Обобщённая разреженная линейная алгебра и высокопроизводительный анализ графов в экосистеме RISC-V

Семён Григорьев

Санкт-Петербургский Государственный Университет

30 сентября 2025г.

Линейная алгебра и анализ графов

- **Анализ больших графов**: графовые БД, анализ кода, поиск уязвимостей, анализ трафика, анализ транзакций, банковская аналитика, социальные сети...
  - Важна производительность
  - **Разнообразные алгоритмы**
- Путь к унифицированной параллельной обработке графов
  - Граф $\iff$ **матрица** смежности
  - Метки на рёбрах $\iff$ **полукольца**, моноиды, ...
  - Линейная алгебра $\iff$ **параллелизм** по данным
- **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**, **ПЛИС**

GraphBLAS API[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,...
  - Запросы с регулярными (RPQ) и контекстно-свободными (CFPQ) ограничениями ...
- Подробнее
  - The GraphBLAS C API Specification[1]
  - GraphBLAS Pointers[2]
  - Introduction to GraphBLAS[3]
  - LAGraph[4]

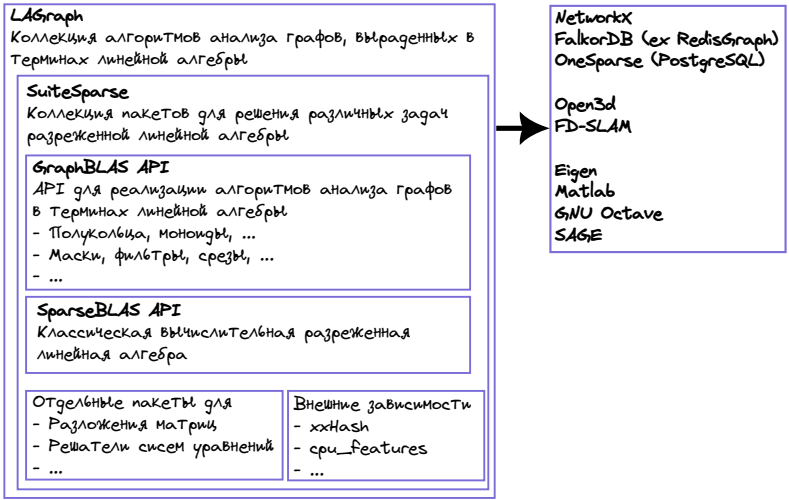
[1] https://graphblas.org/docs/GraphBLAS_API_C_v2.1.0.pdf
[2] https://graphblas.org/GraphBLAS-Pointers/
[3] https://zenodo.org/record/4318870/files/graphblas-introduction.pdf
[4] https://github.com/GraphBLAS/LAGraph
[5] https://graphblas.org/

Реализации GraphBLAS-подобных API

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

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

SuiteStarse[12]



[12] https://github.com/DrTimothyAldenDavis/SuiteSparse

Векторизация умножения матриц в SuiteSparse:GraphBLAS[13]

- Оборудование
  - X86_64
    - ★ **CPU**: Intel Core i7-12700H 800MHz с векторами размером 1024 битов
    - ★ **RAM**: LPDDR4, 16GB
    - ★ **Compiler**: GCC 14.2.0
  - RISC-V
    - ★ **SoC**: SPACEMIT K1/M1, Octa-core X60™(RV64GCVB), RVA22, RVV1.0 1600MHz с векторами размером 2048 битов
    - ★ **RAM**: LPDDR4X, 16GB
    - ★ **Compiler**: GCC 14.2.0 (cross)
- SuiteSparse matrix collection: матрицы разных размеров и разной степени разреженности
- Сравнивали изменение величины среднего времени выполнения 400 запусков умножения матриц

[13] Соответствующий PR: https://github.com/DrTimothyAldenDavis/GraphBLAS/pull/381

Результаты экспериментального исследования векторизованного кода[14]

№	Matrix name	Rows number	Nonzeros	AVX2 (ms.)	No AVX2 (ms.)	RVV (ms.)	No RVV (ms.)	AVX speedup (%)	RVV speedup (%)
1	olafu	16146	515651	5327.7	6629.7	43080.7	52940.1	19.6	18.6
2	fd18	16428	63406	476.4	482.0	2212.6	2181.2	1.2	-1.4
3	sme3Da	12504	874887	4236.9	5124.9	32008.0	42763.8	17.3	25.2
4	stokes64	12546	74242	508.8	564.4	2629.7	2814.1	9.8	6.6
5	sinc12	7500	294986	632.6	864.0	5970.1	8593.8	26.8	30.5
6	fd12	7500	28462	90.4	92.3	484.1	555.3	2.0	12.8
7	bcsstk15	3948	60882	87.8	117.9	1271.5	1770.8	25.6	28.2
8	tols4000	4000	8784	17.1	18.2	184.0	203.5	5.9	9.6
9	ex36	3079	53843	28.5	41.0	574.2	584.8	30.5	1.8
10	iprob	3001	9000	25.2	34.7	279.3	344.9	27.5	19.0
11	MISKnowledgeMap	2427	28511	31.3	38.5	401.6	490.0	18.8	18.0
12	LeGresley_2508	2508	16727	10.4	12.1	106.5	97.7	14.3	-8.9
13	reorientation_2	1544	9408	5.6	9.8	117.9	125.4	42.7	6.0
14	netscience	1589	2742	1.5	2.8	31.4	28.5	47.0	-10.0
15	mcfe	765	24382	2.3	5.5	51.1	65.3	58.8	21.8
16	orbitRaising_3	761	3256	0.6	1.6	10.5	13.1	63.0	19.5

