Поиск кратчайших путей в графе и его применение

Результаты реализации

Сабрина Мусатян, 371 группа,
Мат-Мех СПБГУ

Детали реализации

- https://github.com/sabrinamusatian/ShortestPathsWithGPU
- Вместо CUDA алгоритмы реализованы на OpenCl и библиотеки pyopencl
- GPU: AMD Radeon R9 M370X
- CPU: 2,8 GHz Intel Core i7

A New GPU-based Approach to the Shortest Path Problem

Hector Ortega-Arranz, Yuri Torres, Diego R. Llanos Ferraris, Arturo Gonzalez-Escribano

“Наивное” CPU решение

- Написано самостоятельно - однопоточная Дейкстра с использованием идеи выделения Frontier set, для которого в GPU реализации вершины будут обрабатываться параллельно

GPU решение

3 openCL kernels(идея в том, что каждая вершина графа имеет свой поток):

- Initialize kernel- параллельно заполняем необходимые массивы до начала работы алгоритма
- Relax kernel - каждая вершина рассматривается параллельно, если она входит во Frontier Set релаксируем расстояния для ее соседей
- Minimum kernel - вычисляем минимальное значение для данной итерации, чтобы найти Frontier set, для каждой вершины параллельно находится это значение. Для работы с глобальным минимумом используются атомарные операции.
- Update kernel - для каждой вершины решаем, будет ли она входить в новый Frontier Set

Про Frontier set и его нахождение было рассказано в обзоре области.

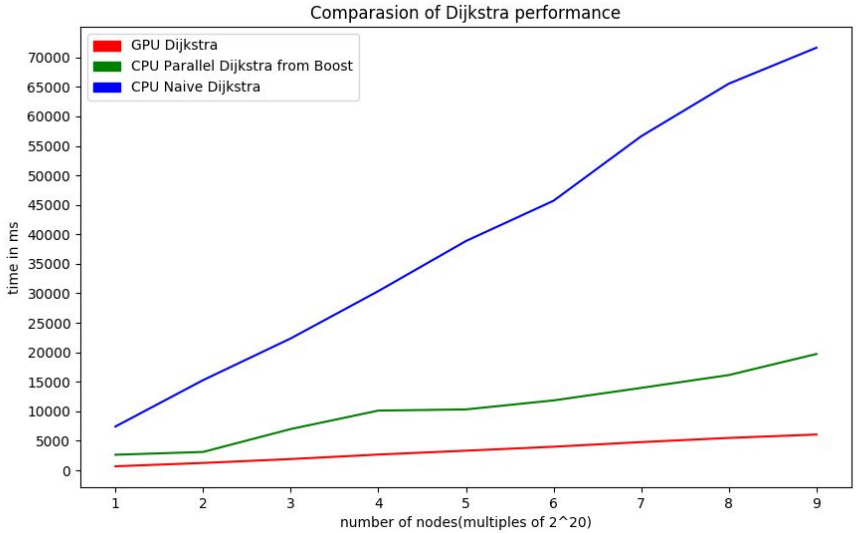
CPU Дейкстра из библиотеки Boost

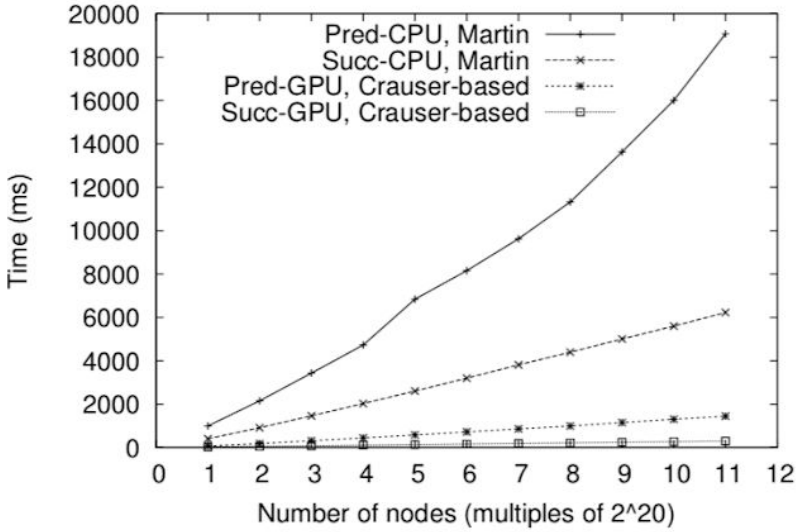
- Реализация параллельного алгоритма Дейкстры на CPU, взятая из библиотеки Boost для C++

