ECE563 Programming Parallel Machines

Tentative Syllabus

Spring 2013 Mo/We/Fr 2:30-3:20 PM, EE 224

Instructor	Prof. R. Eigenmann
Tel	49-41741
Email	eigenman@purdue.edu
Office	EE325
Office Hours	Tu 8:30-10:00 AM and by appointment
Secretary:	Wanda Dallinger, EE326B
Course Web Page:	http://engineering.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. 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, accelerators (such as GPGPUs), multiprocessors and distributed-memory multiprocessor systems. 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 sizeable computational application or develop/improve tools that help in 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 contructs 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	50%. (20% midterm, 30% final exam)
Participation in class	15%
Projects	35%

Notice the high percentage given to class participation. Good class participation means volunteering answers to questions and contributions to discussions without being called on.
For project grades see the projects section below.

Regrading policy: any information that you feel will affect your grade, including grade disputes and information about emergencies that prevent you from attending class or submitting reports on time 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. You may discuss your class standing and tentative grade with the instructor before the end of the course. Once the grades have been finalized, no remedial actions can be granted.

Course schedule:

```
                                    Week
Monday   Jan  7 Sem starts    M W F    1
             14               M W F    2
             21               . W F    3    Mon, Jan 21: M.L.King Day
             28               M W F    4
         Feb  4               M W F    5
             11               M W F    6
             18               M W F    7
             25               M W F    8
         Mar  4               M W F    9    Wed, Mar 6 Midterm exam
             11               . . .         Spring Break
             18               M W F   10
             25               M W F   11
         Apr  1               M W F   12
              8               M W F   13
             15               M W F   14
             22               M W F   15

         Apr 29      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, proposed by the students. There are two options for proposals:

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

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

A project proposal draft is to be prepared by the end of the second week (Friday 11:59 PM); the final proposal is due at the end of the third week of class. Application tuning project proposals will describe, in two pages, the application to be chosen, (including a brief application description and its relevance, the programming language, and the number of lines), the machine(s) to be used in the performance study, the parallel programming model(s) to be used, opportunities for parallelization and tuning (i.e., the programming and performance improvement steps you expect to apply), and the expected performance results from the proposed work. Tool project proposals will include a brief description of the tool that will be extended (if any), the tool features that will be developed, rationales for these features, and a tool evaluation section (metrics for tool evaluation and benchmarks). All proposals should include and a timeline, indicating when you expect to have completed what part of the project (3-4 milestones during the semester).

An intermediate report is due at the end of week 10 and the final report is due on the last day of class. The due dates are also posted on the course web page. The final (intermediate) project report is approximately 10 (5) pages, excluding code, 11pt font, double spaced. The front page of each report must include the project title, student names and email addresses, the course number (ECE663), and whether it is the draft proposal, final proposal, intermediate, or final project report.

Send the project proposals and reports in PDF format to eigenman@purdue.edu before the due date.

Do not request due date extensions. If there are circumstances beyond your control that delay a submission, send a separate note of explanation (with evidence, if applicable) to the instructor, right *after* you have made the late submission. Such explanations will be considered as described under *regrading policy*. Late submissions of the final project reports cannot be accepted, however.

For a good project grade, it is important that the project reports provide evidence that the students understand their performance results. The reports should also express the effort that went into the project (indicate the average hours/week spent on the project, for each student, and the distribution of this time over the 15 weeks)