

Реализация алгоритма поиска путей в графовых базах данных через тензорное произведение на GPGPU

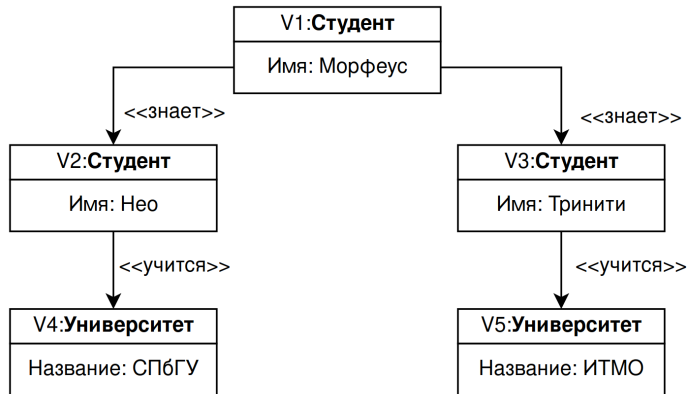
Орачев Егор Станиславович, 17.Б11-мм

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

Рецензент: разработчик биоинформатического ПО, ЗАО "БИОКАД" А.С. Хорошев

Программная инженерия

29 апреля 2021



- Графовая модель данных
- Графовые базы данных
- Граф программы, потока управления, данных и т.д
- Запрос к графовой БД: ограничение на пути

Figure: Пример графовой БД

Запросы с КС ограничениями

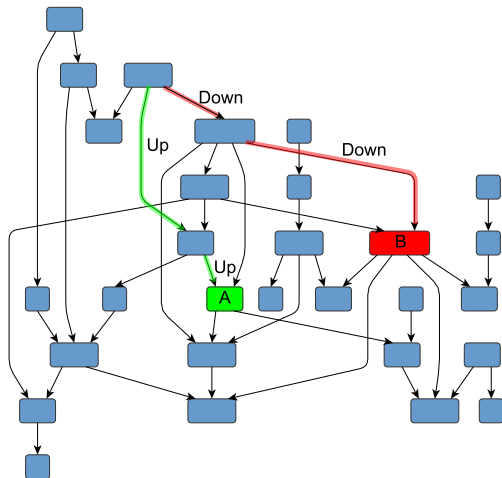


Figure: Пример графа

- Мотивация: навигация в графе
- Пример: находятся ли вершины A и B на одном уровне иерархии?
- Применение: анализ RDF данных, биоинформатика, статический анализ кода
- Проблема: отсутствие поддержки КС запросов как в современных СУБД, так и в отдельных инструментах

Алгоритмы поиска путей с КС ограничениями

- На основе линейной алгебры:
 - ▶ Алгоритм Рустама Азимова
 - ▶ **Алгоритм на основе тензорного произведения¹**
- Операции: матричное умножение, поэлементное сложение, произведение Кронекера и т.д. в булевом полукольце
- Структура матриц: разреженная
- Для эффективной реализации требуется библиотека примитивов разреженной булевой линейной алгебры на GPGPU

¹Context-Free Path Querying by Kronecker Product, Egor Orachev, Ilya Epelbaum, Rustam Azimov, Semyon Grigorev. Дата обращения: 9.04.2021. Ссылка на статью:
https://link.springer.com/chapter/10.1007/978-3-030-54832-2_6

Целью данной работы является реализация алгоритма поиска путей в графовых базах данных через тензорное произведение на GPGPU.

Задачи:

- Разработка архитектуры библиотеки примитивов разреженной линейной булевой алгебры для вычислений на GPGPU
- Реализация библиотеки в соответствии с разработанной архитектурой
- Реализация алгоритма поиска путей с КС ограничениями через тензорное произведение с использованием разработанной библиотеки
- Экспериментальное исследование полученных результатов

Требования к библиотеке

- Поддержка вычислений на Cuda-устройстве
- Поддержка вычислений на CPU
- C-совместимый API для работы с библиотекой
- Python-пакет для работы с примитивами и операциями библиотеки в управляемой высокоуровневой среде языка Python
- Поддержка логирования, функций для отладки и прототипирования конечных пользовательских алгоритмов

