Mistral Usage

Hendryk Bockelmann
Deutsches Klimarechenzentrum (DKRZ)

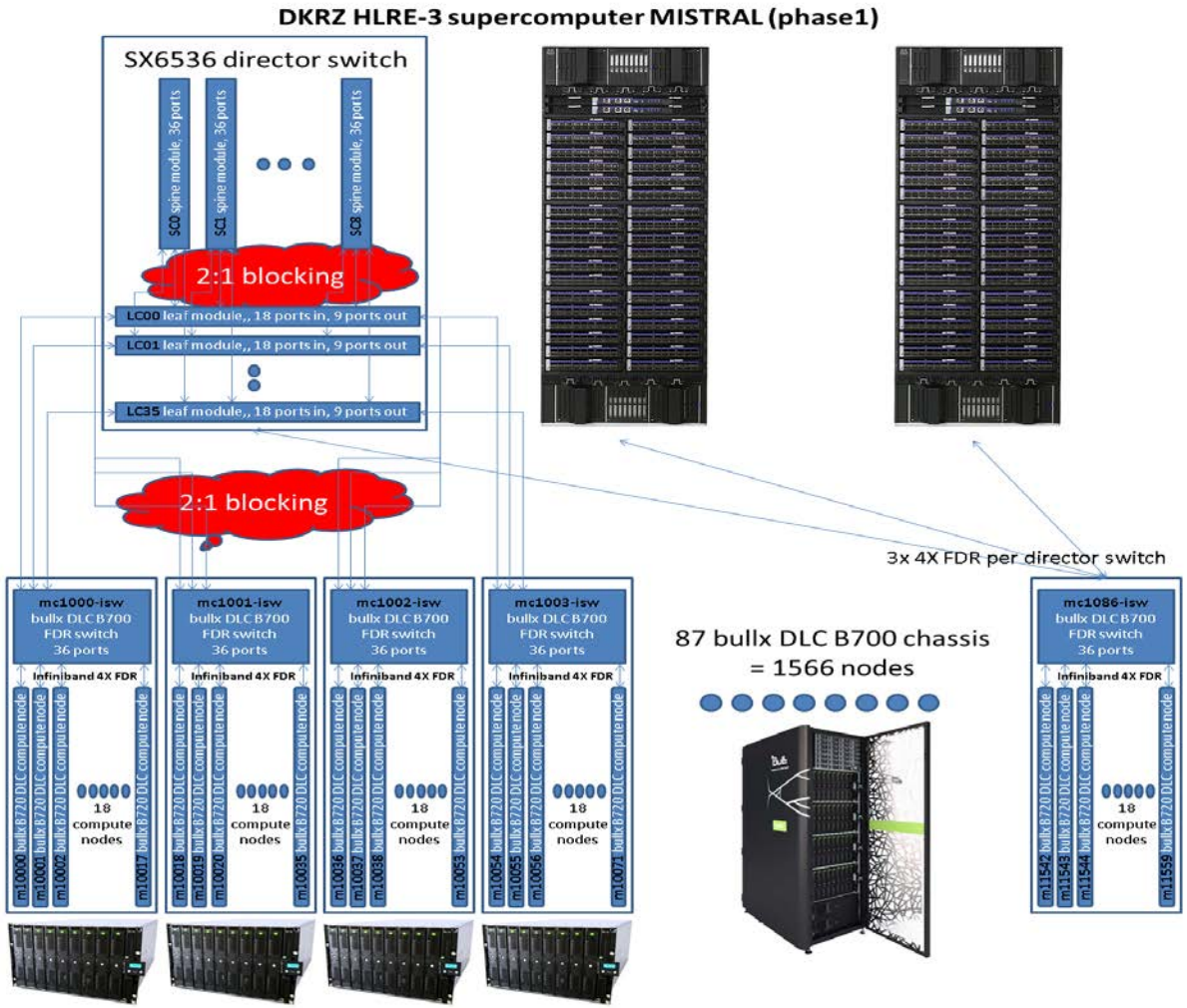
Mistral Characteristics

System	# nodes	Mem(TB)	# cores	TFLOP	Disk storage
mistral (phase 1) total	**1555**	**115**	**37344**	**1493 peak**	**20 PByte**
compute (1386 x 64GB, 110 x 128GB)	1496	100	35904		
prepost (256GB)	38	9.5	912		
visualization (256GB + 2x Nvidia Tesla K80 GPUs)	12	3	288		
mistral (phase 2) additional	**>1500**	**>100**	**?**	**1500 peak**	**30 PByte**

HSW node (24 cores): 16 DP flop/cycle at 2.5 GHz (2 x 256-bit FMA instr. – AVX2)
=> 960.0 GFlop/s per node (806.4 GFlop/s truly)

Configuration

- bullx B700 DLC (Direct Liquid Cooling) blade system with two nodes forming one blade
- each node has two sockets, equipped with an Intel Xeon E5-2680 v3 12-core processor (Haswell)
- four kinds of nodes are available to users:
  - 6 login nodes + 5 interactive nodes (mistralpp.dkrz.de)
  - 1496 compute nodes for running scientific models (thin & big memory)
  - 38 nodes for pre-/postprocessing (fat memory)
  - 12 visualization/GPU nodes
- all nodes are integrated in one FDR Infiniband Fabric with fat tree topology (measured 5.9 GB/s bandwidth, 2.7µs latency)
- operating System is Red Hat Enterprise Linux release 6.4


DKRZ HLRE-3 supercomputer MISTRAL (phase1)
SX6536 director switch
SC0 spine module, 36 ports
SC1 spine module, 36 ports
SC9 spine module, 36 ports
2:1 blocking
LC00 leaf module,, 18 ports in, 9 ports out
LC01 leaf module,, 18 ports in, 9 ports out
LC35 leaf module,, 18 ports in, 9 ports out
2:1 blocking
3x 4X FDR per director switch
mc1000-isw
bullx DLC B700
FDR switch
36 ports
mc1001-isw
bullx DLC B700
FDR switch
36 ports
mc1002-isw
bullx DLC B700
FDR switch
36 ports
mc1003-isw
bullx DLC B700
FDR switch
36 ports
Infiniband 4X FDR
18
compute
nodes
87 bullx DLC B700 chassis
= 1566 nodes
mc1086-isw
bullx DLC B700
FDR switch
36 ports

