ECE 498SL (409 pending): Engineering Parallel Software Spring 2012
Course Overview

Instructor: Steven S. Lumetta (lumetta@illinois.edu)
209 Coordinated Science Lab
phone: 244-5564
office hours: Tu 1-3 upstairs at Caribou (miaZa's when it opens) starting 24 January

The class explores the evolution of programming languages as an engineering process and examines the challenges that face the hardware and software industries with increasing numbers of processors on a chip in order to prepare students to understand and contribute to future language evolution. Students will learn how relationships between language, compiler, runtime, and architecture are used to provide a coherent context in which software developers can reason about correctness and performance, and will gain firsthand experience with tools and techniques through programming assignments. Using the development of C++ as a focal point, the course first examines the expression and implementation of abstractions such as access control, inheritance, templates, and exception handling, identifying both the advantages and the potential pitfalls of these tools. The course continues with an overview of the challenges posed by parallelism, how the high-performance computing community has tried to address those challenges, why many of the solutions have not been adopted by the broader industry, and the relationship with current offerings for desktop parallelism.

You should complete ECE391 or have similar experience before enrolling in this course. Talk with me if you're concerned about your background.

Rationale for the Course:

This course solidifies and extends material to which you have been briefly exposed in earlier programming classes such as ECE190/ECE391 and complements the material in ECE408 (Applied Parallel Programming, the CUDA class, formerly 498AL). The 190 material provides a clear mapping from languages such as C down to hardware, and the lab programs in 391 illustrate implementation aspects of more modern languages as they appear in C. This class will extend that understanding to the level of programming languages now in use by the bulk of the software industry (using C++ for illustration purposes). The 498AL class provides hands-on experience in mapping applications to parallel platforms. This class will complement that experience by illustrating the relationship between the challenges in the parallel software development process and the engineering tradeoffs (e.g., loss of performance) imposed by a range of approaches for meeting those challenges.

Textbooks and Materials:

- B. Stroustrup, "The Design and Evolution of C++," Addison-Wesley, 1994, ISBN 0-20-154330-3.
- G.F. Pfister, "In Search of Clusters: The Ongoing Battle in Lowly Parallel Computing," 2nd edition, Prentice-Hall, 1998, ISBN 0-13-899709-8.

Both books are fairly light reading, and will be supplemented by class notes and an occasional paper from the literature. Papers will be rare, given that I expect a fair number of undergraduates; and will be supplanted by summaries of the papers when possible.

What you should expect:

There will be **three or four combined homework and lab assignments** involving a combination of exercises and programming. You may collaborate on written exercises at your discretion so long as you credit all students involved (turn in one copy with all names); you should be mature enough to recognize that adding your name without thinking about the assignment is likely to reduce your exam scores. The programming components will be less flexible, although we will have a combination of team and individual assignments. You may not share programming solutions in any form outside of your team (or with anyone, for individual assignments).

There will be **one midterm exam** and **one final exam**. The midterm exam will be held **in-class** on Tuesday 6 March. The final exam is on Monday 7 May from 1:30 to 4:30 p.m. Any conflict that you have with either exam **must be reported** to me **at least one week before the exam**, but please report such conflicts as early as possible.

Grading mechanics:

Homework/Lab Assignments:	40%
Midterm Exam:	25%
Final Exam:	35%

Web board and page:

The ECE498SL "web board" is available through the web boards link at http://my.ece.illinois.edu, and will serve as our primary means for communication. As in your earlier classes, the web board serves as a forum for students to post and answer questions, discuss issues, warn of pitfalls, etc. You should read the board at least once a day. I will read and post to the web board to focus discussions and to provide more definitive answers to posted questions. The class web page can be found at http://courses.engr.illinois.edu/ece498/SL/.

Academic Integrity:

Although I encourage you to study together, work products of this course (programming assignments and examinations) must be your own individual work unless explicitly stated otherwise. Do not, for example, exchange code with people outside of your team. If you cheat, you violate the soul of the University, which I take very seriously, and will not compromise. First offense will, in the least, result in a 0 on the assignment or exam. The policy for the course is based on Article 1.4 of the *Student Code* (available at http://www.admin.illinois.edu/policy/code/).

Tentative Syllabus

- introduction and motivation — 1 lecture
- historical overview — 1 lecture
- module design: from C to C++ — 2 lectures
  - best practices in C for data hiding, modular design, reusability, and interface abstractions (seen in 391 labs, but not discussed in lectures)
  - C++ features to overcome C's pitfalls: classes as modules, access control, namespaces, inheritance, virtual functions, and initialization
  - comparative example based on network connections
- modern language abstractions: what, why, and how — 5 lectures
  - exception handling
  - overloading, references, and variable declarations
  - memory management
  - templates and the Standard Template Library
- abstraction vs. performance — 5 lectures
  - interaction between design time and performance goals
  - profiling and tuning: case studies and tools
  - performance reasoning for shared memory threads
- challenges for parallel programming (concepts and examples) — 3 lectures
  - expressing atomicity and precedence/dependence
  - architectural impact on algorithm selection
  - synchronization and the inheritance anomaly
  - determinism and debugging: the cost of verification
- overview of common approaches to parallelism — 3 lectures
  - dimensions for parallel models
  - categorization of approaches: data parallel, shared memory/OpenMP, message passing/MPI, Unified Parallel C, and actors
  - concepts, misconceptions, and useful ideas
- shared memory: theory and practice — 3 lectures
- task models: Intel's TBB, Cilk, Charm, OpenMP, Rigel — 2 lectures
- transactional memory and guarded atomic execution — 2 lectures
- message-passing with distributed memories; design tradeoffs in message handling — 1 lecture
- midterm exam — 1 lecture