

BEAST Lab Preliminary Meeting

LRZ

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

LMU

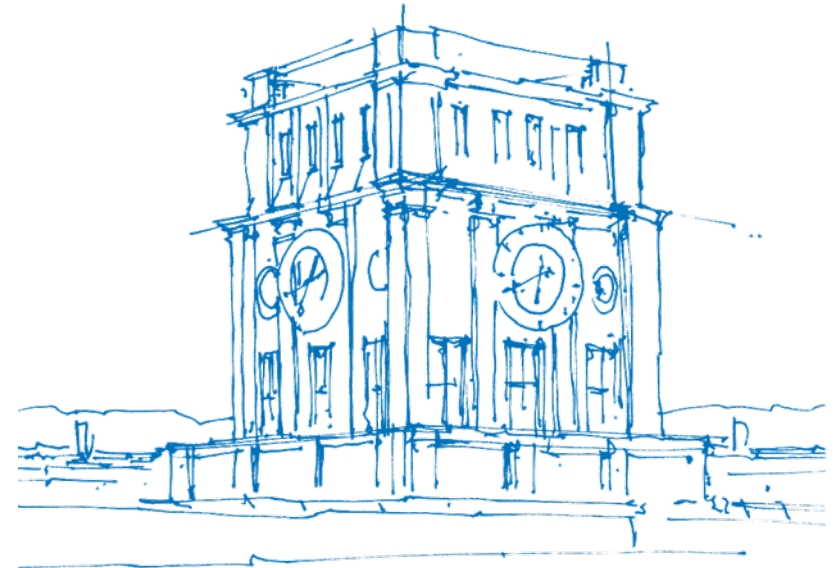
Sergej Breiter, sergej.breiter@nm.ifi.lmu.de

