

# Нахождение компонент связности графа с использованием GPU

Обзор существующих решений

Выполнил: Смиренко Кирилл, 371 группа

Санкт-Петербург  
2017 г.

# Постановка задачи

- Исследовать параллельные алгоритмы нахождения компонент связности графа
- Реализовать 2 алгоритма с использованием техники GPGPU

# Особенности задачи

- GPGPU хорошо подходит для алгоритмов с регулярными обращениями к памяти (regular data access)
- Данная задача предполагает нерегулярные обращения к памяти (irregular data access)
- Эффективность решения сильно зависит от представления графа и алгоритма

# Фундаментальные результаты в области (1)

- Shiloah, Vishkin. An  $O(\log n)$  parallel connectivity algorithm (1982)
- Используется модель Parallel Random Access Machine (PRAM)
- Сложность  $O(\log n)$ , используется  $n + 2m$  процессоров
- Для каждой вершины  $v$  хранится указатель  $D(v)$

# Дополнительные определения

- Звезда – дерево с одним внутренним узлом (корнем) и  $k$  листьями
- Операции на графе указателей:
  - “Short-cut”:  $D(v) \leftarrow D(D(v))$
  - “Hooking”:  $D(r_1) \leftarrow v_2$ , где
    - $r_1$  – корень дерева, которому принадлежит  $v_1$
    - $v_1$  и  $v_2$  принадлежат разным деревьям

# Алгоритм Шилоха-Вишкина

1. Short-cut:

$$D(v) \leftarrow D(D(v))$$

2. Hooking:

у каждого ребра  $uv$  ( $u \neq v$ ) одна из инцидентных вершин меняет родителя

3. Привязка (hooking) звёзд к другим деревьям:

корню каждой звезды назначается в качестве родителя вершина из другого дерева

4. Если граф родителей состоит из звёзд, остановка

## Фундаментальные результаты в области (2)

- Awerbuch, Shiloah. New Connectivity and MSF Algorithms for Shuffle-Exchange Network and PRAM (1983)
- Используются модели SE и PRAM
- Для PRAM: сложность  $O(\log n)$ ,  $n + m$  процессоров
- В гонке на запись ячейки памяти побеждает “сильнейший” процессор

# A Fast GPU Algorithm for Graph Connectivity (1)

- J. Soman, K. Kishore, P. J. Narayanan (2010)
- Модификация алгоритма Шилоха-Вишкина:
  - привязка корней звёзд только к корням других звёзд
  - многоуровневый short-cut (pointer jumping)
  - сокращение графа посредством деактивации рёбер
- Оптимизации для GPU
  - снижение количество операций чтения из памяти
  - отказ от атомарных операций



# A Fast GPU Algorithm for Graph Connectivity (2)

- J. Soman, K. Kothapalli, P. J. Narayanan. Some GPU algorithms for graph connected components and spanning tree (2010)
- L. Wang. An Implementation of Connected Component Algorithm on GPU (2013)

# A Simple and Practical Linear-Work Parallel Algorithm for Connectivity

- J. Shun, L. Dhulipala, G. E. Blelloch (2014)
- Рекурсивный алгоритм сложности  $O(m)$  и глубины  $O(\log^3 n)$
- $(\beta, d)$ -разложение графа  $V$  ( $0 < \beta < 1$ ) –  $V_1, \dots, V_k$ :
  - кратчайший путь между вершинами в  $V_i$  не длиннее  $d$
  - в разных  $V_i, V_j$  лежат концы не более  $\beta m$  рёбер
- Реализации  $(\beta, d)$ -разложения:
  - параллельный (покомпонентно) BFS
  - две оптимизации параллельного BFS

# Better Speedups Using Simpler Parallel Programming for Graph Connectivity and Biconnectivity

- J. A. Edwards, U. Vishkin (2012)
- Авторы рассматривают проблему *двусвязности*
- Алгоритмы:
  - Хопкрофта-Тарьяна (pDFS) – время  $O(n)$ ,  $\lceil m/n \rceil + 1$  процессоров
  - Тарьяна-Вишкина – время  $O(\log n)$ ,  $O(n + m)$  процессоров
  - Тарьяна-Вишкина с использованием BFS – время  $O(h \log n)$ ,  $O(n + m)$  процессоров
- Платформы: Explicit Multi-Threading (XMT), GPGPU

## Итоги: выбранные статьи

- A Fast GPU Algorithm for Graph Connectivity
- A Simple and Practical Linear-Work Parallel Algorithm for Connectivity