Chair of Computer Architecture and Parallel Systems
Department of Informatics
Technical University of Munich

BEAST Lab Organization

LRZ

Dr. Josef Weidendorfer, josef.weidendorfer@lrz.de

Amir Raoofy, amir.raoofy@tum.de

LMU

Sergej Breiter, sergej.breiter@nm.ifi.lmu.de

Minh Thanh Chung, minh.thanh.chung@ifi.lmu.de

Dr. Karl Fürlinger, karl.fuerlinger@ifi.lmu.de

TUM

Bengisu Elis, bengisu.elis@tum.de

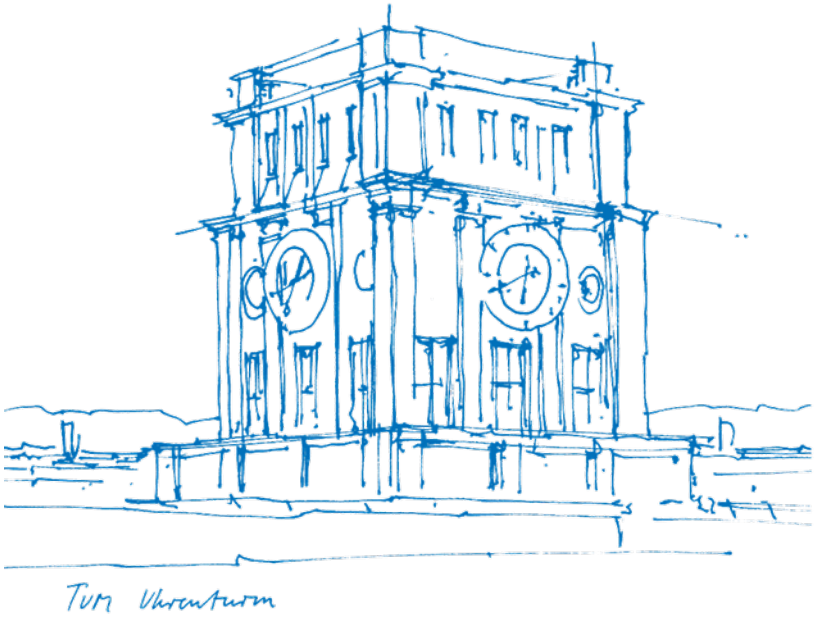


Table of Contents

Tentative Course Structure

- 14 meetings in total (via Zoom and in class)
  - Last meeting on 9th February 2022
  - Dec 1 - TUM Dies Academicus
- 9 Assignments (Assignment 0 : optional)
  - 1 week each
- 1 Project
  - 2 weeks
- Student groups of 3 BA students and 2 MA students
- Two groups will present their reports at the meeting right after deadline
  - Presentation notification will be sent 2 days before presentations
  - No slides, go through your report and talk about your findings

Repository Structure

Gitlab main repository: `https://gitlab.lrz.de/beastlab22ws`

- Each team has a repository, which includes:
  - Lecture slides
  - Assignment material
  - Code template
  - Your code submissions and report (once you place it there)
- Only solutions on the `main` branch will be graded !
  - At the due date, you current master branch state is automatically tagged and archived
  - Make sure your code and report is there by the deadline
- Groups are already set.
- Machine account information will be sent via e-mail - Change passwords please !

