Graduate Students and Super Undergraduates...

- Look for other sources of information.
- Make decisions, because all research problems are under-specified.
- Evaluate their own work.
- Write, write, write.
- Question everything.

"So long, and thanks for the Ph.D.!” a.k.a.
"Everything I wanted to know about C.S. graduate school at the beginning but didn’t learn until later."
http://www.cs.unc.edu/~azuma/hitch4.html

Plan for Today

**Introductions**
- Name; undergraduate, masters, or phd student
- Career goal and name of your favorite TV show

**Logistics**
- Course details
- How to succeed in this course

**Motivation**
- Why study the analysis and transformation of programs?
- Look at some sample program optimizations and assorted issues

Course Outline

- Concepts
- Expectations
- Next Time

CS 553: Algorithmic Language Compilers (PLDI)

**Instructor:** Michelle Strout
mstrout@cs.colostate.edu
Computer Science Building 244
Office hours:
- Monday 1-2pm, grad director
- Thursday 1-2pm, grad director
- Thursday 2-3pm, CS 553

**URL:** http://www.cs.colostate.edu/~cs553

**Schedule:**
Send email by tomorrow noon if you CANNOT switch to MW 3-4:15pm.
Would have another office hour F 3-4pm.

**Logistics**

**Course website** (http://www.cs.colostate.edu/~cs553/)
- Progress page will have everything (schedule, notes, assignments)
- Assignments are also listed on assignments page

**RamCT**
- Grade book
- Submitting assignments (also send to mstrout@cs.colostate.edu)
- Discussions on forum to help navigate LLVM learning curve
- Mail will be sent through this interface, check your email address

**Succeeding in this course**
- Attend class and participate.
- Check the progress and assignments pages on the website every day.
- Spend at least 1-2 hours per day outside of class doing reading and assignments.
- Embrace the RTFM (read the fine manual) concept.
Class Approach (see Syllabus for more details)

4 programming assignments and corresponding reports (40%)
– Each assignment include a report that will be formatted using latex (a latex template has be provided).
– PA1 has been posted and is due September 8th.
– Writeup discusses possible extension if you use revision control.
Midterm (20%) October 6th in class
– Will ask questions about the reading material.
– Will include concepts discussed in class.
– Example questions from previous midterms and finals will be posted.
Term Project (40%)
– Includes a proposal, demo, final paper, and final presentation.
– Will be investigating a program analysis and/or transformation tool of your choice and related research.
– Details about the term project have been posted.

Expectations

DO
– Expect to spend more time on this course than on a challenging undergraduate course.
– Write more than one draft for your assignment reports. Spelling mistakes will be penalized. Correct grammar is also expected. Consider visiting the writing center.
– Make decisions when the assignment is underspecified. Describe the reasoning for your decisions in the assignment report.
– Read assigned reading. Much of it will take more than two readings and anything in the readings might be on the midterm.
– Break your assignments into small pieces, thus enabling easier progress and debugging.
– Ask questions and come to (or call into) office hours sooner rather than later.

Thinking is important and should be done frequently.

Professional Conduct Slide

Talk with other students about the assignment, but do NOT share code or text. Do not take notes while talking with other students.

Be respectful and polite to others in the class and me.

There is more detail in the Syllabus.

Course Objectives

Short Term
– Learn how to implement program analyses and transformations within the LLVM compiler infrastructure.
– Navigate the learning curve for an existing software tool and start adding features to it.

Long Term
– Learn about research questions in the programming languages and compilers area.
– Learn how program parallelization and optimization is done automatically.
– Evaluate existing automation tools and identify gaps and future research questions.
**Motivation**

Why program analysis and transformation?
- Bug finders: security bugs or otherwise
- Code refactoring tools
- Program understanding tools
- Runtime monitoring systems
- Program verifiers
- To implement new programming languages: Julia, Chapel, Go, etc.

Why the focus on parallelism and data locality?
- The free ride of increasing clock rates is over.
- The power wall has pushed computer architecture to multiple cores. Parallelism will be the main way to improve performance.
- Many threads accessing lots of memory all at once causes a memory bottleneck.

