COMP 4510 – Introduction to Parallel Computing

Course Description

Calendar entry

An overview of the architectures of current parallel processors and techniques used to program them. Not to be held with ECE 4530. Prerequisites: COMP 3370 and COMP 3430.

General Course Description

This is a senior level course focusing on the current issues of parallel computing. We take an in-depth look at techniques for the design and analysis of parallel algorithms, discuss commonly available commercial platforms to program such algorithms (e.g., OpenMP, MPI, CUDA), and look at the current state and future directions in parallel computing technology.

We will study various issues in developing parallel algorithms (e.g., load balancing, communication/synchronization latencies) on different multi-core parallel architectures along with design techniques (e.g., pipelining, producer-consumer, fork-join, single program multiple data, single instruction multiple threads) and their implementation for topics such as:

- sorting (e.g., hyper quicksort),
- searching (e.g., DFS, BSF),
- graph algorithms (e.g., shortest path, all pair shortest path, minimum spanning tree),
- scientific computing (e.g., dense matrix algorithms, Cannon's algorithm, Gaussian elimination),
- dynamic programming (e.g., matrix-parenthesization problem, longest common subsequence), and
- science and engineering applications (e.g., n-body problem, computing pi).

We will also cover various interconnection networks (e.g., mesh, hypercube, ring) and their impact on the performance and scalability of the parallel program using metrics such as speedup and efficiency.

In addition to these core aspects of parallel computing, a specific offering of this course may select from any number of interesting parallel computing related topics. Please see an offering's course website for details.

Detailed Prerequisites

Before entering this course, a student should be able to:

- Explain how memory organization can both positively and negatively impact the performance of their code.
- Explain modern CPU design and how those design choices lead to our ability to implement sophisticated algorithms.
- Analyze the algorithm and data structure choices made to implement resource management in the form of CPU scheduling and memory partitioning.
- Implement software that makes use of core operating system functionality through concurrent shared memory and message passing programming models.
- Use a debugger to inspect program state and step through code line-by-line.
- Manipulate memory buffers through pointers and address arithmetic.
- Implement algorithms using the algorithmic design techniques found in COMP 2080 and COMP 3170.

Core Course Goals

By the end of this course students will:

- Be introduced to programming using the message passing paradigm, shared address space platforms, and accelerators.
- Design and implement parallel versions of a variety of common algorithms (taken from 2080 and 3170) using message passing and threading models.
- Analyze the performance and scalability of parallel algorithms based on the programming platform used.
- Discuss a variety of applications of parallel computers.

Core Learning Outcomes

Models of Parallel Computers/Computation

Students should be able to:

1. Compare and contrast single program multiple data model (message passing and shared and distributed address space) and single instruction multiple threads model (accelerators).
2. Explain how simultaneous multithreading, supported through architectural designs such as multicore processors, provides parallel processing.
3. Explain how non-uniform memory access, supported through interconnection networks such as hypercube, provides parallel processing.

Designing Parallel Algorithms

Students should be able to:

1. Apply partitioning, communication, agglomeration, and mapping techniques to a parallel design.
2. Analyze and solve synchronization issues arising in a parallel algorithm.

3. Identify load balancing opportunities and apply them to a parallel algorithm.

Message Passing Computing

Students should be able to:

1. Implement parallel algorithms using MPI.
2. Design and implement parallel message passing solutions to algorithms from COMP 2080.
3. Design and implement parallel message passing solutions to algorithms from COMP 3170.

Shared and Distributed Shared Memory Models

Students should be able to:

1. Explain the impact of parallelism on cache memory and how cache coherence protocols solve these problems.
2. Implement parallel algorithms using OpenMP.
3. Design and implement shared memory solutions to algorithms from COMP 2080.
4. Design and implement shared memory solutions to algorithms from COMP 3170.
5. Implement parallel algorithms using hybrid programming (i.e., MPI with OpenMP).

Performance and Scalability

Students should be able to:

1. Explain Amdahl's and Gustafson's laws.
2. Analyze the scalability of a message passing algorithm in terms of communication and computation time.
3. Analyze the scalability of a shared memory algorithm in terms of communication and computation time.

Additional Topics in Parallel Computing

Along with the core learning outcomes listed above, the instructor can choose from some of the following to supplement their offering of the course. The instructor is also free to include parallel computing related topics of current interest that are not listed here.

- Implementing 2080/3170 algorithms using CUDA
- Accelerators and GPUs
- Heterogenous multi-core processors
- Quantum computers
- Read and review articles in the parallel computing literature