Draft Syllabus:

CS194-15 Engineering Parallel Software

Kurt Keutzer

EECS, University of California, Berkeley

Fall 2016: 2-3:30 Soda 306

As the basic computing device ranging single cell phones to racks of hardware in cloud computing, parallel processors are emerging as the pervasive computing platform of our time. This course will enable advanced undergraduate students to design, implement, optimize, and verify programs to run on present generations of parallel processors.

There are four principal themes that are pursued in this course:
- Software engineering
- Performance Programming
- Programming in Parallel Languages
- Course project

Software Engineering and Software Architecture

Our approach to this course reflects our view that a well-designed *software architecture* is a key to designing parallel software, and a key to software architecture is design patterns and a pattern language. Our course will use Our Pattern Language as the basis for describing how to design, implement, verify, and optimize parallel programs. Following this approach we will introduce each of the major patterns that are used in developing a high-level architecture of a program. Descriptions of these ten structural and thirteen computational patterns, together with other readings, may be found at: https://patterns.eecs.berkeley.edu/.

Performance Programming

Writing efficient parallel programs requires insights into the hardware architecture of contemporary parallel processors as well as an understanding as to how to write efficient code in general. As a result a significant amount of time in the course will be spent on looking “under the hood” of contemporary sequential and multiprocessors and identifying the key architectural details, such as non-uniform memory architecture (NUMA), that are necessary to write high performance code.

Programming in Parallel Languages
Other lectures and laboratories of the course will focus on implementation using contemporary parallel programming languages, verification of parallel software using invariants and testing, and performance tuning and optimization. Particular languages covered typically include OpenMP, MPI, and OpenCL.

Course Projects
The final third of the course will be an open-ended course project. These projects allow students to demonstrate their mastery of the course concepts mentioned above. Students will create their own projects in project teams of 4-6 students.

Prerequisites

Students should have taken the following or equivalents:

- Basic programming course using Java, C or C++
- Undergraduate course on computer organization
- Linear algebra

It is recommended that students have taken:

- At least one upper division course that includes significant programming assignments (e.g. Compilers, Operating Systems, or Software Engineering)

Course Work and Grading

The course consists of twice-weekly lectures in a “flipped classroom” format. Each Tuesday class session will be a lab session. Each Thursday session will review the class, quizzes, and homework assignments. For the first two thirds of the course, there will be a series of programming assignments. There will be two examinations during the course.

- Grading (If you’re following your GPA *pay close attention*!!!!):
  - 20% Assignments (aka Machine Problems (MPs))
  - 30% Two examinations
  - **35% Final Project: individual performance assessed**
    - **5% proposal; 10% final content and presentation and ppt; 20% project**
  - On-line Quizzes: 10%
  - 5% Bonus: Attendance, class participation – can bump you up ½ grade, e.g. B+ → A-

Course Staff

Professor: Kurt Keutzer

Guest Lecturer: Tim Mattson, Intel

TA: Russell Nibbelink

Recommended Course Textbook

Updated version of: *Patterns for Parallel Programming*, T. Mattson, B. Sanders, B. Massingill, Addison Wesley, 2005. Will be distributed in class.

Course Lab and Discussions:

Lab computing facilities will be in Soda 330. Other facilities will be available for projects.

Course Assignments Will Be Selected From Among this List

1. ***Computer Architecture -*** Measure L1/L2/L3 bandwidth and latency on our lab machines. Also, investigate measured ILP for a handful of different SGEMM implementations. Performance in MFlops/s increases, but ILP drops. Also serves as a warmup / refresher for the small subset of C++ we use for the lab assignments. Follows the material from lecture 3 (sequential processor performance)

2. ***Parallel Matrix Multiply (DGEMM) -*** Write naive parallel DGEMM using OMP for loops, OMP tasks, and pthreads. Serves as a simple warm-up for the basic threading libraries. Advanced question on how GCC converts code with OpenMP pragmas into parallel code. Follows the material from lecture 2/4 (parallel programming on shared memory computers)

3. ***Optimize Matrix Multiply (DGEMM) -*** Optimize the naive parallel matrix multiply for both locality and data parallelism (using SSE2). Students get familiar with SSE2 intrinsics if they want to use them for their final projects. Follows the material from lecture 6/8 (memory subsystem performance)

4. ***Introduction to OpenCL -*** Students write both VVADD and SGEMM in OpenCL. They will write the kernels. Follows lecture 9 / 10. (Data parallelism and CUDA).

5. ***OpenCL + OpenGL -*** Students perform a handful of simple graphics operations on an image. Follows lecture 9 / 10. (Data parallelism and CUDA).

6. ***Advanced OpenCL -*** Students write a reduction routine using the ideas presented in class. They also write array compaction using scan. Follows lecture 9 / 10. (Data parallelism and CUDA).

