Санкт-Петербургский государственный университет

Кафедра системного программирования

Реализация алгоритма достижимости с регулярными ограничениями на графическом ускорителе

Козенко Дмитрий Сергеевич, группа 23.Б08-мм

Научный руководитель: : к.ф.-м.н. С.В. Григорьев, доцент кафедры системного программирования

Санкт-Петербург
2025

Графовые базы данных и запросы с регулярными ограничениями

- Графовые базы данных набирают большую популярность в последние дни, например в создании баз знаний[1]
- Запросы с регулярными ограничениями активно используются в графовых базах данных
- Нет реализаций алгоритмов для регулярных запросов на GPU
- Алгоритм RPQ[2] основан на операциях линейной алгебры, которые отлично подходят вычислений на GPU

[1] Zhang Zuopeng Justin. Graph Databases for Knowledge Management https://ieeexplore.ieee.org/document/8123475

[2] RPQ с линейной алгеброй: https://se.math.spbu.ru/thesis/texts/Beljanin_Georgij_Olegovich_Spring_practice_2nd_year_2024_text.pdf

Библиотека cuBool[4]

- Достоинства библиотеки
  - Она имеет лучшие результаты производительности[3]
  - Простота и удобность интеграции и использования
- Недостатки библиотеки
  - Отсутствие поддержки на протяжении долгого времени
  - Отсутствие одной операции, необходимой для алгоритма

[3] Тесты производительности cuBool: `https://github.com/SparseLinearAlgebra/cuBool?tab=readme-ov-file#performance`

[4] Репозитория cuBool: `https://github.com/SparseLinearAlgebra/cuBool`

Постановка задачи

Целью работы является адаптация алгоритма достижимости с регулярными ограничениями для GPGPU вычислений и его перенос на видеокарту

Задачи:

- Обновить библиотеку cuBool для поддержки последней версии CUDA
- Реализовать в cuBool недостающие для алгоритма операции с булевыми матрицами
- Реализовать алгоритм, используя обновленную библиотеку
- Провести эксперименты: найти узкие места в алгоритме и уменьшить среднее время исполнения, сравнить время работы реализации алгоритма на GPU с его реализацией на CPU и проанализировать результаты

Обновление cuBool

- Библиотека cuBool была обновлена с версии CUDA 10.1 от 28 июля 2019 года до версии 12.6 от 14 августа 2024 года
- Удалена локальная версия Cub, который в последней версии CUDA устанавливается вместе с ней в систему
- Удалена опция компиляции *CUDA_SEPARABLE_COMPILATION* [5] из CMake, из-за которой многие тесты вели себя недетерминировано и падали

[5] Документация cmake: `https://cmake.org/cmake/help/latest/prop_tgt/CUDA_SEPARABLE_COMPILATION.html`

Реализация недостающей опции

- В cuBool не хватает операции умножения на инвертированную матрицу, необходимую для применения маски посещенных состояний.
- Все матрицы в cuBool в силу разреженности хранятся как множество индексов ненулевых элементов
- Пусть $C = A * \overline{B}$. Посмотрим, когда конкретный элемент C ненулевой

$$C_{ij} = 1 \iff \begin{cases} A_{ij} = 1 \\ \overline{B_{ij}} = 1 \end{cases} \iff \begin{cases} A_{ij} = 1 \\ B_{ij} = 0 \end{cases} \iff \begin{cases} A_{ij} = 1 \\ B_{ij} \neq 1 \end{cases}$$

- Множество ненулевых индексов C равно разности множеств ненулевых элементов матриц A и B

Реализация алгоритма

- Обновлена библиотека cuBool, результат работы можно увидеть в форке[6] на GitHub на ветке *update-cuda-12.6*
- Реализован алгоритм, код находится в репозитории[7] GitHub
- Настроен CI в репозитории:
  - Запуск тестов на версии функций cuBool, делающих вычисления на CPU
  - Запуск проверки стиля кода анализатором *clang-format*

[6] Форк репозитории cuBool: `https://github.com/mitya-y/cuBool`

[7] Репозитория с алгоритмом: `https://github.com/mitya-y/rpq`

Условия эксперимента

- В качестве данных была выбрана база знаний Wikidata[8] — граф размером 270 миллионов вершин, 15 миллиардов рёбер, а также 600 регулярных запросов к нему
- Для замера времени работы реализации на CPU оригинальный тест производительности из репозитория GitHub[9]
- Для реализации на GPU был написан схожий тест производительности[10]
- Тесты проводились на машине с Intel i5-10400f и Nvidia RTX 3050 (8 Gb VRAM), 16 Gb RAM

[8] База знаний WikiData: https://www.wikidata.org/wiki/Wikidata:Database_download (Дата доступа 11.12.24)