Login and Environment

```
$ ssh mistral.dkrz.de
<userID>@mistral's password:
Last login: Mon Jun  8 18:13:02 2015 from yourpc.xyz.de
*******************************************************************************
*                                                                             *
*  Welcome to MISTRAL @ DKRZ!                                                 *
*                                                                             *
*                                                                             *
*  Please use the Login Nodes for compiling, submitting Jobs, etc. but keep   *
*  in mind they are not for interactive Job processing.                       *
*                                                                             *
*  If you encounter problems, need assistance or have any suggestion,         *
*  please write an email to                                                   *
*                                                                             *
*  --  beratung@dkrz.de --                                                    *
*                                                                             *
*-----------------------------------------------------------------------------*
*                                                                             *
* Notice:                                                                     *
* -------                                                                     *
* You HAVE to set a valid account/project in your job scripts, e.g.           *
*                                                                             *
* #SBATCH --account=<my_project>                                              *
*                                                                             *
* #SBATCH -A <my_project>                                                     *
*                                                                             *
*******************************************************************************
[<userID>@mistral ~]$
```

Login and Environment

- users need to be member in at least one active project
- log in via ssh, replacing <userid> by your username

  ssh <userid>@mistral.dkrz.de
- mistral.dkrz.de is a round robin to one of the login nodes mlogin100-105
- login nodes serve as a frontend to the HPC cluster
  - file editing and compilation of source code
  - submission, monitoring and canceling of batch jobs
  - should only be used for simple and not time- and memory-intensive operations … otherwise:
    - use pre-/postprocessing nodes via SLURM
    - use interactive nodes (explained later)

Software Environment

- modules environment is used
- no hierarchical modules just naming convention

  <modname>/<modversion>
- internal consistency checks should warn if modules are incompatible
- by default only modules concerning the workflow are loaded – no “default compiler”
- latest software list daily updated at

https://www.dkrz.de/Nutzerportal-en/doku/mistral/softwarelist

Software Environment

Management via module sub-commands:

- module avail: show list of all available modules
- module add: load a specific module – use full description <modname>/<modversion> or latest version <modname>
- module list: list currently loaded modules
- module rm <modname>/<modversion> : unload module
- module purge: unload all modules

Data Management – Filesystem

Data Management – Filesystem

- parallel filesystem lustre with 3 data spaces
  - HOME
  - WORK
  - SCRATCH
- all data spaces are available on all nodes
- no differentiation between small and big files needed (cmp. blocksize on GPFS / blizzard)

Data Management – Filesystem

system	HOME	WORK	SCRATCH
Path	/pf/[a,b,g,k,m,u]/<userid>	/work/<projectid>	/scratch/[a,b,g,k,m,u]/<userid>
Description	- Assigned to user account - Storage of personal files, src code, etc	- Assigned to project account - Interim storage of output from experiments and frequently used data	- Assigned to user account - Temporary storage and processing of large data sets
Quota	24 GB	According to annual project allocation	15 TB
Backup	Yes	No	No
Data lifetime	Until user account deletion	1 month after project expiration	14 days since last file access

Data Management – Filesystem

- data from GPFS was mirrored to lustre (HOME,WORK)

  /mnt/lustre01/rsync/[pf|work]

- what to do with the data?
  - just read it, do not mv or cp into /work or /home
  - current lustre client has severe problems in handling data: mv should be a meta data operation only, but is cp instead
  - DKRZ will keep the data as long as the lustre bug is not fixed and inform you whenever there is an update

Lustre

I/O architecture for short

- File system: lustre 2.5
- 29 I/O server and 5 metadata server
- max performance per server is 5.4 GiB/s

Details, measurements and best practices

- following talk by Julian Kunkel

Every day work and scripting

- use lfs commands for faster response!

e.g. ‘lfs ls –l’ instead of ‘ls –l’


slurm
workload manager

SLURM Resource Management

- Partitions and QoS (Quality of Service) are used in SLURM to group nodes and jobs characteristics
- The use of Partitions and QoS entities in SLURM is orthogonal:
  - Partitions for grouping resources characteristics
  - QoS for grouping limitations and priorities

Partition *compute*: 512 nodes max, 8h runtime

Partition *gpu*: 2 nodes max, Nvidia GPU

Partition *prepost*: 2 nodes max, high memory

QoS *express*:
higher priority,
20 min runtime limit,
4 nodes max

Other QoS possible e.g.

long (lower priority, higher runtime)
…
summerschool (higher priority, limited number of nodes)

SLURM Partitions

Partition	compute (default)	prepost	shared	gpu
MaxNodes	512	2	1	2
MaxTime	8 hours	4 hours	7 days	4 hours
Shared	exclusive	yes:4 max 4 jobs share resources	yes:4 max 4 jobs share resources	exclusive (for GPU computing)
MaxMemPerCPU	nodelimit	5 Gbyte	1.25 GByte	5 Gbyte

General limits:

- 20 jobs running in parallel (no limit for queued jobs)
- might be extended

SLURM Associations

```
$ sacctmgr list user format=user,defaultaccount where user=$USER
      User   Def Acct
---------- ----------
   k202082  noaccount
$ sacctmgr list associations format=user,account,partition,qos where
user=$USER
      User    Account  Partition                  QOS
---------- ---------- ---------- --------------------
   k202082  noaccount                          normal
   k202082     k20200              bench,express,normal
   k202082     kt9999                    express,normal
```

- you have to specify the account for each job !
- list partitions: ‘scontrol show partitions’
- list qos: ‘sacctmgr show qos’

SLURM Command Overview

- **sinfo** – show information about partition and nodes
- **squeue** – list pending and running jobs
- **sbatch** – submit job script for execution (batch mode)
- **salloc** – create job allocation and start a shell to use it (interactive mode)
- **srun** – create a job allocation and directly launch a job step (typically an MPI job)
- **scancel** – cancel pending or running job or job step
- **scontrol** – show or modify jobs, partitions, nodes, …

