CS205

HIGH PERFORMANCE COMPUTING FOR SCIENCE AND ENGINEERING

CLASS SYLLABUS

2021-12-17

Fabian Wermelinger

Harvard University

Objective:

With manufacturing processes reaching the limits in terms of transistor density on today's computing architectures, efficient utilization of computing resources must exploit parallel execution to maintain scaling. The use of computers in academia, industry and society is a fundamental tool for solving (scientific) problems while the "think parallel" mindset of code developers is still lagging behind. The aim of this course is to introduce the student to the fundamentals of parallel programming and its relationship on computer architectures. Various forms of parallelism are discussed and exploited through different programming models with focus on shared and distributed memory programming.

After completing this class, the student is capable of identifying parallelism in algorithms and exploit it with appropriate techniques. The student will further learn about methods to analyze the performance and parallel scaling of programs and know how to apply optimizations in software such that it can be deployed on large scale high performance computing (HPC) architectures.

Prerequisites:

The course assumes that the student is comfortable with reading and writing code in the `C` or `C++` (or `Fortran`) programming languages. Homework, lab and examples in class will be presented using `C++`. Familiarity with the Linux command line tools, `ssh`, `git` and editing source code is assumed. Preparatory classes include CS50, CS107 or AC207. *The course will not teach basics of programming.*

Textbooks:

The class does not follow a specific textbook. The following textbooks are suitable for additional reference:

- *"Introduction to High Performance Scientific Computing"*, V. Eijkhout, free pdf 3rd edition 2020
- *"Parallel Programming for Science and Engineering"*, V. Eijkhout, free pdf 2nd edition 2020
- *"An Introduction to Parallel Programming"*, P. Pacheco, Morgan Kaufmann 2011
- *"Introduction to High Performance Computing for Scientists and Engineers"*, G. Hager and G. Wellein, CRC Press 2011
- *"Computer Organization and Design"*, D. Patterson and J. Hennessy, Morgan Kaufmann 2018 (RISC-V edition)
- *"Computer Architecture"*, J. Hennessy and D. Patterson, Morgan Kaufmann 2019
- *"Programming Massively Parallel Processors"*, D. Kirk and W. Hwu, Morgan Kaufmann 2017

Course Format:

The course contains six main components:

1. ***Lectures:*** Deliver the main content of the class. Attendance is mandatory.
2. ***Readings:*** Accommodate lecture material. The reading assignments are discussed in class. Questions to individual students may be asked.
3. ***Quizzes:*** In-class quizzes intended to assess the learning progress.
4. ***Labs:*** Lab sessions offer practice on topics addressed in class and help support homework assignments.
5. ***Homeworks:*** Homework assignments deepen the lecture material and include coding exercises (skeleton codes are provided in `C++`).
6. ***Projects:*** The class is accompanied by a project (teams of 3-4 students) to practice the methods learned in class on a real application. Topics are proposed by the teams and may involve research problems of individual team members.

Grading:

Homework:	40%
Project:	35%
Quizzes:	10%
Labs:	10%
Communal Contributions: Via the class communication platforms.	5%
Bonus: Can be exploited only by participation in discussions during lectures. This may include answering or asking questions or discussions for reading assignments. Maximum is 5%.	5%

The class does not have standard midterm or final exams. The project work involves presentations.

Collaboration and Class Policies:

You are welcome to discuss the course material and homework with others in order to better understand it, but the work you turn in must be your own (with exception of the project where collaborative work is permitted). Any work that is not your own, without properly citing the original author(s), is considered plagiarism. Failure to follow the academic integrity and dishonesty guidelines outlined in the Harvard Student Handbook will have an adverse effect on your final grade. This includes the removal of copyright notices in code. You may not submit the same or similar work to this course that you have submitted or will submit to another without permission.

The course related material will be distributed through `git` repositories on the https://code.harvard.edu/ platform. Membership in the CS205 organization on that platform is required to access the material.

