



Санкт-Петербургский государственный университет

Выпускная квалификационная работа

Реализация библиотеки примитивов разреженной линейной алгебры с поддержкой вычислений на GPU

Орачев Егор Станиславович, 21.M07-мм

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

Рецензент:
эксперт, ООО “Техкомпания Хуавей” С.В. Моисеев

Введение

- Графовая модель данных
- Реальные графы разреженные и насчитывают миллионы вершин, миллиарды ребер
- Высокопроизводительный анализ графов
- Реализация алгоритмов на графах в терминах разреженной линейной алгебры

Проблема: отсутствие полноценных библиотек разреженной линейной алгебры для высокопроизводительного анализа графов с переносимой поддержкой ускорения вычислений на GPU

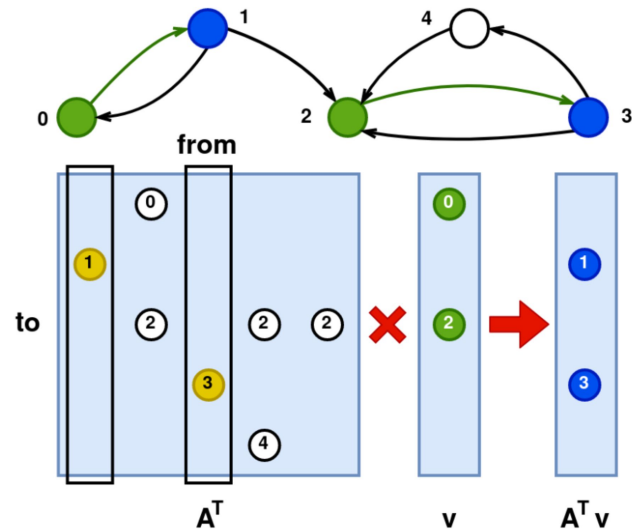


Рисунок: Обход графа с использованием произведения матрицы на вектор.

Цель и задачи

Целью данной работы – реализация библиотеки примитивов обобщенной разреженной линейной алгебры с переносимой, высокопроизводительной и не специфичной для производителя поддержкой вычислений на видеокарте.

Для достижения цели были поставлены следующие задачи.

- Провести обзор предметной области.
- Разработать архитектуру библиотеки.
- Реализовать библиотеку в соответствии с разработанной архитектурой.
- Провести начальное исследование производительности полученных артефактов.

Существующие решения

Решения для анализа графов:

- SuiteSparse GraphBLAS и LaGraph, Huawei GraphBLAS, IBM GraphBLAS
- GraphBLAST, GBTL
- Gunrock, CuSha, MapGraph, Medusa

Математические библиотеки разреженной линейной алгебры:

- cuSPARSE
- clSPARSE
- bhSPARSE
- Cusp

Предлагаемое решение

Проект Spla – библиотека примитивов разреженной линейной алгебры.

Принципы разработки:

- Опциональность ускорения вычислений.
- Пользовательские поэлементные функции.
- Предопределенные скалярные типы данных.
- Гибридный формат хранения данных.
- Экспортируемый интерфейс.
- Интроспекция.