SLURM Job and Node States

- Job States (squeue)
  - **Pending PD** (awaiting resources allocation)
  - **Running R** (has an active allocation)
  - **Cancelled CA** (explicitly cancelled by user or admin)
  - **Timeout TO** (terminated reaching time limit)
  - **Completing CG** (some nodes may still be active)
  - **Completed CD** (all processes on all nodes terminated)
- Node States (sinfo)
  - **Down** (unavailable for use)
  - **Idle** (not allocated, awaiting job allocations)
  - **Allocated** (by one or more jobs)
  - **Completing** (jobs are completing, epilog might run)
  - **Draining** (currently running job, but will not be available afterwards)
  - **Mixed** (some CPUs allocated while others idle)

* after state means node is not reachable for slurmctld

Side by Side Comparison to Other WLM

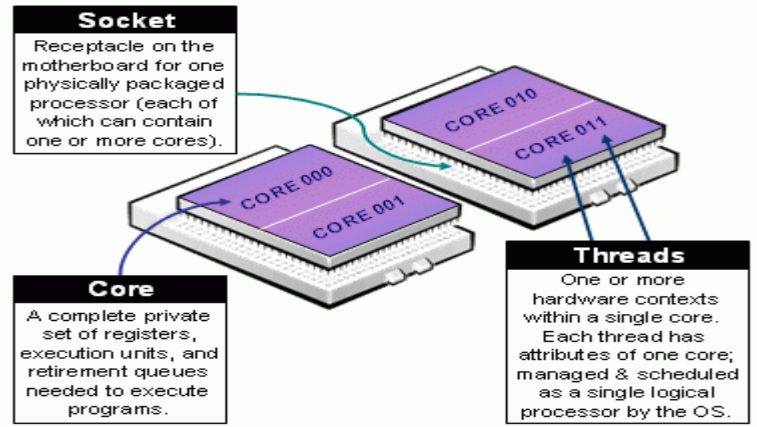
http://slurm.schedmd.com/rosetta.pdf

					28-Apr-20
User Commands	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**
Job submission	qsub [script_file]	sbatch [script_file]	bsub [script_file]	qsub [script_file]	llsubmit [script_file]
Job deletion	qdel [job_id]	scancel [job_id]	bkill [job_id]	qdel [job_id]	llcancel [job_id]
Job status (by job)	qstat [job_id]	squeue [job_id]	bjobs [job_id]	qstat -u * [-j job_id]	llq -u [username]
Job status (by user)	qstat -u [user_name]	squeue -u [user_name]	bjobs -u [user_name]	qstat [-u user_name]	llq -u [user_name]
Job hold	qhold [job_id]	scontrol hold [job_id]	bstop [job_id]	qhold [job_id]	llhold -r [job_id]
Job release	qrls [job_id]	scontrol release [job_id]	bresume [job_id]	qrls [job_id]	llhold -r [job_id]
Queue list	qstat -Q	squeue	bqueues	qconf -sql	llclass
Node list	pbsnodes -l	sinfo -N OR scontrol show nodes	bhosts	qhost	llstatus -L machine
Cluster status	qstat -a	sinfo	bqueues	qhost -q	llstatus -L cluster
GUI	xpbsmon	sview	xlsf OR xlsbatch	qmon	xload
Environment	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**
Job ID	$PBS_JOBID	$SLURM_JOBID	$LSB_JOBID	$JOB_ID	$LOAD_STEP_ID
Submit Directory	$PBS_O_WORKDIR	$SLURM_SUBMIT_DIR	$LSB_SUBCWD	$SGE_O_WORKDIR	$LOADL_STEP_INITDIR
Submit Host	$PBS_O_HOST	$SLURM_SUBMIT_HOST	$LSB_SUB_HOST	$SGE_O_HOST	
Node List	$PBS_NODEFILE	$SLURM_JOB_NODELIST	$LSB_HOSTS/LSB_MCPU_HOST	$PE_HOSTFILE	$LOADL_PROCESSOR_LIST
Job Array Index	$PBS_ARRAYID	$SLURM_ARRAY_TASK_ID	$LSB_JOBINDEX	$SGE_TASK_ID	
Job Specification	**PBS/Torque**	**Slurm**	**LSF**	**SGE**	**LoadLeveler**
Script directive	#PBS	#SBATCH	#BSUB	#$	#@
Queue	-q [queue]	-p [queue]	-q [queue]	-q [queue]	class=[queue]
Node Count	-l nodes=[count]	-N [min[-max]]	-n [count]	N/A	node=[count]
CPU Count	-l ppn=[count] OR -l mppwidth=[PE_count]	-n [count]	-n [count]	-pe [PE] [count]	
Wall Clock Limit	-l walltime=[hh:mm:ss]	-t [min] OR -t [days-hh:mm:ss]	-W [hh:mm:ss]	-l h_rt=[seconds]	wall_clock_limit=[hh:mm:ss]
Standard Output File	-o [file_name]	-o [file_name]	-o [file_name]	-o [file_name]	output=[file_name]
Standard Error File	-e [file_name]	e [file_name]	-e [file_name]	-e [file_name]	error=[File_name]
Combine stdout/err	-j oe (both to stdout) OR -j eo (both to stderr)	(use -o without -e)	(use -o without -e)	-j yes	
Copy Environment	-V	--export=[ALL | NONE | variables]		-V	environment=COPY_ALL
Event Notification	-m abe	--mail-type=[events]	-B or -N	-m abe	notification=start|error|complete|never|alwa

