HLRN-IV User Workshop

Thomas Steinke

ZUSE INSTITUTE BERLIN

Berlin, November 3rd 2020

1. The HLRN-IV System

2. Using the HLRN-IV Complex "Lise"

3. Process & Thread Placement

North German Supercomputing Alliance

hlrn.de

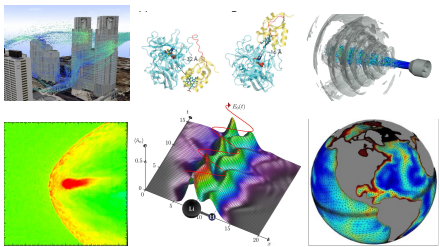
HLRN-IV in (Some) Numbers

16 PFlop/s
peak performance (5× over HLRN-III)

3.5 ×
HLRN Benchmark performance over HLRN-III

3,000+ users

200+ large projects



- **Chemistry** incl. **Material Science**
- **Earth Science** incl. **Climate Research**
- **Engineering**
- **Life Science** (biology, medicine)
- **Physics** incl. **Astro** and **High-Energy Physics**

The HLRN-IV System - Compute Nodes

HLRN Complex "Lise"		HLRN Complex "Emmy"
1270	total compute nodes	**1422**
1236 CLX-AP	**384 (192) GB** standard nodes	**956** CLX-AP + **432** SKL
32 CLX-AP	**768 GB** fat nodes	**16** CLX-AP + **16** SKL
2 CLX-AP	**1.5 TB** huge-mem nodes	**2** CLX-AP
—	GPU nodes	**1** with **4**× **Nvidia V100**

CLX-AP: 2× Intel Xeon Platinum 9242, **96 cores / node**
SKL: 2× Intel Xeon Gold 6148, **40 cores / node**

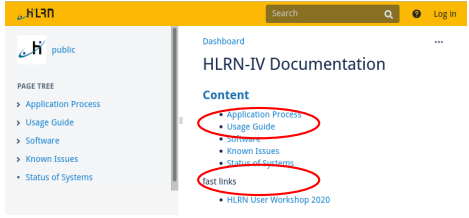
Network: Intel Omni-Path, **100 Gb/s**, **fat tree** topology

The HLRN-IV System - File Systems

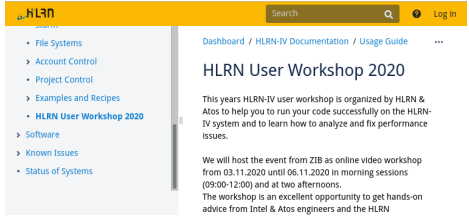
HLRN Complex "Lise"	HLRN Complex "Emmy"
Global File Systems	
HOME: 340 TB, access via NFS (IBM Spectrum Scale Appliance)	
WORK: 8.1 PB, Lustre, aggregated bandwidth 80 GB/s	
48 / 16 MB super stripe size	1 MB stripe size
Intel DAOS, 0.5 PB	**50 TB** Burst Buffer
PERM tape library for archiving, several PBytes	
Node-Local File Systems	
`/tmp`: in-memory local scratch, ≈ **100 GB**	
32-64 nodes, 2 TB local SSD	**480 GB** (Phase 1)

More Workshop Material …

- HLRN User Documentation: `hlrn.de/doc`



- HLRN User Workshop page



empty slide

HLRN-IV User Workshop

Matthias Läuter

ZUSE INSTITUTE BERLIN

Berlin, November 3rd 2020

1 The HLRN-IV System

2 Using the HLRN-IV Complex "Lise"

3 Process & Thread Placement

Hello Lise

- data transfer

```
myPC> ls
mydata.tar ...
myPC> scp mydata.tar -i my_private_key_file myaccount@blogin.hlrn.de:
```

- login

```
myPC> ssh -i my_private_key_file myaccount@blogin.hlrn.de
blogin>
```

- extract examples

```
blogin> cp /sw/userws2020/2020zibws.tar .
blogin> tar xvf 2020zibws.tar
blogin> ls 2020zibws
zibws01 zibws02 zibws03
```

Editor

- X11 forwarding

```
myPC> ssh -X -i my_private_key_file myaccount@blogin.hlrn.de
blogin>
```

- editors

```
blogin> emacs &
blogin> gedit &
blogin> vi
```

Software

- manage shell environment variables (PATH, ...) using `module`
- selected `module` options:

```
> module avail
...
```

list all modules (arranged by HLRN software categories)

```
> module avail intel
intel/18.0.6  intel/19.0.5(default)  intel/19.1.3
```

more selective view

```
> module help intel/19.1.3
```

get specific information

```
> module load intel/19.1.3
```

use Intel compiler

```
> module list
1) sw.skl  2) slurm  3) HLRNenv  4) intel/19.1.3
```

modules currently loaded

Compilation

Example `zibws01`

```
> cd zibws01
```

```
> cat compile.bash

module load intel
module load impi

icc -qopenmp ...

mpiicc ...
```

build script

load the programming environment

OpenMP code

MPI code

```
> ./compile.bash
build hello_omp.bin
build hello_mpi.bin
```

build the code

SLURM Partitions on "Lise"

- Nodes
  - CPU configuration on all nodes: dual-socket CLX-AP, 96 cores
  - main memory configuration

memory [GB]	384	768	1536
node num. [#]	1236	32	2
partition	`standard96`	`large96`	`huge96`

- SLURM partitions

partition	**walltime [h]**	**nodes per job [#]**	
`standard96`	12	512	
`standard96:test`	1	16	high prio
`large96:test`	...		high prio
`huge96`	...		