Wk	Tuesday	Thursday	Labs	Events
1(4)	***Lecture 1:*** *2022-01-25* • Class introduction/organization • Moore's Law • Transistor density and power limit • Parallel computing • Flynn's taxonomy • Overview of parallelism treated in class: DLP, ILP, TLP, shared memory and distributed memory	***Lecture 2:*** *2022-01-27* • Computer architecture • von Neumann architecture • Memory pyramid • Linux process anatomy • Introduction to compute cluster: access, job submission • ***Reading:*** Leiserson paper	***Sign-up:*** Select one of the offered lab session days according to your schedule	***Note:*** The "***Reading***" assignments are relevant for the lecture and due ***on the day of the lecture!*** Questions may be asked to individual students. 1. Doodle for lab day selection due (2022-01-28)
2(5)	***Lecture 3:*** *2022-02-01* • Cache memories: why are they there, how they work • Cache lines and the 3 C's • What is temporal and spatial locality • Cache associativity: fully, n-way, direct mapped • Memory access patterns (differences row-major / column-major)	***Lecture 4:*** *2022-02-03* • Shared memory introduction • Examples of concurrency and concurrent memory access • Why is shared memory programming hard: what is a race condition and why/how does it happen • ***Quiz 1***	***Lab 1:*** Accessing cluster, SLURM, Linux, compiler and C++ tutorials.	1. HW1 release (2022-02-01)
3(6)	***Lecture 5:*** *2022-02-08* • Memory model for shared memory programming and its implications on compilers • Sequential consistency • Mutual exclusion / critical sections / locks • Overview of thread libraries	***Lecture 6:*** *2022-02-10* • Introduction to OpenMP: why OpenMP and how to use it in new or existing codes • OpenMP: fork/join parallel regions • OpenMP: work sharing constructs • OpenMP: data environment • ***Reading:*** OpenMP specification 5.2 Chap. 1 (until 1.4 inclusive)		1. Lab 1 due (2022-02-11) 2. Project team formation due (2022-02-08)
4(7)	***Lecture 7:*** *2022-02-15* • OpenMP: synchronization constructs • OpenMP: library routines • OpenMP: environment variables • OpenMP: processor binding	***Lecture 8:*** *2022-02-17* • UMA/NUMA memory architectures • What is cache coherency and why is it required in shared memory programming • Cache coherency protocols (focus on MESI) • False sharing • ***Quiz 2***	***Lab 2:*** OpenMP locks, critical sections and atomic clauses.	1. HW1 due (2022-02-15) 2. HW2 release (2022-02-15)
5(8)	***Lecture 9:*** *2022-02-22* • Performance analysis (single node) • Relationship of compute performance (flop) to memory bandwidth • Roofline model • ***Reading:*** Williams paper	***Lecture 10:*** *2022-02-24* • Introduction to distributed programming (recap Flynn's taxonomy) • What is the Message Passing Interface (MPI) • Simple parallel MPI program example	***Lab 3:*** False sharing and cache thrashing.	1. Lab 2 due (2022-02-25) 2. Project high-level description due (2022-02-22)
6(9)	***Lecture 11:*** *2022-03-01* • MPI: blocking point-to-point • MPI: blocking collective • ***Reading:*** MPI 4.0 Standard 3.1, 3.2, 3.4, 3.5	***Lecture 12:*** *2022-03-03* • MPI: non-blocking point-to-point • MPI: non-blocking collective • ***Reading:*** MPI 4.0 Standard 3.7		1. Lab 3 due (2022-03-04)

Wk	Tuesday	Thursday	Labs	Events
7(10)	***Lecture 13:*** *2022-03-08* • MPI: I/O file management • MPI: I/O read and write routines • Parallel I/O for data compression example	***Lecture 14:*** *2022-03-10* • Hybrid MPI and OpenMP • Overhead associated with sending messages • Message packing • Working with scientific libraries (BLAS/LAPACK/Eigen) • ***Quiz 3***	***Lab 4:*** MPI reductions and scans.	1. HW2 due (2022-03-08) 2. HW3 release (2022-03-08)
8(11)	***Spring break:*** *2022-03-15*	***Spring break:*** *2022-03-17*		
9(12)	***Presentations for project proposals:*** *2022-03-22*	***Presentations for project proposals:*** *2022-03-24*		1. Lab 4 due (2022-03-25) 2. Project proposals due
10(13)	***Lecture 15:*** *2022-03-29* • Parallel scaling analysis • Strong scaling / Amdahl's law • Weak scaling	***Lecture 16:*** *2022-03-31* • Instruction set architecture (ISA) / RISC / CISC • Assembly language (`x86_64`) • Processor pipelining (ILP) • ***Reading***: Hennessy and Patterson Turing lecture	***Lab 5:*** Linking your code with third party libraries. Examples for BLAS and LAPACK.	1. HW3 due (2022-03-29) 2. HW4 release (2022-03-29)
11(14)	***Lecture 17:*** *2022-04-05* • Recap Flynn's taxonomy: SIMD • Instruction set architecture extensions • What is vectorization and why is it important • Memory alignment and relation to cache lines	***Lecture 18:*** *2022-04-07* • Manual vectorization • Intel intrinsics • Bit masking/shuffling • Examples for manual vectorization and performance impact (DLP in roofline)		1. Lab 5 due (2022-04-08)
12(15)	***Presentations for project designs:*** *2022-04-12*	***Presentations for project designs:*** *2022-04-14*		1. Project designs due
13(16)	***Lecture 19:*** *2022-04-19* • Compiler auto vectorization • SPMD programming model • Intel ISPC compiler • ***Reading***: Pharr paper • ***Quiz 4***	***Lecture 20:*** *2022-04-21* • GPU computing I: ▪ Streaming processors ▪ Main difference between CPU and GPU architectures ▪ SIMD and SIMT ▪ Introduction to CUDA	***Lab 6:*** Understanding machine instructions by learning how to debug code.	1. HW4 due (2022-04-19) 2. HW5 release (2022-04-19)
14(17)	***Lecture 21:*** *2022-04-26* • GPU computing II (CUDA): ▪ CUDA warps and threads ▪ Streaming multiprocessor and Little's Law ▪ Example CUDA kernel for vector addition • Class summary	***Reading period:*** *2022-04-28*		1. HW5 due (2022-05-01) 2. Lab 6 due (2022-04-29)

Wk	Tuesday	Thursday	Labs	Events
15(18)	***Reading period:*** *2022-05-03*	***Exam period:*** *2022-05-05*		
16(19)	***Exam period:*** *2022-05-10* • Project final presentations (date TBD)	***Exam period:*** *2022-05-12* • Project final presentations (date TBD)		1. Project deliverables due 2. Project final presentations due