Санкт-Петербургский государственный университет

Кафедра системного программирования
Программная инженерия

Влаев Никита Владиславович

Реализация операций над разреженными булевыми матрицами на OpenCL

Отчет по производственной практике

Научный руководитель:
к.ф.-м.н., доцент кафедры информатики СПбГУ Григорьев С.В.

Санкт-Петербург
2021

Оглавление

Введение

Поиск путей в графе с ограничением в виде формальных языков — это задача, в которой формальные языки [13] используются для создания ограничений на множество путей между вершинами. Каждому пути между вершинами в графе соответствует слово, состоящее из меток на пройденных рёбрах, а ограничением на путь является принадлежность соответствующего ему слова некоторому заданному формальному языку.

Наибольший интерес среди классов формальных языков по иерархии Хомского [17] представляют контекстно-свободные языки по причине большей выразительности, чем регулярные языки. Так, в задаче поиска путей контекстно-свободные ограничения позволяют задавать более сложные отношения между вершинами. Например, важный класс запросов поиска вершин, лежащих на одном уровне иерархии [1], задаётся только контекстно-свободными, но не регулярными ограничениями. Запросы такого вида, как и другие запросы с контекстно-свободными ограничениями имеют широкое применение в графовых базах данных [16], статическом анализе кода [14], в биоинформатике [18] и при обработке rdf-файлов [4].

Задача поиска путей в графе с ограничением в виде формальных языков была впервые сформулирована Михалисом Яннакакисом (Mihalis Yannakakis) в 1990 году [21]. Многие алгоритмы для решения этой задачи были предложены с тех пор, но недавно, Йохем Куиперс и др. (Jochem Kuijpers) показали [9] что современные алгоритмы недостаточно эффективны для практического использования.

Одним из способов создать высокопроизводительные решения для задач анализа графов является сведение их к операциям линейной алгебры. Матричный [2] и тензорный [5] алгоритмы, согласно [8], являются одними из самых эффективных алгоритмов решения задачи контекстно-свободной достижимости(CFPQ), при этом они основаны на операциях линейной алгебры над булевыми матрицами. Использование GPGPU для вычисления операций с разреженными матрицами,

как известно из результатов [8], приводит к повышению эффективности алгоритмов. Однако на данный момент нет библиотек, предоставляющих соотвествующие примитивы, используя стандарт OpenCL [11], который предназначен для создания кроссплатформенных реализаций алгоритмов, что повышает переносимость библиотеки и вариативность в выборе оборудования, в том числе в вопросах стоимости. По этим причинам есть необходимость в реализации алгебраических примитивов для работы с разреженными булевыми матрицами на OpenCL.

Существует несколько способов реализовать алгебраические примитивы для работы с разреженными булевыми матрицами. Например, уже существуют реализации на CPU и CUDA C [8], для OpenCL есть возможность как использовать его в чистом виде, так и взять библиотеку clSPARSE [12] как объект для расширения функциональности, так как она уже предоставляет примитивы для работы с разреженными матрицами на OpenCL.

В рамках этой работы рассматривается реализация на основе clSPARSE, однако единовременно с ней другими авторами разрабатываются реализации с нуля на OpenCL и CUDA C. По этой причине конечным результатом этой работы будет также являться сравнительный анализ различных реализаций.

1. Постановка задачи

Целью данной работы является реализация операций линейной алгебры для работы с разреженными булевыми матрицами на GPGPU и сравнение ее с другими реализациями.

Для достижения данной цели в рамках работы были поставлены следующие задачи.