http://slurm.schedmd.com/pdfs/summary.pdf

SLURM Workflow

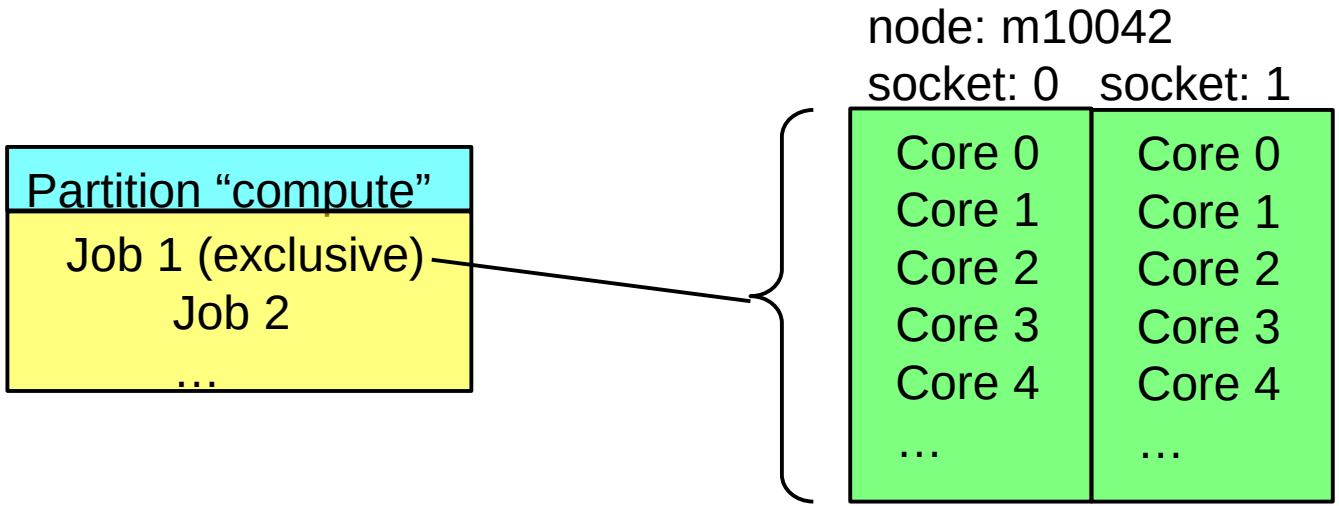
1. **partition** : which resources do I need? What limits (runtime, memory) do I have?
2. **allocation** : batch job or interactive usage?
3. **node usage** : NUMA board
   - 2 sockets with 12 cores each
     - each core 2 CPUs/HyperThreads
   - Memory per CPU?



SLURM Workflow

Example to demonstrate job allocations and job steps:

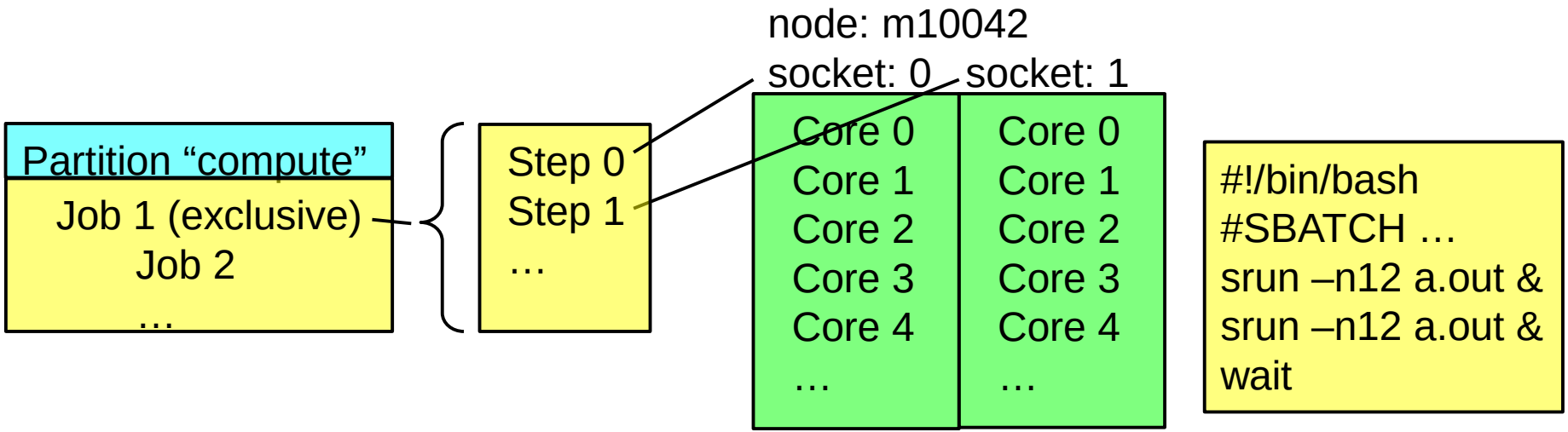
- Jobs are allocated resources (sbatch or salloc)



SLURM Workflow

Example to demonstrate job allocations and job steps:

- Jobs spawn job steps, which are allocated resources from within the job's allocation (srun)



SLURM Spawning Tasks

- both sbatch and salloc only allocate resources
- use srun to start parallel (MPI) application
- for testing purposes one might call srun directly from login node:

```
$ srun –N2 –n2 –Ak20200 -l hostname
srun: job 125356 queued and waiting for resources
srun: job 125356 has been allocated resources
0: m11225
1: m11226
```

SLURM Batch Jobs

SLURM Batch Jobs

- submit job scripts using **sbatch** command
- shell script executed on the first node of the allocation
- SLURM submission options are prefixed with

  #SBATCH
- short or long option, e.g. --nodes=<n> or –N <n>
- for long form: **NO** space around = allowed
- #SBATCH ignored after first executable command in script
- options can also be given on command line and have highest priority

SLURM Batch Jobs

#SBATCH option	Default value	Description
--nodes=<n>	1	Number of nodes for the allocation
--ntasks=<n>	1	Number of (MPI) tasks
--ntasks-per-node=<n>	1	Number of (MPI) tasks per node
--cpus-per-task=<n>	1	Number of threads (logical CPUs) per task
--time=<walltime>	partition dependent	Requested wallclock limit
--partition=<name>	compute	Partition for the allocation
--account=<projectid>	NONE	Project Id for the accounting
--[no-]requeue	requeue	Requeue batch job if node fails? Script is initiated from its beginning in this case!

