Обобщённая разреженная линейная алгебра и высокопроизводительный анализ графов: проблемы и перспективы

Graph Analytics Club

Семён Григорьев

Санкт-Петербургский Государственный Университет

10 сентября 2025

Семён Григорьев

- Доцент кафедры системного программирования Санкт-Петербургского Государственного Университета
- Руководитель исследовательской группы
- Области интересов
  - **Высокопроизводительная линейная алгебра** для анализа графов
    - ⋆ **Обобщённая**: матрицы и вектора параметризованы типом элемента, операции над ними могут быть заданы пользователем
    - ⋆ **Разреженная**: специализированные структуры для хранения матриц и векторов, специализированные алгоритмы для их обработки
    - ⋆ В том числе, с использованием **графических ускорителей**
  - **Высокопроизводительный анализ графов**

- Email: s.v.grigoriev@mail.spbu.ru
- GitHub: gsvgit
- Google Scholar: Semyon Grigorev
- DBLP: Semyon V. Grigorev

Как создать «достаточно универсальный» фреймворк для разработки высокопроизводительных (параллельных) алгоритмов анализа графов?

GraphBLAS[5]

- API для создания алгоритмов анализа графов на основе линейной алгебры
  - Различные операции над матрицами и векторами (**разреженными**)
  - Параметризация алгебраическими структурами: полукольцами, моноидами и т.д.
- Позволяет выражать **различные** алгоритмы
  - Обход в ширину, поиск кратчайших путей, достижимость, ...
  - Подсчёт треугольников, PageRank, остовные деревья, кластеризация, ...
  - Навигационные запросы: **RPQ, CFPQ,** ...
- Подробнее
  - The GraphBLAS C API Specification[1]
  - GraphBLAS Pointers[2]
  - Introduction to GraphBLAS[3]
  - LAGraph[4]

[1] https://graphblas.org/docs/GraphBLAS_API_C_v2.1.0.pdf
[2] https://graphblas.org/GraphBLAS-Pointers/
[3] https://zenodo.org/record/4318870/files/graphblas-introduction.pdf
[4] https://github.com/GraphBLAS/LAGraph
[5] https://graphblas.org/

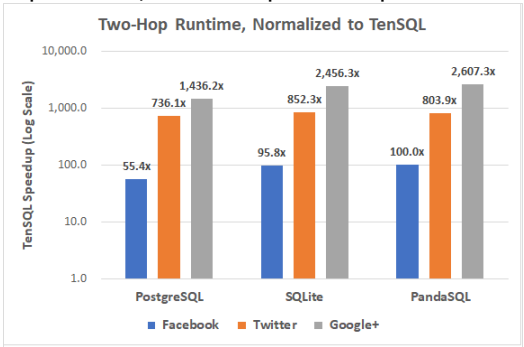
Реализации GraphBLAS-подобных API

- **SuiteSparse:GraphBLAS**[6] — **эталон** на чистом C[7] (Intel, Nvidia)
- Huawei's GraphBLAS[8] — частичная реализация на C++
- CombBLAS[9] — распределённая, частичная реализация на C++
- GraphBLAST[10] — поддержка GPGPU, Cuda C, частичная реализация
- **Spla**[11] — поддержка GPGPU, OpenCL C, частичная реализация
- GraphLily[12] — подмножество GraphBLAS на FPGA
- Обёртки для различных языков: Python, Rust, ...

[6] https://github.com/DrTimothyAldenDavis/GraphBLAS
[7] Ведётся работа над использованием GPGPU через Cuda
[8] https://gitee.com/CSL-ALP/graphblas
[9] https://github.com/PASSIONLab/CombBLAS
[10] https://github.com/gunrock/graphblast
[11] https://github.com/SparseLinearAlgebra/spla
[12] GraphLily: Accelerating Graph Linear Algebra on HBM-Equipped FPGAs

Использование

- FalkorDB (ex RedisGraph)[13] — графовая БД, основанная на SuiteSparse:GraphBLAS
- OneSparse[14] — расширение PostgreSQL, позволяющее использовать разреженную линейную алгебру (для обработки графов)
- NetworkX[15] — SuiteSparse:GraphBLAS для реализации некоторых алгоритмов анализа графов
- TenSQL[16,17] — SQL + GraphBLAS



[13] https://www.falkordb.com/
[14] https://onesparse.com/
[15] https://networkx.org/
[16] https://github.com/sandialabs/TenSQL
[17] TenSQL: An SQL Database Built on GraphBLAS

И всё было бы хорошо, но ...