Архитектура библиотеки

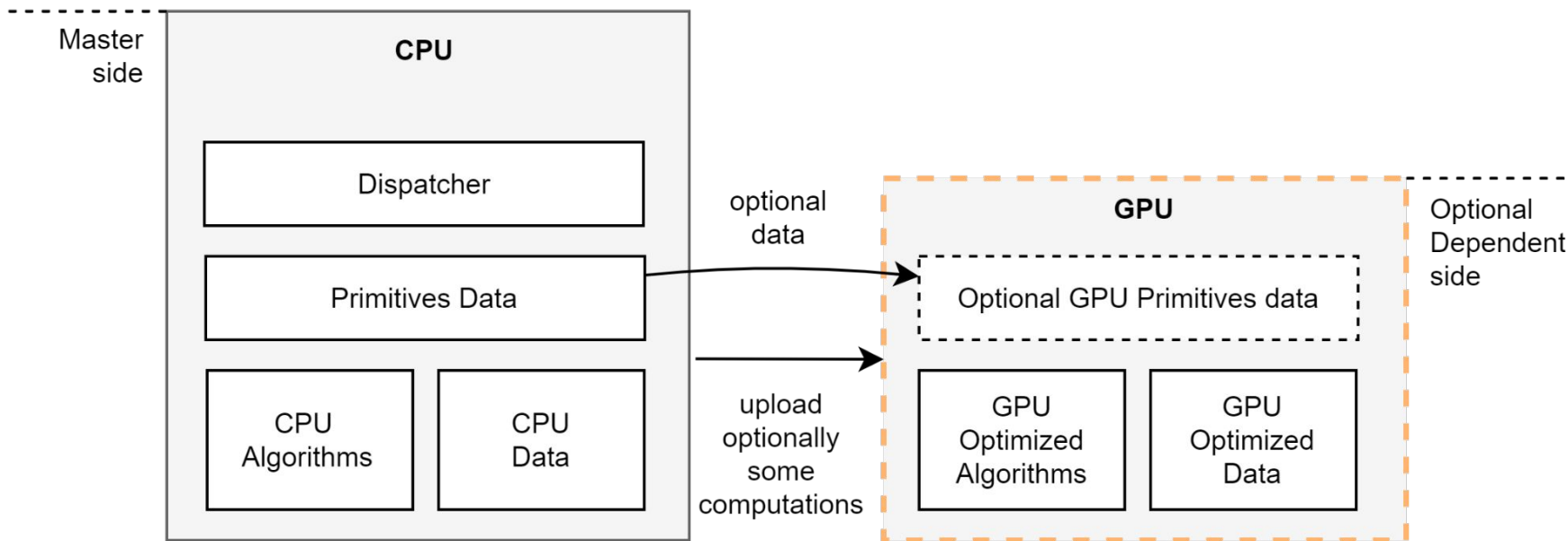


Рисунок: Высокоуровневый дизайн библиотеки Spla. Сторона библиотеки на CPU контролирует общую работу и способна функционировать самостоятельно. Вычисления на GPU опциональны. Алгоритмы и хранилище данных опциональны.

Детали реализации

- **Технологии**
 - C++17, CMake, OpenCL 1.2
- **Форматы хранения данных**
 - Векторные: DoK, Dense (CPU/GPU), COO (CPU/GPU)
 - Матричные: DoK, LiL, CSR (CPU/GPU)
- **Операции линейной алгебры**
 - Masked SpV x SpM
 - Masked SpM x V
 - Masked SpM x SpMT
- **Алгоритмы на графах**
 - BFS, SSSP, PR, TC

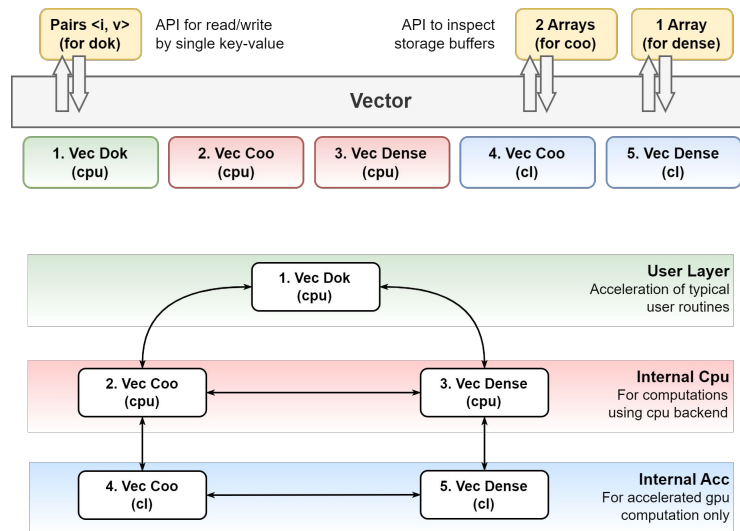


Рисунок: Гибридное хранилище данных на примере контейнера Вектор.

