Vectorisation and Portable Programming using OpenCL

Andreas Beckmann, Ilya Zhukov, Willi Homberg, JSC
Wolfram Schenck, FH Bielefeld

Course no. 95/2017 in the trainings programme of Forschungszentrum Jülich
Supercomputer Systems: Dual Track Approach

IBM Power 4+ JUMP, 9 TFlop/s
IBM Blue Gene/L JUBL, 45 TFlop/s
IBM Blue Gene/P JUGENE, 1 PFlop/s
IBM Blue Gene/Q JUQUEEN 5.9 PFlop/s

IBM Power 6 JUMP, 9 TFlop/s
JUROPA 200 TFlop/s
HPC-FF 100 TFlop/s

JURECA Cluster (2015) 2.2 PFlop/s
JURECA Booster (2017) ~ 5 PFlop/s
JURECA Cluster Successor Module (2020)

Modular Data Centric System
General-Purpose Systems
Hierarchical Storage Server
Highly Scalable Systems

Modular Tier-0/1 System towards Exascale

GCS Cluster Module (2018) > 10 PFlop/s
GCS Booster Module (2019) ~ 20 ... 40 PF

20. November 2017
JURECA – Juelich Research on Exascale Cluster Architectures

General purpose cluster

- 2.24 Pflop/s
- #50 Top500
- 1872 compute nodes
  - Two Intel Xeon E5-2680 v3 Haswell CPUs per node
  - 2 x 12 cores, 2.5 GHz
- 75 compute nodes equipped with two NVIDIA K80 GPUs
  - 2 x 4992 CUDA cores (4 visible devices per node)
  - 2 x 24 GiB GDDR5 memory
- 12 visualization nodes
  - Two Intel Xeon E5-2680 v3 Haswell CPUs per node
  - Two NVIDIA K40 GPUs per node
  - 2 x 12 GiB GDDR5 memory
JUROPA-3 Accelerated Nodes
(NVIDIA Tesla K20x, Intel Xeon Phi 5110P)

4x K20x accelerated compute nodes
- 2 Intel Xeon E5-2650 (Sandy Bridge-EP)
- eight-core processors
  - 2.0 GHz
  - SMT (Simultaneous Multithreading)
- 64 GB memory (DDR3, 1600 MHz)
- IB FDR HCA
- 2 NVIDIA Tesla K20X GPU
  - 2688 CUDA cores
  - 6 GB memory

4x Xeon Phi accelerated compute nodes
- 2 Intel Xeon E5-2650 (Sandy Bridge-EP)
- eight-core processors
  - 2.0 GHz
  - SMT (Simultaneous Multithreading)
- 64 GB memory (DDR3, 1600 MHz)
- IB FDR HCA
- 2 Intel Xeon Phi 5110P
  - 60 cores
Agenda Day 1

09:00   Welcome
09:15   Introduction to parallel computing
10:00   Login to test systems and query OpenCL capable devices
10:30   Coffee break
10:45   OpenCL programming concepts (I)
11:45   Lunch at Casino
12:45   OpenCL programming concepts (II)
13:45   Exploit OpenCL vectorisation features
14:45   Coffee break
15:00   Example: parallel reduction
Agenda Day 2

9:00  Portable performance
10:30 Coffee break
10:45 Matrix multiplication (I)
11:45 Lunch at Casino
12:45 Matrix multiplication (II)
13:45 Heterogeneous multi-device programming
14:45 Coffee break
15:00 Appendix
   OpenCL: History & Future
   OpenCL compared with CUDA
Acknowledgements

- Many previous presentations on OpenCL have been read in preparation for this course. You are encouraged to look at these:
  - OpenCL – An Introduction for HPC Programmers, ISC2011, Tim Mattson, Intel, Udeepta Bordoloi, AMD
  - Introduction to OpenCL, Training course June 2012, George Leaver, University of Manchester
  - OpenCL: An Introduction, Simon McIntosh-Smith, University of Bristol
    - http://handsonopencl.github.io
  - NVIDIA OpenCL SDK
  - AMD Accelerated Parallel Processing SDK
  - Intel SDK for OpenCL Applications
Course test platforms

- OpenCL – Open Computing Language
  - Open, royalty-free standard
  - For cross-platform, parallel programming of modern processors
  - An Apple initiative
  - Specified by the Khronos group
- Code hardware agnostic and portable (contrary to CUDA)
- Intended for accessing heterogeneous computational resources
  - CPUs (Intel processors, Intel Xeon Phi coprocessors (KNC), ...)
  - GPUs (NVIDIA Fermi, Kepler, ...)
- not available during course:
  - AMD APUs/CPUs/GPUs, ARM SoCs, FPGAs, DSPs, ...
JURECA – Juelich Research on Exascale Cluster Architectures