SLURM commands

- job script

```
> cat myjob.bash
#!/bin/bash ...
```

- job submission

```
> sbatch myjob.bash
Submitted batch job 5504228
```

- job interrupt

```
> scancel 5504228
```

- job status

```
> squeue -u myaccount
JOBID   STATE
5504228 PENDING
```

```
JOBID   STATE
5504228 RUNNING
```

- results

```
> cat slurm-5504228.out
hello world ...
```

SLURM options

- command line for `sbatch`, `srun`, `salloc`

```
> sbatch --nodes=1 ...
```

- job script

```
> cat myjob.bash
#BATCH --nodes=1 ...
```

option	description
`--partition=<name>`	slurm partition name
`--time=<hh:mm:ss>`	max. wallclock time
`--nodes=<value>`	number of nodes
`--ntasks=<value>`	number of proc.
`--ntasks-per-node=<value>`	proc. per node (ppn)

- ... more options

```
> man sbatch
```

Code Execution OpenMP

Example `zibws01`

```
> cat omp.bash

#SBATCH --partition=standard96:test
#SBATCH --time=00:10:00
#SBATCH --nodes=1

export OMP_NUM_THREADS=2                    # number of threads

./hello_omp.bin                             # program call
```

```
> sbatch omp.bash
```

Code Execution MPI

Example `zibws01`

1. variant `srun`

```
> cat mpi_srun.bash
#SBATCH --ntasks=4                  # number of MPI tasks
#SBATCH --ntasks-per-node=2         # MPI tasks per node

srun ./hello_mpi.bin
```

2. variant `mpirun`

```
> cat mpi_mpirun.bash
#SBATCH --nodes=2
module load impi/2019.5             # Intel MPI

export SLURM_CPU_BIND=none          # ignore SLURM placement
export I_MPI_PIN_DOMAIN=core        # use Intel MPI placement
export I_MPI_PIN_ORDER=scatter

mpirun -ppn 2 ./hello_mpi.bin
```

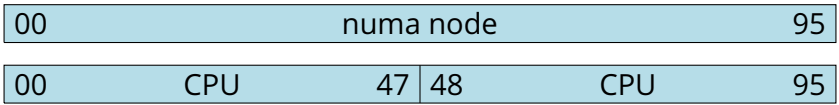
1 The HLRN-IV System

2 Using the HLRN-IV Complex "Lise"

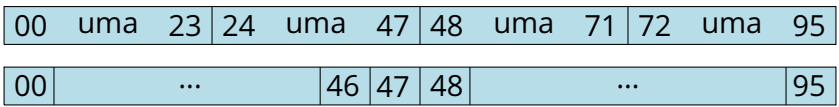
3 Process & Thread Placement

NUMA node with CLX-AP CPU

- NUMA – non uniform memory access



- UMA – uniform memory access
- core – cache access



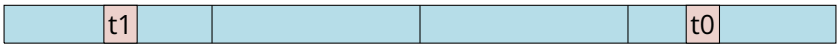
Placement for OpenMP threads

Example `zibws02`

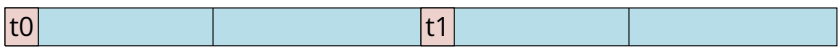
```
> cd zibws02
> ./compile.bash
```

- placement control with `xthi.c`

```
> sbatch omp_xthi.bash
> cat slurm-xxxxxx.out
unset OMP_PROC_BIND                          # placement off
xthi thread 1 (core affinity = 0-191)
xthi thread 0 (core affinity = 0-191)
```




```
export OMP_PROC_BIND=spread                  # placement on
xthi thread 0 (core affinity = 0,96)
xthi thread 1 (core affinity = 48,144)
```



daxpy, OpenMP benchmark

$$y = y + a \cdot x \quad \text{for} \quad x, y \in \mathbb{R}^n,\ a \in \mathbb{R}$$

- floating point operations $+, \cdot$ memory operations read/write

$$N_{\text{dp}} = 2n\,\text{Flop} \qquad N_{\text{mem}} = 3n \cdot 8\,\text{Byte}$$

- arithmetic intensity

$$I_{\text{a}} = \frac{N_{\text{dp}}}{8\,N_{\text{mem}}} = \frac{1}{12}\frac{\text{Flop}}{\text{Byte}}, \qquad \frac{N_{\text{dp}}}{T} = I_{\text{a}} \cdot \frac{N_{\text{mem}}}{T}$$

- HLRN node, 2× Intel Xeon 9242

$$\frac{N_{\text{dp}}}{T} < 3500\,\text{GFlop/s}, \qquad \frac{N_{\text{mem}}}{T} < 456\,\text{GB/s}$$

daxpy performance

Example `zibws02`

- daxpy results

```
> sbatch omp_daxpy.bash
> cat slurm-xxxxxx.out

placement off
threadnum= 96 : GFlop/s= 18.95 GB/s= 227.43

placement on
threadnum= 96 : GFlop/s= 32.93 GB/s= 395.26
```

First touch policy

1. OpenMP threads pinned to CPU cores
2. Is memory access distributed to all threads?

- array initialization

```
#pragma omp parallel for
for(i=0; i<n; i++) {
 x[i] = 0.0;
 y[i] = 0.0;
}
```

- working loop

```
#pragma omp parallel for
for(i=0; i<n; i++)
 y[i] = y[i] + a * x[i];
```