ENEE 759A: Parallel Processing Computer Architectures

Course Goals:

Today, small parallel computers, consisting of a few to a dozen processors, are common place given the commercial demand for high-performance servers. It is natural to extend these modest parallel architectures to large-scale systems. Such large-scale multiprocessors are cost-effective alternatives to more traditional supercomputers for highly computation-intensive applications. Indeed, over the past decade, we have seen significant activity in the computer architecture community related to building and studying large-scale multiprocessors. This course teaches how to program, design, and evaluate large-scale multiprocessors.

This course addresses fundamental issues in the design and use of large-scale multiprocessors. Both software and hardware issues are addressed. In the software area, the course will examine parallel applications and their computation requirements, including how they are expressed using parallel programming languages. The course will also look at runtime software that provides the system-level support needed in a parallel architecture. In the hardware area, the course will examine all facits of the design of multiprocessors, including processor support for parallelism, memory system design, and interconnection networks. An emphasis will be placed on design for balance; therefore, when studying hardware issues, the course will also look back at the application requirements to determine the relative investments needed in processor speed, memory size, and network bandwidth and latency. Finally, the course will also examine the tools and techniques necessary for evaluating multiprocessors. The course will survey the different simulation technologies used for studying multiprocessors, and will look at several instrumentation techniques commonly used to study performance.

Course Prerequisite(s):

ENEE 646, or equivalent.
ENEE 114, or equivalent.
Students with questions regarding their course background should consult the instructor.

Topics Prerequisite(s):

This course assumes a basic understanding of computer architecture at the first-year graduate level. Students should have a grasp of basic topics in computer architecture including pipelining, cache memory hierarchies, virtual memory, and I/O subsystems. Also, in order to complete the required programming assignments (see *Course Requirements* *and Grading* in section *Information for Spring 99 Semester* below), students should be familiar with the C programming language, and have taken an undergraduate-level course on software engineering.

Textbook(s):

The material for the course will consist of a single course pack which will include 1). technical papers from the parallel processing literature, and 2). course notes provided by the instructor. No formal text will be used in the course. Students will read papers from the course pack, about four per week. The papers on any given week will pertain to the topics covered in the lectures. The lectures themselves will follow the course notes provided in the course pack.

Reference(s):

Core Topics:

The course is divided into three broad categories: software, evaluation, and hardware design. In the software category, the course will look at applications and computational models. The evaluation category looks at simulation methods and instrumentation techniques. Finally, the hardware category looks at several aspects of multiprocessor design.

1. Applications
    - Continuum
    - Particle
    - Graph
    - Parallelizing algorithms
    - Processing and communication requirements
2. Computation Models
    - Communication and synchronization
    - Shared memory MIMD
    - Message passing MIMD
    - Data parallel SIMD
    - Dataflow MIMD
    - Example machines and languages
3. Simulation
    - Trace driven
    - Execution driven
    - Direct execution and binary translation
    - Example simulators
4. Instrumentation
    - Performance metrics
    - Implementing statistics
    - Analytical modeling

5. Impact of technology
    - Physical limits and constraints
    - VLSI opportunities
    - Packaging, wires
6. Processors
    - Block multithreading
    - Synchronization support
7. Memory systems
    - Scalable memory organizations
    - Caches and the coherence problem
    - Snooping
    - Scalable directories
8. Interconnection networks
    - Direct mesh and cube networks
    - Indirect omega networks
    - Bandwidth and latency

Optional Topics:

Information for Spring 99 Semester:

Instructor: D. Yeung	Web page: http://www.ee.umd.edu/~yeung/
Phone: 405-3649	e-mail: yeung@eng
Hours: By appointment	Office: AVW 2713
Time of Course: MW 11-12:15	Room: TBD

Course Structure: In addition to completing all reading assignments from the course pack, the course requirement consists of homework assignments, a midterm, and a final project. Student performance on each of the three requirements will contribute to the student's final grade using the following breakdown:

Homework assignments: 30%
Midterm: 30%
Final project: 40%

The homework assignments will include application and algorithm implementation problems, hardware design problems, and evaluation of design tradeoffs (all on paper). The most significant portion of the homework assignments, however, will center around a series of programming assignments that will require the students to implement parallel code, compile it, and run it on a simulator of a multiprocessor. Significant hands-on experience will be gained in the design of parallel software, how to debug it, and how to evaluate its performance.

The midterm will be a closed-book exam given in-class, and will test the student's knowledge of the material covered. The questions on the midterm will resemble those in the paper-based homework assignments.

Finally, each student will be required to undertake a significant final project. The instructor will suggest several broad topics, but it will be up to the students to propose their own projects, subject to the approval of the instructor. Students will be given substantial latitude in designing their projects, but each project must demonstrate novel design related to parallel processing either in applications, system software, or hardware. Furthermore, the project must conduct an evaluation of the design in terms of performance or functionality to ascertain the success of the design in meeting proposed design goals. Students will be allowed (and encouraged) to work in teams on their projects, but multi-member projects must demonstrate substantially greater effort than single-member projects.

Last Updated:

17 June 1998, Donald Yeung

khodary@eng.umd.edu