Student HPC Hackathon 8/2018

J. Simon, C. Plessl

22. + 23. August 2018

• Get the most performance out of our new HPC system Noctua1
• Performance is measured with given Benchmark programs (+ your favorite App)
• Form groups of about max. two individuals
• HPC experts from PC² will support you

Agenda
– 1st day starts with
  • an introduction to the Noctua1 system, and
  • an overview of the Benchmark programs and running rules
– Do your performance optimizations/measurements
– 2nd day ends with
  • a presentation of the results of each group

Date
– August, 22nd + 23rd, 10:00-16:00 / 9:00-16:00
– Room 02.267
Student HPC Hackathon 8/2018

Challenges

- **New Hardware**
  - Processor architecture (cache hierarchy, AVX512 instructions,..)
  - Communication network (OmniPath 100Gbps)
  - Storage System
- **New Software**
  - Programming Tools
  - Libraries
  - Resource management system “SLURM”
  - Lustre file system
- **Planning your batch jobs**
  - Optimal use of available resources

Noctua1

Frontside: Cold air intake

Backside: Cooled backdoors
Noctua1: Cray CS 500 Storm

- **256 compute nodes**
  - 2x Intel Xeon Gold 6148, each 20C, 2.4 GHz
  - 192 GiB
- **16 FPGA nodes**
  - Intel Xeon 6148+6148F, 192 GiB
  - each with 2 Nallatech Stratix 10
- **Parallel file system**
  - Lustre
  - 720 TB disk capacity
- **Interconnect Intel Omni-Path**
  - 100 Gbit/s network
  - Blocking factor 1:1.4

Noctua1: Architecture

![Diagram of Cray CS 500 Storm architecture](image)
Noctua1: Rack Layout

5x water cooled rack
1x air cooled rack
~ 161 kW

Intel Xeon Scalable Processor

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket</td>
<td>Socket P</td>
</tr>
<tr>
<td>Scalability</td>
<td>2S, 4S, 8S, and &gt;8S (with node controller support)</td>
</tr>
<tr>
<td>CPU TDP</td>
<td>70W – 205W</td>
</tr>
<tr>
<td>Chipset</td>
<td>Intel® C620 Series (code name Levisburg)</td>
</tr>
<tr>
<td>Networking</td>
<td>Intel® Omni-Path Fabric (integrated or discrete)</td>
</tr>
<tr>
<td></td>
<td>4x100Gbe (integrated w/chipset)</td>
</tr>
<tr>
<td></td>
<td>100G/40G/25G discrete options</td>
</tr>
<tr>
<td>Compression and Cryptos Acceleration</td>
<td>Intel® QuickAssist Technology to support 100G/s compress/decompress</td>
</tr>
<tr>
<td></td>
<td>100G RSA/2K public key</td>
</tr>
<tr>
<td>Storage</td>
<td>Integrated QuickData Technology, VMD, and NTB Intel® Optane® SSD, Intel® 3D-NAND NVMe &amp; SATA SSD</td>
</tr>
<tr>
<td>Security</td>
<td>CPU enhancements (MBE, PPI, MX) Manageability Engine Intel® Platform Trust Technology Intel® Key Protection Technology</td>
</tr>
<tr>
<td>Manageability</td>
<td>Innovation Engine (IE) Intel® Node Manager Intel® Datacenter Manager</td>
</tr>
</tbody>
</table>
OmniPath – Integrated Fabric

Intel Fabric through carrier card  Intel Fabric passive cable  CPU with integrated Fabric

Noctua1 vs. OCuLUS

<table>
<thead>
<tr>
<th></th>
<th>Noctua1</th>
<th>OCuLUS</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td># nodes</td>
<td>272</td>
<td>616</td>
<td>- 56%</td>
</tr>
<tr>
<td># cores per node</td>
<td>2x20</td>
<td>2x8</td>
<td>x 2.5</td>
</tr>
<tr>
<td>Total # cores</td>
<td>10.880</td>
<td>9.856</td>
<td>+ 10%</td>
</tr>
<tr>
<td>Memory per node [GiB]</td>
<td>192</td>
<td>64</td>
<td>x 3</td>
</tr>
<tr>
<td>Total memory [TiB]</td>
<td>52,2</td>
<td>41,2</td>
<td>+ 27%</td>
</tr>
<tr>
<td>HP-Linpack [TFlop/s]</td>
<td>&gt; 535</td>
<td>188,7</td>
<td>x 2.8</td>
</tr>
<tr>
<td>Accu. STREAM [GiB/s]</td>
<td>50.150</td>
<td>43.700</td>
<td>+ 15%</td>
</tr>
<tr>
<td>Accu. SpecFP 2006 base</td>
<td>381.000</td>
<td>295.700</td>
<td>+ 29%</td>
</tr>
<tr>
<td>Accu. SpecINT 2006 base</td>
<td>516.800</td>
<td>381.900</td>
<td>+ 35%</td>
</tr>
<tr>
<td>Scaled Spec MPI</td>
<td>&gt; 560</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>MPI latency [µs]</td>
<td>&lt;1,34</td>
<td>2,1</td>
<td>- 36%</td>
</tr>
<tr>
<td>MPI bandwidth [GiB/s]</td>
<td>24,5</td>
<td>7</td>
<td>x 3.5</td>
</tr>
</tbody>
</table>