1. Реализовать операции ввода-вывода для разреженных булевых матриц на основе clSPARSE
2. Реализовать операцию поэлементного сложения разреженных булевых матриц на основе clSPARSE
3. Реализовать операцию умножения разреженных булевых матриц на основе clSPARSE
4. Провести сравнительный анализ полученной реализации с другими

2. Обзор

2.1. Используемые технологии

2.1.1. OpenCL

OpenCL [11] — это технология, связанная с параллельными компьютерными вычислениями на различных типах графических и центральных процессоров. Главное ее преимущество — возможность создания кроссплатформенных приложений для работы как с CPU, так и с GPU, поддерживающими стандарт OpenCL. Это качество позволяет как получить переносимую между платформами реализацию программных продуктов, так и подобрать специализированные аппаратные средства, подходящие под требования для разработки продукта.

2.1.2. GraphBLAS

GraphBLAS [10] — это спецификация API, которая определяет стандартные строительные блоки для графовых алгоритмов на основе линейной алгебры. GraphBLAS построен на идее, что разреженная матрица может быть использована для представления графов либо как матрица смежности, либо как матрица инцидентности. Спецификация GraphBLAS описывает, как графовые операции (например, обход и преобразование графов) могут быть эффективно реализованы с помощью методов линейной алгебры (например, умножение матриц) над различными полукольцами.

2.2. Библиотеки операций с разреженными булевыми матрицами

Необходимость в операциях с разреженными матрицами возникла достаточно давно, и на данный момент достигнуты существенные успехи в разработке соответствующих алгоритмов. Некоторые операции были реализованы в библиотеках линейной алгебры в рамках стандарта

BLAS. Такие библиотеки не предоставляют специальных версий алгоритмов для булевых матриц. Это является значимым недостатком, так как для булевой матрицы требуется меньше памяти и арифметических вычислений. Сравнение библиотек пройдет по следующим критериям:

1. Поддерживаемые форматы разреженных матриц
2. Набор поддерживаемых операций
3. Технологии: CUDA, OpenCL

В первую очередь важны поэлементное сложение и матричное умножение разреженных матриц, так как эти операций являются основополагающими для интересующих алгоритмов CFPQ.

2.2.1. clSPARSE

clSPARSE [12] — это библиотека, использующая OpenCL для реализации операций линейной алгебры для разреженных матриц. Этот проект является результатом коллаборации AMD Inc. и Vratis Ltd.. Эта библиотека предоставляет доступ через интерфейс на C и C++ к таким операциям над разреженными и плотными матрицами, как матричное умножение, умножение на вектор, преобразование матриц из одного формата в другой и считывание матриц. clSPARSE поддерживает форматы CSR и COO, и не поддерживает формат разреженных булевых матриц.

2.2.2. SuiteSparse

SuiteSparse [20] — это полная реализация стандарта GraphBLAS на CPU, который определяет набор разреженных матричных операций на расширенной алгебре полуколец с использованием, большого разнообразия операторов и типов. Применительно к разреженным матрицам смежности эти алгебраические операции эквивалентны вычислениям на графах. GraphBLAS предоставляет мощную и выразительную основу для создания графовых алгоритмов, основанных на математике

разреженных матричных операций на полукольце. Ведется активная работа над реализацией на CUDA при поддержке NVIDIA.

2.2.3. Cusp

Cusp [6] — это библиотека с открытым кодом для вычислений на графах с реализацией операций разреженной линейной алгебры, основанная на CUDA и Thrust. Поддерживает форматы CSR, COO, диагональных и смешанных разреженных матриц. Реализовано поэлементное сложение и матричное умножение разреженных матриц.

2.2.4. cuSparse

cuSPARSE [7] — проприетарная библиотека от NVIDIA с закрытым кодом, основанная на CUDA, которая содержит набор основных операций линейной алгебры, используемых для работы с разреженными матрицами. Поддерживает форматы CSR, COO, CSC, BSR разреженных матриц. Реализовано матричное умножение и поэлементное сложение с линейными коэффициентами.

2.3. Разработки научной группы

Параллельно работе над реализация операций над разреженными булевыми матрицами на основе clSPARSE, в рамках лаборатории языковых инструментов JetBrains шла разработка аналогичных инструментов с нуля на CUDA и OpenCL.