SLURM Batch Jobs

Resource selection and resource allocation options can be mixed in sbatch …

- Selection:

  #SBATCH --nodes=32

  #SBATCH --cores-per-socket=12

- Allocation:

  #SBATCH --ntasks=12

  #SBATCH --ntasks-per-socket=4

  #SBATCH --cpus-per-task=8

- srun inherits option from sbatch if not overwritten

CAUTION: HyperThreading

- mistral Haswell processors support HyperThreading
- i.e. each node has 24 physical cores, 48 logical CPUs or Hardware Threads (HWT)
- SLURM offers option --threads-per-core to distinguish between nodes in a heterogeneous system
- BUT: on mistral --threads-per-core=2 is enforced! Even if you do not want to use HT
- Respect this when setting --cpus-per-task

CAUTION: Frequency Scaling

- mistral Haswell processors allow for CPU frequency scaling
- Operating system can scale CPU frequency up or down in order to save power
- default srun behaviour: use fixed 2.5 GHz
- you might experiment with different settings

  export SLURM_CPU_FREQ_REQ=1200000

  or

  srun --cpu-freq=1200000
- frequency has to be given in kiloHertz (kHz)
- allowed frequencies (1.2, 1.3, …, 2.5 GHz)

Example: OpenMP Job Using Intel Compiler

```
#!/bin/bash
#SBATCH --job-name=OpenMPjob
#SBATCH --partition=shared
#SBATCH --ntasks=1
#SBATCH --cpus-per-tasks=8
#SBATCH --time=00:30:00
#SBATCH --account=x12345

export OMP_NUM_THREADS=8
export KMP_AFFINITY=verbose,granularity=thread,compact,1
export KMP_STACKSIZE=64m

cdo –P 8 <operator> <ifile> <ofile>
```

i.e. 8 CPUs (HyperThreads) = 4 cores are allocated

thread binding … next slides

Example: MPI Job

```
#!/bin/bash
#SBATCH --job-name=MPIjob
#SBATCH --partition=compute
#SBATCH --nodes=4
#SBATCH --ntasks-per-node=24
#SBATCH --cpus-per-tasks=2
#SBATCH --time=00:30:00
#SBATCH --account=x12345

module load intelmpi
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

srun –l --cpu_bind=verbose,cores ./myapp
```

i.e. 2 CPUs (HyperThreads) = 1 core per MPI task are allocated

task binding … next slides

Example: MPI Job with and w/o HT

```
#!/bin/bash
#SBATCH --job-name=MPIjob
#SBATCH --partition=compute
#SBATCH --nodes=4
#SBATCH --time=00:30:00
#SBATCH --account=x12345

module load intelmpi
export I_MPI_PMI_LIBRARY=/usr/lib64/libpmi.so

# first run without HyperThreading
srun –l --cpu_bind=verbose --hint=nomultithread --ntasks-per-node=24 ./myapp

# second run with HyperThreading
srun –l --cpu_bind=verbose --hint=multithread --ntasks-per-node=48 ./myapp
```

only reservation of 4 nodes without specific allocation

allocation to MPI tasks given for each job-step (srun)

SLURM MPI Support

- many different MPI implementations are supported
  - IntelMPI
  - bullxMPI
  - MVAPICH2
  - OpenMPI
- always use srun to launch the tasks – **no** mpirun, mpiexec, mpiexec.hydra, …
- some MPI option might be set via environmental variables: e.g. I_MPI_DEBUG=4 for Intel MPI
- see DKRZ website and/or user's guide for examples

SLURM MPMD Support

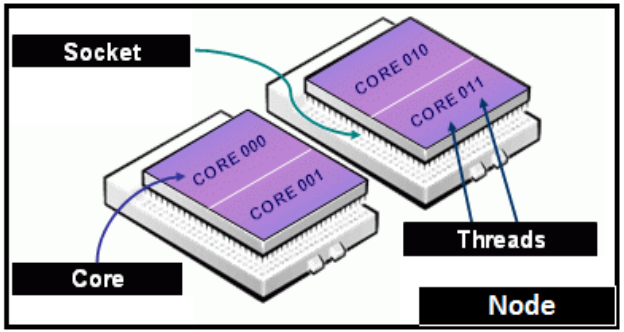
- Different programs may be launched by task ID with different program arguments
- Use “--multi-prog” option and specify configuration file instead of executable program
- Configuration file lists task IDs, executable programs, and arguments (“%t” mapped to task ID and “%o” mapped to offset within task ID range)

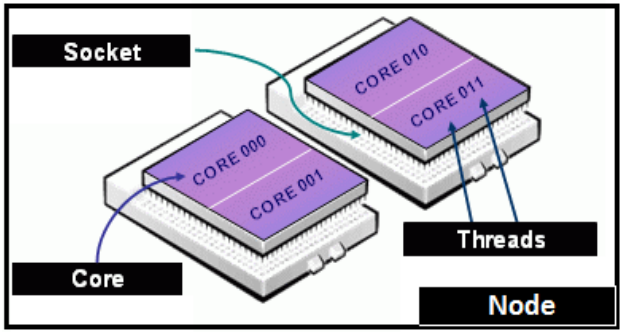
```
> cat mpmd.conf
#TaskID  Program                    Arguments
0-23     /pf/z/z123456/test/ocean
24-47    /pf/z/z123456/test/atmos   --rank=%o

> srun –l --ntasks=48 --multi-prog mpmd.conf
```

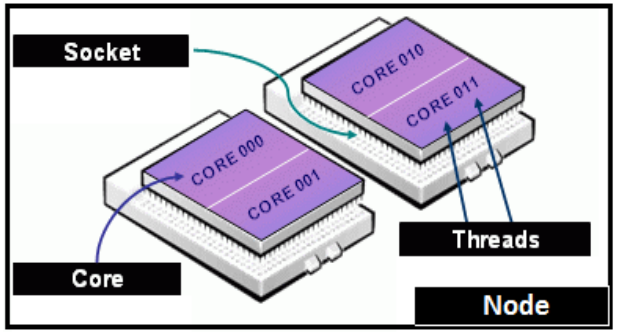
SLURM Process and Thread Binding

- **HSW node** : NUMA board
  - 2 sockets with 12 cores each
  - each core 2 CPUs/HyperThreads
- **Mapping processes** : distribution of ranks on nodes

  srun --distribution=<block|cyclic[:block|cyclic]>
  - first argument: block or cycle on successive nodes
  - second argument: block or cycle on successive sockets