Пример использования

```
1 SPLA_API Status spla::bfs(const ref_ptr<Vector>& v,
2                           const ref_ptr<Matrix>& A,
3                           uint s,
4                           const ref_ptr<Descriptor>& desc) {
5     const auto N = v → get_n_rows();
6
7     ref_ptr<Vector> front_prev = make_vector(N, INT);
8     ref_ptr<Vector> front      = make_vector(N, INT);
9     ref_ptr<Scalar> front_size = make_int(1);
10    ref_ptr<Scalar> depth       = make_int(1);
11    ref_ptr<Scalar> zero        = make_int(0);
12    int current_level = 1;
13
14    front_prev → set_int(s, 1);
15
16    while (!(front_size → as_int() == 0)) {
17        depth → set_int(current_level);
18
19        exec_v_assign_masked(v, front_prev, depth, SECOND_INT, NQZERO_INT, desc);
20        exec_vxm_masked(front, v, front_prev, A, BAND_INT, BOR_INT, EQZERO_INT, zero, desc);
21        exec_v_reduce(front_size, zero, front, PLUS_INT, desc);
22
23        current_level += 1;
24
25        std::swap(front_prev, front);
26    }
27
28    return Status::Ok;
29 }
```

Фрагмент кода: Реализация алгоритма поиска в ширину (BFS) с использованием C++ API библиотеки Spla.

Экспериментальное исследование

Исследовательские вопросы:

- **B1:** Какова производительность разработанной библиотеки в сравнении с существующими решениями на GPU?
- **B2:** Какова производительность и каково масштабирование решения на GPU различных производителей?
- **B3:** Какова производительность решения на встроенных GPU в сравнении с существующими CPU инструментами?

Алгоритмы: BFS, SSSP, PR, TC

Инструменты: Gunrock, GraphBLAST, LaGraph (SuiteSparse), Spla

Набор данных

Graph	Vertices	Edges	Out Degree		
			Avg	Sd	Max
coAuthorsCit	227.3K	1.6M	7.2	10.6	1.4K
coPapersDBLP	540.5K	30.5M	56.4	66.2	3.3K
amazon2008	735.3K	7.0M	9.6	7.6	1.1K
hollywood2009	1.1M	112.8M	98.9	271.9	11.5K
comOrkut	3.1M	234.4M	76.3	154.8	33.3K
citPatents	3.8M	33.0M	8.8	10.5	793.0
socLiveJournal	4.8M	85.7M	17.7	52.0	20.3K
indochina2004	7.4M	302.0M	40.7	329.6	256.4K
belgiumosm	1.4M	3.1M	2.2	0.5	10.0
roadNetCA	2.0M	5.5M	2.8	1.0	12.0
rggn222s0	4.2M	60.7M	14.5	3.8	36.0
rggn223s0	8.4M	127.0M	15.1	3.9	40.0
roadcentral	14.1M	33.9M	2.4	0.9	8.0

Таблица: Набор данных для эксперимента. Тринадцать матриц графов из коллекции разреженных матриц университета Флориды.