2.3.1. cuBool

cuBool [22] — это библиотека примитивов и операций линейной булевой алгебры для работы с разреженными матрицами, написанная на платформе NVIDIA CUDA. Библиотека предоставляет C-совместимый API, написанный в стиле GraphBLAS.

Библиотека поставляется с пакетом python pycubool - оболочкой для библиотеки cuBool C API. Этот пакет экспортирует функции и прими-

тивы библиотеки в формате высокого уровня с автоматическим управлением ресурсами и синтаксически удобным API.

Библиотека предоставляет наиболее популярные операции для работы с булевыми матрицами в формате CSR и COO, такие как ввод-вывод, транспонирование, извлечение субматрицы/субвектора, редуцирование матрицы в вектор, поэлементное сложение, матричное умножение и произведение Кронекера.

2.3.2. clBool

clBool [23] — это библиотека для операций с разреженными булевыми матрицами, реализованная на основе технологии OpenCL. В ней поддержаны такие форматы матриц, как CSR, COO, DCSR и реализованы операции умножения, поэлементного сложения, произведения Кронекера булевых матриц.

2.4. Архитектура clSPARSE

Так как предложенное решение основано на clSPARSE, необходимо кратко описать архитектуру этой библиотеки.

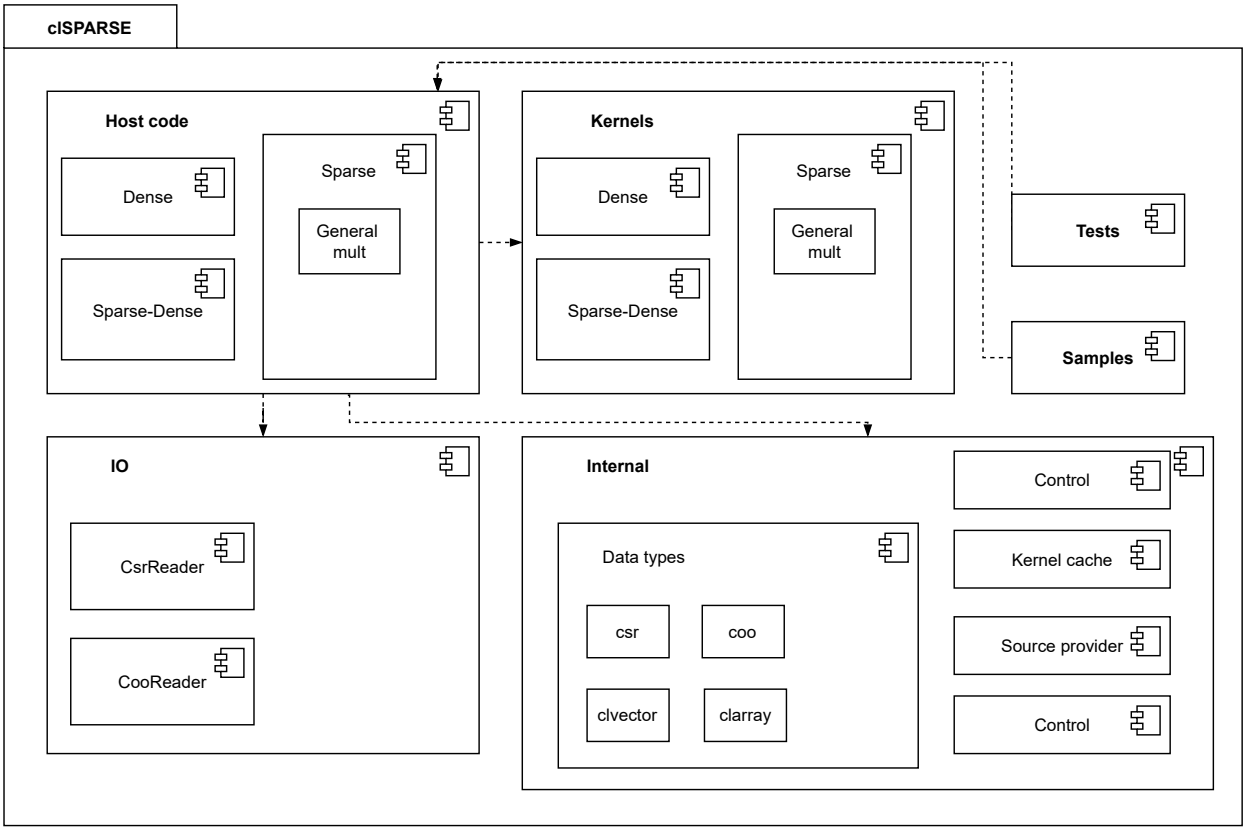
Библиотека clSPARSE написана на языке C++14. Для её работы необходимо устройство, поддерживающее стандарт OpenCL 1.0 или 1.1 с установленным окружением, которое можно найти на сайте вендора.