[9] Код для теста производительности алгоритма: `https://github.com/georgiy-belyanin/RPQ-bench`

[10] Код теста: `https://github.com/mitya-y/rpq/blob/master/benchmark.cpp`

Результаты сравнения с реализацией на CPU

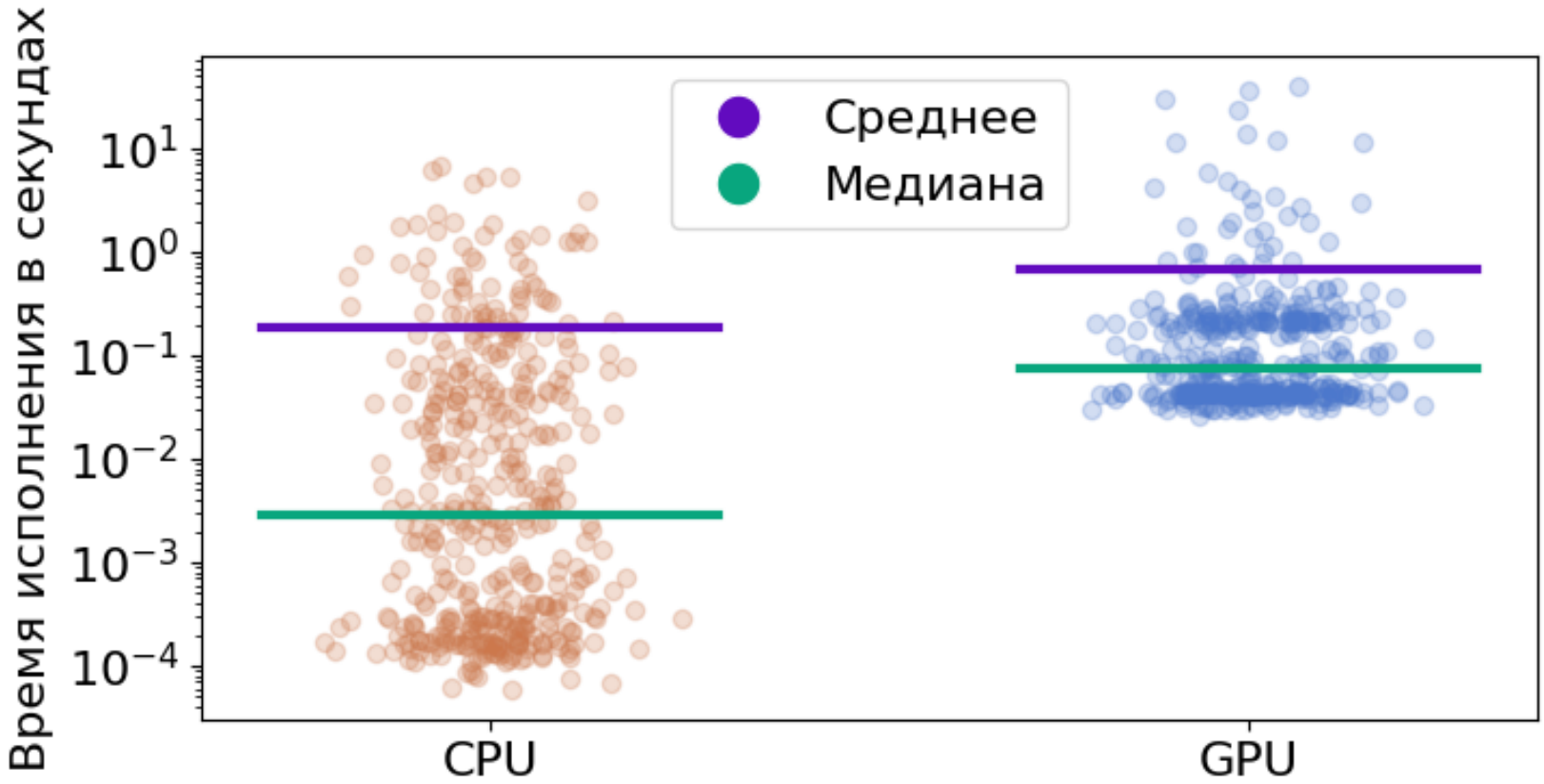

Рис.: Облако распределения времени исполнения запросов

Результаты сравнения с реализацией на CPU

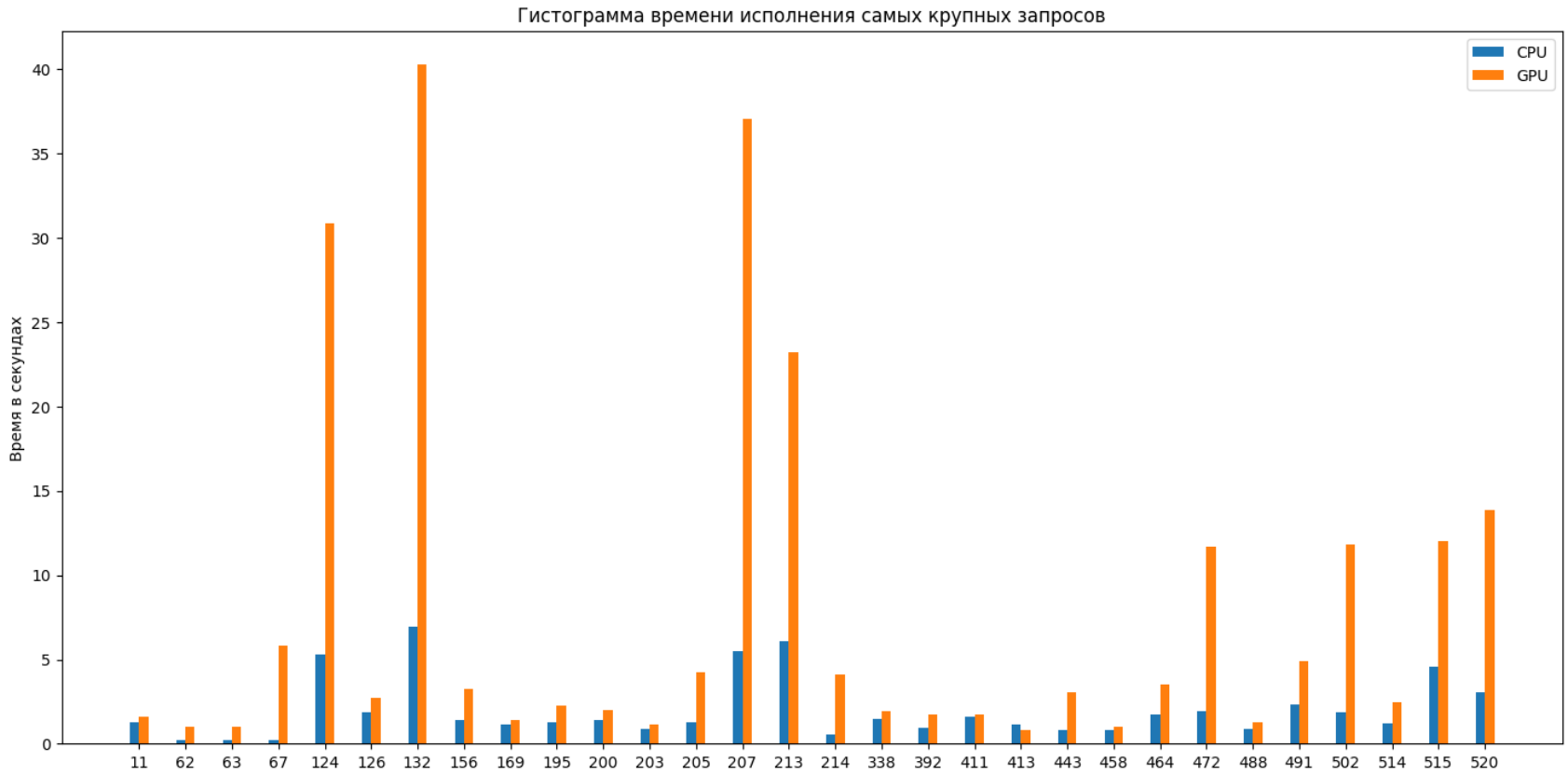

Рис.: Гистограмма времени исполнения на самых крупных запросах (занявших более 1 секунды)

5 лучших и 5 худших запросов в сравнении с CPU

Среднее замедление	3.59
Медиана всех ускорений	0.0379
Среднее аримф. по всем ускорениями	0.232

Номер запроса	Время GPU	Время CPU	Ускорение
353	0.182	0.263	1.444
421	0.294	0.441	1.501
416	0.431	0.704	1.633
346	0.277	0.46	1.658
311	0.179	0.3	1.682

Таблица: 5 лучших запросов

Номер запроса	Время GPU	Время CPU	Ускорение
429	0.170	0.000081	0.000476
511	0.200	0.000111	0.000553
484	0.202	0.000127	0.000627
433	0.175	0.000118	0.000671
442	0.174	0.000119	0.000683

Таблица: 5 худших запросов

Результаты

- Библиотека cuBool была обновлена до последней версии CUDA 12.6
- В cuBool реализована операция поэлементного умножения на инвертированную матрицу
- При помощи обновленной версии cuBool алгоритм достижимости с регулярными ограничениями был реализован для запуска на GPU
- Был проведен эксперимент, который показал, что текущая реализация на GPU проигрывает в производительности реализации на CPU: абсолютное среднее замедление равно 3.59