

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.846: Parallel Computing

Spring 2008, Agarwal

Handout #1

Course Description and General Information

Prof. Anant Agarwal
Phone: (617) 253-1448
email: agarwal@mit.edu
Office: 32-G782, MIT CSAIL
Hours: by appointment

Teaching Assistant:

Henry (Hank) Hoffmann
Phone: (617) 253-0674
email: hank@cag.csail.mit.edu
Office: 32-G740, MIT CSAIL
Hours: TBA

Secretary:

Simone Nakhoul
Phone: (617) 253-2629
email: snakhoul@mit.edu
Office: 32-G784A

Web page:

<http://www.cag.csail.mit.edu/classes/6.846>

Time:

Tues Thurs 2:30–4:00PM
Place: 26-210

1 Overview of the Course

With the aggressive adoption of multicore by our computer industry, parallel computing is no longer a forward looking research topic, rather it has come mainstream. This course focuses on how to program and to design multicores and other parallel computers. Accordingly, the course caters both to students interested in learning how to use multicores

effectively, and to those interested in parallel computing architectures.

The desire to efficiently solve important problems drives both the multicore user and the parallel computer designer. Consequently, we will start with discussions of several typical numeric, combinatorial, and embedded applications and focus on one or more of them to illustrate concretely the tradeoffs in parallel programming.

After the overview of typical problem classes and specific applications, the course will cover the following: computational models and parallel programming—shared-memory, message-passing, data-parallel, and dataflow; parallel machine organization—processors, cache and memory systems, coherence, interconnection networks, synchronization; performance evaluation methodologies—probabilistic analyses, queuing analysis, simulation; compiler technologies—fine-grain and coarse grain techniques; run-time and operating systems—load balancing, dynamic partitioning, and protection in multiuser systems.

The course includes a laboratory component which will involve programming the 64-core Tile multicore processor.

2 Lectures

The course will consist of lectures and discussion. A few readings (typically two or three) will be assigned each week. We will discuss the assigned papers during part of the lectures.

3 Grade

Grades will be assigned according to the following weighting factors

- Homeworks/Labs—30%
- Class Participation—20%
- Project—50%

4 Homework/Labs

There will be roughly one homework every two weeks. Although students are encouraged to work together on homeworks and laboratories, each student is expected to turn in their own complete writeup. Some of the homeworks will require writing small programs and using a multicore computer and simulators that we will provide. A laboratory handout describes how the multicore computer and simulators can be accessed. Students will be given accounts on our local machines for the laboratories.

5 Readings

You will be expected to read several papers or lecture notes each week.

6 Projects

The course will involve a substantial project. Students will also be able to compete in an optional programming contest as part of the project. Joint projects are encouraged. You will be required to turn in a paper before the last week of the term describing the project. You will also be required to make a short presentation of your results to the class during the last week of the course.

Here are some ideas for projects:

- Write a parallel application. Variations on this theme include: (1) Comparing various approaches to parallel programming such as shared memory, message passing, or streams. This variation works well for team projects in which each student codes the same application in a different style; (2) studying the application's synchronization or communication behavior; (3) studying its performance when written using different partitioning approaches; (4) Finally, simply turbo charging the application for speed.
- Analytically investigate the optimal grain size of a core in multicore processors.
- Develop a new architectural idea and evaluate its effectiveness by implementing a simulator.
- Propose a new compiler algorithm and implement it as part of an existing system to evaluate its usefulness.
- Critically evaluate an architecture or an architectural feature.
- Perform analytical studies of some architectural mechanisms. For example, using an analytical model, compare the performance of a multicore processor containing single threaded versus multithreaded cores.

Feel free to stop by and discuss ideas for projects with the course staff. *Get started on the project early.*