Infrastructure Usage

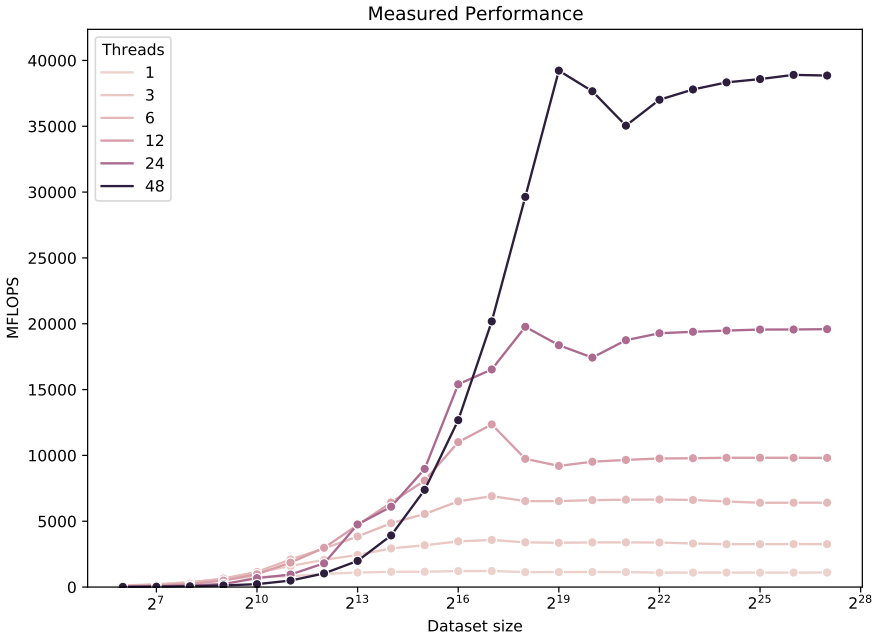
GitLab

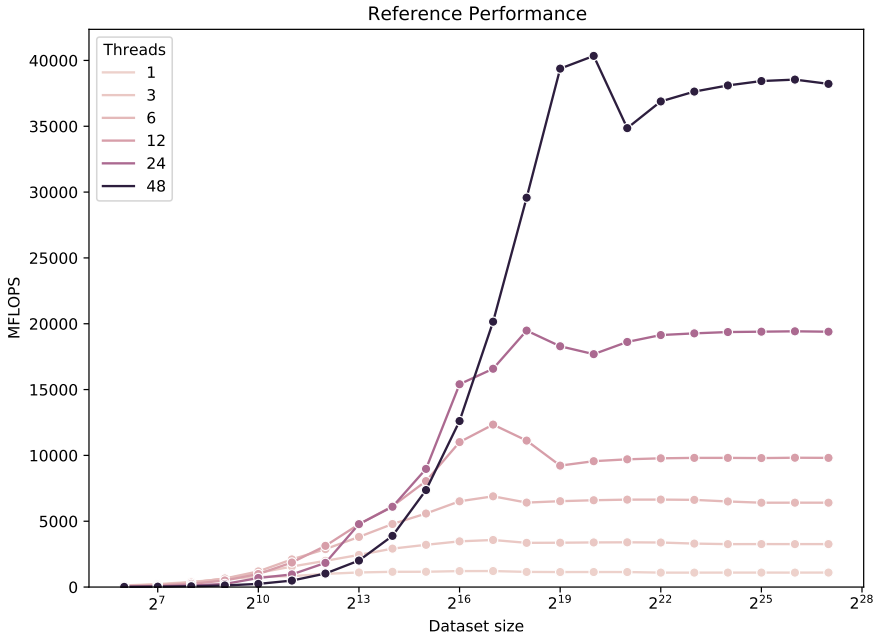
- We need you to sign your commits.
- Tutorial: `https://docs.gitlab.com/ee/user/project/repository/gpg_signed_commits/`
- If GitLab shows a `Verified` label on your commits you are good to go.
- We will tag your last commit before each deadline.

Continuous Integration

- Allows for rapid feedback to your code solutions, helps you get set up.
- Code runs on Fujitsu machines for CPU code . Performance data is automatically collected and visualized.

