Minor Project Proposal

On

GRAPH BETWEENNESS CENTRALITY FOR SPARSE GRAPHS USING CUDA

Submitted by

171IT208 ANIKETH ANGANWADI
171IT212 AYUSH GUPTA
171IT236 SAI SIDDARTH YV

Under the Guidance of

Dr NEELIMA BAYUPPU
Assistant Professor

Dept. of Information Technology,
NITK, Surathkal

Date of Submission: 27 September 2019

Department of Information Technology
National Institute of Technology Karnataka,
Surathkal.

Certificate

This is to certify that the Proposal entitled **Graph Betweenness Centrality for Sparse Graphs using CUDA** has been presented by Aniketh Anganwadi, Ayush Gupta and Sai Siddarth YV, students of V semester B.Tech. (I.T), Department of Information Technology, National Institute of Technology Karnataka, Surathkal, on 27 September 2019, during the odd semester of the academic year 2019 - 2020, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology.

Place: NITK, Surathkal
Date:27 September 2019

Declaration

We hereby declare that the Seminar (IT300) Proposal entitled **Graph Betweenness Centrality for Sparse Graphs using CUDA** which is being submitted to the National Institute of Technology Karnataka Surathkal, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in the department of Information Technology, is a bona fide proposal of the work carried out by us. The material contained in this project report has not been submitted to any University or Institution for the award of any degree.

Place : NITK, SURATHKAL
Date : 27 September 2019

Aniketh Anganwadi
171IT208

Ayush Gupta
171IT212

Sai Siddarth YV
171IT236

Department of Information Technology

Contents

Certificate

Declaration

1 Abstract **1**

2 Introduction **2**

3 Problem Statement And Motivation **3**
3.1 Problem Statement 3
3.2 Motivation 3

4 Literature Review **4**

5 Methodology **5**

6 Implementation Strategy **6**

7 Individual Contribution **7**

8 Lacuna in existing work **8**

9 Results and Analysis **9**

10 Timeline Plan and Challenges involved **11**
10.1 Timeline Plan: 11
10.2 Challenges: 11

11 Conclusions **12**

References **13**

1 Abstract

Betweenness centrality is a measure of the influence of a vertex in a graph. It measures the ratio of shortest paths passing through a particular vertex to the total number of shortest paths between all pairs of vertices. The naive method to calculate betweenness centrality involves a lot of computation time and memory space .In our proposed work we introduce the Brande‘s algorithm which is faster and states a mathematical model to reduce the calculations required to calculate Betweenness Centrality by introducing the concept of partial dependencies between vertices of the graph. The required computations are done using GPUs using CUDA in order to apply parallelism to solve the problem.

2 Introduction

Graph analysis is an essential tool for domains such as social networks, computational biology, and machine learning. Real-world applications of graph algorithms involve tremendously large networks that cannot be inspected manually. Betweenness Centrality (BC) of a vertex in a given graph is a useful metric that determines the influence of the vertex in the graph. Betweenness Centrality has many practical applications including finding the best locations for stores within cities, power grid contingency analysis, and community detection.

Betweenness centrality for a given vertex v in a given graph is the ratio of two quantities. The first one is the number of shortest paths between any two vertices s and t ,such that s,t,v are unique and the shortest path traversal includes v.The second quantity is the total number of existing shortest paths between any two vertices in the given graph. The first quantity can be calculated by applying the Floyd Warshall algorithm to the given graph and counting the number of shortest paths between two unique vertices that include v.The second quantity is equal to total number of existing shortest paths between all vertices, taken two at a time. The following equation illustrates the calculation for Betweenness Centrality :

$$BC(v) = \sum_{s \neq t \neq v} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

3 Problem Statement And Motivation

3.1 Problem Statement

- A recursive relation for defining the partial dependency() w.r.t root node s is:

$$\delta_s(v) = \sum_{w:v \in pred(w)} \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_s(w))$$

- This is the Brande's Partial Dependency equation
- The project aims to parallelise the Brandon's algorithm using CUDA and show the improvements with respect to a serial implementation of the same.

3.2 Motivation

- Betweenness centrality is widely used to find best locations for stores within cities, power grid contingency analysis and community detection.
- The naive implementation of solving the all-pairs shortest paths problem is of the order of O(nˆ3) by Floyd-Warshall technique and Betweenness Centrality is calculated by counting the necessary shortest paths.
- Brande's algorithm reuses the already calculated shortest distances by using partial dependencies of shortest paths between pairs of nodes for calculating BC of a given vertex w.r.t a fixed root vertex.