В1: Сравнение производительности

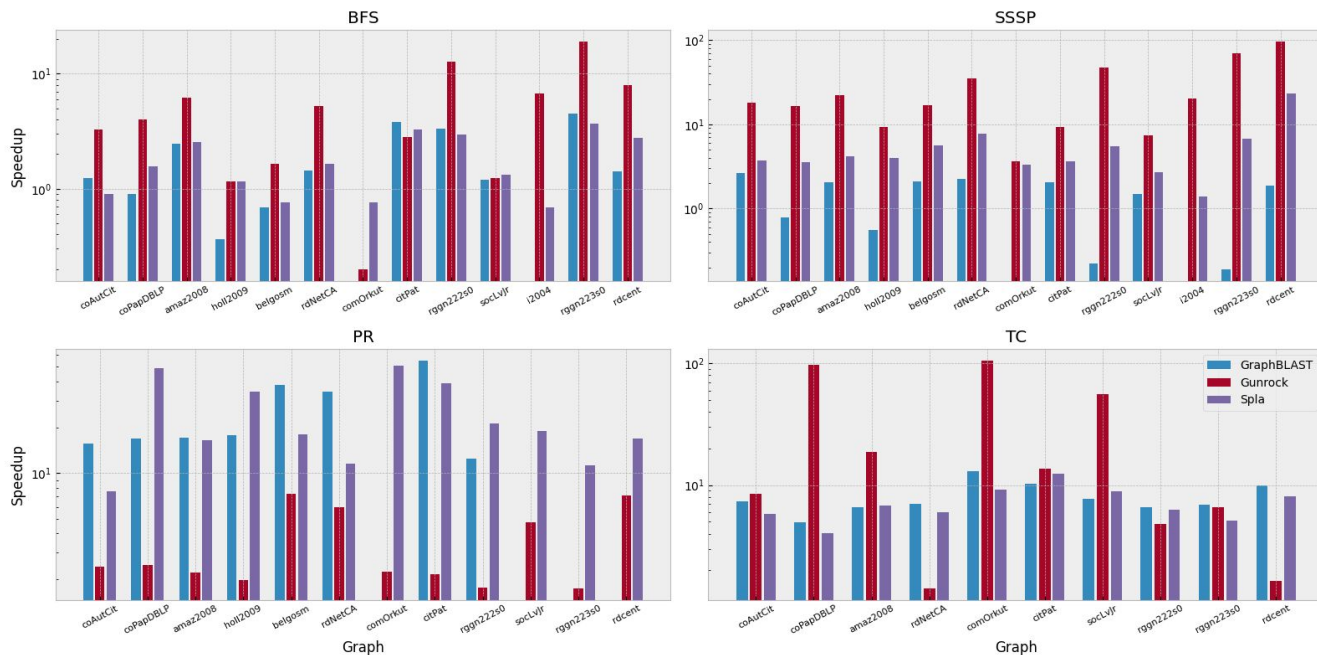


График: Сравнение производительности GraphBLAST, Gunrock, Spla относительно LaGraph (SuiteSparse). Логарифмический масштаб. Конфигурация: Ubuntu 20.04, Core i7-6700 4-core, DDR4 64Gb, GeForce GTX 1070, 8Gb VRAM.

B2: Масштабируемость и переносимость

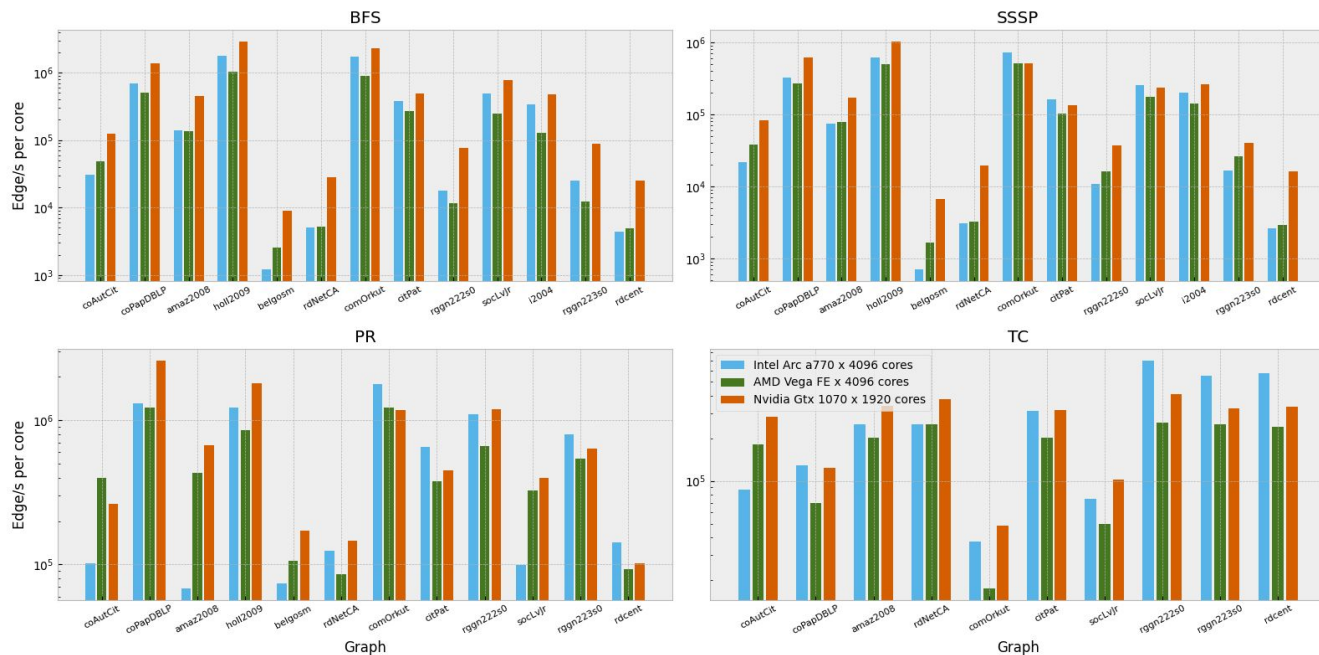


График: Производительность Spla библиотеки показанная как число ребер/сек. на ядро GPU. Логарифмический масштаб. Конфигурация: GeForce GTX 1070 with 8Gb VRAM, Arc A770 with 8GB VRAM and Radeon Vega FE with 16GB VRAM.