Архитектура библиотеки

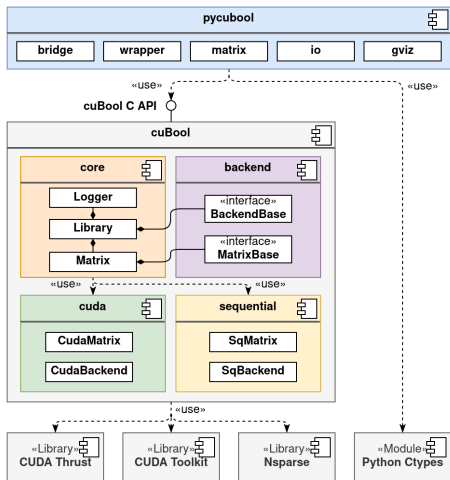


Figure: Архитектура разработанной библиотеки

Последовательность обработки операций

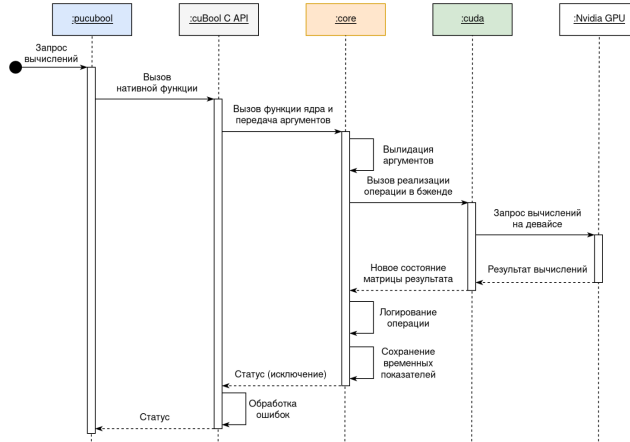


Figure: Последовательность выполнения вычислительной матричной операции на Nvidia GPU с использованием rucubool

- C, C++, CUDA C/C++, Python
- Разреженная матрица булевых значений хранится формате CSR
- Размер в памяти $\text{sizeof}(\text{unsigned int}) * (\text{rows} + 1 + \text{nvals})$
- Операции:
 - ▶ Умножение
 - ▶ Поэлементное сложение
 - ▶ Произведение Кронекера
 - ▶ Транспонирование
 - ▶ Редуцирование к вектор-столбцу
 - ▶ Извлечение подматрицы

Пример использования C API

```
1 #include <cubool/cubool.h>
2
3 cuBool_Status TransitiveClosure(cuBool_Matrix A, cuBool_Matrix* T) {
4     cuBool_Matrix_Duplicate(A, T);           /* Копируем матрицу смежности A */
5
6     cuBool_Index total = 0;
7     cuBool_Index current;
8     cuBool_Matrix_Nvals(*T, &current);       /* Количество ненулевых значений */
9
10    while (current != total) {                /* Пока результат меняется */
11        total = current;
12        cuBool_MxM(*T, *T, *T, CUBOOL_HINT_ACCUMULATE); /* T += T x T */
13        cuBool_Matrix_Nvals(*T, &current);
14    }
15
16    return CUBOOL_STATUS_SUCCESS;
17 }
```

Figure: Вычисление транзитивного замыкания для ориентированного графа без меток с использованием **cuBool C API**

Пример использования Python API

```
1 import pycubool
2
3 def transitive_closure(a: pycubool.Matrix):
4     t = a.duplicate()           # Копируем матрицу смежности A
5     total = 0                   # Количество ненулевых значений результата
6
7     while total != t.nvals:     # Пока результат меняется
8         total = t.nvals
9         t.mxm(t, out=t, accumulate=True) #  $t += t \times t$ 
10
11     return t
```

Figure: Вычисление транзитивного замыкания для ориентированного графа без меток с использованием **pycubool**

Алгоритм поиска путей с КС ограничениями через тензорное произведение

- Алгоритм реализован с использованием разработанного пакета русibool
- Реализация доступна в рамках проекта CFPQ-PyAlgo
- Вход: ориентированный граф с метками на ребрах и КС грамматика в виде рекурсивного автомата
- Выход: матрица достижимости, а также индекс для восстановления всех пути в графе соответствии с входной грамматикой