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

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

TUM

Bengisu Elis, bengisu.elis@tum.de



TUM Uhrenturm

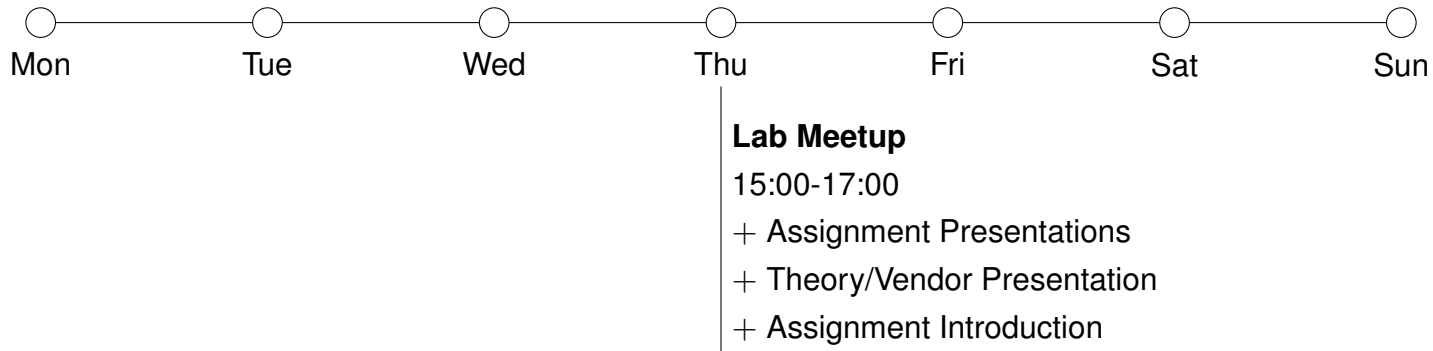


Table of Contents

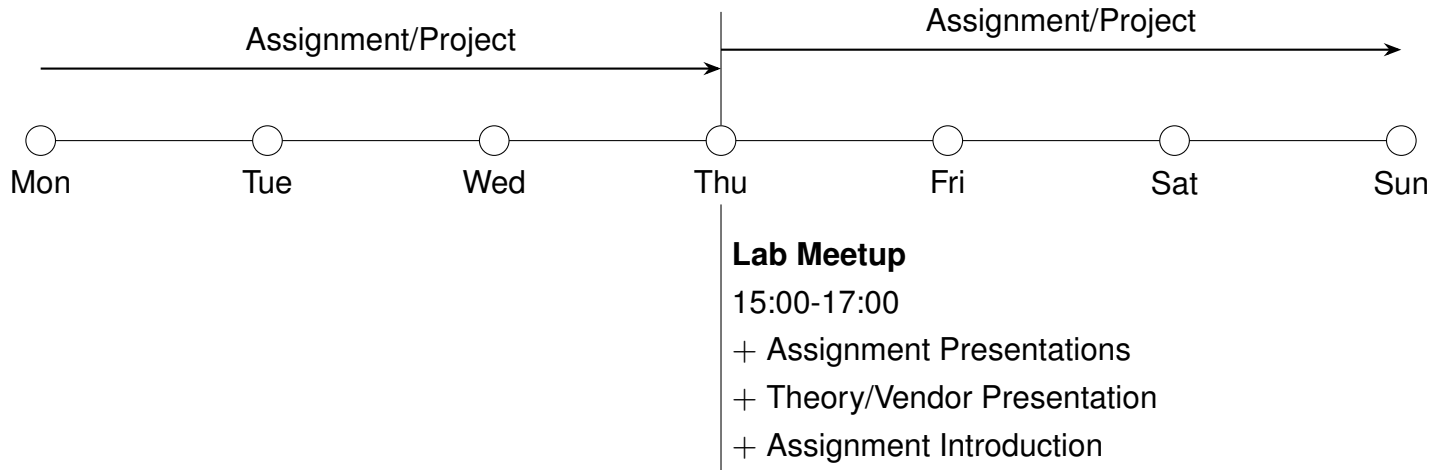
Course Organization

Introduction to BEAST

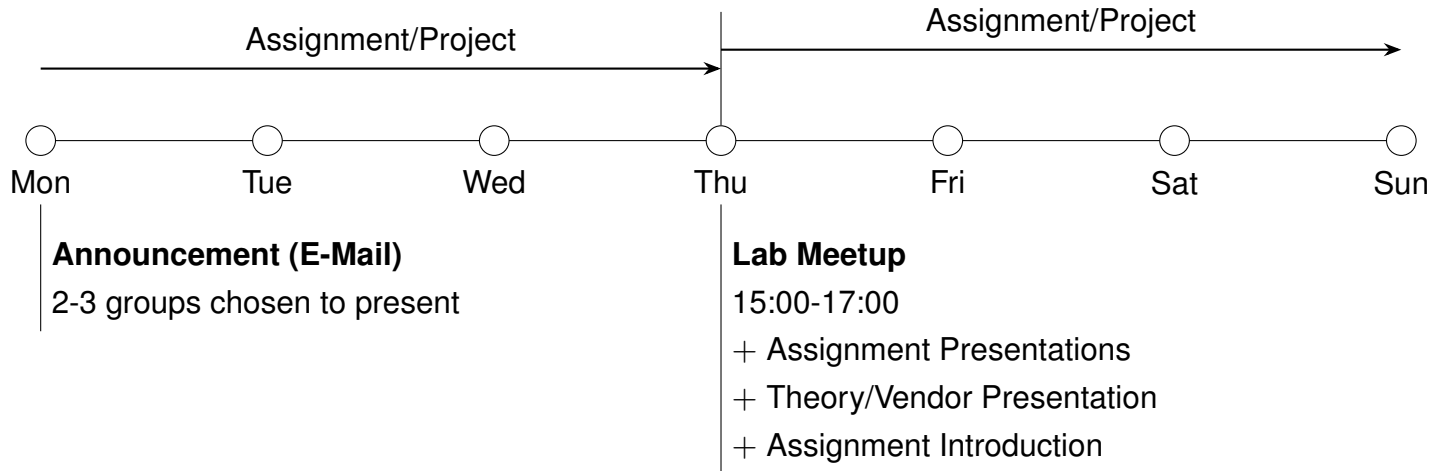
Weekly Schedule



Weekly Schedule

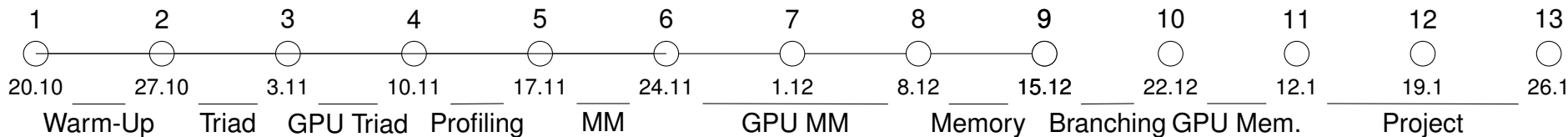


Weekly Schedule



Tentative Semester Overview - Last Year

Weeks



Organization

- Note: This is preliminary based off of last semester and is subject to improvements
- 8 Assignments
- Assignment 0 (of today) is not optional
 - 1 week each (except on holidays)
- Project
 - 2 weeks

