10 minutes. Please note that interactive SMP Jobs must use only one task and the number of threads is specified with the option **`--cpus-per-task`**. X11 forwarding may be enabled by adding the option **`--x11`** to **`salloc`**.

6 Development environment

The three major parts of the development software environment are compilers, libraries and tools. Currently, the installation looks like that (default versions in boldface):

- Compilers
  - Intel compilers (version 17.0, 18.0, **19.0**, 19.1, 2021)
  - GNU compilers (version **4.8.5**, 5.1.0, 7.4.0, 9.4.0)
  - PGI compilers (version 17.7, **19.10**)
  - Nvidia compilers (version 20.11)
  - Julia language (version 1.6.5)
- Libraries
  - Intel MKL (Math Kernel Library, version 2017, 2018, **2019**, 2020, 2021)
  - Intel TBB (Threading Building Blocks, version 2017, 2018, **2019**, 2020, 2021)
- MPI libraries
  - Intel MPI (version 2017, 2018, **2019**, 2019.9, 2021)
  - Open MPI (version 1.8.6, **4.1.1**)
- Debugger
  - ARM Forge/DDT (version 22.0.2)
  - Perforce TotalView (version 2022)
  - Intel Inspector (version 2021)
- Profiler
  - PAPI (Performance Application Programming Interface, version 5.2.0)
  - Intel Advisor (version 2021)
  - Intel VTune Amplifier (version **2019**, 2020)
  - Intel Trace Analyzer and Collector (version 2017, 2018, **2019**, 2020, 2021)
  - ARM Forge/MAP (version 22.0.2)
  - Vampir (version 10.0.2) with Score-P (version 7.1)

If there is no preference on how to build an application, the Intel software should be used.

6.1 Compilers

Environment modules provide the environment of the compilers. After loading one of these modules several commands invoke the compilers for according programming languages:

Compiler	Modules	C/C++	FORTRAN 77/90
Intel Compilers	**`intel/*`**	**`icc/icpc`**	**`ifort`**
New Intel Compilers	**`intel_new/*`**	**`icx/icpx`**	**`ifx/ifx`**
GNU Compilers	**`gnu/*`**	**`gcc/g++`**	**`gfortran`**
PGI Compilers	**`pgi/*`**	**`pgcc/pgc++`**	**`pgf77/pgf90`**
Nvidia Compilers	**`nvhpc/*`**	**`nvc/nvc++`**	**`nvfortran/nvfortran`**

Nehalem specific code can be generated by **-xhost** with the Intel compilers and by **–march=native** with the GNU compilers. Common optimizations are performed by the general option **–O2** for all compilers.

Links:
https://www.intel.com/content/www/us/en/develop/documentation/oneapi-dpcpp-cpp-compiler-dev-guide-and-reference/top.html
https://www.intel.com/content/www/us/en/develop/documentation/fortran-compiler-oneapi-dev-guide-and-reference/top.html
https://gcc.gnu.org/onlinedocs/
https://gcc.gnu.org/onlinedocs/gfortran/
https://docs.nvidia.com/hpc-sdk/pgi-compilers/19.10/x86/index.htm
https://docs.nvidia.com/hpc-sdk/archive/20.11/index.html

6.1.1 Julia language

Julia is a programming language for scientific computing. By Just-In-Time (JIT) compilation using LLVM, it combines the ease of use from interpreter languages like Python or R with efficiency and performance provided by common C compilers.
All versions of Julia were built with more recent GNU compiler versions than the operating system provides. In our environment modules framework, we consider the modules **julia** as compiler modules. Each of these modules expands the environment the needed GNU compiler version inline and does not allow for other compiler modules loaded because of conflicts. If you need to compile C/C++ code with your Julia application, please use **gcc/g++** coming with the modules **julia**.
After loading a module **julia**, you can either start an interactive session on a login node

```
cheops1$ julia
julia> 1 + 2
3
julia> exit()
cheops1$
```

or run the script of your applicaton in a batch job on a compute node by

```
julia script.jl arg1 arg2 ...
```

in your job script.