Структура библиотеки представлена на рисунке 1. Компоненты отображают файловую структуру исходного кода.

В компоненте **Kernels** содержатся ядра OpenCL, необходимые для запуска параллельных вычислений на используемом графическом ускорителе. В их число входят как стандартные обобщенные операции, такие как scan и reduce, так и более содержательные, как вычисление приближенного сверху кол-ва ненулевых элементов для каждого ряда результирующей матрицы.

Host code содержит код, предоставляющий интерфейс для запуска алгебраических операций над плотными и разреженными матрицами и векторами в C++.

Рис. 1: Архитектура clSPARSE



IO содержит методы для считывания поддерживаемых форматов матриц из файла. В частности, поддерживается формат MTX.

Компонент **Internal** содержит:

1. Определения основных типов C++, соответствующих векторам, массивам, и разреженным матрицам в clSPARSE

2. Класс Control, отвечающий за управление ресурсами используемого устройства, поддерживающего OpenCL

3. Класс Kernel cache, выполняющий кеширование скомпилированных ядер

4. Класс Source provider, предоставляющий доступ к информации об используемом устройстве с OpenCL и частичному регулированию конфигурации библиотеки во время исполнения

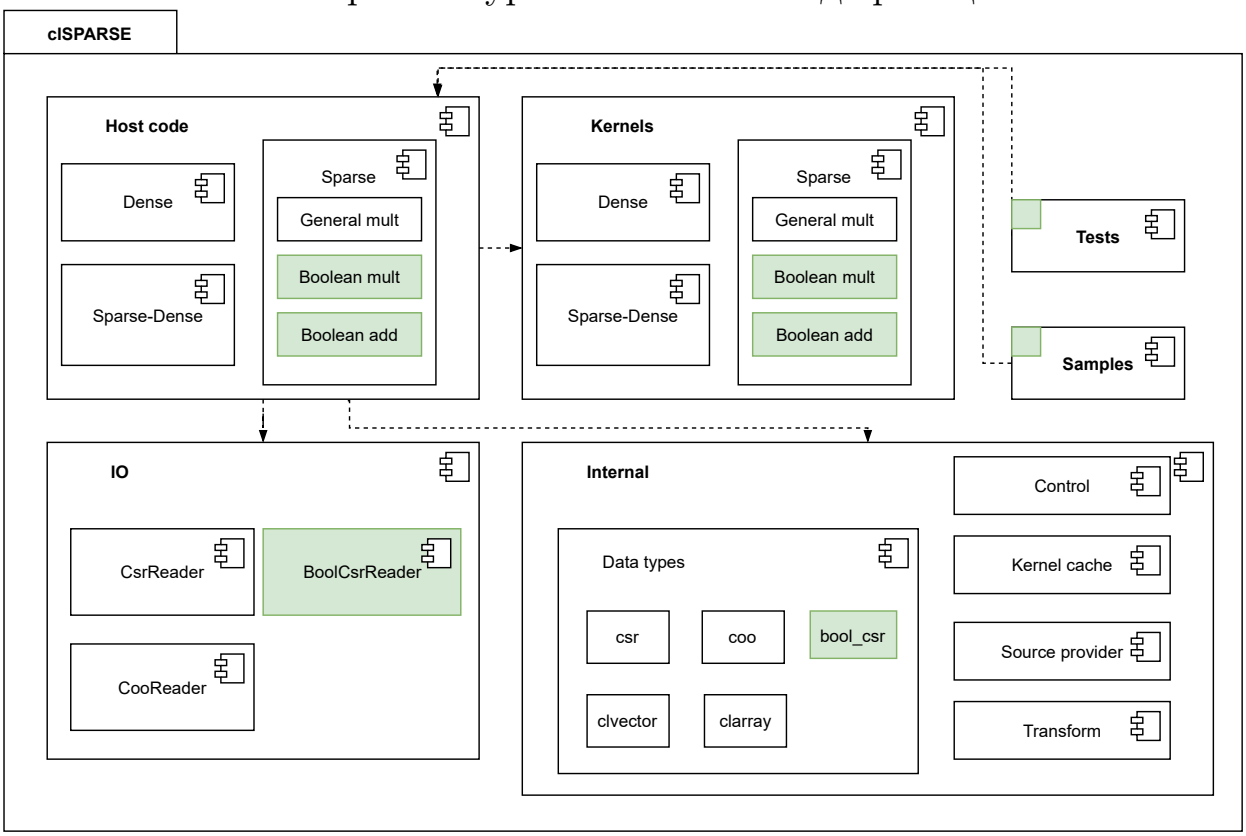
5. Компонент Transform, предоставляющий операции для преобразования одного типа матрицы в другой