Syllabus, Fall 2016: Classes are at 2:00 – 3:30PM in Soda 306

Lectures are available on-line at: https://cvw.cac.cornell.edu/eps/default

Lectures should be viewed and quizzes taken BEFORE the corresponding class in the syllabus.

Week	***Date***	***What***	***Topic***
Week 1	Tuesday 8/23	No class	
	Thursday 8/25	Lecture 1	First Lecture: Intro, Background, Course Objectives and Course Projects --Keutzer
Week 2	Tuesday 8/30	Lecture 2	A programmer's introduction to parallel computing: Amdahl's law, Concurrency vs. Parallelism, and the jargon of parallel computing. Getting started with OpenMP and Pthreads. --Mattson
	Thursday 9/1	Lecture 3	Sequential Processor Performance 1: Example; Pipelining, Superscalar, etc.; Compiler Optimizations; --Keutzer
	Tuesday 9/6	Discussion 1	Intro to the Lab Environment. Assignment 1 goes out.
	Tuesday 9/6	Lecture 4	Sequential Processor Performance Part 2 --Keutzer
	Thursday 9/8	Lecture 5	Parallel Processor Architectures and the Future of Computing --Keutzer

	Tuesday 9/13	Discussion 2	**Assignment 1 due**. Assignment 2 goes out.
	Tuesday 9/13	Lecture 6	Optimizing Program Performance and the Roofline Model - Keutzer
	Thursday 9/15	Lecture 7	Architecting Parallel Software with Patterns: Another way to think about parallelism - Keutzer
	Wednesday 9/20	Discussion 3	**Assignment 2 due.** Study for midterm.
	Tuesday 9/20	Lecture 8	Performance of sequential processors: exploring the memory system with matrix multiplication --Mattson
	Thursday 9/22	Lecture 9	A programmers introduction to parallel computing: making concurrency safe -- Mattson
Week 6	Tuesday 9/27	No discussion	Assignment 3 goes out.
	Tuesday 9/27	Midterm	***Midterm 1***
	Thursday 9/29	Lecture 10	Data Parallelism -- video lecture by David Sheffield

Week 7	Tuesday 10/4	Discussion 5	**Assignment 3 due.** Assignment 4 goes out.
	Tuesday 10/4	Lecture 11	Heterogeneous computing using and OpenCL (CUDA also) -- video lecture by David Sheffield
	Thurs 10/6	Lecture 12	Structured grid, Geometric Decomposition, and MPI --Mattson
Week 8	Wednesday 10/11	Discussion 6	**Assignment 4 due.** Assignment 5 goes out. Project proposal due at Midnight
	Tuesday 10/11	Lecture 13	Mid-semester review in Video
	Thursday 10/13	Lecture 14	Structural Patterns and Parallelism Part 1 – Keutzer Video Project proposals presented in class
Week 9	Tuesday 10/18	Discussion 7	**Assignment 5 due.** Assignment 6 goes out.
	Tuesday 10/18	Lecture 15	Structural Patterns and Parallelism Pt2 – Keutzer video
	Thurs 10/20	Lecture 16	Parallelizing logic optimization using patterns - part 1 Project progress presented in class
Week 10	Tuesday 10/25	Discussion 8	**Assignment 6 due.** Midterm 2 review and discussion.
	Tuesday 10/25	Lecture 17	Parallelizing logic optimization using patterns - part 2 – Keutzer video – Project progress presented in class

	Thursday 10/27	Midterm 2	***Midterm 2***
Week 11	Tuesday 11/1	Discussion 9	Project meetings: show up with evidence of work!
	Tuesday 11/1	Lecture 18	Computational patterns: dense linear algebra - pt1 Videos by Michael Anderson
	Thursday 11/3	Lecture 19	Computational patterns: Dense linear algebra - pt2 Video by Michael Anderson
Week 12	Tuesday 11/8		Holiday
	Tuesday 11/8	Lecture 20	Computational patterns - Sparse linear algebra - pt1 – video by Michael Anderson Project software architecture presented in class
	Thursday 11/10	Lecture 21	Computational Patterns: Sparse linear algebra - pt2 – Video by Michael Anderson Project software architecture presented in class
Week 13	Tuesday 11/15	Discussion 11	Project meetings: show up with evidence of work!
	Tuesday 11/15	Lecture 22	Parallelizing speech recognition - pt1 –Keutzer video
	Thurs 11/17	Lecture 23	Parallelizing speech recognition - pt2 --Keutzer
Week 14	Tues 11/22	Lecture 24	Your career in software - Keutzer
Week 15	Tuesday 11/29	Discussion 12	discuss projects
	Tuesday 11/29	Presentations/ Or defer	Project-related office hours
	Thurs 12/1	Presentations/ Or defer	Project-related office hours

Week 16	Tuesday 12/6	Presentations/if necessary	Project presentations
	Thurs 12/8	Presentations/ If necessary	Project presentations