Links:
https://julialang.org
https://docs.julialang.org

6.2 Libraries

6.2.1 Intel MKL

The MKL (Math Kernel Library) includes the functionality of many other libraries (BLAS, LAPACK, FFT, Sparse BLAS) addressing mathematical operations and algorithms often needed in scientific computing. Both shared memory and distributed memory is supported by multi-threaded routines and implementations using message passing (ScaLAPACK, PBLAS, BLACS), respectively. Again, MKL is made available by the according environment module **mkl**. In C/C++ the MKL header **mkl.h** has to be included. The common linkage scheme for sequential processing is compiler specific because of the FORTRAN runtime libraries needed:

Compiler	Linkage in C/C++	Linkage FORTRAN 77/90
Intel Compilers	**-lmkl_intel_lp64** **-lmkl_sequential** **-lmkl_core -lpthread**	**-lmkl_intel_lp64** **-lmkl_sequential** **-lmkl_core -lpthread**
GNU Compilers	**-lmkl_intel_lp64** **-lmkl_sequential** **-lmkl_core -lpthread** **-lm**	**-lmkl_gf_lp64** **-lmkl_sequential** **-lmkl_core -lpthread**
PGI Compilers	**-lmkl_intel_lp64** **-lmkl_sequential** **-lmkl_core -lpthread**	**-lmkl_intel_lp64** **-lmkl_sequential** **-lmkl_core -lpthread**

The common linkage scheme for multi-threaded processing is more complicated due to the compiler specific threading:

Compiler	Linkage in C/C++	Linkage FORTRAN 77/90
Intel Compilers	**-lmkl_intel_lp64** **-lmkl_intel_thread** **-lmkl_core** **-liomp5 -lpthread**	**-lmkl_intel_lp64** **-lmkl_intel_thread** **-lmkl_core** **-liomp5 -lpthread**
GNU Compilers	**-lmkl_intel_lp64** **-lmkl_intel_thread** **-lmkl_core** **-liomp5 -lpthread** **-lm**	**-lmkl_gf_lp64** **-lmkl_intel_thread** **-lmkl_core** **-liomp5 -lpthread**
PGI Compilers	**-lmkl_intel_lp64** **-lmkl_intel_thread** **-lmkl_core** **-liomp5 -lpthread**	**-lmkl_intel_lp64** **-lmkl_intel_thread** **-lmkl_core** **-liomp5 -lpthread**

The number of threads can be set by the environment variable **MKL_NUM_THREADS** or by the more general OpenMP variable **OMP_NUM_THREADS**.

Links:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html
https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl-link-line-advisor.html

6.3 MPI libraries

6.3.1 Intel MPI

Intel MPI provides support of the MPI-3.1 standard. Again, Intel MPI is made available by the according environment module **intelmpi**. The compiler scripts including header files and linking libraries have compiler specific names:

Compiler	C/C++	FORTRAN 77/90

Intel Compilers	**`mpiicc/mpiicpc`**	**`mpiifort/mpiifort`**
GNU Compilers	**`mpigcc/mpigxx`**	**`mpif77/mpif90`**

Integers of 8 bytes size are supported by the option **`-ilp64`**. Specific modules (version suffix **`mt`** and **`dbg`**) provide thread safe and debugging libraries, respectively.

Links:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/mpi-library.html

6.3.2 Open MPI

Open MPI is an open source implementation of the MPI-3.1 standard. It is made available by the according environment module **`openmpi`**. Specific modules pointing to according builds provide both support of 8 bytes integers (version suffix **`ilp64`**) and thread safety (version suffix **`mt`**). The compiler scripts including header files and linking libraries have the same naming for all compilers:

Compiler	C/C++	FORTRAN 77/90
All Compilers	**`mpicc/mpicxx`**	**`mpif77/mpif90`**

Links:
https://www.open-mpi.org/doc/

6.3.3 MPI runtime environment

With the batch system SLURM, MPI applications can only be executed within jobs. The batch system provides its own launcher **`srun`**, which tightly integrates the application processes. Therefore, only the number of MPI processes has to be specified

```
srun -n $SLURM_NTASKS ./a.out
```

and is provided by SLURM as the number of tasks. Interactive invocation of this command is only possible within interactive jobs. The SLURM runtime environment also provides pinning and affinity of application processes according to the assigned resources. Therefore, options or environment variables from the MPI libraries changing the runtime environment like pinning are ignored and should not be used anymore (see Section 5.3).