Кроме этого, на схеме представлены компоненты, отвечающий за тесты и примеры использования.

3. Решение

3.1. Архитектура

Рис. 2: Архитектура clSPARSE с модификациями



Архитектура библиотеки clSPARSE дополнена [24] модулями для поддержки операций над разреженными булевыми матрицами, согласно схеме 2. В них входят:

- Ядра, реализующие необходимые примитивы для алгоритмов поэлементного сложения и умножения разреженных булевых матриц
- Модуль для считывания булевых CSR матриц
- Описание типа данных булевой CSR матрицы: тип clsparseBoolCsrMatrix и необходимые метаданные
- Код на CPU, необходимый для запуска соответствующих кернелов: функции clsparseBoolScsrSpGemm и clsparseBoolScsrElemAdd

- Примеры использования и тесты

3.2. Реализация

3.2.1. Ввод-вывод

Для удобной работы и отладки была реализована функциональность ввода-вывода булевых разреженных матриц в формате Matrix Market [3] и встроена в библиотеку clSPARSE.

3.2.2. Умножение

Реализован алгоритм матричного умножения булевых разреженных матриц на основе оригинального алгоритма SpGEMM [15], представленного в библиотеке clSPARSE. Этот алгоритм был предназначен для работы с матрицами, в ячейках которых лежали числа, целочисленные или с фиксированной запятой. Поэтому его необходимо было адаптировать для эффективной работы с разреженными булевыми матрицами, в формате представления которых этих чисел нет, так как для работы с ними достаточно знать местоположение ненулевых элементов.

3.2.3. Поэлементное сложение

Поэлементное сложение реализовано для булевых CSR матриц аналогично реализации в cuBool [22].

Основное отличие заключается в том, что базовая реализация создана на основе CUDA и использует готовые примитивы из Thrust и CUDA-специфичные процедуры для работы с данными в одном ворпе. По этой причине было необходимо реализовать соотвествующие примитивы на OpenCL и преобразовать изначальный алгоритм таким образом, чтобы новая реализация была сравнима по производительности с оригинальной и встраивалась в архитектуру clSPARSE.

