Blurb: Programming for Performance
ECE459, Winter 2013

Patrick Lam

The course webpage for Winter 2011 is at `http://patricklam.ca/p4p`. Next year's offering ought to be similar.

Brief Overview

Many modern software systems must process large amounts of data, either in the form of huge data sets or vast numbers of (concurrent) transactions. This course introduces students to techniques for profiling, rearchitecting, and implementing software systems that can handle industrial-sized inputs. The key idea is to leverage parallelism, which is available today in the form of multicore, multi-CPU, and distributed systems. Experience with these techniques will enable you to design and build high-performance software.

While you may have seen some of these ideas in the context of operating systems (eg ECE354/CS350) and system performance evaluation (CS457), this course gives you tools to make code run faster. The focus in ECE354/CS350 is understanding and implementing the primitives; our focus is on using them effectively. CS457 supplements this course well in that it tells you what you need to improve, while this course tells you how to improve it.

We will sometimes see implementation details that you need to get right to write certain applications, but as with any university-level course, this course focusses more on the concepts than magic invocations, so that you can continue to apply the basic ideas after the technologies inevitably change.

Objectives. More specifically, after this course you will be able to:

- investigate a software architecture, propose ways to improve its performance, and estimate the impact of your changes;
- understand what the compiler is doing to automatically improve your code's performance, both in terms of regular optimizations and automatic parallelization, and how you can help the compiler;
- use common mechanisms to manually exploit parallelism: manual specification of parallelism (e.g. OpenMP); multithreading (and locks); vectorization (including GPU programming); and distributing the problem over many systems (e.g. via MapReduce).

The **key objective** of this course is to instill an understanding of how to make your programs faster, mostly by parallelization. To do so, we will integrate concepts from architectures, compilers, and operating systems.

Course Description

Topics. Here is a list of topics.

- Introduction; Designing and Profiling for Performance
- Processor Architecture Review
- Programming for Multicores

- Embarassingly Parallel Problems
- Concurrent Programming Review
- Compilers, Optimizations and Automatic Parallelization
- Manual Parallelization and Keeping it Safe: Locks (e.g. using fine-grained locks), Races, Reordering, Memory Barriers, and Thread Pools
- OpenMP, MPI
- Special-purpose languages: X10, etc
- Special-purpose languages: MapReduce/Hadoop
- Programming GPUs with CUDA/OpenCL

I posted reasonably complete lecture notes, which you can consult on the course webpage.

Assignments. Since this course has "programming" in the title, you will be expected to write code for these assignments. Here is last year's list of assignments.

1. parallelizing a Monte Carlo estimation
2. comparing automatic and manual parallelization of C code; using OpenMP
3. parallelizing the implementation of a genetic algorithm
4. GPU programming (OpenCL)

For the 2013 offering, I'll try to have more frequent, but smaller, assignments.