Sample CI Visualization

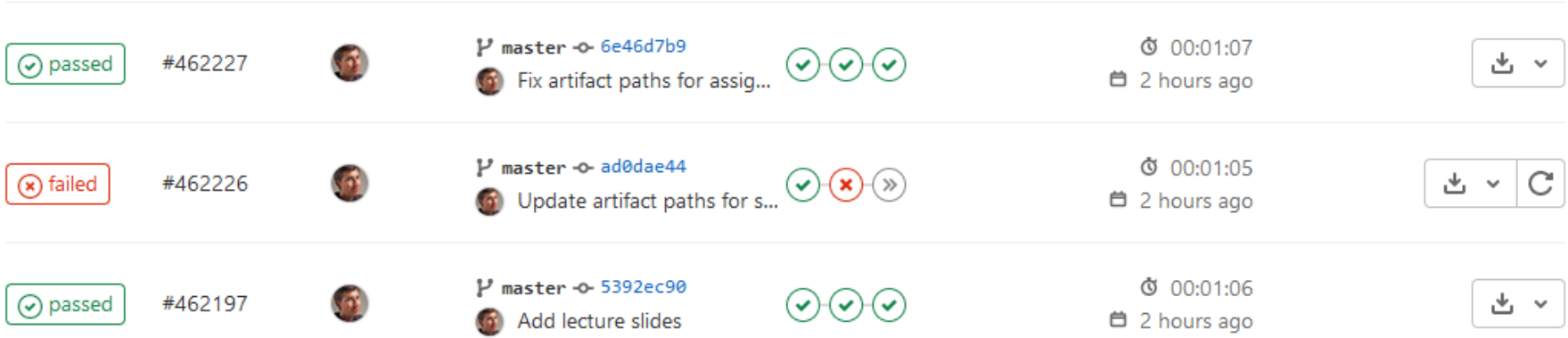




Performance data is stored in the build job artifacts (right hand column when viewing build job).

CI Usage

Benefit: Shows you when your code fails to produce correct results or performance.



CI ensures you have the measuring basics correct. You still need to conduct most of the experiments and explain your findings.

Up Next: Introduction to BEAST

BEAST Lab

Introduction to Systems

April 28, 2022 | Josef Weidendorfer

Collaboration among 3 institutions

LMU
TUM
LRZ

LMU – MNM/Prof. Kranzlmüller
(Karl Fürlinger, Minh Chung, Sergej Breiter)

TUM – CAPS/Prof. Schulz
(Bengisu Elis, Dennis-Florian Herr, Vincent Bode)

LRZ - Future Computing Group
(Josef Weidendorfer, Amir Raoofy)

Focus: Experimental Evaluation

We want you to learn about **performance properties of current architectures**

- Be able to understand and explain performance effects seen from measurements
- Get a deeper understanding of current system designs (CPU / GPU)

Part 1: get started with small codes across systems

- We show key hardware design concepts + a parallel programming model (OpenMP)
- We give you typical small HPC code examples
- You run measurements of different scenarios across systems, compare / discuss results
- We all discuss results in weekly meetings, starting with presentations of groups

Structure:

Memory on CPU (Triad / Traversal) ➔ Compute on CPU (MM) ➔ … on GPU ➔ Tools

Focus: Experimental Evaluation

We want you to learn about **performance properties of current architectures**

- Be able to understand and explain performance effects seen from measurements
- Get a deeper understanding of current system designs (CPU / GPU)

Part 2: make use of gained knowledge

- We assign randomly one system to each group
- We give you some larger typical HPC code examples
- You tune the code to get best single-node performance (3 week time)
- We discuss intermediate/final experiences/results in weekly meetings

Evaluation of Single-Node Performance

Target Architectures for the Lab

CPUs
- Intel Icelake (ISA: x86-64 + AVX512)
- AMD Rome (ISA: x86-64 + AVX2)
- Marvell ThunderX2 (ISA: ARM AArch64 + Neon)
- Fujitsu A64FX (ISA: ARM AArch64 + SVE)

GPUs
- NVidia V100
- AMD MI-100

SuperMUC-NG
Top500 (Nov 2018): #8
Lenovo Intel (2019)
311,040 cores
Intel Xeon Skylake
26.9 PetaFlops Peak
19.5 PetaFlops Linpack
719 TeraByte Main Memory
70 PetaByte Disk

