ECE459: Programming for Performance
Winter 2012

Jon Eyolfson

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. Experience with these techniques will enable you to design and build critical software infrastructure.

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;
- 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).

General Information

Course Website: `http://ece459.eyolfson.com/`

Schedule:

Lectures	MWF	10:30 - 11:20 AM	DWE 3522
Midterm	March 2	6:30 - 7:50 PM	RCH 105/110
Tutorials	F	1:30 - 2:20 PM	DWE 3522

Tutorials will be held as-needed and announced ahead of time.

Instructor:

Jon Eyolfson
Office: DC2553
Office Hours: Announced in class, or by appointment
Email: `jeyolfso@uwaterloo.ca`

Teaching Assistants:

Shay Berkovich
Office: E5 4124
Email: `sberkovi@uwaterloo.ca`

Mohammad Rostami
Office: DC2553
Email: `m2rostam@uwaterloo.ca`

Course Description

Topics. Here is a detailed list of topics and estimated lecture hours for each topic. The number of hours is actually more of an educated guess than a reality-based datum. Note: the topics and times may change slightly as well.

Introduction	1
Processor Architecture Review	1.5
Programming for Multicores	3
Embarassingly Parallel Problems	3
Compilers, Optimizations and Automatic Parallelization	4.5
Concurrent Programming Review	1.5
Locks and Memory Barriers	1.5
OpenMP	3
Lock-free Algorithms and common CPU primitives	3
Designing and Profiling for Performance	3
Concurrency hacks and concurrent data structures	2
Loop perforation	1.5
Programming GPUs	3
Special-purpose languages: X10, etc	1.5
Special-purpose languages: Functional languages	1.5
Special-purpose languages: MapReduce	1.5

Reference Material

While I'm not requiring this book, I'll be using it for a substantial portion of the course:

> Darryl Gove. Multicore Application Programming for Windows, Linux and Solaris. Addison-Wesley, 2010.

I intend to post reasonably complete lecture notes for the material that I'm lecturing on.

Evaluation

This course includes assignments, a midterm, and a final examination. Too many fourth-year courses have projects, so I'm not going to pile on another project here.

Assignments	40%
Midterm	10%
Final exam	50%

Assignments. Since this course has "programming" in the title, you will be expected to write code for these assignments. Here is a projected list of assignments for this course. I plan to have 4 assignments.

1. using manual parallelization with Pthreads
2. using compiler-provided automatic parallelization and OpenMP
3. identifying opportunities for parallelization and estimating their impact
4. GPU programming

Assignment handin will be done via links on the course webpage.

Exams. Exams will be closed-book.

Schedule. The assignment dates are not set in stone, but this in the general guideline.

January 18	A1 out
February 1	A1 due, A2 out
February 15	A2 due, A3 out
February 29	A3 out
March 2	Midterm
March 14	A3 due, A4 out
March 28	A4 due
Exam period	Final exam

Group work. You may discuss assignments with others, but I expect each of you to do each assignment independently. Acceptable collaboration includes discussing ideas and structures with others, as well as helping others debug their code. If your code is too close in structure to someone else's code, you are going to have a problem. The best way to avoid such problems is by (1) not sending your code around; and (2) not writing down anything beyond general notes (pseudocode) about other peoples' code. I will follow UW's Policy 71 if I discover any cases of plagiarism (and I have). I will not use turnitin, but I may use other plagiarism detection software.

Lateness. You have 4 days of lateness to use on submissions throughout the term. Each day you hand in something late consumes one of the days of lateness. The fifth day of lateness causes your lowest assignment mark to be halved, while the sixth day causes both assignment marks to be halved. If you hand in something and you have more than 6 days of lateness, I'll start converting marks to 0 and dropping the associated late days. You can only hand in a submission up to the time I return all of the submissions. You don't get any credit for unused late days.

For example, you may hand in A1 one day late, A2 two days late, A3 1 one day late, and everything else on time. Or you can hand A2 four days late, if you hand in everything else on time. Finally, if you hand in A1 3 days late, A2 1 day late, A3 3 days late, and everything else on time, I will either give you a 0 for A1, leaving you with 4 late days, or give you a 0 for A2, leaving you with 5 late days and causing your mark for A1 to be halved. I'll choose the option which gives you more marks.

Required inclusions

Academic Integrity: In order to maintain a culture of academic integrity, members of the University of Waterloo community are expected to promote honesty, trust, fairness, respect and responsibility. [Check `www.uwaterloo.ca/academicintegrity/` for more information.]

Grievance: A student who believes that a decision affecting some aspect of his/her university life has been unfair or unreasonable may have grounds for initiating a grievance. Read Policy 70, Student Petitions and Grievances, Section 4, `www.adm.uwaterloo.ca/infosec/Policies/policy70.htm`. When in doubt please be certain to contact the departments administrative assistant who will provide further assistance.

Discipline: A student is expected to know what constitutes academic integrity [check `www.uwaterloo.ca/academicintegrity/`] to avoid committing an academic offence, and to take responsibility for his/her actions. A student who is unsure whether an action constitutes an offence, or who needs help in learning how to avoid offences (e.g., plagiarism, cheating) or about rules for group work/collaboration should seek guidance from the course instructor, academic advisor, or the undergraduate Associate Dean. For information on categories of offences and types of penalties, students should refer to Policy 71, Student Discipline, `www.adm.uwaterloo.ca/infosec/Policies/policy71.htm`. For typical penalties check Guidelines for the Assessment of Penalties, www.adm.uwaterloo.ca/infosec/guidelines/penaltyguidelines.htm.

Appeals: A decision made or penalty imposed under Policy 70 (Student Petitions and Grievances) (other than a petition) or Policy 71 (Student Discipline) may be appealed if there is a ground. A student who believes he/she has a ground for an appeal should refer to Policy 72 (Student Appeals) `www.adm.uwaterloo.ca/infosec/Policies/policy72.htm`. Note for Students with Disabilities: The Office for Persons with Disabilities (OPD), located in Needles Hall, Room 1132, collaborates with all academic departments to arrange appropriate accommodations for students with disabilities without compromising the academic integrity of the curriculum. If you require academic accommodations to lessen the impact of your disability, please register with the OPD at the beginning of each academic term.