3.3. Сравнительный анализ

Для сравнения операций над булевыми матрицами в различных реализациях использовался набор матриц, описанный в таблице 1. Для замеров были выбраны сильно разреженные матрицы из коллекции SuiteSparse [19], отличающиеся плотностью ненулевых элементов и размерами. Матрицы описывают совершенно разные данные: связанные покупки в интернет-магазинах, веб-соединения, дорожные сети.

№	Matrix M	# Rows	Nnz of M	Nnz of M^2	Nnz of $M + M^2$
0	wing	62,032	243,088	714,200	917,178
1	luxembourg_osm	114,599	239,332	393,261	632,185
2	amazon0312	400,727	3,200,400	14,390,544	14,968,909
3	amazon-2008	735,323	5,158,388	25,366,745	26,402,678
4	web-Google	916,428	5,105,039	29,710,164	30,811,855
5	roadNet-PA	1,090,920	3,083,796	7,238,920	9,931,528
6	roadNet-TX	1,393,383	3,843,320	8,903,897	12,264,987
7	belgium_osm	1,441,295	3,099,940	5,323,073	8,408,599
8	roadNet-CA	1,971,281	5,533,214	12,908,450	17,743,342
9	netherlands_osm	2,216,688	4,882,476	8,755,758	13,626,132

Таблица 1: Описание матриц.

Было произведено сравнение производительности операций умножения и поэлементного сложения булевых разреженных матриц. В таблице с результатами сравнения операций поэлементного сложения булевых матриц не представлена исходная библиотека clSPARSE, так как в ней изначально не реализована данная операция.

Оборудование: Ubuntu 20.04, Intel core i7-6700 CPU, 3.4GHz, DDR4 64Gb RAM, Geforce GTX 1070 GPGPU 8Gb RAM.

M	Clbool	clSPARSE-Bool	Cubool	Cusp	cuSPARSE	SuiteSparse
0	1.89±0.32	1.96±1.04	1.12±0.02	1.54±0.20	2.40±0.04	3.91±2.06
1	1.63±0.32	3.18±1.05	1.84±0.10	1.11±0.21	0.86±0.04	1.86±0.29
2	23.53±0.63	14.12±1.79	12.14±0.46	16.26±0.40	23.74±0.04	37.12±0.13
3	36.68±0.88	21.34±2.53	20.06±2.67	29.86±1.65	27.75±1.65	65.02±1.35
4	43.28±2.89	22.78±2.01	24.20±0.90	32.08±1.34	88.48±0.28	77.68±1.66
5	12.47±0.13	27.96±1.73	16.69±0.59	11.07±0.28	11.62±0.03	36.50±1.01
6	15.29±0.05	35.35±1.89	19.86±0.65	14.42±0.98	17.43±0.03	46.33±3.07
7	10.43±0.14	36.17±1.77	19.55±0.12	9.83±0.24	11.57±0.99	28.44±1.00
8	22.31±0.11	50.07±2.25	31.61±1.01	18.90±0.40	20.50±0.82	65.74±1.39
9	18.91±2.08	55.38±2.14	30.65±1.89	14.75±0.31	18.15±0.03	50.29±0.93

Таблица 2: Поэлементное сложение.