V. Bodur, C. H. G. (Bachelor) or 2 (Master)

Previous Vendor Talks



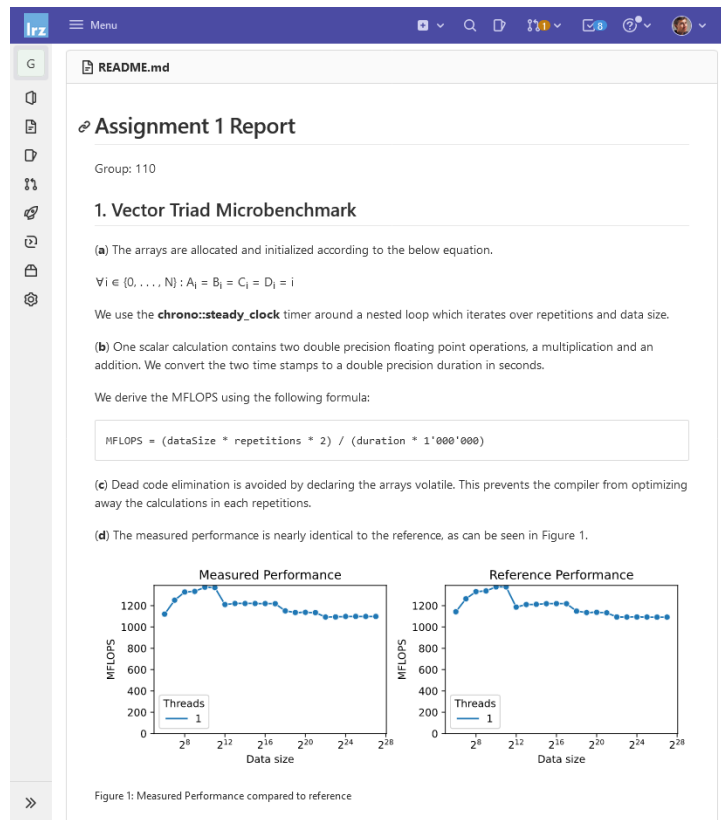
Deliverables/Grading

Git Repository

- **Assignment/Project Report** in Markdown
- Your Code
- CI Jobs (not graded)

Presentation

- No slides. Go through the report
- Talk about what you learned
- Get feedback from advisors



Assignment 1 Report

Group: 110

1. Vector Triad Microbenchmark

(a) The arrays are allocated and initialized according to the below equation.

$$\forall i \in \{0, \dots, N\} : A_i = B_i = C_i = D_i = i$$

We use the `chrono::steady_clock` timer around a nested loop which iterates over repetitions and data size.

(b) One scalar calculation contains two double precision floating point operations, a multiplication and an addition. We convert the two time stamps to a double precision duration in seconds.

We derive the MFLOPS using the following formula:

$$\text{MFLOPS} = (\text{dataSize} * \text{repetitions} * 2) / (\text{duration} * 1'000'000)$$

(c) Dead code elimination is avoided by declaring the arrays volatile. This prevents the compiler from optimizing away the calculations in each repetitions.

(d) The measured performance is nearly identical to the reference, as can be seen in Figure 1.

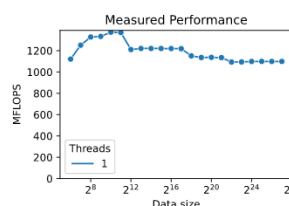
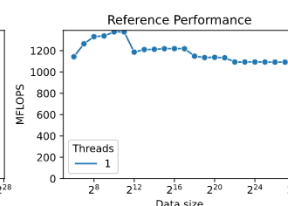