Проблемы дизайна GraphBLAS[19]

- «Естественная» выразимость алгоритмов
  - BFS vs DFS[18]
- Необходимы специфичные оптимизации
  - Kernel Fusion для разреженных данных
  - В целом, нерегулярный параллелизм — это тяжело
- Неявные нули
- «Громоздкость» из-за ручных оптимизаций и необходимости тонких настроек
- (Не)универсальность
- ...

[18] Linear Algebraic Depth-First Search

[19] What did GraphBLAS Get Wrong?, John Gilbert, UC Santa Barbara, GraphBLAS BoF at HPEC, September 2022

Kernel Fusion

$$\underbrace{M_1 + M_2}_{M_4} + M_3 \rightsquigarrow add(add(M_1, M_2), M_3)$$

- ✔ Stream Fusion — для «одномерных» данных
- ✔ XLA — для плотных данных
- ⚙ MLIR[20]
- ? Для разреженных вычислений в общем виде
  - ☹ Для обобщённых вычислений
  - ☹ Для GPGPU и других специализированных ускорителей

[20] Например, mlir-graphblas

Неявные нули[21,22]

- Разреженный матрицы и вектора — не храним «нули»
- C API — сложно внятно описать на уровне типов, кто такие эти «нули»
  - Часто появляются выделенные значения («давайте считать, что `0:Int` не храним»)
  - Пользовательские функции: не достаточно чёткие типы, сложно обрабатывать крайние случаи
  - Переход между доменами: в одном домене выделенное значение не храним, а в другом должны хранить, потому что это «значимый» элемент, а выделенное значение другое

```
add (x:Int, y:Int):Int = x + y
```

Можем получить 0. Должны сохранить?

[21] https://github.com/lessthanoptimal/ejml/pull/145#issuecomment-888293732
[22] https://github.com/GraphBLAS/LAGraph/issues/28

«Громоздкость»

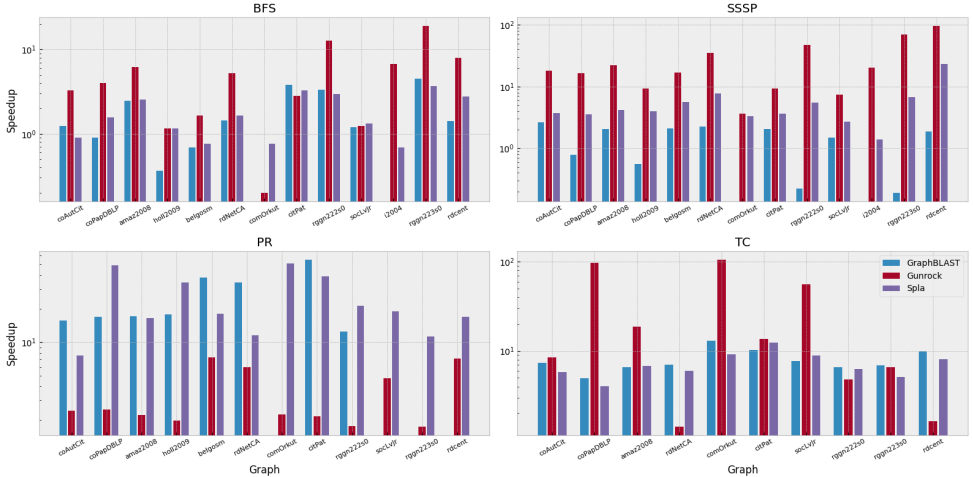
- Ручное слияние ядер (kernel fusion) — разрастается количество аргументов
  - Маска — аргумент умножения[23,24] и не только
  - Маска может быть инвертированной или нет
- Поэлементные операции
  - `ewise_add`, `ewise_mult`, mask, . . . [25]
- Дескрипторы — средство тонкой ручной настройки операций
- . . .

[23] $mask(M_1 * M_2, M_3) \rightsquigarrow mult(M_1, M_2, M_3)$

[24] Да, как `multiply_add`

[25] Почему не `map2` или что-то аналогичное? Отчасти, потому что проблема с нулями.