M	Clbool-Hash	Clbool	clSPARSE-Bool	clSPARSE	Cubool	cuSPARSE	Cusp	SuiteSparse
0	1.88±0.34	4.23±0.36	1.57±0.28	2.02±0.34	1.78±0.06	20.04±0.04	5.39±0.12	7.92±0.27
1	2.06±0.36	6.41±0.36	2.31±0.30	2.64±0.34	2.50±0.04	1.72±0.05	3.79±0.07	3.72±1.57
2	52.02±0.04	62.39±0.06	25.19±0.98	32.73±1.57	24.19±0.82	408.71±0.54	108.98±0.45	258.04±8.39
3	81.43±0.29	96.62±0.44	40.25±1.36	51.19±2.01	34.63±1.53	184.64±10.23	172.74±0.36	378.94±15.28
4	126.36±0.34	Out of memory	140.54±4.12	165.03±8.30	42.19±1.26	4726.33±0.52	Out of memory	712.54±20.07
5	14.02±0.15	34.56±0.18	19.16±1.08	23.03±1.60	18.26±0.19	37.24±0.13	42.38±0.17	67.36±1.57
6	16.64±0.12	41.90±0.20	23.62±1.24	28.43±1.88	22.73±0.18	46.40±0.15	51.77±0.30	81.27±1.80
7	16.62±0.23	38.61±0.23	22.28±0.99	25.31±1.47	23.37±0.17	26.54±0.07	33.09±0.20	58.46±1.76
8	23.10±0.06	58.99±1.62	32.99±1.77	39.26±2.52	32.09±0.34	65.14±4.47	76.72±3.23	116.88±3.99
9	24.57±0.11	57.07±0.13	33.89±1.39	38.51±2.12	35.30±0.22	50.68±0.14	51.28±0.35	93.47±2.09

Таблица 3: Матричное умножение.

По результатам проведенных экспериментов были сделаны следующие выводы:

- Реализации на CUDA показывают лучшие результаты в замерах производительности на NVIDIA, но не поддерживаются широким спектром устройств, в отличие от реализаций на OpenCL
- Сложение
    1. На матрицах с Nnz of $M + M^2 > 10000000$ реализация не уступает CPU
    2. На OpenCL альтернативый алгоритм из Clbool показывает лучшие результаты, чем представленный

- Умножение

    1. Представленный алгоритм превосходит по скорости CPU
    2. Алгоритм превышает по скорости аналогичную реализацию с нуля на OpenCL в clBool
    3. Алгоритм с использованием хеш-таблиц в clBool превосходит по эффективности другие реализации на OpenCL

- Кодовая база clSPARSE перестала активно поддерживаться с 2015 года, и как показала практика, интеграция ее как низкоуровневого элемента в инструменты для решения CFPQ является очень трудоемкой задачей по причине распространенной последовательной связанности операций в архитектуре библиотеки. При этом, согласно сравнительному анализу, созданы эффективные аналоги с архитектурой, более подходящей для интеграции их в высокоуровневые инструменты.

4. Заключение

В результате проделанной работы были выполнены следующие задачи.

1. Реализованы операции ввода-вывода для разреженных булевых матриц на основе clSPARSE.

2. Реализована операция матричного умножения разреженных булевых матриц на основе clSPARSE.

3. Реализована операция поэлементного сложения разреженных булевых матриц на основе clSPARSE.

4. Проведен сравнительный анализ реализаций, в результате которого была оценена производительность итогового решения относительно его аналогов. Итоги:

   - Реализация операций умножения и сложения разреженных матриц на OpenCL на основе clSPARSE не сильно уступает ее аналогам на OpenCL без дополнительных зависимостей по производительности
   - Результат данной работы полезен только с точки зрения проверки гипотезы о том, что на основе clSPARSE можно построить эффективную реализацию операций над разреженными булевыми матрицами
   - Опыт использования библиотеки показал, что она не предпочтительна для дальнейшей работы

Список литературы

[1] Abiteboul Serge, Hull Richard, Vianu Victor. Foundations of Databases. — 1995. — 01. — ISBN: 0-201-53771-0.

[2] Azimov Rustam, Grigorev Semyon. Context-free path querying by matrix multiplication. — 2018. — 06. — P. 1–10.

[3] Boisvert Ronald, Pozo Roldan, Remington Karin. The Matrix Market Exchange Formats: Initial Design. — 1997. — 03.

[4] Context-Free Path Queries on RDF Graphs / Zhang Xiaowang, Zhiyong Feng, Xin Wang, Guozheng Rao. — 2015. — 06.

[5] Context-Free Path Querying by Kronecker Product / Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev. — 2020. — 08. — P. 49–59. — ISBN: 978-3-030-54831-5.

