CS 421 Project Proposal

Team Members :

Harsha Vardhan	-	150050073
Mani Shankar	-	150050081
Joshi	-	150094

Aim : Count number of triangles in an undirected graph

(More generally number of k-cliques)

Project Description :

Number of triangles in a graph is a useful metric in network analysis. It is particularly useful in analysis of social networks. As the size of social networks is increasing day by day, such metrics of the graphs need to be computed quickly and efficiently to analyse them properly.

We want to implement this in serial, MPI and CUDA versions and then check the achieved speedups over fairly large data sets. (Tentatively from Stanford Network Analysis Project)

Initially we wish to go over the doctoral thesis of Thomas Schank and know about some basic and advanced (sequential) algorithms like node-iterator, edge-iterator, fast matrix multiplication in dense graphs, compact-forward and other square root decomposition methods used in practice. We'll implement the sequential algorithms first as a benchmark (and also use openmp to measure speedup) . Then we'll think of parallelising the above algorithms (for instance, each edge, node can be checked independently for triangles in edge-iterator, node-iterator algorithms respectively) and writing CUDA versions on our own. (This work was formerly done for CMU parallelism competition by Shu-Hao and Yicheng Qin, their report and references are available online but no references to code are present.) We'll read their work and implement sequential (also with openmp) and CUDA versions on our own.

We will try to optimise the CUDA version as much as we can (or at least up to the ideas described in the above link, for instance Workload Balance vs Cache Utilization)

There is also a MPI-version of the project. We initially wish to understand how the mpi version is written (assigning each connected component to a processor is too naive. Divide the graph into subgraphs, count the number of triangles in each subgraph and avoid over and under counting). We wish to understand the idea and use this implementation in our comparisons. If we have any ideas of improving or rewriting the MPI version we would do it, if time permits.

In case of excess time, we will try to see if any implemented algorithms can be extended for finding k-cliques in arbitrary undirected graphs or find (if possible implement) any known efficient algorithms in literature.

Work Division :

Initially all of us will read the required algorithms from literature. After that, each of us will implement a (or more) sequential algorithm(s). Later, we will discuss upon which of them is parallelizable to a greater extent and work on it. The work division of the CUDA version *per se* is not decided now, but we will submit it in the final project report. We'll also see through what justice can be done to the MPI version!