6.4 Debugger

6.4.1 ARM Forge/DDT

Forge is a debugger/profiler from ARM for parallel applications (both multi-threaded and MPI). As Forge is integrated with the Intel MPI and Open MPI libraries, the according environment module **`forge`** requires an Intel MPI or Open MPI module. When using the debugger DDT with the Intel MPI library, components of the Intel Trace Analyzer and Collector are needed for the Intel Message Checker plugin and the according Trace Analyzer and Collector module is required as well. Executables to be analyzed are instrumented by

```
mpiicc|mpigcc|mpiifort|mpif90… -g -O0
```

including debugging information and disabling any optimization. Then, the debugger DDT is launched with the graphical user interface **forge**. With the DDT action Run, applications are executed via batch system by default. However, changing to interactive execution is possible, e.g. within an interactive job on a compute node. With the DDT action Attach, the debugger connects to processes of already running jobs when started on the executing compute node. In the analysis, single source lines can be resolved for different threads and processes, respectively.

Links:
https://developer.arm.com/documentation/101136/22-0-2

6.4.2 Perforce TotalView

TotalView is a debugger from Perforce for parallel applications (both multi-threaded and MPI). As TotalView is integrated with the Intel MPI and Open MPI libraries, the according environment module **totalview** requires an Intel MPI or Open MPI module. Executables to be analyzed are instrumented by

```
mpiicc|mpigcc|mpiifort|mpif90… -g -O0
```

including debugging information and disabling any optimization. Then, the debugger is launched with the graphical user interface **totalview**. With the action Debug a (Parallel) Program, applications are executed via batch system by default. However, changing to interactive execution is possible, e.g. within an interactive job on a compute node. With the action Attach to Process, the debugger connects to processes of already running jobs when started on the executing compute node. In the analysis, single source lines can be resolved for different threads and processes, respectively.

Links:
https://help.totalview.io/

6.4.3 Intel Inspector

The Inspector is a memory debugger for multi-threaded applications. As it relies on the Intel compilers, the according environment module **inspector** requires an Intel compiler module. Executables to be analyzed are instrumented by

```
icc|icpc|ifort -g -O0
```

Then, the Inspector can be launched with the command line user interface **inspxe-cl** or the graphical user interface **inspxe-gui**. Within these environments, the application is executed and analyzed. Issues in memory access (memory leaks, race conditions, etc.) are pointed out. In the analysis, single source lines from the relating code sequences can be resolved for different threads and processes, respectively.

Links:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/inspector.html

6.5 *Profiler*

6.5.1 PAPI Library

The PAPI library collects information from the performance counters in the processor hardware. The operating system already provides PAPI. There is no module to load. The utilization of PAPI by High Level Functions is straightforward:

C	FORTRAN 90
`#include <papi.h>`	`#include "f90papi.h"`

`PAPI_flops(&rtime, &ptime,`
`&flpops, &mflops );`
	`call PAPIF_flops(real_time,&`
	`proc_time, flpops, mflops, check)`

`/* a lot of work */`	`! a lot of work`

`PAPI_flops(&rtime, &ptime,`
`&flpops, &mflops );`
`printf("%f %f %lld %f\n", rtime,`
`ptime, flpops, mflops);`
	`call PAPIF_flops(real_time,&`
	`proc_time, flpops, mflops, check)`
	`print *, real_time, proc_time,&`
	`flpops, mflops, check`

Include the header via preprocessing, place according calls around the source code of interest and link with the library (link option **`-lpapi`**).

Links:
https://bitbucket.org/icl/papi/wiki/Home

6.5.2 Intel Advisor

Advisor is a profiler, which gives advice where to introduce vectorization and threading into the application. The executable needs to be compiled with debugging information

```
icc|icpc|ifort -g -O2
```

for the source code resolved analysis. All optimizations have to be specified explicitly since the compiler option **`-g`** sets no optimization by default. As Advisor also relies on the Intel compilers, the according environment module **`advisor`** requires an Intel compiler module. After loading this module, Advisor can be launched with the command line user interface **`advixe-cl`** or the graphical user interface **`advixe-gui`**. Within these environments, the application is executed and analyzed. Critical source code lines are marked with recommendations how to change to code.