4 Literature Review

In [1], This post describes how we used CUDA and NVIDIA GPUs to accelerate the BC computation, and how choosing efficient parallelization strategies results in an average speedup of 2.7x, and more than 10x speedup for road networks and meshes versus a naïve edge-parallel strategy.

In [2],This paper tells us that prior GPU implementations had suffered from large local data structures and inefficient graph traversals that limit scalability and performance.This paper presented several hybrid GPU implementations,providing good performance on graphs of arbitrary stucture rather than just scale-free graphs as was done in previous researches.This paper achieved upto 13x speedup on high-diameter graphs and an average of 2.71x speedup overall over the best existing GPU algorithm.Here the have used the GPU-FAN Package for load balancing acroos threads by using edge-parallel method.The GPU-FAN focuses on fine grained parallelism,using all threads from all thread blocks to traverse edges in parallel for one source vertex of the betweenness centrality computation at a time.

In [3], The history of centrality is explained where it encompasses first attempts in the late 1940s at MIT (Bavelas 1946), in the framework of communication patterns and group collaboration; It also tells us about the Social Network Analysis and how it can be represented in graph representation and from the graph, we can study the properties of its structure,and the role,position, and prestige of each social entity. We can also find various kinds of sub-graphs, e.g., communities formed by groups of actors.

It gives an insight to the types of centralities in a graph notably:

- measures based on distances[Lin‘s index].
- measures based on paths[Katz‘s index].

And also gives a hierarchial clustering approach for computing degree ,closeness and between centrality.

5 Methodology

Effectively the calculation of BC values can be divided into 2 steps:

1. Calculating the partial dependencies w.r.t all nodes. This is done by fixing a root and doing a forward Breadth First Search (BFS) to calculate the depth of other nodes w.r.t the fixed root s

2. Accumulating the partial dependencies for all root nodes to calculate the BC value. can be done using a reverse Breadth First Search by level-order traversal from the deepest level towards the root to calculate partial dependencies of the shortest paths between the current node to the fixed root which passes for all the predecessor nodes, i.e. the nodes which are immediately one level above of the current node and the shortest path from root to the current elements passes through them.

This gives a running Time of O(V*E)

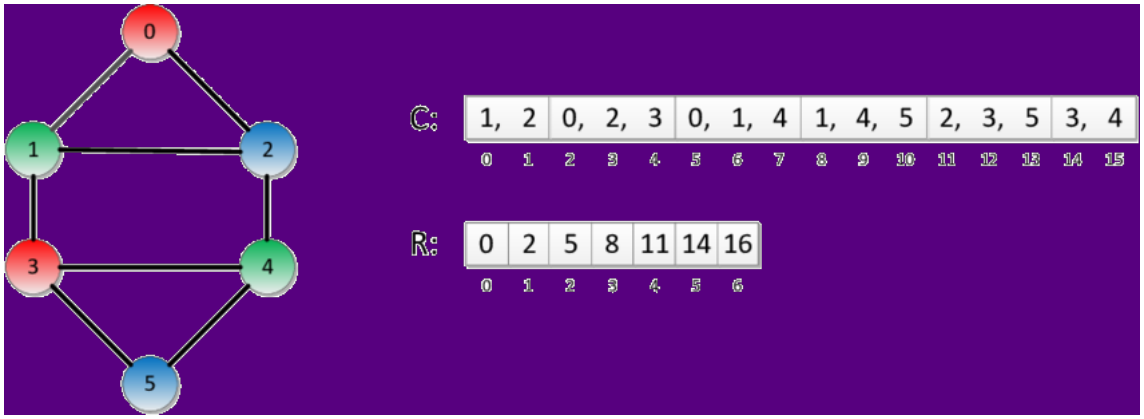
6 Implementation Strategy

Serial Implementation Of Brande's Algorithm.
Multiple blocks in a CUDA grid can be used for parallelising partial dependencies for independent roots.
This can achieve a coarse grained parallelism by processing each source independently in parallel.
Fine grained parallelism can be achieved by running a parallel BFS using optimal work distribution strategies.
A top down BFS can be used to calculate the distance and number of shortest paths to each node with respect to a given source.
Using these values, dependency values can be calculated by traversing the graph bottom up, using a multi-sourced bottom up Breadth First Search in parallel.

