Parallel Distributed-Memory Based Community Detection for Large Graphs

Diana Palsetia, William Hendrix, Ankit Agrawal, Wei-keng Liao, Alok Choudhary
Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60208
{*drp925,whendrix,ankitag,wkliao,choudhar*}*@eecs.northestern.edu*

Abstract

Community detection is a well-studied problem in graph data analytics. As graph sizes have increased, more attention is turning to parallel techniques. In general, graph algorithms may be parallelized by dividing the data or by dividing the workload. One of the main challenges in designing a parallel algorithm is partitioning the data. Since the data access pattern in graph algorithms is often irregular and highly dependent on the network structure, a poor partitioning scheme can cause high communication cost and severely affect the quality of clustering. Here we describe our ongoing work on developing a distributed-memory based parallel algorithm for community detection. The algorithm adopts a data-based decomposition strategy with duplication, which is expected to achieve good scalability without sacrificing cluster quality.

1. Introduction

Data clustering organizes a collection of data points into groups based on their similarity. Clustering graphs usually refers to the identification of vertex subsets (clusters) that have significantly more internal edges than external ones. For example, in social networks, cluster analysis helps to identify the formation of trends or user communities (relevant to marketing studies) and social group behavior. In the past few years, the volume of data has surpasses the capabilities of traditional graph-based techniques. For instance, a popular community detection algorithm based on maximum modularity called CNM (Clauset, Newman, Moore) [1] takes approximately 18 hours to process a social network data set containing 2,238,731 users and 14,608,137 connections. Such poor performance has encouraged researchers to develop heuristic and parallel clustering approaches for tackling large data sets.

Graph problems are data-driven, i.e., the memory access pattern in graph algorithms is often irregular and highly dependent on the network structure itself. Unlike spatial-based data clustering algorithms where the affinity of two data points can be determined by their distance, most graph algorithms must traverse the graph via edges to calculate the affinity of a vertex to another. Thus, scalable graph algorithms based on partitioning of computation can be difficult to develop, as the structure of computations in the algorithm is not known a priori. While a few parallel implementations of graph clustering algorithms exist, they often suffer from frequent synchronization [2], and their quality is affected by the processing order of assigning vertices to the communities.

To tackle the irregularity of data access nature exhibited in graph algorithms, most parallel implementations are designed for shared-memory architectures, such as multi-core CPUs or GPUs with multi-threading capabilities. For instance, the fastest parallel community

detection algorithm published in the literature achieved a speedup of 13.7 using 80 compute cores on a shared-memory machine [3]. Due to irregular data patterns, data partitioning is one of the main challenges in parallelizing graph problems in a distributed environment and a key factor for determining their performance and scalability. This is also evident from recent work proposed using a distributed-memory based algorithm using Louvain method [4]. This method is said to have achieved a 5x performance speedup on 128 processors.

2. Proposed Approach

Our proposed distributed-memory parallel algorithm for detecting communities in large-scale graphs is named PMEP, built around the idea of maximizing equilibrium and purity (MEP) of communities. MEP [5] has been demonstrated to produce high quality of results for medium to large graphs. This motivates us to adopt its community detection properties for determining local communities in our parallel algorithm design. Parallelizing MEP by dividing its computation would limit parallelization based on the structure of the graph, and it would suffer from frequent synchronization. Therefore, we choose to adopt a data-based decomposition approach which divides a graph into sub-networks, detects communities within each subnetwork independently using MEP, and combines the results.

Our parallel algorithm design focuses on an efficient data partitioning strategy. We use an existing graph partitioning algorithm to partition the graph. However, the novel contribution in our work is the duplication of partition data, so as to improve cluster quality and scalability. Merging the local communities would obviously affect the final clustering result quality. In our parallel design, a vertex may belong to more than one process, thereby resulting in repetitive computation. However, this strategy allows us to run MEP algorithm on subgraphs independently and concurrently. It also avoids the frequent inter-process communication that would appear in the direct parallelization approach. Once the local communities have been identified, PMEP runs a global resolution phase in parallel to combine the local communities into a final global solution.

3. Preliminary Results & Ongoing Work

We have implemented our proposed algorithm using MPI for Python [6]. Preliminary experimentation on large synthetic graphs with millions of vertices and edges on 3160 processes indicate that our parallelization approach is able to give us high quality clusters in a scalable fashion. Moreover, this approach is still effective when we increase the number of partitions. Our original design assumed the partitioned data with duplication as input. We are currently working on developing a complete end-to-end solution, which can read just the original graph, and perform the data duplication and partitioning in parallel.

References

[1] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. Phys. Rev. E, 70(6):066111, 2004.

[2] S. Bansal, S. Bhowmick, and P. Paymal. Fast community detection for dynamic complex networks. In L. F. Costa, A. Evsuko_, G. Mangioni, and R. Menezes, editors, Complex Networks, volume 116 of Communications in Computer and Information Science, pages 196-207. Springer Berlin Heidelberg, 2011.

[3] E. J. Riedy, H. Meyerhenke, D. Ediger, and D. A. Bader. Parallel community detection for massive graphs. In Graph Partitioning and Graph Clustering, pages 207-222, 2012.

[4] C. Wickramaarachchi, M. Frincu, P. Small, and V. Prasanna. Fast parallel algorithm for unfolding of communities in large graphs. In High Performance Extreme Computing Conference (HPEC), 2014 IEEE, pages 1- 6, Sept 2014.

[5]H. Zardi and L. B. Romdhane. An $O(n^2)$ algorithm for detecting communities of unbalanced sizes in large scale social networks. Know.-Based Syst., 37:19-36, Jan. 2013

[6] L. Dalcin, R. Paz, and M. Storti. Mpi for python. Journal of Parallel and Distributed Computing, 65(9):1108 - 1115, 2005.