

ECE 498SL, Spring 2009

Course Overview

Instructor: Steven S. Lumetta (lumetta@illinois.edu)
209 Coordinated Science Lab
phone: 244-5564
office hours: Tu 1-3 upstairs at Za's (may change later)

Objectives of the Course:

This course examines the relationships between language, compiler, runtime, and architecture in providing a coherent context in which software developers can reason about correctness and performance. The material covers both state-of-the-art design and explores the options for future design processes. The class begins with an overview of the abstractions provided by modern programming languages such as C++, the rationale for their design, and the mechanisms used to implement them. Next, we examine the challenges that face the hardware and software industries with increasing numbers of processors on a chip. Finally, the course reviews the approaches used by the high-performance computing community and illustrate how the architecture, runtime, and compiler can be leveraged to abstract away some of these challenges.

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 190 and 391, and complements the material in Wen-mei Hwu's 498AL (which is pending permanence). 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 **four combined homework and lab assignments** involving a combination of exercises and programming. As with 391, 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 probably 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 around Tuesday 10 March. The final exam is on Thursday 14 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%

Graduate students may also elect to enroll for an additional unit (probably as independent study under my guidance) in which students work in self-identified teams of two to four to develop a final project related to the course material. This additional work will be incorporated into the final grade in a way that does not affect students not taking the additional unit.

Web board and page:

The ECE498SL “web board” is available through the web boards link at <http://my.ece.uiuc.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.

I will also set up a class web page that you will be able to find from the ECE courses page (<http://courses.ece.uiuc.edu/ece498/sl/> is the default location, but it’s currently occupied) once it becomes useful.

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.uiuc.edu/policy/code/>).

Tentative Syllabus

- C methods for data hiding, modular design, reusability, and interface abstractions (seen in 391 labs, but not discussed in lectures) 2 lectures
- modern language abstractions: what, why, and how 5 lectures
 - inheritance and type benefits
 - type polymorphism and virtual functions
 - strong types and pointer analysis
 - templates and iterators
- abstraction vs. performance 3 lectures
 - interaction between design time and performance goals
 - composition and interface specification
 - planned extensibility; software process engineering
- challenges for parallel programming 4 lectures
 - expressing atomicity and precedence/dependence
 - architectural impact on algorithm selection
 - the inheritance anomaly
 - determinism and debugging: the cost of verification
- overview of system models and implications for reasoning about program behavior
 - review of processor virtualization (from 391); parallel threads using coherent shared memory 3 lectures
 - message-passing with distributed memories; design tradeoffs in message handling 3 lectures
 - bulk synchronous structure 1 lecture
 - task models: Cilk, Charm, OpenMP, Rigel 3 lectures
 - implicit parallelism: thread-level speculation and control independence 3 lectures
 - transactional memory and guarded atomic execution 2 lectures