General purpose cluster

- 2.24 Pflop/s
- #50 Top500
- 1872 compute nodes
  - Two Intel Xeon E5-2680 v3 Haswell CPUs per node
  - 2 x 12 cores, 2.5 GHz
- 75 compute nodes equipped with two NVIDIA K80 GPUs
  - 2 x 4992 CUDA cores (4 visible devices per node)
  - 2 x 24 GiB GDDR5 memory
- 12 visualization nodes
  - Two Intel Xeon E5-2680 v3 Haswell CPUs per node
  - Two NVIDIA K40 GPUs per node
    - 2 x 12 GiB GDDR5 memory
JUROPA-3 – JUROPA Update Prototype System

Heterogeneous prototype system: 60 compute nodes

- 52 nodes
  - 2 Intel Xeon E5-2650 (Sandy Bridge)
  - 8-core processor, 2-2.8 GHz
  - 64 – 256 GB memory
- 4 nodes
  - 2x Sandy Bridge, 64 GB memory
  - 2x NVIDIA K20x GPUs (Kepler)
  - 3.94/1.31 TFLOPS SP/DP
- 4 nodes
  - 2x Sandy Bridge, 64 GB memory
  - 2x Intel Xeon Phi 5110P
  - 2.02/1.01 TFLOPS SP/DP
OpenCL Course Cheat Sheet

Workstations
Logging in
Username: train0??, where ?? is the number assigned to you.
Password: see whiteboard
Here, you find course materials under: -train060/OpenCL_Course
Please copy the required files to the remote systems.

Editing
On JURECA and JUROPA3 you will find the same GPFS-based HOME filesystem!
kate provides remote editing, thus you could use:
sftp://juropa3/homea/hpclab/train0??
to access your files on JURECA and JUROPA3 and edit them locally.

Command line
Start a terminal window: <ALT>+<F2> opens a window to enter commands, then type konsole:
cd <dir> - switches the working directory
ls - lists files in current directory
cp - same as ls but gives more detail
rn <file> - deletes (removes) a file.
Cannot be undone!
ls <file> - shows the context of a file.

JURECA
Logging in:
Type `ssh-add` and passphrase/word then
`ssh jureca.zam.kfa-juelich.de`
Choosing an SDK
OpenCL comes with the NVIDIA platform:
module load CUDA
Accessing a compute node
JURECA uses the slurm batch system.
Allocate compute resources:
salloc --nodes=1 --gres=gpu:4
"partition=gpus
"reservation=openc1
Start interactive shell on allocated node:
srun --pty /bin/bash -i
Device query
For all test systems you find a script under
-train060/OpenCL_Course/Device_Query
to query all available OpenCL capable GPU devices
module load CUDA
make
export CUDA_VISIBLE_DEVICES=0,1,

JUROPA3
Logging in:
Type `ssh-add` and passphrase/word then
`ssh juropa3.zam.kfa-juelich.de`
Accessing a compute node
We have a reservation of 4 NVIDIA K20X-based GPU
nodes and 4 Intel Xeon Phi accelerated MIC nodes.
JUROPA3 uses the slurm batch system:
salloc -N1 -n16 -p q_gpus --reservation=openc1_gpus
or
salloc -N1 -n16 -p q_mics --reservation=openc1_mics
then
srun --pty /bin/bash -i
Device query
For all test systems you find a script under
-train060/OpenCL_Course/Device_Query
to query all available OpenCL capable devices on an
MIC node
module load Intel
make
LIB=/usr/local/intel/OpenCL/intel/
opencl-1.2-3.2.1.16712/lib64
export LD_LIBRARY_PATH=$LIB
Exercise OpenCL_Platforms/device-query

- Login to all test systems (cf. Cheat Sheet on previous slide)
  - JURECA node
  - JUROPA-3 GPU node
  - JUROPA-3 MIC node
- and query platform information to find all OpenCL capable devices
  - copy local directory Device_Query to the remote host
  - inspect corresponding run script and run it
    - ./run_device_query_*.sh
OpenCL Architecture: Platform Model

- A host is connected to one or more (possibly heterogeneous) OpenCL devices
  - A device is divided into compute units (CUs)
  - CUs are subdivided into processing elements (PEs)