(Не)универсальность



- Ускорение относительно LAGraph
- Графы из SuiteSparse Matrix Collection
- GraphBLAST и Spla — GraphBLAS-подобные
- Gunrock — BSP-like подход на основе BFS

Выводы

- Обобщённая разреженная линейная алгебра — вполне рабочий инструмент высокопроизводительного анализа графов
  - FalkorDB, NetworkX, LAGraph, OneSparse, ...
- Пока хорошо работает только на одном узле, in memory, многоядерные CPU
  - SuiteSparse: GraphBLAS
- Не лишён недостатков
  - Если не пользуетесь готовыми прикладными алгоритмами, то думать о графах в терминах линейной алгебры — отдельный навык
  - Придумывать специальные полукольца и моноиды — нетривиальное занятие

Направления развития

- Использование (специализированных) ускорителей
  - В SuiteSparse:GraphBLAS ведётся работа по реализации некоторых ядер на Cuda
  - Spla использует GPGPU через OpenCL
- Разработка специализированных ускорителей для разреженной линейной алгебры
  - Расширения для RISC-V
  - Более «экзотические» решения[26]
- Больше прикладных решений: анализ сетевого трафика[27], анализ кода[28], ...
- Усовершенствование API
  - Ведётся работа над C++ API[29]
  - Попытки применить идеи из функционального программирования[30]

[26] Dedicated Hardware Accelerators for Processing of Sparse Matrices and Vectors: A Survey

[27] Deployment of Real-Time Network Traffic Analysis Using GraphBLAS Hypersparse Matrices and D4M Associative Arrays

[28] Universal High-Performance CFL-Reachability via Matrix Multiplication

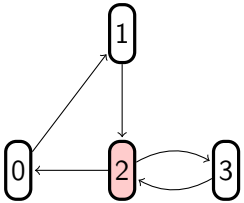
[29] GraphBLAS C++ API Specification v1.0

[30] Алгебраические типы для работы «нулями»

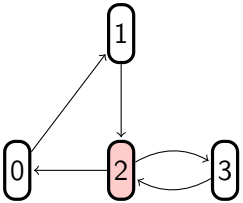
Графы для экспериментов

Name	Vertices	Edges	Avg Deg	Sd Deg	Max Deg
coAuthorsCiteseer	227.3K	1.6M	7.2	10.6	1372.0
coPapersDBLP	540.5K	30.5M	56.4	66.2	3299.0
amazon-2008	735.3K	7.0M	9.6	7.6	1077.0
hollywood-2009	1.1M	112.8M	98.9	271.9	11467.0
belgium_osm	1.4M	3.1M	2.2	0.5	10.0
roadNet-CA	2.0M	5.5M	2.8	1.0	12.0
com-Orkut	3.1M	234.4M	76.3	154.8	33313.0
cit-Patents	3.8M	33.0M	8.8	10.5	793.0
rgg_n_2_22_s0	4.2M	60.7M	14.5	3.8	36.0
soc-LiveJournal	4.8M	85.7M	17.7	52.0	20333.0
indochina-2004	7.4M	302.0M	40.7	329.6	256425.0
rgg_n_2_23_s0	8.4M	127.0M	15.1	3.9	40.0
road_central	14.1M	33.9M	2.4	0.9	8.0

Пример: обход в ширину



Пример: обход в ширину



Текущий фронт

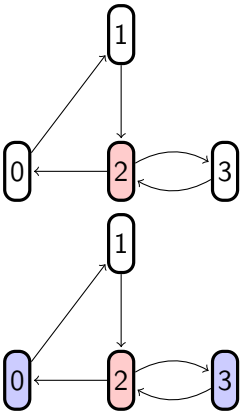
Матрица смежности

Новый фронт

$$\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$$

Полукольцо

Пример: обход в ширину



Текущий фронт

Матрица смежности

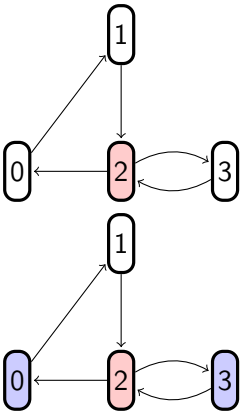
Новый фронт

$$\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$$

Полукольцо

$$\begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$$

Пример: обход в ширину



Текущий фронт

Матрица смежности

Новый фронт

$$\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}$$

Полукольцо

$$\begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}$$