EE 361C/382C **Vijay Garg** **Spring 2020**

Term Paper Topics

EE 361C

As part of EE 361C you are required to do a term project, submit the source code to the instructor and give a demo to the class on April 28. The term project should be done in groups of four. Every team will get 3 minutes to give a demo. A longer demo may be uploaded to youtube (optionally). The source code of the project will be due on **April 28**.

EE 382C

If you are enrolled in EE 382C, then you are required to submit a term paper and make a class presentation in addition to the requirements of EE 361C. The term project should be done in groups of two. The term paper (and the source code of the project) will be due on **April 28**. It should be approximately 6 pages (excluding references), typed and double spaced. The paper should have an abstract, introduction, description of the project, various design alternatives considered during implementation, performance results and conclusions. Every team is required to make a presentation for 8 minutes. Each team member must present and will get about 4 minutes for presentation. The class presentations will begin on April 28.
It is your responsibility to form teams. It is expected that all team members will make an equal effort in completion of the project. The entire team will get a single grade on the term paper. You are encouraged to use TACC for your term project.

A good reference for implementation of parallel algorithms is

Please sign-up by **March 5**. At most two teams can sign-up for any project.

1. *Concurrent Tree based Algorithms*
   Implement fine grained lock-based and lock-free binary search trees, and red-black trees. Compare the performance of lock-based and lock-free algorithms.

   Team 1 members:

   Team 2 members:Yizhuo Du, Yu Liu

2. *Parallel Priority Queue Algorithms*
   Implement fine grained lock-based and lock-free priority queue algorithms. Compare the performance of lock-based and lock-free algorithms.

   Team 1 members: Zachary Chilton, Ankur Kaushik, Juan Paez, Kieran Hannon

   Team 2 members: Benson Huang, Steven Wang, Kyle Zhou, Wesley Klock

3. *Concurrent Skiplist Algorithms*
   Implement fine grained lock-based and lock-free skiplist algorithms. Compare the performance of lock-based and lock-free algorithms.

   Team 1 members: Clara Johnson, Sammy Chien, David Chao, Huy Le
   Team 2 Members: Jost Luebbe, Jerad Robles, Anthony, Gagan

4. *Concurrent Hash Tables*
   Implement concurrent hashing algorithms. You should try multiple ways of hashing such as hashing with chains, Cuckoo Hashing, and Hopscotch Hashing. Compare the performance of various lock-based and lock-free algorithms.

   Team 1 members: Jianwen Dong, Yuesen Lu

   Team 2 members: Xiyu Wang, Jun Wang

5. *Parallel CPU based Shortest Path in Graph Algorithms*
   Implement parallel CPU based algorithms to compute shortest paths (single-source, or all pairs).

   Team 1: Abhiroop Kodandapursanjeeva, Muhammed Mohaimin Sadiq
   Team 2: Angel Cheng, Sean Wang, Jaino Vennatt, Vishruthi Ramaswamy

6. *Parallel GPU based Shortest Path in Graph Algorithms*
   Implement parallel GPU based algorithms to compute shortest paths (single source, or all pairs).

   Team 1: Daniel, Kayvan, Allen, David
   Team 2: Grace Lee, Hsiao-Yuan Chen

7. *Parallel CPU based Spanning Tree Algorithms*
   Implement parallel CPU based algorithms to compute minimum spanning trees and connected components

   Team 1: Aman Hemani, Geoffrey Tian, David Wolf, Tony Li
   Team 2 : Ryan Kim, David Zehden, Casey Hu, Tarek Allam

8. *Parallel GPU based Spanning Tree Algorithms*
   Implement parallel GPU based algorithms to compute minimum spanning trees and connected components
   Team 1:
   Team 2:

9. *Parallel CPU based Max-Flow Algorithms*
   Implement parallel CPU based algorithms to compute max-flow

   Team 1: Joseph Dieciedue, Shashwat Pandey, Andrew Pardubsky
   Team 2 : Neha Agarwal, Cameron Chalk

10. *Parallel GPU based Max-Flow Algorithms*
    Implement parallel GPU based algorithms to compute minimum spanning trees, or connected components (or max-flow algorithms).
    Team 1:
    Team 2:

11. *Parallel CPU based Sorting Algorithms*
    Implement parallel CPU based algorithms to sort large arrays. Implement mergesort, quicksort, radix sort, brick sort, and bitonic sort. Compare their performance.

    Team 1: Suhas Raja, Matt Golla, Matthew Jones, Alan Penichet
    Team 2: Donovan McCray, Lauren Jones, Shriprama Rao, Luke Norrell

12. *Parallel GPU based Sorting Algorithms*
    Implement parallel GPU based algorithms to sort large arrays. Implement mergesort, quicksort, radix sort, brick sort, and bitonic sort. Compare their performance.

Team 1: Will Easterby, John Koelling, Jack Burrus, James Sullivan
Team 2: Musa Rafik, Aly Ashraf, Zach Sisti

13. *Parallel CPU based Algorithms for Matrix Computations*
    Implement parallel CPU based algorithms to solve linear systems, invert matrices, compute determinants, and LU factorizations.

    Team 1: Qihua Yang, Yang Hu

14. *Parallel GPU based Algorithms for Matrix Computations*
    Implement parallel GPU based algorithms to solve linear systems, invert matrices, compute determinants, and LU factorizations.

    Team 1: Yating Wu, Jiayi Yang
    Team 2: Sneha Pendharkar, Siddhartha Shetkar, Sami Khandker, Kavya Duvedi

15. *Parallel GPU based Algorithms for Image Processing*
    Implement parallel GPU based algorithms to process images such as blurring them or applying various filters.

    Team 1: Thomas Plantin and Nader Al Awar
    Team 2: Shih-Lung Hsu, Pengfei Ran.

16. *Parallel Algorithms for Scientific Computations*
    Implement parallel algorithms related to scientific computation. Some possible candidates are: fast fourier transform, polynomial multiplication and division, primality testing etc.

    Team 1:

17. *Parallel Algorithms for Data Mining*
    Implement parallel algorithms related to data mining such as for clustering, classification or regression.

    Team 1:Utsha Khondkar, Ammar Sheikh, Erick Machado, Douglas Riggs

Team 2: Brandon Pham, Neel Drain, Matthew Machado, Jonathan Mounsif

18. *Parallel Algorithms for Text Analysis*
    Implement parallel algorithms related to text analysis such as searching for substrings or patterns.

    Team 1: Xin Geng, Zitian Xie
    Team 2: Edie Zhou, Jacob Grimm, Rish Bhatnagar, Josh Kall

19. *Parallel Algorithms for Computational Geometry*
    Implement parallel algorithms related computational geometry such as computing convex hulls, computing intersection of convex sets, computing visibility of points etc.

    Team 1: (Convex Hulls) Connor Fritz, Qinzhe Wu
20. *Parallel Poset Algorithms*
    Implement parallel offline and online algorithms to compute width of a poset, irreducibles of the poset, and to check if the given poset is a lattice.

21. *Parallel Algorithms for the Assignment problem*
    Implement parallel algorithms to maximum weighted bipartite matching
    Team1:

22. *Parallel Algorithms for the Satisfiability problem*
    Implement parallel algorithms to solve the satisfiability problem

    Other Problems:

    Betweenness Centrality
    Team 1: Melanie Feng, Joseph Dean, Dylan McCoy, Simon Hoque

    Barnes-Hut Simulation

Gomory-Hu Tree Construction

PageRank Algorithm
Team1: Jinxin Ma, Weiming Yang

Maximal Independent Set in a Graph

K-way partitioning of a graph

Housing Allocation Problem

Bipartite Matching Problem