--- Title ---

Counting k-cliques in parallel: David Wise, Jason Li

--- Summary ---

We are going to count the number of k-cliques in large graphs on multi-core platforms such as the GHC machines.

--- Background ---

There was a triangle-counting project which won the 2013 parallelism competition; we will attempt to further improve the parallelism, and also generalize the triangle/3-clique counting to 4-cliques and maybe 5-cliques. Clique finding is important in determining how clustered a graph is, because the number of edges alone is often not sufficient. It has applications to social networking problems, since determining how clustered a social network graph is is important to social network companies.

--- Challenge ---

It is not obvious how to count even the number of 3-cliques in less than O(n^2) time. In fact, the best known result for counting k-cliques runs in time O(m^{k/2}) (m = number of edges), which is very large for million-sized graphs. In addition, the same algorithm takes O(m^{3/2}) space, so even the space constraint needs to be optimized.

We hope to exploit the low dependency on distinct parts of the graph. If we split vertices into groups, then we even have the advantage of locality. However, we still need to handle the cliques with vertices in different groups, and avoiding overcounting will be a challenge. At the end, we hope to experiment with many different algorithms and see which exhibit the best parallelism.

--- Goals and Deliverances ---

We plan to implement a k-clique counting algorithm that is highly parallel in at least the k=3 (triangle) case. We are aiming for at least 10x speedup, which is what the previous team was able to achieve. We hope to generalize this code for higher k and observe similar speedups.

We hope to obtain even faster algorithms than the previous team, and hope that we can solve within a reasonable time the k=4 or k=5 cases, even on million-sized graphs. If time permits, we may also investigate more efficient approximation algorithms to the counting k-cliques problem.

At the demo, we will show our algorithm in action on a small graph, as well as the graphs of our speedup factors for various test cases. If the speedup is rather dependent on the graph size, then we will attempt to explain why.

--- Platform ---

We will be programming in C++, since it is a fast language, and since most of the tools we have learned in class are used in C++. We will be testing mostly on the GHC machines, since computational services are readily available. To obtain results, we plan to use the latedays cluster, which has more cores and therefore more potential for parallelism.

--- Schedule ---

Week 1: Find test cases online, and code up a brute-force to determine the correct number of k-cliques. Read papers on computing k-cliques, and implement some of them sequentially.
Week 2: Research on ways to parallelize C++ code, and try to implement one or two of them. Think of alternative algorithms that work efficiently in parallel.
Week 3: Implement the algorithms, and analyze their running times for various test cases on the GHC computers. Make further optimizations to the algorithms that seem promising.
Week 4: Run the algorithms on latedays, and tweak the algorithms to be more optimized for machines with more cores.
Week 5: Finish up testing and obtain the final results. Analyze why the graphs are the way they are, and write the final report.