BEAST – Bavarian Energy Architecture and Software Testbed

The LRZ Future Computing Testbed

Testbed Objectives

- Help decide about next large system
  - Get experience on benefits of various future architectures for LRZ codes
  - Find best configuration: how much money to spend on compute / memory / network?
  - Enable migration planning: educate own staff / port LRZ tools / prepare courses
  - Support vendor collaboration

- Enable research studies on new technologies
  - Forward looking: LRZ services around future platforms, novel usage models
    - more experimental: FPGAs, AI accelerators, integration of heterogeneity (QC)
  - In partnership with selected researchers from Munich universities

Lot of work to do! Engage students for student work (BA, MA): This Lab!

The Testbed – Available Hardware

2 racks, each with 6 PDUs (for power measurements)

- Max power consumption per rack: 35 kW

Top to bottom (picture from last year)

- 3 switches (Infiniband 200Gb/s HDR), 2x 48port 1Gb/s Ethernet
- Login 1U “testbed.cos.lrz.de”
- **2x AMD Rome** GPU server 2U: “rome1” **/ “rome2”**
- Storage 2U with homes
- **2x Marvell ThunderX2** GPU server 2U: “thx1” **/ “thx2”**

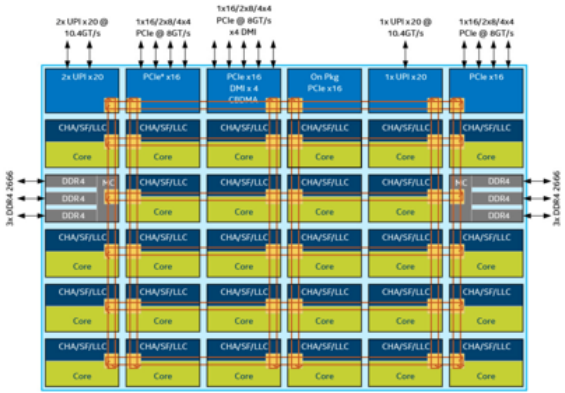
Not shown:

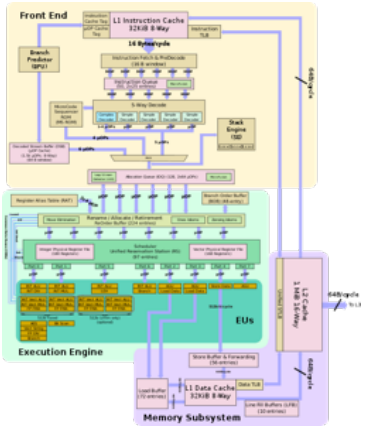
- HPC CS500 Management 2U + **8 nodes A64FX “cs1”** – “cs8”
- **2x Intel IceLake** GPU server 2U: “ice1” / **“ice2”**

Intel Skylake (available as fallback)

SuperMUC-NG

- Here: only Single-Node experiments
- Node
  - 2 sockets with Intel Skylake Xeon Platinum 8174
  - 2x 24 = 48 cores
    - 2x 512bit vector units per core (8 x DP FMA)
    - 2 threads per core (”Hyper-Threading”)
    - 2.3 GHz base (currently: 2.5 GHz), 14nm
- 96 GB main memory





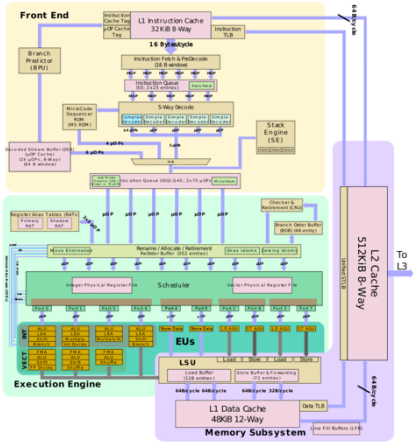
Links

- https://doku.lrz.de/display/PUBLIC/Hardware+of+SuperMUC-NG
- https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(server)

