10/22/2023

Team: Prasanna Akolkar, Neel Gupta

https://github.com/gunrock/gunrock/blob/main/benchmarks/tc_bench.cu

https://github.com/gunrock/gunrock/tree/main/examples/algorithms/tc

https://github.com/wzbxpy/TRUST

EE 451

Accelerating Approximate Triangle Counting in Sparse Graphs

Introduction

Triangle counting in graphs is a classic problem in the field of graph algorithms and network analysis. Triangles, which are subgraphs with three nodes and three edges, and their counts reveal network denseness, identify tight-knit communities, and inform metrics like clustering coefficients that characterize the interconnectedness and structure of the graph. In situations where input graphs exceed single-node storage constraints, obtaining an exact count of triangles can be time-consuming. Thus, approximate methods provide a close estimate in much faster time known as Approximate Triangle Counting (ATC). ATC involves sampling a subset of the graph, counting triangles within this subset, and then scaling or extrapolating the results to estimate the number of triangles in the entire graph.

Algorithm 7. Approximate-Triangle-Counting(G=(V,E))

$R \leftarrow 0$
parfor $i \leftarrow 1 \ldots \mathcal{M}$ **do**
  Let V_i be a random subset of V ▷ See Section 6.2 for details about the sampling
  if size of $G[V_i]$ exceeds machine space S **then**
    Ignore this sample and set $\hat{T}_i \leftarrow 0$
  else
    Let $\hat{T}_i$ be the number of triangles in $G[V_i]$
    $R \leftarrow R + 1$
end parfor
Let $\hat{T} = \sum_{i=1}^{\mathcal{M}} \hat{T}_i$
return $\frac{1}{p^3 R}\hat{T}$

However, despite the algorithm's theoretical performance guarantees of terminating after a constant number of rounds using $O(n)$ space overall and $O(m)$ space per machine where n and m are the number of vertices and edges in the graph, ATC's real world parallel implementation on the CUDA platform has not been explored on sparse graphs, leaving a hole in theoretical guarantees that have been established for large graphs but not for graphs which can effectively fit on single machines.

Description of Implementation

The implementation will be on a heterogeneous computing architecture consisting of a host CPU, AMD Ryzen 7 3700X, and a device GPU, NVIDIA RTX 2070 Super with 8 GB VRAM under the assumption that the entire input graph can fit in this space but many cores can work on this data in different rounds of execution of the parallel for loop in line 2 of the algorithm.

- First, we will implement the algorithm on dense graphs to get baseline performance and confirm real world constant performance as theorized.
- Second, we will implement the algorithm on sparse graphs in order to check whether theoretical constant performance guarantees remain.

We hope to further optimize the performance of ATC on sparse graphs via approximating certain data dependencies in the sampling part of the algorithm for more parallel activity.

Evaluation

For the implementation of the ATC algorithm, we will need datasets of dense and sparse graphs. The dataset that would be used for the analysis of the algorithm will be retrieved from the Stanford Large Network Dataset Collection (SNAP), one of the most popular resources for large datasets. A total of 2 datasets will be selected from the Social Network collection - one dense graph and another sparse.

We would be interested in plotting comparisons for $\log_n(T)$ versus the $\log_n(\text{time})$ (where T is the number of triangles and n is the number of vertices) for the ATC Algorithm on both the dense and sparse graphs. Our primary criteria for evaluation is to confirm that the time required to count triangles in sparse graphs is significantly less than that in dense graphs.

References

Biswas, A. S., Eden, T., Liu, Q. C., Mitrović, S., & Rubinfeld, R. (2022). Massively Parallel Algorithms for Small Subgraph Counting. arXiv preprint arXiv:2002.08299.

Tětek, J. (2021). Approximate Triangle Counting via Sampling and Fast Matrix Multiplication. arXiv preprint arXiv:2104.08501.

Leskovec, J., & Krevl, A. (2014). SNAP Datasets: Stanford Large Network Dataset Collection. Retrieved from http://snap.stanford.edu/data.

https://github.com/rapidsai/cuhornet/tree/main