

ECE563 Programming Parallel Machines

Tentative Syllabus

Spring 2009 Tu/Th 4:30-5:50 PM, EE 226

Instructor Prof. R. Eigenmann
Tel 49-41741
Email eigenman@purdue.edu
Office EE334C
Office Hours Tu 8:30-10:00 AM and by appointment
Secretary: Jill Comer, EE339
Course Web Page: <http://www.ece.purdue.edu/~eigenman/ECE563/>

Prerequisites: EE565 (Computer Architecture). Substantial programming experience.

Text: Research papers and course handouts.

Description:

This course presents methods and techniques for programming parallel computers, such as multicore and high-end parallel architectures. Various parallel algorithms are presented to demonstrate different techniques for identifying parallel tasks and mapping them onto parallel machines. Realistic science/engineering applications and their characteristics will be discussed. Parallel architectures to be considered are multicores, multiprocessors and distributed-memory multiprocessor systems. Several emerging architectures will also be discussed, e.g., GPUs and Cell processors. Programming paradigms for these machines will be compared, including directive-based (OpenMP), message passing (MPI) and thread-based (Posix threads) methods. Methodologies for analyzing and improving the performance of parallel programs will be discussed. There will be a class project in which students parallelize and tune the performance of a large computational application or develop/improve a tool that helps this process.

Outcomes:

A student who successfully fulfills the course requirements will have demonstrated:

- an understanding of the basic features of parallel computer architectures and their relationship to parallel program design.

- an ability to analyze a program for parallelism and express this parallelism for both shared-memory and distributed-memory machines.
- an understanding of parallel models and programming constructs for OpenMP, MPI, and Posix threads.
- an understanding of performance factors of parallel program executions and their relationship to application characteristics and parallel programming constructs.
- an ability to use parallelizing compilers to parallelize and tune the performance of application programs.

Course Grading:

Tests	60%. (30% midterm, 30% final exam)
Participation in class	15%.
Projects	25%

Regrading policy: any information that you feel will affect your grade, including grade disputes and emergencies that prevent you from attending class, must be sent by email to the instructor. In general, the instructor will not respond to these notes or change your intermediate grades. However, the information will be factored into your final class grade.

Course schedule:

				Week	
Monday	Jan 12	Sem starts	T R	1	
	19		. R	2	Mon, Jan 19: M.L.King Day
	26		T R	3	
	Feb 2		T R	4	
	9		T R	5	
	16		T R	6	
	23		T R	7	
	Mar 2		T R	8	
	9		T R	9	Thu, Mar 12 Midterm exam
	16		. .	.	Spring Break
	23		T R	10	
	30		T R	11	
	Apr 6		T R	12	
	13		T R	13	
	20		T R	14	
	27		T R	15	
	May 4	final exams week			Final exam: TBD

Tentative Course Outline:

	Topic	#weeks
1.	Introduction and Motivation	1
2.	Program parallelization techniques	1
3.	Tuning parallel programs	1
4.	Explicit vs. automatic parallelization	2
5.	OpenMP	1
7.	MPI	1
8.	Pthreads	1
9.	Other models	1
10.	Programming methodologies and tools	2
11.	Application studies	1
12.	Project Discussions	2

Class Projects: Groups of two students will conduct a class project. There are two options:

1. An application of the students' choice will be converted to a parallel program and executed on a parallel machine. The project reports will document the original and tuned performance for each program section as well as intermediate results of each program tuning step.
2. The students can develop a new or improve an existing program development tool. The goal is to create a tool that will be of use to the students who conduct performance tuning projects. The project report must contain evidence for the improvement of the performance tuning process due to the tool. Appropriate performance metrics must be defined.

Important findings of the projects will be presented in a 10-20 minute talk at the end of the semester.

A project proposal draft is to be prepared by the end of the second week and the final proposal by the end of the third week of class. Application tuning project proposals will describe, in two pages, the application to be chosen, (including programming language, number of lines, application area, relevance for application field), the machine(s) to be used in the performance study, the parallel programming model(s) to be used, opportunities for parallelization, and the expected performance (i.e., speedup) of the parallel application. Tool project proposals will include a brief description of the existing tool (if any), the tool features that will be developed, rationales for these features, and a tool evaluation section (metrics for tool evaluation and benchmarks). The schedule for intermediate and final reports are posted on the course web page.