Intel Icelake

Two systems in BEAST

- 2 sockets Intel Xeon (Icelake) Platinum 8360Y
  - 2x 36 = 72 cores
    - 2x 512bit vector units per core (8 x DP FMA)
    - 2 threads per core (”Hyper-Threading”)
    - 2.4 GHz base, Intel 10nm
- 512 GB main memory, 1.5 TB Optane NVRam



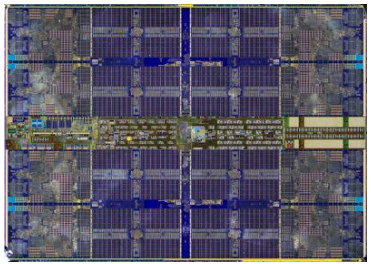
Links

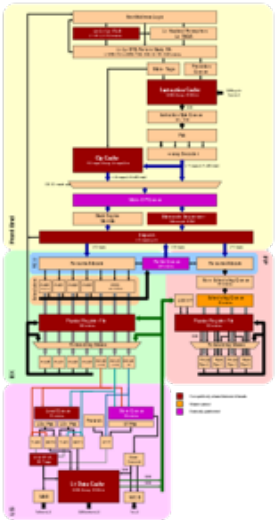
- https://en.wikichip.org/wiki/intel/microarchitectures/ice_lake_(server)
- https://en.wikichip.org/wiki/intel/microarchitectures/sunny_cove

AMD Rome

Two systems in BEAST

- 2 sockets with EPYC 7742
- 2x 64 = 128 cores (“Zen2”)
  - Chiplet design: IO-Die + 8x CCX-Dies (2x 4-core)
  - 2x 256-bit vector units per core (4 x DP FMA)
  - 2 threads per core
  - 2.25 GHz base, TSMC 7nm
- 512 GB main memory
- 2x AMD Radeon MI-100 GPUs
  - 7nm, 32GB HBM, PCIe4





Link

- https://en.wikichip.org/wiki/amd/microarchitectures/zen_2

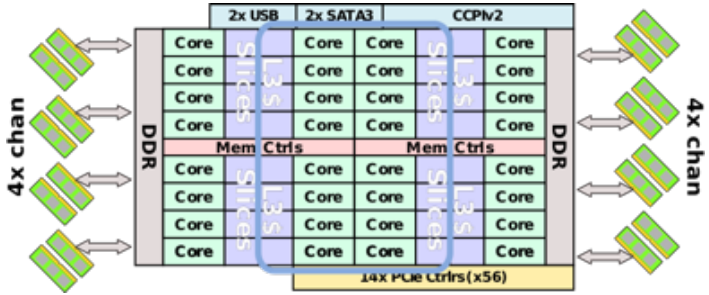
Marvell ThunderX2

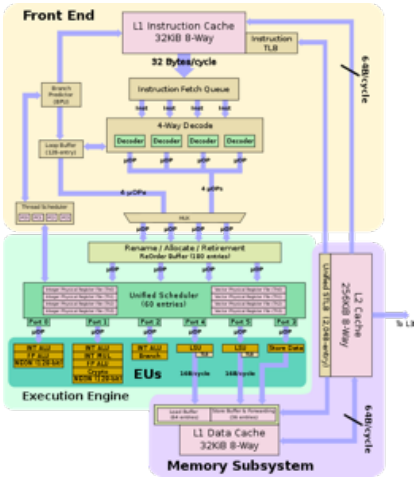
Two systems in BEAST

- 2 sockets with ThunderX2 CN9980
- 2x 32 = 64 cores (“Vulcan”)
  - 128-bit vector units (2 x DP FMA)
  - 4 threads per core
  - 2.2 GHz base, 16nm
- 512 GB main memory
- 2x Nvidia V-100
  - Volta, 32GB HBM, PCIe3