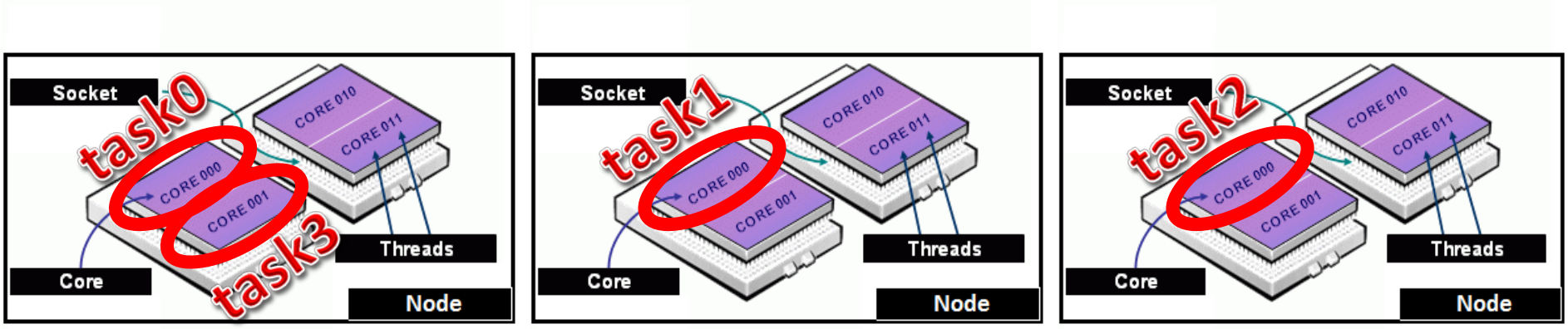






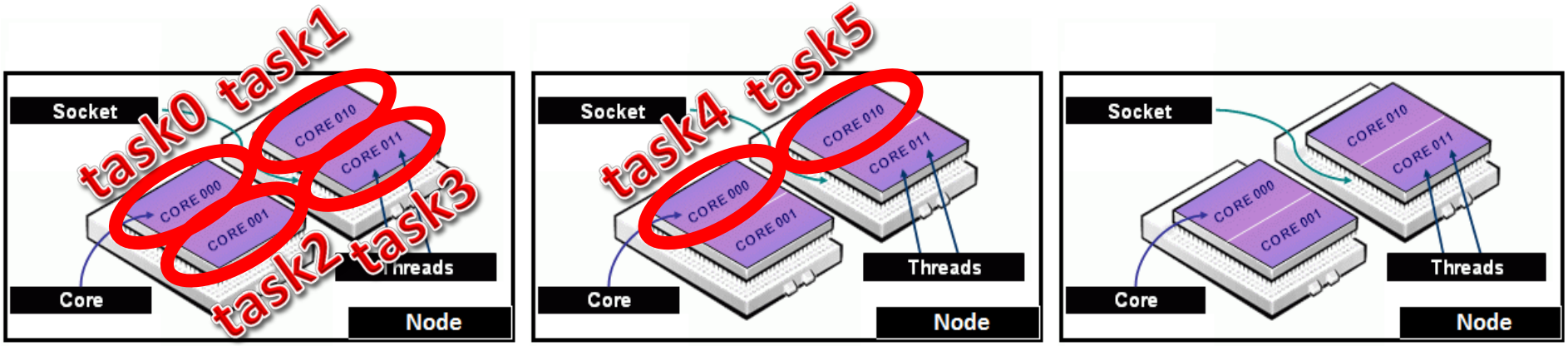
SLURM Process and Thread Binding

srun --distribution=cyclic:block



SLURM Process and Thread Binding

srun --distribution=block:cyclic



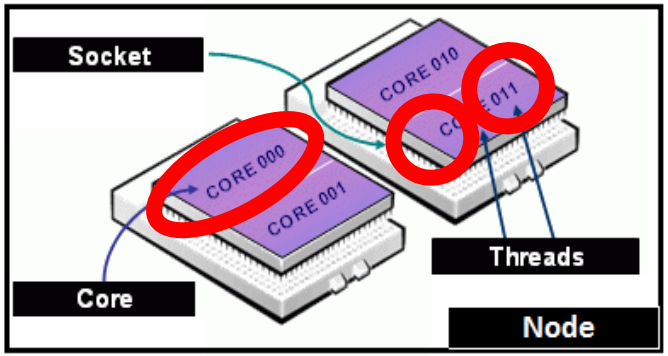
mistral default is block:block

SLURM Process and Thread Binding

- Binding ranks to cores, cpus, ...

  srun --cpu_bind=<[verbose,]type>

  - bind to a single core: <type> = cores
  - bind to a single CPU/HyperThread: <type> = threads
  - custom bindings: <type> = map_cpu:<list>



SLURM Process and Thread Binding

- Binding OpenMP threads to cores, cpus

  export KMP_AFFINITY=[<modifier>,…]<type>[,<permute>]

  - modifier
    - verbose : very helpful !
    - granularity=core : reserve full physical core
    - granularity=thread : reserve logical CPU/HyperThread
  - type
    - compact : place threads as close as possible
    - scatter : distribute threads as evenly as possible
  - permute : sets most significant level of topology map, i.e. 0=CPUs (default), 1=cores, 2=socket/LLC

SLURM Process and Thread Binding

For hybrid MPI/OpenMP combine both:

1 node, 6 MPI tasks and 4 OpenMP threads per task, no use of HyperThreading:

```
#SBATCH --nodes=1
#SBATCH --tasks-per-node=6
#SBATCH --cpus-per-task=8
export OMP_NUM_THREADS=4
export KMP_AFFINITY=granularity=core,compact,1
srun -l --cpu_bind=cores ./myapp
```

