Filtering: A methods for Solving Graph Problems in MapReduce – First Draft

He Xinyu

May 8, 2024

Abstract

The graph data from both industry and academia can be too large to compute in one machine's RAM. To be able to process this data, the MapReduce framework was created to split the graph data into multiple small data and then process them in parallel. In the original paper, the author presents a general algorithmic design technique in the MapReduce framework called *filtering*. Filtering reduces the size of the input into distributed forms and uses distributed machines to solve the targeted problems. The first draft of this paper only showcases the understanding of the problem and the research plan. The full understanding of the given paper is not accomplished in this phase, and errors are likely to be present in this draft. This first draft is completed based on my understanding of the requirements listed on Moodle.

1 Introduction

As the development of tech giants in the world, more and more data in gathered though our daily driver apps. How to process this huge amount of data is becoming a real concern. Although the average RAM sizes in servers have improved substantially, from GB to TB, it remains woefully inadequate to process large scale data. Under such background, the MapReduce framework was developed to processing massive data. In MapReduce framework, data is stored as tuple <key, value> pairs and computation is proceeded in recursive and distributed ways. Mapper decides how data is distributed among machines and Reducer performs non-trivial computation locally on those machines. Mapper and Reducer work only on subgraphs, which means no machine can see the full input graph.

In the original paper, the author focus on class MRC^0 and assume to work with dense graph $m = n^{1+c}, c > 0$[2]. The authors propose *filtering*, which means filtering part of the input graph on the first stage in parallel and add some patchwork to ensure a proper solution. The author then implement the method on these problems: minimum spanning trees; unweighted maximal matchings; maximum weighted matching; minimum edge cover and minimum cut. The empirical results shows superiority compares to streaming methods in terms of running time.

2 Overview

To better understanding the problem discussed in this paper, we give those notations. let $G = (V, E)$ be the input graph and N be the input size; n, m indicate the number of nodes and edges; μ is the memory available on each machine.

Firstly, lets recap on the model: MRC^i. In this model, N is the input size and $\epsilon > 0$ is some fixed constant. Now there is less than $N^{1-\epsilon}$ machines, and each machine has less than $N^{1-\epsilon}$ memory. As a result, the total amount of memory in the complete system is $N^{2-2\epsilon}$. An algorithm in MRC^i runs in the worst case $O(log^i N)$ rounds[1]. Due to the massive data size need to be transfer between machines, the cost for a round in MapReduce framework is very high. This is usually the dominant cost in a MapReduce framework.

In the original paper, the authors focus on class MRC^0, and work with dense graph $m = n^{1+c}, c > 0$. The author finds constant times of rounds algorithms for

- Maximal matching
- Minimum cut
- 8-approximation for the maximum weighted matching
- 2-approx for vertex cover
- 3/2-approx for edge cover

The method proposed in this paper selectively filter or drop parts of the input and reduce the problem size until it fit on a single machine's memory. The machine computed the reduced input and produce final result. Of course, the filtering process is differ among problems in different forms. And the main challenge is to filter enough data to process on a single machine as well as preserve the precisions. As listed above, the author showcases that many fundamental graph algorithms can be computed using filtering method and finish computation in a constant times of rounds.

3 Outline of the future work

This first draft of the essay present my understanding of the MapReduce framework, the targeted problems and the method the author took to solve them. Due to the fact that each sub-problems' filtering methods is different among each other, the fully understanding of the filtering methods is still absent in this phase.

It is very difficult to understand the paper without prior knowledge of each sub-problems and background. My first focus on reading related papers and gain insight into those sub-problems and background papers. This part of work will become the background section of the final essay. The first phase is planing to finish before 25. May. The concerning topics are listed as follows.

- minimum Spanning tree
- unweighted maximal matchings
- maximum weighted matching
- minimum edge cover
- minimum cut
- class MRC

Secondly, I shall look into the specific techniques the author used to solve those problems. Because the filtering algorithm differ among problems, this step is only possible after gaining fully understanding of the specific problem. This part will be the relevant part of the final essay. The second part, together with second-phase essay writing, is planing to finish before 14. June. The second draft should contain full essay structure. After finish the second draft of the paper, the paper shall be reviewed completely to prepare for the third draft.

References

[1] Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 938–948. SIAM, 2010.

[2] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 85–94, 2011.