Инфраструктура CFPQ-PyAlgo

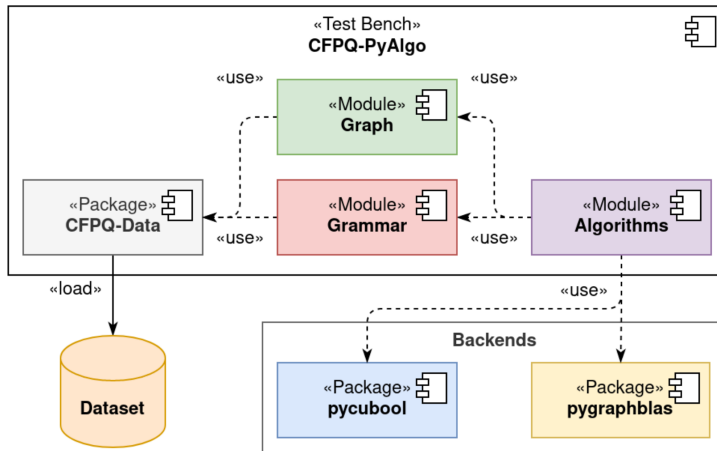


Figure: Архитектура CFPQ-PyAlgo стенда для тестирования алгоритмов

- Рабочая станция: Intel Core i7-6790, 3.40GHz, RAM DDR4 64Gb, GeForce GTX 1070 с 8Gb VRAM, ОС Ubuntu 20.04
- Данные, необходимые для замеров, предварительно загружаются в RAM или VRAM в требуемом формате
- Исследовательские вопросы:
 - B1:** Какова производительность отдельных операций реализованной библиотеки примитивов разреженной линейной булевой алгебры на GPGPU по сравнению с существующими аналогами?
 - B2:** Какова производительность реализованного алгоритма поиска путей через тензорное произведение на GPGPU по сравнению с существующими аналогами, также полагающимися на примитивы линейной алгебры?

B1: Набор данных

Матрица M	Кол-во Строк R	Nnz M	Nnz/ R	Max Nnz/ R	Nnz M^2	Nnz $M + M^2$
wing	62,032	243,088	3.9	4	714,200	917,178
luxembourg_osm	114,599	239,332	2.0	6	393,261	632,185
amazon0312	400,727	3,200,400	7.9	10	14,390,544	14,968,909
amazon-2008	735,323	5,158,388	7.0	10	25,366,745	26,402,678
web-Google	916,428	5,105,039	5.5	456	29,710,164	30,811,855
roadNet-PA	1,090,920	3,083,796	2.8	9	7,238,920	9,931,528
roadNet-TX	1,393,383	3,843,320	2.7	12	8,903,897	12,264,987
belgium_osm	1,441,295	3,099,940	2.1	10	5,323,073	8,408,599
roadNet-CA	1,971,281	5,533,214	2.8	12	12,908,450	17,743,342
netherlands_osm	2,216,688	4,882,476	2.2	7	8,755,758	13,626,132

Figure: Разреженные матричные данные

B1: Матричное умножение

Матрица M	cuBool		CUSP		cuSPRS		clSPRS		SuiteSprs	
	t	m	t	m	t	m	t	m	t	m
wing	1.9	93	5.2	125	20.1	155	4.2	105	7.9	22
luxembourg_osm	2.4	91	3.7	111	1.7	151	6.9	97	3.1	169
amazon0312	23.2	165	108.5	897	412.8	301	52.2	459	257.6	283
amazon-2008	33.3	225	172.0	1409	184.8	407	77.4	701	369.5	319
web-Google	41.8	241	246.2	1717	4761.3	439	207.5	1085	673.3	318
roadNet-PA	18.1	157	42.1	481	37.5	247	56.6	283	66.6	294
roadNet-TX	22.6	167	53.1	581	46.7	271	70.4	329	80.7	328
belgium_osm	23.2	151	32.9	397	26.7	235	68.2	259	56.9	302
roadNet-CA	32.0	199	74.4	771	65.8	325	98.2	433	114.5	344
netherlands_osm	35.3	191	51.0	585	51.4	291	102.8	361	90.9	311

Figure: Матричное умножение, время (t) в миллисекундах, память (m) в мегабайтах, отклонение в пределах 10%

В1: Поэлементное матричное сложение

Матрица M	cuBool		CUSP		cuSPRS		clSPRS		SuiteSprs	
	t	m	t	m	t	m	t	m	t	m
wing	1.1	95	1.4	105	2.4	163	-	-	2.3	176
luxembourg_osm	1.7	95	1.0	97	0.8	151	-	-	1.6	174
amazon0312	11.4	221	16.2	455	24.3	405	-	-	37.2	297
amazon-2008	17.5	323	29.5	723	27.2	595	-	-	64.8	319
web-Google	24.8	355	31.9	815	89.0	659	-	-	77.2	318
roadNet-PA	16.9	189	11.2	329	11.6	317	-	-	36.6	287
roadNet-TX	19.6	209	14.5	385	16.9	357	-	-	45.3	319
belgium_osm	19.5	179	10.2	303	10.5	297	-	-	28.5	302
roadNet-CA	30.5	259	19.4	513	20.2	447	-	-	65.2	331
netherlands_osm	30.1	233	14.8	423	18.3	385	-	-	50.2	311

Figure: Поэлементное матричное сложение, время (t) в миллисекундах, память (m) в мегабайтах, отклонение в пределах 10%