Link

- https://en.wikichip.org/wiki/cavium/microarchitectures/vulcan

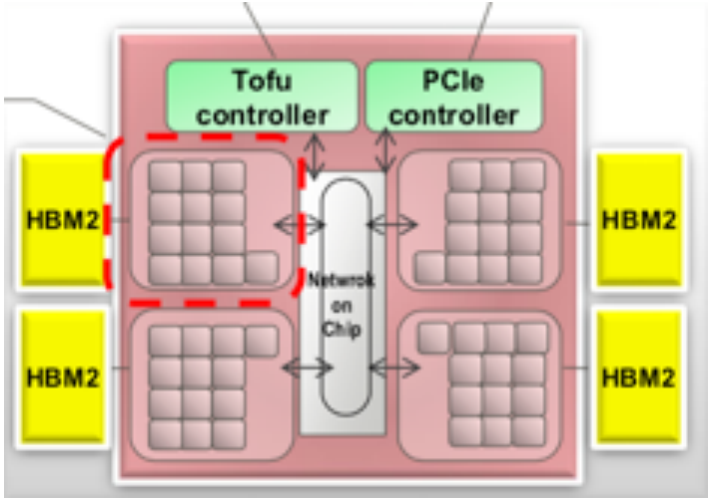




Fujitsu A64FX

HPE CS500 in BEAST

- 8 nodes with one A64FX CPU (“NSP1”)
- 48 cores per CPU
  - 2x 512bit vector units per core
  - 1.8 GHz, TSMC 7nm
  - 4 NUMA domains
- 32 GB HBM2



[Fujitsu: The 1st SVE Enabled Arm Processor: A64FX and Building up ARM HPC Ecosystem, 2019]

Link

- https://en.wikipedia.org/wiki/Fujitsu_A64FX

Access and Usage: Intel Skylake / SuperMUC-NG (Fallback)

Access

- via “ssh rbgaccount@lxhalle.informatik.tu-muenchen.de”
- ssh XXX@skx.supermuc.lrz.de

Compilers

- via “module”, see available packages with “module avail” / load with ”module load”
- ICC (“icpc” for C++, “icc” for C, Intel compiler, enable OpenMP: “-openmp”)
- GCC (“g++” C++, “gcc” C, GNU Compiler Collection, enable OpenMP: “-fopenmp”)

Node allocation (test queue, 30 minutes, e.g. project h039y - see “groups”)

- salloc -Ah039y -ptest -t30

Access and Usage: BEAST Systems

Access via Linux Cluster login nodes

- ssh XXX@lxlogin8.lrz.de (or lxlogin1@lrz.de)
- ssh testbed.cos.lrz.de
- ssh <system>

If testbed.cos.lrz.de is not reachable, retry after 1 hour

- probably just a reboot

Compilers

- system: “gcc”
- via modules: see “module avail”, then “module load <package>”

Access and Usage: Intel Icelake @ BEAST

Access

- ssh XXX@lxlogin8.lrz.de (or lxlogin1@lrz.de)
- ssh testbed.cos.lrz.de
- ssh ice2

Compilers

- gcc, icc (Intel compiler)

Access and Usage: AMD Rome @ BEAST

Access

- ssh XXX@lxlogin8.lrz.de (or lxlogin1@lrz.de)
- ssh testbed.cos.lrz.de
- ssh rome2

Compilers

- gcc, clang (from AMD RocM)

Access and Usage: ThunderX2 @ BEAST

Access

- ssh XXX@lxlogin1.lrz.de
- ssh testbed.cos.lrz.de
- ssh thx2

Compilers

- gcc
- (via „module load cuda/11.1.1 llvm“) clang

Access and Usage: AMD A64FX @ BEAST

Access

- ssh XXX@lxlogin1.lrz.de
- ssh testbed.cos.lrz.de
- ssh cs1 / cs2

Compilers

- gcc (8)
- gcc 11 (via „module load gcc/11.0.0“)
- Cray compiler: “cc”, enable OpenMP: “-h omp”

Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

Up Next: Assignment 0: Introduction