Figure 1: Measured Performance compared to reference

Next Steps

Register on Matching System

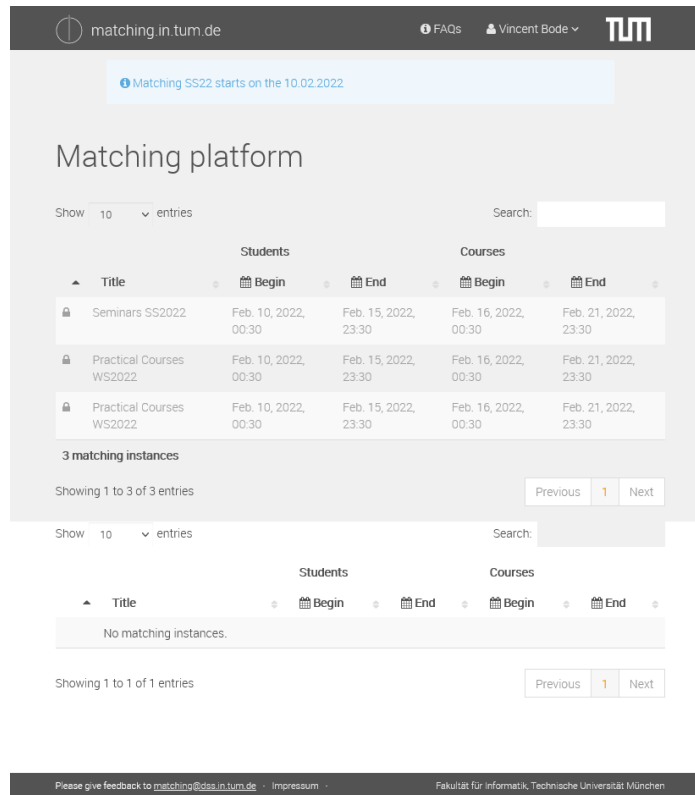
- We will prioritize you if you attended today
- Open until 14.02.2023
- Wait for announcement of matching results (23.02.2023)

Group Preferences

- Only after matching has ended
- Send us by e-mail (bengisu.elis@tum.de)
- No preferences submitted → we will match you

Attend Course Kickoff

- At university if everything goes according to plan
- We hope to see you there :)



The screenshot shows the 'matching.in.tum.de' website. At the top, there's a navigation bar with 'matching.in.tum.de', 'FAQs', 'Vincent Bode', and the TUM logo. A blue banner below the header states 'Matching SS22 starts on the 10.02.2022'. The main section is titled 'Matching platform'. It features a search bar and a 'Show 10 entries' dropdown. Below this, there are two tables: 'Students' and 'Courses'. The 'Students' table has columns for Title, Begin, and End. The 'Courses' table has columns for Title, Begin, and End. The 'Students' table lists three entries: 'Seminars SS2022', 'Practical Courses WS2022', and 'Practical Courses WS2022'. The 'Courses' table lists three entries: 'Seminars SS2022', 'Practical Courses WS2022', and 'Practical Courses WS2022'. Below the tables, it says '3 matching instances' and 'Showing 1 to 3 of 3 entries'. There are 'Previous', '1', and 'Next' buttons. The bottom of the page has a footer with 'Please give feedback to matching@csa.in.tum.de', 'Impressum', and 'Fakultät für Informatik, Technische Universität München'.

Up Next: Introduction to BEAST

BEAST Lab Summer Term 2023

Preliminary Meeting

February 9, 2023

Collaboration among 3 institutions

LMU
TUM
LRZ

TUM – CAPS/Prof. Schulz
(Bengisu Elis)

LRZ - Future Computing Group
(Josef Weidendorfer)

We want you to learn about **performance properties of modern 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 the weekly meetings, from presentations of 2 groups

Structure:

Memory on CPU (Triad / Traversal) → Compute on CPU (MM) → ... on GPU → Tools

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
- You tune the code to get best single-node performance (2 weeks time)
- We all discuss progress in weekly meeting

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



- Work in student groups
 - we expect you to split up the work equally
- Assignments
 - every week (Part 2: 2 weeks)
 - code / reports (MarkDown) via Gitlab repos, partially CI feedback in the beginning
- Weekly meetings (Thursday afternoon)
 - talks around assignment tasks (microarchitecture, parallel prog. models, ...)
 - student group presentations for every assignment (randomly selected)
 - discussions around results

Prerequisites



- Good knowledge of C (C++) on Linux
- Basic knowledge of computer architecture. You should know terms such as
 - Multi-core, L1/L2/L3 caches, TLB, pipelining, SIMD, SMT
- Interest in computer architecture, benchmarking, low-level code optimization