CME 213 Syllabus Spring 2019

Instructors

- Eric Darve, ME, ICME, darve@stanford.edu
- Colfax, and NVIDIA engineers guest lectures
- Teaching assistants:
    - William Jen
    - Chenzhuo Zhu

Classes will be primarily on Monday and Wednesday. Friday will be used only for special or catch-up lectures.

Class forum and web site

- https://stanford-cme213.github.io/ class material, homework, final project
- Forum site: http://piazza.com. You need to register on piazza.
- Homework will be graded using gradescope. The entry code is **M42X7G**. Please register at https://gradescope.com.

Grading, homework, project

- 1 pre-requisites homework + 4 homework assignments: 65% of grade
- One final project: 35% of grade

All homework for this class will be prepared electronically. Homework papers will consist of:

1. Computer code. We will use primarily C++ and CUDA.
2. Written report with answers, plots, and figures in PDF format

The first homework is due Wednesday April 10.

Honor code:

> "that they will not **give** or **receive** aid in examinations; that they will not **give** or **receive** unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading"

You cannot copy someone else's computer code. Answers in all the papers you are submitting for a grade must be your own. Concerns will be reported to the Office of Community Standards. In case a student is found guilty, the Office of Community Standards will decide on sanctions. In recent years, the most frequent Honor Code violations arise when a student submits another's work as his or her own or gives or receives unpermitted aid. The standard sanction for a first offense includes a one-quarter suspension from the University and 40 hours of community service. In addition, the instructor will automatically issue a "No Pass" or "No Credit" grade. The standard sanction for multiple violations (e.g., cheating more than once in the same course) is a three-quarter suspension and 40 or more hours of community service.

Final Project

You will be given a final project to work on towards the end of the quarter. The project will be on implementing a neural network to recognize hand-written digits. The project will involve CUDA and MPI programming.

Books

Good news: most books are available electronically from the Stanford Library. Just go to: http://searchworks.stanford.edu/

Parallel computing books

- *Parallel Programming for Multicore and Cluster Systems,* Rauber and Rünger. Applications focus mostly on linear algebra.
- *Introduction to Parallel Computing,* Grama, Gupta, Karypis, Kumar. Wide range of applications from sort to FFT, linear algebra and tree search.
- *An introduction to parallel programming,* Pacheco. More examples and less theoretical.
- *C++ High Performance: Boost and optimize the performance of your C++17 code,* Andrist, Sehr; focused on recent C++ features but also contains a discussion of parallel computing for shared memory machines
- *Introduction to High Performance Computing for Scientists and Engineers,* Hager, Wellein; MPI and openMP; discussion of hybrid parallel computing

OpenMP and multicore books

- *Using OpenMP: portable shared memory parallel programming,* Chapman, Jost, van der Pas. Advanced coverage of OpenMP.
- *Parallel Programming in OpenMP,* Chandra, Menon, Dagum, Kohr, Maydan, McDonald; a bit outdated now.
- *The art of multiprocessor programming,* Herlihy, Shavit. Specializes on advanced multicore programming.
- *Using OpenMP—The Next Step: Affinity, Accelerators, Tasking, and SIMD,* van der Pas, Stotzer, Terbo; covers recent extensions to OpenMP and some advanced usage

CUDA books

- *Professional CUDA C Programming,* Cheng, Grossman, McKercher; has more advanced usage like multi-GPU programming
- *Programming Massively Parallel Processors: A Hands-on Approach,* Kirk, Hwu; in its 3rd edition now; covers a wide range of topics including several in numerical linear algebra, many applications, and parallel programming patterns relevant to CUDA
- *CUDA for Engineers: An Introduction to High-Performance Parallel Computing,* Storti, Yurtoglu; interesting examples relevant to engineers
- *CUDA by Example: An Introduction to General-Purpose GPU Programming,* Sanders, Kandrot
- *CUDA Handbook: A Comprehensive Guide to GPU Programming,* Wilt
- Best reference: *CUDA C programming guide* on the NVIDIA web site.

MPI books

- *Parallel Programming with MPI,* Pacheco; classic reference; somewhat dated at this point
- *Using MPI: Portable Parallel Programming with the Message-Passing Interface,* Gropp, Lusk, Skjellum; very complete reference
- *Using Advanced MPI: Modern Features of the Message-Passing Interface,* Gropp, Hoefler, Thakur, Lusk; same authors as previous entry; discusses recent and more advanced features of MPI

What this class is about

- We will focus on how to program:
  - Multicore processors, e.g., desktop processors: Pthreads, C++ threads, OpenMP.
  - NVIDIA graphics processors using CUDA.

    - Computer clusters using MPI.
- We will cover some numerical algorithms for illustration: sort, linear algebra, and basic parallel primitives.

What this class is not about

- Parallel computer architecture
- Parallel design patterns and programming models
- Parallel numerical algorithms. See *CME 342: Parallel Methods in Numerical Analysis*

Requirements

- Some basic knowledge of UNIX (ssh, makefile, git, etc.)
- Good knowledge of C and C++ (including pointers, templates, standard library)
- Proficiency in scientific programming, including testing and debugging

Overview of class content

- Pthreads, OpenMP – Eric Darve
- OpenMP – Colfax engineer
- CUDA – Eric Darve
- CUDA – NVIDIA engineers
- MPI – Eric Darve