Catalog Data: Concurrent Scientific Computing (3)
Problems of synchronization, concurrent execution, and their solution techniques. Design and implementation of concurrent software in a distributed system. Prerequisite CMPSC 121, CMPSC 201 or CMPSC 202; MATH 220; MATH 230 or MATH 231.

Typical Textbook: Parallel Programming in C with MPI and OpenMP by M. J. Quinn (published by Addison Wesley). The students will also read 3-4 papers on topics related to scientific computing and parallel programming.

Course Objectives: CMPSC 450 introduces students to the fundamentals of parallel programming. Topics include learning overview on parallel machines, introduction to the art of developing parallel algorithms for discrete (combinatorial) and continuous (PDE) problems, their implementation in MPI and evaluation of the algorithms. The objective of the course is to enable the students to exploit the inherent parallelism of an application. At the end of the course they should be able to detect concurrent structures in a sequential algorithm and design a parallel technique for the same. They should be comfortable in implementing parallel algorithms in MPI and also know how to evaluate (in terms of speedup, isoefficiency, etc.) parallel algorithms. Since the focus is on scientific computing they should also be able to discretize simple PDE equations and transform them to matrices as a preprocessing step to parallelism.

Primary Course Outcomes: Upon completion of their course the students should possess the following skills:

- Identifying Concurrency: Given a problem statement, the students should be able to identify the concurrent operations and design a parallel algorithm for the problem.
- Analysis of Parallel Program: The students should be able to evaluate the effectiveness of a parallel code such as by measuring the such as measuring the speedup (weak and strong scaling), isoefficiency and communication costs.
- Program Implementation: The students should be able to implement a parallel algorithm in MPI. They should also be able to perform simple discretization operations on PDEs.
- Overall Knowledge of Parallelism: The students should have an overall knowledge of the science of parallelism including overviews on parallel architectures and parallel programming languages.

Relationship to Undergraduate Program Outcomes: CMPSC 450 supports the following program outcomes:

- Demonstrate an ability to analyze the space/time complexity of algorithms using discrete mathematics, including the appropriate use of $O$-nation and recurrence relations.
- Write clear and effective technical prose.
- Demonstrate independent learning by using unfamiliar computer systems, test equipment, and software tools to solve technical problems.
- Be able to discuss major trends in industry and current research activities within the discipline.
- Demonstrate an ability to work effectively in multi-disciplinary teams. The term multi-disciplinary is used here in the broader sense to include teams of computer professional having different skills; e.g., one team member might be familiar with web development, whereas another might have experience with microprocessor systems.
Required Topics: 1 week = 3 hours – Total 15 weeks
(38 hrs. total)
• Introduction, overview of parallel programming (1 week)
• Overview of parallel architectures (1 week)
• Analysis of parallel programs with examples from sample applications (2 weeks)
• Introduction to MPI (.5 week)
• Discrete Parallel Problems (some examples: Sieve of Eratosthenes, Floyd's Algorithm, Sorting, Game Trees) (3 weeks)
• Matrix Computations (2 weeks)
• Parallelism of PDE equations (3 weeks)
• Shared memory programming (1 week)
• Revision Classes and Discussion (Scattered through the semester) (1.5 weeks)

Class Format: Two lectures (along with in class programming assignments) per week (duration of each lecture is 75 minutes)

Professional Component: The goal of CMPSC 450 is to enable the students to tap the potential of parallel algorithms. They will be trained to design implement and evaluate programs using parallel heuristics.

Evaluation: There will be one test and one final, each worth 20% making a total of 40%. There will be one programming project worth 25%, assignments (which might include small programming tasks) worth 20% and presentation of papers worth 15%.

Author: Sanjukta Bhowmick
Last Revised: May 1, 2008