Links:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/advisor.html

6.5.3 Intel VTune Amplifier

VTune Amplifier is a profiler, which offers time and event based profiling. Like PAPI, VTune Amplifier collects information from the performance counters in the processor hardware. Therefore, the executable does not have to be instrumented. However, VTune Amplifier also needs debugging information

```
icc|icpc|ifort -g -O2
```

for the source code resolved analysis. As VTune Amplifier relies on the Intel compilers, the according environment module **vtune** requires an Intel compiler module. After loading this module VTune Amplifier can be launched with the command line user interface **amplxe-cl** or the graphical user interface **amplxe-gui**. Within these environments, the application is executed and analyzed. Single source lines or even assembly lines can be resolved within the analysis of the performance data. Moreover, a call graph with critical path can be determined.

Links:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html

6.5.4 Intel Trace Analyzer and Collector

The Trace Analyzer and Collector (TAC) is a profiler for MPI applications. Currently it offers time based profiling by linking with a profiling library. As TAC relies on the Intel MPI library, the according environment module **tac** requires an Intel MPI module. After loading one of these modules the whole MPI application is instrumented at compile time by linking with option **-trace**. An execution of the instrumented MPI application results in a trace file, which then can be analyzed

```
traceanalyzer a.out.stf
```

with the Trace Analyzer.

6.5.4.1 How to filter tracing

The instrumentation by compiling with the compiler option **-tcollect** enables filtering of routines to be traced. The filter rules are then defined in an appropriate configuration file for the Trace Collector during runtime. For example, a file **myconfig** containing the lines

```
SYMBOL * FOLD
SYMBOL mykernel* ON UNFOLD
```

restricts tracing to the routine **mykernel** and those routines that it calls. The asterisk after **mykernel** matches a possible underscore in compiler's naming convention. Providing this configuration file during runtime by the environment variable **VT_CONFIG**

```
env VT_CONFIG=myconfig srun -n $SLURM_NTASKS ./a.out
```

results in tracing only MPI functions called in **mykernel**. In the same way, specific MPI functions can be filtered throughout the whole MPI application by using wildcard expressions matching a subset of MPI functions. Please refer to the manual page of **VT_CONFIG** for more details on the Trace Collector configuration file.

Links:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/trace-analyzer.html

6.5.5 ARM Forge/MAP

Forge is a debugger/profiler from ARM for parallel applications (both multi-threaded and MPI). As Forge is integrated with the Intel MPI and Open MPI libraries, the according environment module **`forge`** requires an Intel MPI or Open MPI module. When using the profiler MAP, executables to be analyzed are instrumented by

```
mpiicc|mpigcc|mpiifort|mpif90… -g -O2
```

including debugging information and recovering optimization. Then, the profiler MAP is launched with the graphical user interface **`forge`**. With the MAP action Profile, applications are executed via batch system by default. However, changing to interactive execution is possible, e.g. within an interactive job on a compute node. In the analysis, single source lines can be resolved for different threads and processes, respectively.

Links:
https://developer.arm.com/documentation/101136/22-0-2

6.5.6 Vampir

Vampir is another profiler for MPI applications. Its instrumentation with the tracing tool Score-P provides time based profiling and event based profiling via the PAPI library. As we have built Score-P with both Intel and Open MPI, the according environment module **`scorep`** requires an Intel MPI or Open MPI module. After loading the module, the whole MPI application is instrumented by compiling with compiler wrappers

Compiler	C/C++
All Compilers	`scorep mpiicc/mpicc/mpiicpc/mpicxx`
	FORTRAN 77/90
	`scorep mpiifort/mpif77/mpif90`

An execution of the instrumented MPI application results in a trace file. After loading the module **`vampir`**, this trace file can be analyzed

```
vampir traces.otf2
```

with Vampir.

Links:
https://tu-dresden.de/die_tu_dresden/zentrale_einrichtungen/zih/forschung/projekte/vampir
https://www.vi-hps.org/projects/score-p