| Storage Capacity [TB] | 720 | 500 | + 44%  |
| Storage bandwidth [GiB/s] | 20 | 25 | - 20% |
| MPI network blocking fact. | OPA 1:1,4 | IB 1:2 |
| MPI network [Gbps]    | 100  | 40   | x 2.5  |
| Power consumption[kW] | 164  | 230  | - 29%  |
Installation Plan

Floorspace of DataCenter Building-O

Legende

NOCTUA: Kühlplatz
Netzwerkschrank
vorhandene Rechnersysteme
Entwurf-Entwurf
LED-Entwurf
Klimacontroller
Zub. elektrische
Unterteilung (20089)
Übergangsartikel Klimacontroller
PC2 Environment

- IMT Account
  - Member of group **HPC-LCO-SIMON**
  - **fileservice** activated (IMT serviceportal)
- Login server and Noctua1 frontend
  - fe-1....
- Standard Environment Settings
  - PC2FS
    - $PC2SW Software installed by PC2
    - $PC2DATA/HPC-LCO-SIMON shared data of group HPC-LCO-SIMON
    - $PC2SCRATCH/HPC-LCO-SIMON/<user> shared filesystem OCuLUS / Noctua1
    - $PC2PFS/HPC-LCO-SIMON Lustre parallel filesystem (use individual dirs)
    - $PC2SYSNAME Noctua
  - SLURM
    - Version 17.11.8

Program Development Environments

- Intel Parallel Studio Cluster Edition ([modulefile intel/18.0.3])
  - C/C++ and Fortran
  - Python
  - Intel Math Kernel Libraries (MKL)
  - Intel MPI
  - Intel Data Analytics Acceleration Library, Integrated Performance Primitives, Threading Building Blocks
  - Intel VTune, Advisor, Inspector, Trace Analyzer and Collector
- Cray Compiler Environment "CCE"
  - C/C++ and Fortran 2008
  - OpenMP 4.1, MPI 2.2, UPC 1.2, OpenACC 2.0
  - LibSci, LibSci_ACC
- Cray Performance Measurement, Analysis, and Porting Tools
  - Performance and Analysis Tool **CrayPAT**
  - Visualization Tool **Cray Apprentice2**
  - Porting Tool **Cray Reveal**
Cray Performance Tools

- CrayPAT profiles executables
  - Timing and hardware performance counter measurements
  - Collect and show program top time consumers and bottlenecks
  - Automatic generation of observations and suggestions
  - Data collection and presentation of computation, communication, I/O, and memory statistics
  - CrayPAT lite is a simplified, easy-to-use version of CrayPAT
- Visualization of performance data with Cray Apprentice2
  - Reports and graphical formats
  - GUI
  - Runs on Windows, MacOS, and Linux using the platform-independent data files
- Code-restructuring assistant Reveal
  - Helps developers to add additional levels of parallelism
  - Assists with parallelizing more complicated loops
  - Combining performance statistics and program source code

SLURM

- User commands
  - sacct, salloc, sattach, sbatch, sbcast, scancel, scontrol, sinfo, smap, squeue, srn, strigger, sview
- Managed entities
  - Jobs (allocation of resource assigned to a user for a specified amount of time)
  - Job steps (sets of parallel tasks within a job)
  - Resources are nodes, processors, memory, ...
  - Nodes are logically organized into (possibly overlapping) partitions
**SLURM Quick Start**