ВЗ: Производительность на встроенных видеокартах

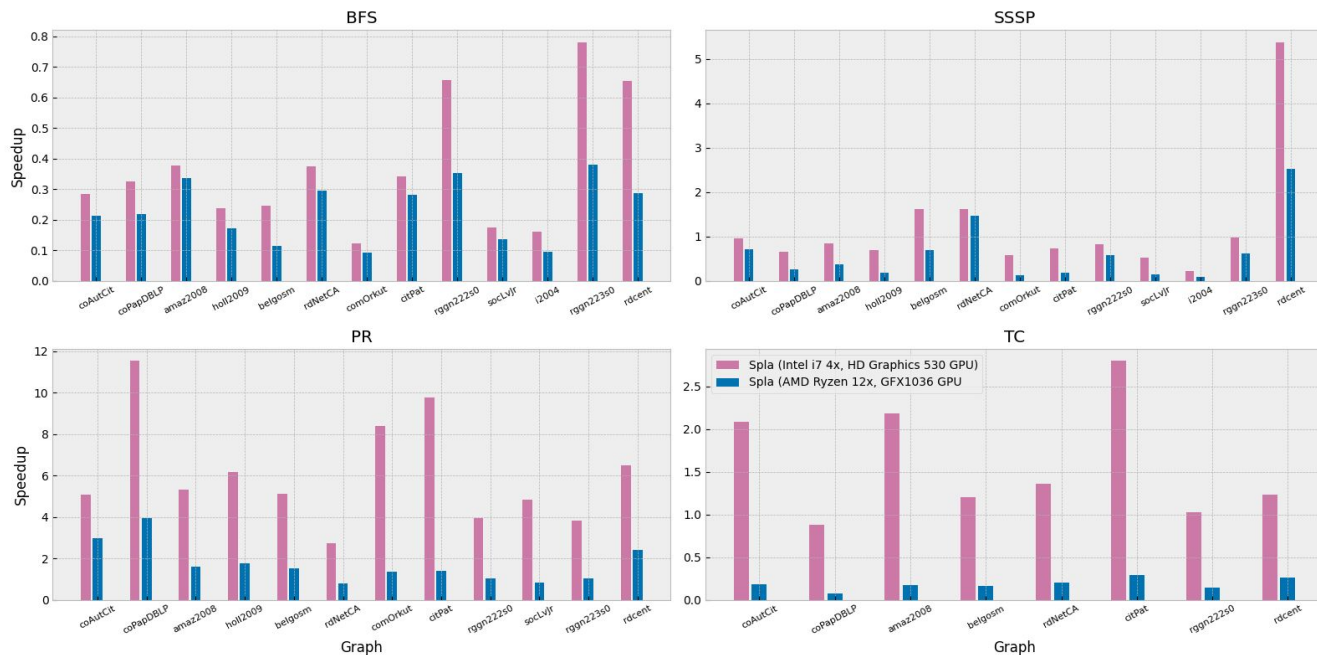


График: Ускорение Spla библиотеки на интегрированной GPU относительно LaGraph (SuiteSparse) на CPU. Конфигурация: Ubuntu 20.04, Core i7-6700, DDR4 64Gb, HD Graphics 530 GPU and Ubuntu 22.04, Ryzen 9 7900x, DDR4 128 GB, GFX1036 GPU.

Заключение

В ходе данной работы были получены следующие результаты.

- Проведен обзор предметной области.
- Разработана архитектура библиотеки примитивов разреженной линейной алгебры с поддержкой вычислений на GPU.
- Реализована библиотека в соответствии с разработанной архитектурой.
- Выполнено начальное исследование производительности полученных артефактов.

Исходный код проекта опубликован на платформе GitHub. Он доступен по ссылке github.com/SparseLinearAlgebra/spla.