SLURM Process and Thread Binding

Usage of HyperThreading allows to double e.g. number of MPI-tasks:

```
#SBATCH --nodes=1
#SBATCH --tasks-per-node=12
#SBATCH --cpus-per-task=4
export OMP_NUM_THREADS=4
export KMP_AFFINITY=granularity=thread,compact,1
srun -l --cpu_bind=threads ./myapp
```

SLURM Epilog Job Report

- further metrics are planned (I/O, MPI, memory bandwidth)
- web based timeseries for detailed analyis (graphite database)

automatic performance issue recognition

- Compare against what?
- you know your job best!

```
* JobID            : 399160
* JobName          : slurm_impi_5.1.sh
* Account          : k20200
* User             : k202082 (21204), k20200 (1567)
* Partition        : compute
* QOS              : normal
* Nodelist         : m[10086-10089] (4)
* AvgCommBlocking  : 0.7500 (in [0,4] less is better)
* Submit date      : 2015-10-06T08:27:22
* Start time       : 2015-10-06T08:27:23
* End time         : 2015-10-06T08:59:28
* Elapsed time     : 00:32:05 (Timelimit=01:00:00)
* Command          : /mnt/lustre01/pf/k/k202082/MPIcomparison/hpcg-2.4/
*                    build_impi/bin/slurm_impi_5.1.sh
* WorkDir          : /mnt/lustre01/pf/k/k202082/MPIcomparison/hpcg-2.4/
*                    build_impi/bin
*
* StepID | JobName      NodeHours ConsEnergy [J]    MaxRSS [Byte] ReqCPUFreq [Hz]
* ------------------------------------------------------------------------------
* 0      | xhpcg             0.54              0     973716K (53)          2.500G
* batch  | batch              2.1              0       7348K (0)               0
* 1      | xhpcg             0.54              0     974172K (41)          2.500G
* 2      | xhpcg             0.53              0     973712K (53)          1.200G
* 3      | xhpcg             0.53              0      973028K (5)          1.200G
* ------------------------------------------------------------------------------
```

Mistral Interactive Usage

Interactive Usage via SLURM

- login to compute nodes allowed if allocation is given
  - for batch jobs

    me@mlogin103:~$ sbatch myjob.sh

    Submitted batch job 375836

    me@mlogin103:~$ squeue –j 375836 –o %N

    NODELIST

    m10665

    me@mlogin103:~$ ssh m10665
  - for interactive jobs via salloc (next slide)
- possible for all nodes, especially prepost and GPU

Interactive Usage via SLURM

```
me@mlogin100:~$ salloc -p prepost –N1
                --exclusive –Ak20200
salloc: Granted job allocation 375494

bash-4.1$ env | grep –i slurm
SLURM_NODELIST=m11543
SLURM_NNODES=1
SLURM_JOBID=375494
SLURM_TASKS_PER_NODE=48
SLURM_SUBMIT_HOST=mlogin100

bash-4.1$ hostname
mlogin100

bash-4.1$ srun –n 2 --label hostname
0: m11543
1: m11543
```

ask for 1 prepost node for exclusive usage

Environment Variables set

Subshell executed on submit host

To execute commands on compute nodes use srun

Interactive Usage via SLURM

```
bash-4.1$ srun --label bash
hostname
0: m11543
exit

bash-4.1$ hostname
mlogin100
bash-4.1$ env | grep JOBID
SLURM_JOBID=375494
bash-4.1$ ssh –X m11543

me@m11543:~$ hostname
m11543
me@m11543:~$ exit
Connection to m11512 closed.
bash-4.1$ exit
salloc: Relinquishing job allocation 375494
```

execute a shell on allocated node

No Prompt is displayed!

Back on submit host
Still inside allocation

Login directly on compute node is possible

Login on Interactive Nodes (new)

- use mistralpp.dkrz.de for interactive work (successor of wizard.dkrz.de)
- no limits set – users might encounter conflicts on resources (especially memory) … mistral login nodes now have cpu time limit of 4 hours per process
- 5 nodes available via round robin:

  m11550 – m11554

Looking for halo.dkrz.de replacement?

- use SLURM allocation on visualization nodes of mistral
- talk by Niklas Röber after break

Summing up: Interactive Usage on Mistral

- like on wizard.dkrz.de before: **shared** resources for all users

```
me@zuhause:~$ ssh <userid>@mistralpp.dkrz.de
me@m11552:~$ who
Sheldon      pts/0      2015-09-25 15:56 (somewhere.out.there)
Leonard      pts/2      2015-09-11 16:28 (somewhere.out.there)
Howard       pts/3      2015-09-22 22:12 (somewhere.out.there)
…
```

- using SLURM allocation on prepost partition: **exclusive** node for you

```
me@zuhause:~$ ssh -t -X <userid>@mistral.dkrz.de \
‘salloc -N1 -pprepost -A<project> -- /bin/bash -c “ssh -t -X \$SLURM_JOB_NODELIST”’
me@m11542:~$ who
me           pts/0      2015-09-28 15:07 (mlogin103.hpc.dkrz.de)
```

Software Environment

Compiler (accessible via modules)