Compressed Sparse Row (CSR) format is a common way to store Graphs for GPU computations
It consists of 2 arrays:

1. Adjacency List array (Column Indices): The Adjacency List array is a concatenation of adjacency list of each vertex into an array of E elements.

2. Adjacency List Pointers array (Row Offset): Array of V+1 element that points at where adjacency list of each vertex begins and ends within the column indices array.



7 Individual Contribution

1. Sai Siddarth YV(171IT236) is handling the parallelizing the BFS and acheiving the Fine Grain Parallelism(Work efficient method) and documentation.

2. Aniketh Anagawadi(171IT208) is handling the running the BFS for multiple root nodes and thus acheiving coarse grain Parallelism(Work distribution method) PPT presentation and documentation.

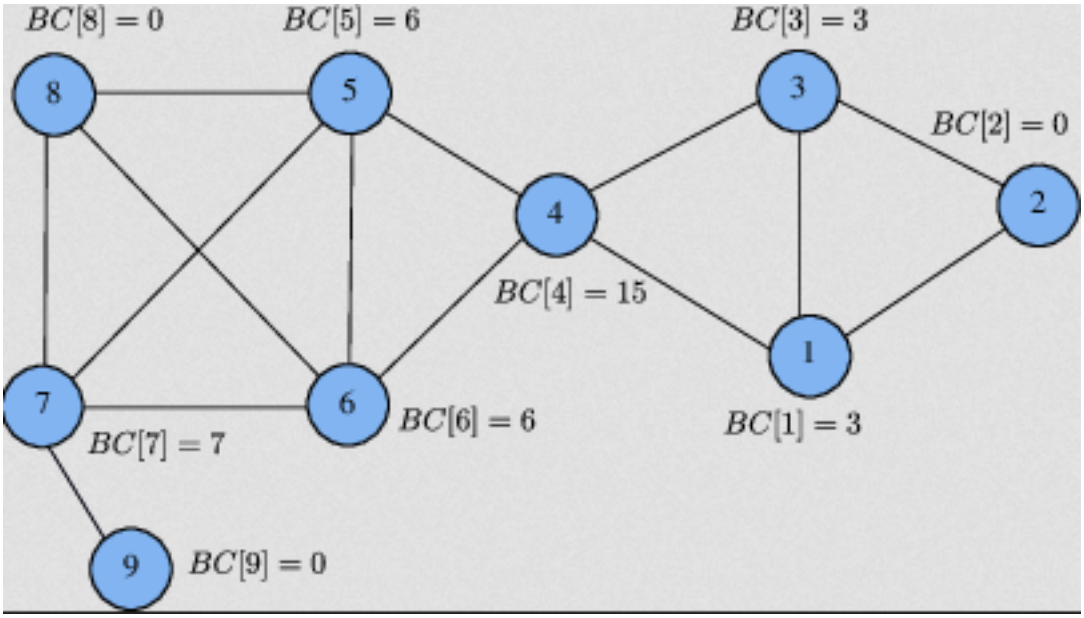
3. Ayush Gupta(171IT212) is handling the serial code implementation and the graph formatting in CSR and COO formats,parallel approach using vertex based and edge based and finding the lacuna in our current work and coming up with possibilities.