B2: Набор данных

Граф \mathcal{G}	$ V $	$ E $	Кол-во <i>sco</i>	Кол-во <i>type</i>	Кол-во <i>bt</i>	Кол-во <i>a</i>	Кол-во <i>d</i>
go-hierarchy	45 007	980 218	490 109	0	—	—	—
enzyme	48 815	109 695	8 163	14 989	—	—	—
eclass_514en	239 111	523 727	90 512	72 517	—	—	—
go	272 770	534 311	90 512	58 483	—	—	—
geospecies	450 609	2 201 532	0	89 062	20 867	—	—
taxonomy	5 728 398	14 922 125	2 112 637	2 508 635	—	—	—
arch	3 448 422	5 940 484	—	—	—	671 295	2 298 947
crypto	3 464 970	5 976 774	—	—	—	678 408	2 309 979
drivers	4 273 803	7 415 538	—	—	—	858 568	2 849 201
fs	4 177 416	7 218 746	—	—	—	824 430	2 784 943

Figure: RDF графы и графы программ для экспериментов

B2: Запросы с КС ограничениями

$$S \rightarrow \overline{subClassOf} \ S \ subClassOf \mid \overline{type} \ S \ type \\ \mid \overline{subClassOf} \ subClassOf \mid \overline{type} \ type \quad (1)$$

$$S \rightarrow \overline{subClassOf} \ S \ subClassOf \mid subClassOf \quad (2)$$

$$S \rightarrow broaderTransitive \ S \ \overline{broaderTransitive} \\ \mid broaderTransitive \ \overline{broaderTransitive} \quad (3)$$

$$S \rightarrow \overline{d} \ V \ d \\ V \rightarrow ((S?)\overline{a})^*(S?)(a(S?))^* \quad (4)$$

Figure: КС грамматики запросов для анализа. Запросы, для анализа RDF данных: G_1 (1), G_2 (2), G_{geo} (3), для анализа указателей в графах программ — G_{ma} (4).

B2: Анализ RDF данных

Граф \mathcal{G}	G_1								G_2							
	Tns_{gpu}		Tns_{cpu}		Mtx_{gpu}		Mtx_{cpu}		Tns_{gpu}		Tns_{cpu}		Mtx_{gpu}		Mtx_{cpu}	
	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m
go-hierarchy	0.15	315	0.17	265	0.11	208	0.08	254	0.27	484	0.24	252	0.06	6	0.09	255
enzyme	0.02	9	0.04	137	0.01	<1	0.01	181	0.01	4	0.02	132	0.01	<1	0.01	181
eclass_514en	0.10	57	0.24	205	0.04	14	0.07	180	0.09	36	0.27	193	0.02	2	0.06	181
go	1.67	176	1.58	282	0.43	68	1.00	244	0.82	119	1.27	243	0.18	12	0.94	246
taxonomy	2.21	1266	4.42	2018	0.71	364	1.13	968	0.90	969	3.56	1776	0.24	91	0.72	1175

Figure: Анализ RDF данных с использованием запросов G_1 и G_2 , время (t) в секундах, память (m) в мегабайтах, отклонение в пределах 10%

Граф \mathcal{G}	Tns_{gpu}		Tns_{cpu}		Mtx_{gpu}		Mtx_{cpu}	
	t	m	t	m	t	m	t	m
geospecies	err	err	26.32	19537	1.17	5272	7.48	7645

Figure: Анализ RDF данных с использованием запроса G_{geo} , время (t) в секундах, память (m) в мегабайтах, отклонение в пределах 10%

В2: Анализ указателей

Граф \mathcal{G}	Tns_{gpu}		Tns_{cpu}		Mtx_{gpu}		Mtx_{cpu}	
	t	m	t	m	t	m	t	m
arch	57.22	1928	262.45	6718	27.90	588	83.75	1842
crypto	57.43	1966	257.52	6720	28.10	596	84.83	1842
drivers	<i>err</i>	<i>err</i>	1309.57	46941	62.49	3999	269.93	5750
fs	83.86	3166	470.49	46941	47.67	932	165.09	5750

Figure: Анализ указателей с использованием запроса G_{ma} , время (t) в секундах, память (m) в мегабайтах, отклонение в пределах 10%

Заключение

- Спроектирована библиотека примитивов линейной булевой алгебры для работы с разреженными данными на GPGPU
- Реализована библиотека cuBool² в соответствии с разработанной архитектурой, опубликован соответствующий python-пакет³
- С использованием русicubool реализован алгоритм поиска путей с КС ограничениями через тензорное произведение
- Выполнено экспериментальное исследование полученных артефактов
- Результаты исследования были представлены на конференции GrAPL 2021⁴

²GitHub cuBool: <https://github.com/JetBrains-Research/cuBool>

³PyPI pycubool: <https://test.pypi.org/project/pycubool/>

⁴GrAPL 2021: Workshop on Graphs, Architectures, Programming, and Learning. Сайт конференции: <https://hpc.pnl.gov/grapl/>.