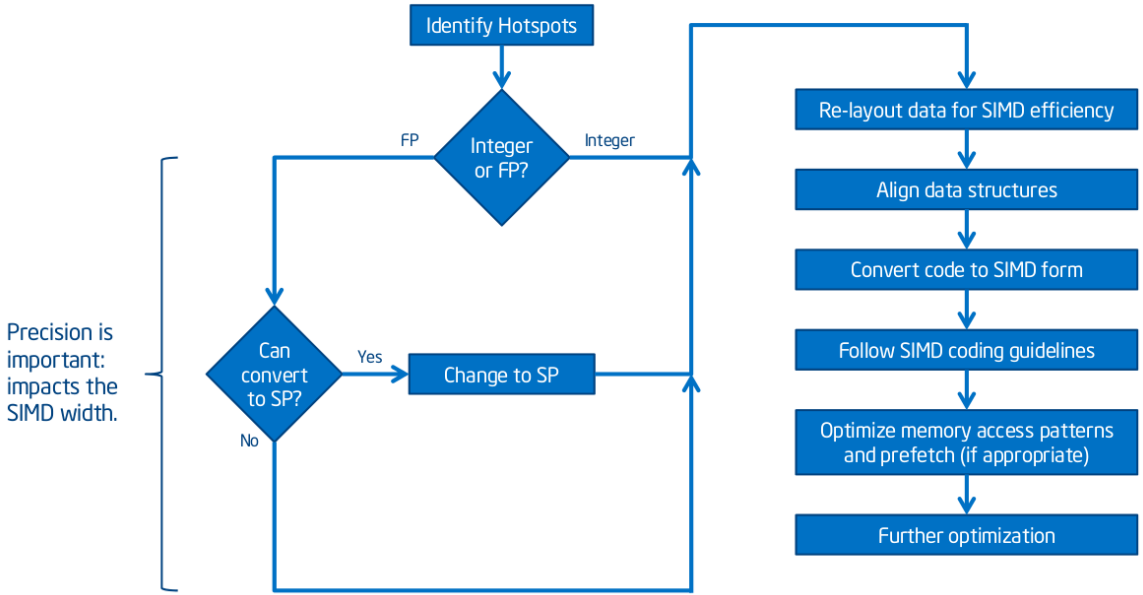
- icc, ifort, icpc 14, 15 and 16 (since last week)
  - to be used for production
  - full support from Intel
- gnu compiler collection
  - 4.8.2 will be system compiler of rhel-7 (phase 2 mistral)
  - 5.x latest for testing
- nag 6.0 for debugging purposes
- pgi 15.7 with OpenACC support for GPGPU computing

Intel Compiler

- CORE-AVX2 vector extensions are crucial for performance on HSW: -xCORE-AVX2 or -xHost
- -fltconsistency (or -mieee-fp) enables improved floating-point consistency, but might cost a factor of 4 since only limited vector operations can be used
- but also compiler might be wrong: performance regression for icc auto vectorization from version 14 to 15 … Intel ticket opened, we keep all old versions

Intel Compiler

- from mistral introduction in July by Micheal Klemm:



- lessons learned: get to know your code better

Intel Compiler

- -fp-model precise or strict only allows for value-safe optimizations – good for testing/debugging, bad for production
- FMA instruction does not round intermediate product result => final result might differ to non-FMA CPUs: -no-fma -fp-model strict
- enabling aggressive vectorization (already –O3) might hinder decomposition independent results due to varying loop lengths

Intel Compiler

FP Model and FMA Summary

debug

production

Key	Value Safety	Expression Evaluation	FPU Environ. Access	Precise FP Exceptions	FMA Use
precise source double extended	Safe	Varies Source Double Extended	No	No	Yes
strict	Safe	Varies	Yes	Yes	No
fast=1 (default)	Unsafe	Unknown	No	No	Yes
fast=2	Very Unsafe	Unknown	No	No	Yes
except except-	*/** *	* *	* *	Yes No	* *

* These modes are unaffected. `-fp-model except[-]` only affects the precise FP exceptions mode.
** It is illegal to specify `-fp-model except` in an unsafe value safety mode.

For more information: “Consistency of Floating-Point Results using the Intel Compiler or Why doesn’t my application always give the same answer?” by Corden and Kreitzer

MPI Versions

- bullxMPI (with and w/o mellanox tools)
  - recommended by BULL (if environment is tuned)
  - full support from BULL
- IntelMPI
  - full support from Intel
  - 5.1 version also with latest DAPL (direct access programming library) for improved MPI message transfer
- mvapich2 and OpenMPI
  - no further support, just for testing and comparison
  - OpenMPI 1.8.4 also CUDA-aware (device memory pointers supported in send and receive APIs)

MPI best Settings and Known Limitations

- bullxMPI and OpenMPI
  - unlimited stacksize might have negative influence on performance
  - better use real needed amount, e.g.

    bash: ulimit -s 102400

    csh: limit stacksize 102400
  - need to be set on all nodes: srun --propagate=STACK
- IntelMPI with DAPL 2.1.6
  - enforce shared memory for MPI intranode and DAPL UD (user datagram) internode communication … improved runtime up to 3%

    export I_MPI_FABRICS=shm:dapl

    export I_MPI_FALLBACK=0

    export I_MPI_DAPL_UD=enable

    export I_MPI_DAPL_UD_PROVIDER=ofa-v2-mlx5_0-1u

MPI best Settings and Known Limitations

- bullxMPI
  - bullxmpi_mlx-1.2.8.3 : OpenMPI 1.6.5 with mellanox tools
  - bullxmpi_mlx_mt-1.2.8.3 : + multithreading support
- How to use:
  - module add intel
  - module add mxm/3.3.3002

    Mellanox Messaging Accelerator: “fully utilizing underlying network infrastructure”
  - module add fca/2.5.2379

    Fabric Collective Accelerator: “significantly reduced MPI collective operations runtime”
  - module add bullxmpi_mlx

MPI best Settings and Known Limitations

- Mellanox tools used via OpenMPI MCA (Modular Component Architecture)
  - Point-to-point management layer

    export OMPI_MCA_pml=cm
  - Matching transport layer (MPI-2 one-sided comm.)

    export OMPI_MCA_mtl=mxm

    OpenMPI default:

    export OMPI_MCA_pml=ob1 # openib

    export OMPI_MCA_mtl=^mxm

MPI best Settings and Known Limitations

- MPI collective algorithms: using FCA (might be useful)

  export OMPI_MCA_coll=^ghc

  export OMPI_MCA_coll_fca_enable=1

  export OMPI_MCA_coll_fca_priority=95

- for all options see: ompi_info --all

“a good environment”

export OMPI_MCA_pml=cm

export OMPI_MCA_mtl=mxm

export OMPI_MCA_coll=^ghc

export MXM_RDMA_PORTS=mlx5_0:1

disable GHC algorithm from BULL for collectives (mistral has full fat tree, no islands)

Any questions?

If not: have fun working on mistral

If not now: contact beratung@dkrz.de

Documentation will be available at

https://www.dkrz.de/Nutzerportal-en/doku/mistral