- Information about the system
  
  > `sinfo`

  
<table>
<thead>
<tr>
<th>PARTITION</th>
<th>AVAIL</th>
<th>TIMELIMIT</th>
<th>NODES</th>
<th>STATE</th>
<th>NODELIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute*</td>
<td>up</td>
<td>infinite</td>
<td>16</td>
<td>alloc</td>
<td>cn-[0001-0016]</td>
</tr>
<tr>
<td>compute*</td>
<td>up</td>
<td>infinite</td>
<td>240</td>
<td>idle</td>
<td>cn-[0017-0256]</td>
</tr>
<tr>
<td>fpga</td>
<td>up</td>
<td>infinite</td>
<td>16</td>
<td>idle</td>
<td>fpga-[0001-0016]</td>
</tr>
<tr>
<td>all</td>
<td>up</td>
<td>infinite</td>
<td>16</td>
<td>alloc</td>
<td>fpga-[0001-0016]</td>
</tr>
<tr>
<td>all</td>
<td>up</td>
<td>infinite</td>
<td>256</td>
<td>idle</td>
<td>cn-[0017-0256],fpga-[0001-0016]</td>
</tr>
</tbody>
</table>

- State of jobs
  
  > `squeue`

  
<table>
<thead>
<tr>
<th>JOBID</th>
<th>PARTITION</th>
<th>NAME</th>
<th>USER</th>
<th>ST</th>
<th>TIME</th>
<th>NODES</th>
<th>NODELIST(REASON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>11618</td>
<td>compute</td>
<td>my.scrip</td>
<td>jens</td>
<td>PD</td>
<td>0:00</td>
<td>1</td>
<td>(Resources)</td>
</tr>
</tbody>
</table>

- Accounting information about active and completed jobs
  
  > `sacct`

  
<table>
<thead>
<tr>
<th>JOBID</th>
<th>JobName</th>
<th>Partition</th>
<th>Account</th>
<th>AllocCPUs</th>
<th>State</th>
<th>ExitCode</th>
</tr>
</thead>
<tbody>
<tr>
<td>11617</td>
<td>IMB-MPI1</td>
<td>compute</td>
<td></td>
<td>120</td>
<td>COMPLETED</td>
<td>0:0</td>
</tr>
<tr>
<td>11618</td>
<td>my.scrip</td>
<td>compute</td>
<td></td>
<td></td>
<td>PENDING</td>
<td></td>
</tr>
</tbody>
</table>

**SLURM Quick Start (2)**

- More information about nodes, partitions, jobs, job steps, configurations
  
  > `scontrol show partitions`

  PartitionName=compute
  AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
  AllocNodes=ALL Default=YES QoS=N/A
  DefaultTime=NONE DisableRootJobs=NO ExclusiveUser=NO GraceTime=0 Hidden=NO
  MaxNodes=UNLIMITED MaxTime=UNLIMITED MinNodes=1 LLN=NO MaxCPUsPerNode=UNLIMITED
  Nodes=cn-[0001-0256]
  PriorityJobFactor=1 PriorityTier=1 RootOnly=NO ReqResv=NO OverSubscribe=NO
  OverTimeLimit=NONE PreemptMode=OFF
  State=UP TotalCPUs=10240 TotalNodes=256 SelectTypeParameters=NONE
  DefMemPerNode=UNLIMITED MaxMemPerNode=UNLIMITED

- Resource allocation and launch the tasks for a job step in a single command
  
  > `srun –N3 –l /bin/hostname`

  cn-0019
  cn-0018
  cn-0017
SLURM Quick Start (3)

- Submit a script for later execution

```bash
> cat my.script
#!/bin/bash
#SBATCH --time=1
/bin/hostname
srun -l /bin/hostname
srun -l /bin/pwd
```

```bash
> sbatch -N4 -o my.stdout my.script
sbatch: Submitted batch job 1234
```

```bash
> cat my.stdout
0: cn-0001
1: cn-0002
2: cn-0003
3: cn-0004
2: /upb/departments/pc2/user/j/jens/Tests/SLURM
0: /upb/departments/pc2/user/j/jens/Tests/SLURM
1: /upb/departments/pc2/user/j/jens/Tests/SLURM
3: /upb/departments/pc2/user/j/jens/Tests/SLURM
```

SLURM Quick Start (4)

- MPI batch script

```bash
#!/bin/bash
# Example with 80MPI tasks and 40tasks per node.
#
# Project/Account (use your own)
#SBATCH -A hpc2n-1234-56
#
# Number of MPI tasks
#SBATCH -n 80
#
# Number of tasks per node
#SBATCH --tasks-per-node=40
#
# Runtime of this jobs is less then 1 hours.
#SBATCH --time=1:00:00
module load intel/18.0.3
srun ./mpi_program
```

```bash
# End of submit file
```
SLURM Quick Start (5)

- Resource allocation and spawn job steps within that allocation

> `salloc -N 4 bash`

$ `sbatch a.out /tmp/pi`

*Granted job allocation 1234*

$ `srun /tmp/pi`

*Result is 3.14159*

$ `srun rm /tmp/pi`

$ `exit`

`salloc: Relinquishing job allocation 1234`