CS 6230 Project Report

Parallel Computation of Centrality

Rui Dai, Sam Olds

May 3, 2017

1 Introduction

In graph theory, centrality indicates the importance of a vertex in the network. This concept is naturally applied on social network analysis. Imagine you are producing a new product and want to find beta users. It's simple to let users with high centrality to use and spread the news to their reachable networks.

There are many different definitions of centrality, eg. degree centrality, closeness centrality and betweenness centrality. In our project, we will use the vertex betweenness centrality, which is defined as

$$g(v) = \sum_{s \neq v \neq t} \frac{\delta_{st}(v)}{\delta_{st}}$$

Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. It's not hard to imagine that the computation of centrality is very expensive with all the operations with shortest paths. Our goal of this project is parallel such computation and leverage the technology of MPI/OPENMP we learned in class.

We will explain two algorithms we use for parallel centrality computation, with experiments, result, and analysis.

2 Related Work

Centrality was first introduced as a measure for quantifying the control of a human on the communication between other humans in a social network by Linton Freeman[2] In his conception. Ever since, the concept has drawn many interests in cross-indiscipline area like network analysis and social science.

Betweenness centralities in a graph involve calculating the shortest paths between all pairs of vertices on a graph, which requires $\Theta(V^3)$ time with the FloydWarshall[3] algorithm. On sparse graphs, Johnson's[4] algorithm may be more efficient, taking $O(V^2 \log V + VE)$ time. In the case of unweighted graphs the calculations can be done with Brandes' algorithm[1] which takes $\Theta(VE)$ time. Normally, these algorithms assume that graphs are undirected and connected with the allowance of loops and multiple edges. When specifically dealing with network graphs, often graphs are without loops or multiple edges to maintain simple relationships (where edges represent connections between two people or vertices). In this case, using Brandes' algorithm will divide final centrality scores by 2 to account for each shortest path being counted twice.

Breadth first search is also a great part in centrality computation as for the shortest path part. Parallel BFS algorithms have

3 Graph Representation and Generation

3.1 Representation

In our work, we represent graph in sparse adjacency matrix.

3.2 Generation

4 Methods

4.1 Simple Solution

4.2 Method Based on Brandes'

Sam

5 Results

Sam

6 Conclusion

References

[1] Brandes, U. A faster algorithm for betweenness centrality. *Journal of mathematical sociology 25*, 2 (2001), 163–177.

[2] Burt, R. S. *Structural holes: The social structure of competition.* Harvard university press, 2009.

[3] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[4] Johnson, D. B. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM) 24*, 1 (1977), 1–13.

7 Appendix