Данные

- Как и у авторов оригинальной статьи был создан скрипт для генерации случайных графов.
- Графы являются связными
- Средняя степень вершины 7

Сравнение результатов





On Ranking Nodes using kNN Graphs, Shortest-paths and GPUs

Ahmed Shamsul Arefin, Regina Berretta, Pablo Moscato

Данные

- В оригинальной статье были использованы медицинские данные, которые не являются открытыми
- Для экспериментов был использован скрипт для генерации случайных графов как в предыдущем алгоритме
- Графы являются связными
- У каждой вершины ровно 3 соседа

“Наивная” CPU реализация

- Применяется BFS для поиска кратчайших путей от заданной вершины до всех остальных
- Такие пути вычисляются для каждой вершины графа, чтобы вычислить centrality
- Реализована самостоятельно на основе алгоритма, описанного авторами статьи без параллельности

GPU реализация

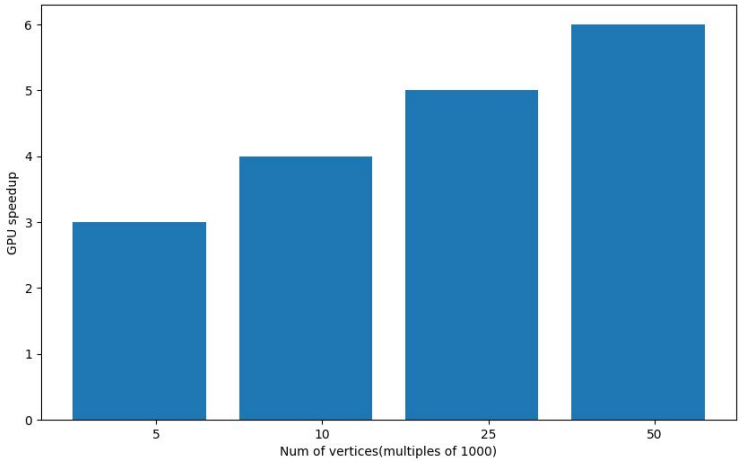
- Ребра хранятся списком смежности
- На каждой итерации ищутся вершины, находящиеся на расстоянии d от начальной
- BFS kernel: рассматриваем на каждой итерации каждое ребро параллельно, если для его исходящей вершины расстояние в графе от искомой вершины равно d, то присваиваем входящему ребру расстояние d+1(в оригинальной статье делается предположение, что веса всех ребер = 1)
- Для каждой вершины проводим операцию BFS и находим centrality

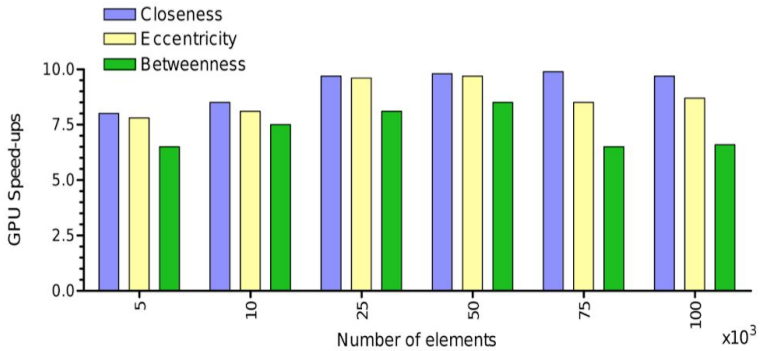
Применение Boost к проблеме Centrality

- Взята реализация параллельного алгоритма Дейкстры из библиотеки Boost
- Для каждой вершины найдены кратчайшие пути и вычислена centrality

Сравнение результатов

Выигрыш GPU реализации по сравнению с наивной CPU реализацией:





Сравнение результатов