Course: Introduction to Parallel Scientific Computing

Semester: Spring

Instructor: Dr. P. Kumar

Type: CS/Open Elective

Requirements: Math-I (Graphs), Math-II (Basic Linear Algebra), Math-III (Basic Probability and Stats.), Programming in C/C++, Familiarity with basic Linux environment, Algorithms, and Data Structures. Knowledge of Python and Matlab can help in quick prototyping and in some assignments.

Credit: 4

Weightages:

- **Mid-sem-1: 10%**
- **Mid-sem-2: 10%**
- **Assignment: 10%**
- **Projects: 40% [5 page report + github code]**
- **End-sem: 30%**

Note the following:

1) Many parallel algorithms and hands-on demonstration is done during tutorials. So, attending Tutorials is a recommended!
2) This course is attendance free, but attending is strongly recommended.
3) A working knowledge of C/C++ is essential.
4) The projects may be started immediately after Mid-1.
5) The classes will alternate between HPC and Scientific Computing.

Scientific Computing: Basic Algorithms for Data Science

Matrix computations show up almost everywhere: scientific computing, data mining, computer vision, machine learning, etc. In this topic, we will learn key ideas and algorithms. These are basic set of matrix/tensor computations that every CSE students must know. Later we will learn how to implement these on parallel architectures.

1. Matrix-Vector, Matrix-Matrix Multiply, BLAS-2 and BLAS-3, FLOPS
2. Discrete fourier transform; Fast Fourier transform
3. LU decomposition; LU with pivoting
4. Rotation and Reflection Matrices; QR factorization
5. Least squares problem; application to machine learning

6. Non negative matrix factorization
7. Matrix Completition and recommender systems
8. Algorithms for computing Eigenvalues and Eigenvectors
9. Spectral Decomposition
10. Singular value decomposition and PCA
11. Matrix Diagonalization
12. Handling Sparsity in matrices
13. Concept of gradients, Hessian
14. Conjugate gradient methods: CG
15. Preconditioning Techniques
16. Tensor decompositions: HOOI algorithm
17. Tensor decomposition: CP Decomposition
18. Basic Optimization Methods. Primal and dual problems.
19. Variants of Stochastic Gradient Methods
20. Newton-CG; Exploiting Sparsity.
21. Monte-Carlo Algorithms
22. Kernel Methods: SVM
23. Neural Network Based Classification
24. Basic Reinforcement Learning Algorithms
25. Basic Philosophy of Quantum Algorithms
26. Quantum Parallelism Vs Classical Parallel
27. Review of Linear Algebra for Quantum Algorithm
28. Review of Complex numbers and Probability
29. Quantum GATES
30. Grover's quantum algorithm for search
31. Quantum Fourier Transform
32. Quantum algorithm for to "solve" Ax=b (sketch only)

C, C++ Discussions (if needed)

1. Review of C/C++
2. New features of C++-11: auto, move semantics, lambda functions, etc

MATLAB Programming (if needed)

Basic Introduction to using MATLAB: colon notation, plotting, for loops, if/else, etc.

High Performance Single, Multi-Core, and Many-Core Programming

Modern day computers including the laptops we have, and even the smart phones we have are multi-core processors. To be able to write programs on such systems, we need to learn multi-threading concepts. In this topic, we will learn basic ideas and theory of shared memory programming. In tutorial, (see below) various tools, and syntax needed to do multi-threading will be introduced with demo codes. But before we launch into multi-threading, we first learn how to write efficient codes for single core. Many of the supercomputes these days are hybrid machines, i.e., they are multi CPU and multi GPU systems. To extract the most out of such machines, it is good to learn GPU.

1. Basic Ideas of Modern Day Architectures: RAM, Cache, Registers
2. Cache Optimization
3. Vectorization
4. Roof-line model
5. Stream benchmarks
6. Spawn-Sync models
7. Parallel for loops
8. Race conditions
9. NUMA issues (multi-socket)
10. Why CUDA? Why Now?
11. Development environment
12. Intro to CUDA: Hello, World
13. Parallel programming in CUDA: AXPY, Dot, Matrix multiply
14. Thread cooperation
15. Constant memory and events
16. Texture Memory
17. Atomics
18. Streams
19. CUDA on multiple GPUs
20. Some example programmes in CUDA

Multithreading in C/C++

We will learn to program multicore systems.

1. Creating threads
2. Parallel for loop implementation
3. Spawn and Sync using pthreads
4. Using atomics and locks
5. Parallel matrix multiplication using pthreads
6. Parallel FFT using pthreads
7. Tools for scalability/profiling studies: Likwid, Vtune

Multithreading Using OpenMP

Another open source tool for multi threading is OpenMP. It provides a higher level framework for multithreading compared to pthrerads. It is also very popular.

1. Parallel for loop
2. Spawn-sync
3. Using locks, atomics, and critical sections
4. Parallel matrix multiplication in OpenMP
5. Parallel QR decomposition in OpenMP

Supercomputing: Introduction to Multi-CPU/GPU Programming

Why Supercomputers Exist? Why so much buzz about them? How are these machines programmed? In this topic, we will learn how to program the “beast”: supercomputers! Students will get access to 10 x 12 = 120 cores HPC machines to run and test the codes.

1. Introduction to supercomputers
2. Architecture of recent supercomputers: Multi CPU + Many core architectures
3. Top 500 list
4. Distributed memory models
5. MPI: point to point communication
6. User defined data types and Packaging
7. Collective Communications
8. Communicators
9. One-Sided Communications
10. Introduction to performance analysis tools: Scalasca and Tau
11. Using MPI for multi CPU and multi GPU systems
12. Demo of conjugate gradient method on multi CPU and multi GPU
13. Demo of solving heat equation on multi CPU and multi GPU systems
14. Tensor compression on multi CPU and multi GPU systems

Case Studies: Parallel Algorithms in Scientific Computing

Some of these may be done as project.

1. Parallel Matrix Factorizations: LU or QR
2. Parallel Least Squares
3. Parallel Numerical Integration
4. Parallel SVD
5. Parallel Monte Carlo methods
6. Parallel Tensor compression: HOOI
7. Parallel Stochastic Gradient Descent
8. Parallel Newton CG
9. Parallel Conjugate Gradient Algorithms
10. Parallel Numerical Simulation of Heat Equation
11. Parallel Numerical Simulation of Wave Equation

Case Studies: Parallel Algorithms in Machine Learning, Data Mining, Image Processing, etc

In this topic, we discuss several applications of matrix computations for real world applications.

Most of these topics are usually implemented as a project.

1. Parallel Classification of Hand Written Digits using Neural Networks
2. Parallel Text Mining
3. Parallel Page Ranking
4. Parallel Face Recognition Using Tensor SVD

5. Parallel Neural Networks for Hand-written Recognition
6. Parallel Logistic Regression with Newton CG
7. Parallel Image Inpainting
8. Parallel CNN
9. Parallel Reinforcement Learning Algorithms
10. Parallel Clustering