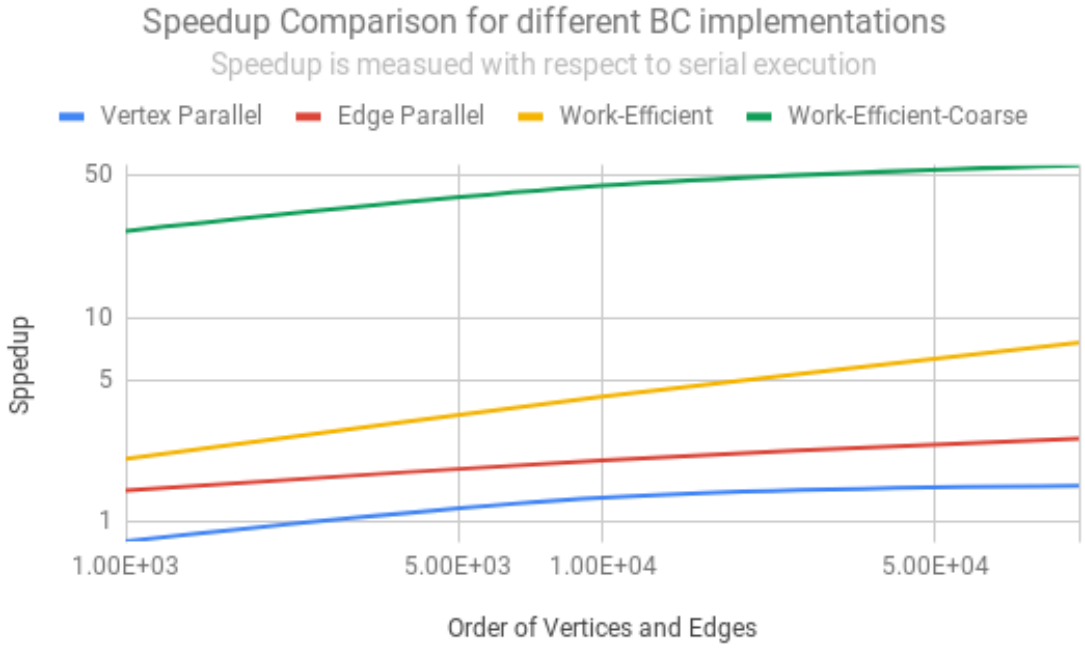
8 Lacuna in existing work

- Big graphs require new algorithms.
- Is polynomial execution time really feasible? NO as, Brandes algorithm not feasible for n = 109
- On big graphs quadratic time is as bad as NP-Hard New, finer-grain, complexity theory needed.
- Need for massively parallel algorithms,out-of-core algorithms, sublinear algorithms, approximated algorithms, randomized algorithms, etc.
- Velocity with which new data keeps coming requires streaming algorithms that only read each data point once (or a few time), specialized small-space data structures (sketches) that maintain basic statistics and can be updated on-the-fly, algorithms which are robust to changes in the data, etc.
- In order to scale graph analysis to real-world applications and to keep up with their highly dynamic nature, we need to devise new approaches specifically tailored for modern parallel stream processing engines that run on clusters of shared-nothing commodity hardware.

9 Results and Analysis

For a large number of nodes, coarse grained parallel implementation should have a higher execution time than fine grained parallel implementation.




Speedup Comparison for different BC implementations
Speedup is measued with respect to serial execution
Vertex Parallel
Edge Parallel
Work-Efficient
Work-Efficient-Coarse
50
10
5
1
Sppedup
1.00E+03
5.00E+03
1.00E+04
5.00E+04
Order of Vertices and Edges

10 Timeline Plan and Challenges involved

10.1 Timeline Plan:

Week 1: Trying to understand the papers we referred to in the literature survey.Learning the Brande‘s Algorithm.

Week 2: Learning Cuda through online sources(Coursera)and trying to find the usefulness of GPU‘s in solving our current problem.

Week 3: Starting Work on our individual contributions and meeting every saturday and addressing our gaps and solutions.

Week 4: Programs compiled with Cuda and will compare the execution times and speedup with respect to serial program and finally presentation of our project and final report.

10.2 Challenges:

Main challenges we are facing is the learning of cuda and giving the blocks the instruction to execute BFS on each of the seperate root nodes and integrating the concept of CSR formats of graphs for GPU calculations.

However, the unprecedented volume, velocity, and variety pose real algorithmic challenges, especially when dealing with expressive and complex representations such as graphs.

11 Conclusions

In this project we discussed various ways to parallelize betweenness centrality calculations on the GPU. Since the structure of real-world graphs can vary tremendously, no single decomposition of threads to tasks provides peak performance for all input. For high-diameter graphs, using asymptotically efficient algorithms is paramount to obtaining high performance, whereas for low-diameter graphs it is preferable to maximize memory throughput, even at the cost of unnecessary memory accesses.

The importance of robust, high-performance primitives cannot be overstated for the implementation of more complicated parallel algorithms. Fortunately, libraries such as Thrust and CUB provide fast and performance portable building blocks to allow CUDA developers to tackle more challenging tasks.

References

[1] https://devblogs.nvidia.com/accelerating-graph-betweenness-centrality-cuda/

[2] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=7013034

[3] http://matteo.rionda.to/centrtutorial/BonchiDeFranciscisMoralesRiondato-CentralityBigGraphsTutorial-Slides.pdf