[6] Corporation NVIDIA. CUDA library for sparse linear algebra and graph computations based on Thrust. — `https://cusplibrary.github.io/`. — 2014.

[7] Corporation NVIDIA. CUDA sparse matrix library. — `https://docs.nvidia.com/cuda/cusparse/index.html`. — 2020.

[8] Evaluation of the Context-Free Path Querying Algorithm Based on Matrix Multiplication / Nikita Mishin, Iaroslav Sokolov, Egor Spirin et al. // Proceedings of the 2nd Joint International Workshop on Graph Data Management Experiences and Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA'19. — New York, NY, USA : Association for Computing Machinery, 2019. — Access mode: `https://doi.org/10.1145/3327964.3328503`.

[9] An Experimental Study of Context-Free Path Query Evaluation Methods / Jochem Kuijpers, George Fletcher, Nikolay Yakovets, Tobias Lindaaker // Proceedings of the 31st International Conference on Scientific and Statistical Database Management. — SSDBM '19. —

New York, NY, USA : Association for Computing Machinery, 2019. — P. 121–132. — Access mode: https://doi.org/10.1145/3335783.3335791.

[10] GraphBLAS Mathematics - Provisional Release 1.0. — Access mode: http://www.mit.edu/~kepner/GraphBLAS/GraphBLAS-Math-release.pdf (online; accessed: 29.05.2020).

[11] Group Khronos. OPEN STANDARD FOR PARALLEL PROGRAMMING OF HETEROGENEOUS SYSTEMS. — https://www.khronos.org/opencl/. — 2020.

[12] Inc. AMD, Ltd. Vratis. OpenCL library implementing Sparse linear algebra routines. — http://clmathlibraries.github.io/clSPARSE/. — 2020.

[13] Jäger Gerhard, Rogers James. Formal language theory: refining the Chomsky hierarchy. — https://doi.org/10.1098/rstb.2012.0077. — 2012.

[14] Kodumal John, Aiken Alex. The Set Constraint CFL Reachability Connection in Practice // SIGPLAN Not. — 2004. — Vol. 39, no. 6. — P. 207–218. — Access mode: https://doi.org/10.1145/996893.996867.

[15] Liu Weifeng, Vinter Brian. An Efficient GPU General Sparse Matrix-Matrix Multiplication for Irregular Data. — 2014. — 05. — P. 370–381.

[16] Mendelzon A., Wood P. T. Finding Regular Simple Paths in Graph Databases // SIAM J. Comput. — 1995. — Vol. 24. — P. 1235–1258.

[17] N. Chomsky. Three models for the description of language. — doi: 10.1109/TIT.1956.1056813. — 1956.

[18] Sevon Petteri, Eronen Lauri. Subgraph Queries by Context-free Grammars // Journal of Integrative Bioinformatics. — 2008. — 06. — Vol. 5.

[19] SuiteSparse Matrix Collection. — Режим доступа: https://sparse.tamu.edu/ (дата обращения: 12.05.2021).

[20] SuiteSparse:GraphBLAS. — Access mode: https://github.com/DrTimothyAldenDavis/GraphBLAS (online; accessed: 16.12.2020).

[21] Yannakakis Mihalis. Graph-Theoretic Methods in Database Theory. — 1990. — 01. — P. 230–242.

[22] https://github.com/EgorOrachyov. Linear Boolean algebra library primitives and operations for work with sparse matrices written on the NVIDIA CUDA platform. — https://github.com/JetBrains-Research/cuBool. — 2021.

[23] https://github.com/mkarpenkospb. Library for operations with sparse boolean matrices, implemented using the technology OpenCL. — https://github.com/mkarpenkospb/clBool. — 2021.

[24] nikitavlaev00@gmail.com. clSPARSE with bool operations. — https://github.com/nikitavlaev/clSPARSE/tree/dev. — 2021.