ECE-451/556 Parallel and Distributed Computing

- 2023 Fall
- **Instructor:** Dov Kruger
- **Meeting Times and Office hours**
- **Resources**
- **Course Web Address**

Prerequisites

- Required good command of C++, good idea to know some machine language
- 16:332:563 (Grads) or 16:332:331 (undergrads)
- 16:332:351 (Programming Methodology-II & Data
- Strongly advised to have a solid background using Linux
  - Tools in this course will use linux, remote machines will be available
- 14:332:434 – Operating Systems (would be a plus)

Software and Hardware Requirements

Because students today may have a Mac running an ARM CPU, we will provide online facilities, but if you can compile on your own intel machine this will lighten the load on the central resources. Everyone is expected to get an account on Amarel, the Rutgers research cluster, and to use the resource carefully as we are guests.

- g++/clang++ compiler capable of executing c++-14 at least
- c++ threading
- Intel 4th generation or better capable of executing AVX2 instructions
- OpenMP-capable compiler (we will use vectorization)
- CUDA running on NVIDIA GTX 10xx or better
- OpenMPI (running on Amarel cluster)
- Icarus Verilog (if time permits)

COURSE DESCRIPTION

This course covers parallel computing

- Overview of computer architecture and speed
  - Current limits to parallel execution speed
    - Memory bandwidth
    - Latency
  - Future Improvements
- Approaches to Parallelism
  - Multithreading
  - vectorization
  - Optimization of memory bandwidth
  - Custom hardware architecture

- Technologies
  - C++ threads
    - locking
  - Vectorization (AVX2, AVX512 or Neon)
  - OpenMP (multithreaded and vectorized)
  - CUDA (massively parallel execution on GPUs)
  - MPI (Message Passing Interface)
  - Verilog (if time permits)

Course Outcomes

After completion of this course, students will be able to * Write code using C++ threads * Write code using openmp * Benchmark code to measure actual performance * Parallelize algorithms * Analyze problems to determine whether they are memory bound * Optimize algorithms to decrease memory utilization where possible * Write SIMD programs using AVX instructions (or ARM Neon if you prefer) * Use a parallelizing compiler like ISPC to generate vectorized code * Write CUDA kernels to run on NVIDIA GPUs * Write parallel programs on clusters using MPI

FORMAT AND STRUCTURE

- Classes include slides and live coding in a number of C/C++ derivative languages and APIs. You are encouraged to actively participate.

COURSE MATERIALS

All textbooks are optional. Most materials for this course are linked in the notes and are freely downloadable.

- Optional textbooks
  - Parallel Computer Architecture: a Hardware/Software approach, David E. Culler, Jaswinder Pal Singh, Anoop Gupta. Morgan Kaufman, ISBN 558603433
- Intel ISPC Compiler
- Intel Intrinsics Guide
- CUDA programming guide
- OpenMP Manual
- Other Readings: Papers available in ref directory of repo

COURSE REQUIREMENTS

- **Attendance:** Attendance is crucial for an effective learning but will not be graded. Your work will speak for itself.

- **Homework:** Coding assignments will be submitted via canvas for individual single files, or via github.

GRADING PROCEDURES

Grades will be based on: * Homework problem sets on theory (5%) * Paired Programming Homeworks (15%) * Mini projects (15%) * Midterm (32.5%) * Final exam (32.5%)

[Grading Policies] (https://github.com/RU-ECE/DovKrugerCourses/grading.md)
[Academic Honesty and Discipline] (https://github.com/RU-ECE/DovKrugerCourses/academichonesty.md)

IMPORTANT DATES

- Midterm ** 2024-TBD **
- Final ** 2024-TBD **