Why scientific computation?
- It’s Science!!

---

**Isn’t Compilation/Automated Parallelism A Solved Problem?**

“Optimization for scalar machines is a problem that was solved ten years ago”
-- David Kuck, 1990

**Machines keep changing**
- New features present new problems (e.g., MMX, SSE, GPU)
- Changing costs lead to different concerns (e.g., loads)

**Languages keep changing**
- Wacky ideas (e.g., OOP and GC and lambda functions) have gone mainstream
- Parallelism is now mainstream

**Applications keep changing**
- Interactive, real-time, mobile, secure, streaming

**Some apps always want more**
- More precision
- Simulate larger systems

**Goals keep changing**
- Correctness
- Run-time performance
- Code size
- Compile-time performance
- Power
- Security

---

**Structure of a Typical Compiler (from CS 453)**

- **Analysis**
  - character stream
    - lexical analysis
    - tokens
    - “words”
    - syntactic analysis
    - AST
    - “sentences”
    - semantic analysis
    - annotated AST
    - interpreter

- **Synthesis**
  - IR code generation
  - IR
  - optimization
  - IR
  - code generation
  - target language

---

**Modern View of Compilers**

- Analysis and performance transformations are useful everywhere
  - Analysis and transformations can be performed at run time and link time, not just at “compile time”
  - Optimization can be applied to OS as well as applications to help make virtualization feasible
  - Analysis can be used to improve security by finding bugs
  - Analysis can be used in software engineering
    - Program understanding, reverse engineering, refactoring
    - Debugging and testing and correctness proofs
  - New parallel programming abstractions can enable the automation of parallelization and performance transformations
  - Bottom line
    - Analysis and transformation play essential roles in computer systems
    - Computation important ⇒ understanding computation important
Some Exciting Current Research in PLDI and PPOPP

Premier conferences for dissemination of compiler and programming languages research
- PLDI (Programming Language Design and Implementation)
- POPL (Principles of Programming Languages)

Parallel Programming Languages
- Most common: C/C++ or Fortran 90+ combined with MPI and/or OpenMP
- Up and coming languages and programming models
  - DARPA HPCS languages: Cray’s Chapel and IBM’s X10
  - PGAS languages like UPC and CoArray FORTRAN
  - CUDA and OpenCL for programming GPUs
  - Google’s Go
  - Julia, which is implemented using LLVM

Yes, but can it help me get a job?

Summer internships in past 10 years
- LLNL with ROSE compiler (2)
- Cray with Chapel group
- NCAR (5)
- Intel working on hand-parallelization based on compiler feedback

Intel, IBM, Mathworks, etc. are all very interested in compilers because they need to provide an effective one to sell their main product.

Many new parallel programming models are in development and use. For example, my group has open graduate research positions for Spring.

Do an internet search on “LLVM job”.

Goal in 553 is to Automate Performance Transformations

Automate parallelization and performance optimization in a compiler
- What is the programming model for the programmer?
- What is the intermediate representation (IR) to enable parallelism and data reuse?
- How can we generate efficient code from the IR?

Run-time Library Support
- The compiler cannot do it all.
- How can the compiler provide information to the run-time system?
- What algorithms should be used to efficiently schedule at runtime?

Course Focus
- The intermediate representation for the compiler.
- Will look at the interaction between the IR and runtime for sparse computations such as molecular dynamics simulations.

What we are going to do in 553

Learn how to implement program analyses and transformations.
- Data-flow analysis and transformation.
- Loop analysis and transformation.

Work within the context of scientific computing.
- What performance transformations are important for these codes?
- What analyses are needed to enable these transformations and how might parallel programming model features simplify the analyses?
- What are the programming patterns that scientists use and how should they affect programming models?

Discover and study research problems in the area of compilers
- Evaluate existing compiler tools (LLVM-based tools, AlphaZ, and Pluto) and read related papers.
- Present research in the form of a paper and a presentation.
Next Time

Reading
- So long and thanks for the PhD
- PA1 writeup

Homework
- Start working on PA1
- Send email if you CANNOT change class time to 3-4:15

Lecture
- Intro to LULESH benchmark and LLVM