

Texas A&M Spring 2008, CPSC 689-608  
Multicore CPU Architectures and Software Development  
Blocker 608M, Tuesday 3:55-6:00

Instructor	Craig C. Douglas (craig.c.douglas@gmail.com)
Office	608G Blocker
Office Hours	By appointment
Class web site	<a href="http://www.mgnet.org/~douglas/Classes/multicore">http://www.mgnet.org/~douglas/Classes/multicore</a>
Textbook	None, the course will focus on reading research papers posted on the course web site.

### Course Description

Central processing units (CPUs) have undergone a revolutionary change recently. CPU vendors ran into technical hurdles (e.g., excessive heat, too high electrical power requirements, and not well understood physics) at increasing the clock speeds that have forced them to switch to putting a number of cores onto CPUs instead of a single core.

The move to a multicore architecture requires a complete change in software development in order to utilize the new CPUs effectively. No longer can programmers just see their programs run faster as clock frequencies increase. Instead coarse grained parallelism becomes of key importance to all new software development, including on single CPU systems.

The goal of the course is to study multicore CPU architectures, implications of hardware designs, software challenges, and emerging technologies relevant to software development for multicore systems. Topics include multicore microprocessors, memory hierarchy, programming models, scheduling, debugging, and memory use. How this relates to software development is the binding issue.

This course will focus on reading, analyzing, and discussing research papers related to multicore systems and developing software for this environment.

### Prerequisites

General knowledge of computer systems and computer architecture will be assumed.

### Grading Criteria

<u>Percentage</u>	<u>For what</u>
10	Class participation
10	Paper evaluations
40	Class presentations
10	Presentation evaluations
30	Project

### *Class participation*

Research papers are not always well written, sometimes make misleading claims, and even sometimes contain errors. Students are expected to contribute to the discussion of the papers in class by subjecting the readings to critical analysis, asking questions, and offering observations.

### *Class presentations*

Students will be required to plan and deliver three class presentations. A presentation on set of papers should include

- Motivation for the work.
- Key algorithms, strategies, techniques, insights, and results.
- Critical evaluation of the work.
- Remaining open issues.

While not required, presenters should show their presentations to the instructor in advance. Each presentation will have one class period (75 minutes). Within 48 hours of the presentation, the presenter will provide the instructor with a version of the presentation suitable for posting on the class web page. Correct English grammar and spelling is required. Errors discovered during the presentation or answers to audience questions should be included in the turned in version.

### *Paper evaluations*

Students are required to evaluate 67% of the papers (your choice of which ones). A paper evaluation consists of

- Your name.
- The paper name.
- A paper summary of at most 5 sentences.
- The three most important strengths (1 sentence on each).
- The three most important weaknesses (1 sentence on each).
- A summary of one issue or open problem of at most 3 sentences.

Paper evaluations will be graded on a scale of 1-3. The default grade is 2. Insightful reviews will receive a 3. Disappointing reviews will receive a 1. Students will email evaluations of papers to the instructor ([craig.c.douglas@gmail.com](mailto:craig.c.douglas@gmail.com)) *before the start of the class* in which the paper is presented. Late or incomplete evaluations will receive a 0.

### *Presentation evaluations*

You are required to evaluate at least 67% of the student presentations in class. There are two motivations for this. First, if you have to write reviews of presentations, you will pay closer attention. Second, if you know that your presentation is being evaluated by your classmates (as well as the instructor), you may try harder to make it engaging. The ability to make engaging presentations to intelligent audiences is one of the most important skills that you need.

A presentation evaluation consists of

- Your name.
- The name of the presenters or presenters.
- How well did the presenters explain why the area matters?
- Did the presenters mumble, fail to make eye contact, speak too quickly or too slowly?
- Were the presentation slides too busy, too ugly, or just right?
- How well did the presenters seem to know the material? Were they honest about admitting when they did not know something?
- Any additional information that you would like to add.

Presentation evaluations will be graded on a scale of 1 to 3. The default grade is 2. Insightful evaluations will receive a 3. Disappointing evaluations will receive a 1. Evaluations for presentations that I believe the reviewer did not attend will receive a 0. To receive credit for a presentation evaluation, email it to the instructor ([craig.c.douglas@gmail.com](mailto:craig.c.douglas@gmail.com)) before Sunday noon in the week in which the presentation occurred. Late or incomplete evaluations will receive no credit.

Presentation evaluations are intended to be helpful. Give constructive criticism, not nasty, mean comments (i.e., do to others as you would have them do to you). Your evaluations will be kept anonymous. I will merge and edit presentation evaluations submitted and forward them to the presenter(s). I will suppress inappropriate or unhelpful comments.

### *Final project*

This course requires a term paper or a final software project with a written project report.

Term papers must be done individually. For a term paper, you may focus on the same topic as one of your presentations, but you should study different papers. Consult the instructor if there is any question.

Final projects will typically be done individually, but they may be done by a pair with instructor permission. Group projects will submit a single paper. A project involving application development or experimentation may build upon existing code, but consultation with the instructor is in order. Code that is obtained from other sources should be clearly identified.

## Outline of Course

Lecture	Description
1	Software and the concurrency revolution, simultaneous multithreading, and the case for chip multiprocessing
2	Maximizing on chip parallelism through multithreading on single chip multiprocessors
3	Scalable single chip multiprocessing systems
4	OpenMP for single chip parallelism
5	MPI for single chip parallelism
6	Scheduling in languages for fine grained parallelism
7	Shared caches among threads and how to use them efficiently
8	Shared memory consistency models: A tutorial
9	The Java memory model
10	Algorithms for scalable synchronization on shared-memory multiprocessors
11	Race detection methods and algorithms for multithreaded programs
12	Debugging shared memory programs
13	Implementing highly concurrent data structures
14	Simple, fast, and practical non-blocking and blocking concurrent queue algorithms
15	Project discussions
16	Threads implementations and scheduling for single chip multiprocessors
17	What do high-level memory models mean for transactions?
18	Software transactional memory
19	Transactional memory coherence and consistency and nonblocking transactions without indirection using alert-on-update
20	Disjoint-access-parallel implementations of strong shared memory primitives
21	Speculative thread decomposition through empirical optimization
22	Tight analysis of the performance potential of thread speculation using SPEC CPU 2006 benchmarks
23	Optimizing memory transactions
24	Conditional memory ordering: Memory model = instruction reordering + store atomicity
25	Techniques for reducing overheads of shared-memory multiprocessing
26	Applying thread-level speculation to explicitly parallel applications
27	Project presentations
28	Project presentations