[14] Во всех экспериментах стандартное отклонение не превосходит 5%

Результаты экспериментального исследования векторизованного кода[14]

№	Matrix name	Rows number	Nonzeros	AVX2 (ms.)	No AVX2 (ms.)	RVV (ms.)	No RVV (ms.)	AVX speedup (%)	RVV speedup (%)
1	olafu	16146	515651	5327.7	6629.7	43080.7	52940.1	19.6	18.6
2	fd18	16428	63406	476.4	482.0	2212.6	2181.2	1.2	-1.4
3	sme3Da	12504	874887	4236.9	5124.9	32008.0	42763.8	17.3	25.2
4	stokes64	12546	74242	508.8	564.4	2629.7	2814.1	9.8	6.6
5	sinc12	7500	294986	632.6	864.0	5970.1	8593.8	26.8	30.5
6	fd12	7500	28462	90.4	92.3	484.1	555.3	2.0	12.8
7	bcsstk15	3948	60882	87.8	117.9	1271.5	1770.8	25.6	28.2
8	tols4000	4000	8784	17.1	18.2	184.0	203.5	5.9	9.6
9	ex36	3079	53843	28.5	41.0	574.2	584.8	30.5	1.8
10	iprob	3001	9000	25.2	34.7	279.3	344.9	27.5	19.0
11	MISKnowledgeMap	2427	28511	31.3	38.5	401.6	490.0	18.8	18.0
12	LeGresley_2508	2508	16727	10.4	12.1	106.5	97.7	14.3	-8.9
13	reorientation_2	1544	9408	5.6	9.8	117.9	125.4	42.7	6.0
14	netscience	1589	2742	1.5	2.8	31.4	28.5	47.0	-10.0
15	mcfe	765	24382	2.3	5.5	51.1	65.3	58.8	21.8
16	orbitRaising_3	761	3256	0.6	1.6	10.5	13.1	63.0	19.5

[14] Во всех экспериментах стандартное отклонение не превосходит 5%

Результаты экспериментального исследования векторизованного кода[14]

№	Matrix name	Rows number	Nonzeros	AVX2 (ms.)	No AVX2 (ms.)	RVV (ms.)	No RVV (ms.)	AVX speedup (%)	RVV speedup (%)
1	olafu	16146	515651	5327.7	6629.7	43080.7	52940.1	19.6	18.6
2	fd18	16428	63406	476.4	482.0	2212.6	2181.2	1.2	-1.4
3	sme3Da	12504	874887	4236.9	5124.9	32008.0	42763.8	17.3	25.2
4	stokes64	12546	74242	508.8	564.4	2629.7	2814.1	9.8	6.6
5	sinc12	7500	294986	632.6	864.0	5970.1	8593.8	26.8	30.5
6	fd12	7500	28462	90.4	92.3	484.1	555.3	2.0	12.8
7	bcsstk15	3948	60882	87.8	117.9	1271.5	1770.8	25.6	28.2
8	tols4000	4000	8784	17.1	18.2	184.0	203.5	5.9	9.6
9	ex36	3079	53843	28.5	41.0	574.2	584.8	30.5	1.8
10	iprob	3001	9000	25.2	34.7	279.3	344.9	27.5	19.0
11	MISKnowledgeMap	2427	28511	31.3	38.5	401.6	490.0	18.8	18.0
12	LeGresley_2508	2508	16727	10.4	12.1	106.5	97.7	14.3	-8.9
13	reorientation_2	1544	9408	5.6	9.8	117.9	125.4	42.7	6.0
14	netscience	1589	2742	1.5	2.8	31.4	28.5	47.0	-10.0
15	mcfe	765	24382	2.3	5.5	51.1	65.3	58.8	21.8
16	orbitRaising_3	761	3256	0.6	1.6	10.5	13.1	63.0	19.5

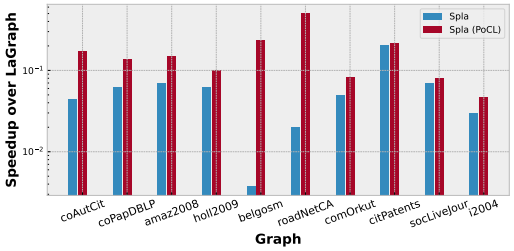
[14] Во всех экспериментах стандартное отклонение не превосходит 5%

RISC-V и GPGPU

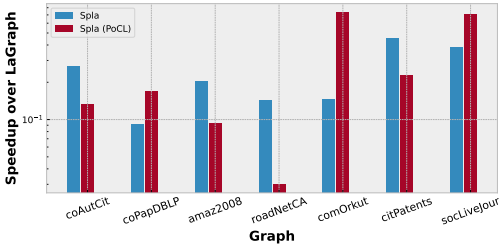
- **RISC-V CPU + Imagination Technologies GPU**: самая распространённая (из доступных) конфигурация
- **RISC-V CPU + AMD GPU**: есть официальная поддержка (Milk-V Pioneer, Milk-V Megrez, RuyiBook)
- **RISC-V CPU + Intel GPU**: ходят слухи, что можно, но официальной поддержки пока нет
- **RISC-V CPU + Nvidia GPU**: анонсировано
- **RISC-V GPU**: Vortex

Spla на SpacemiT M1 (RISC-V) с IMG BXE-2-32 GPU

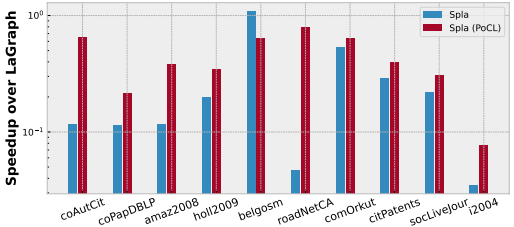
BFS



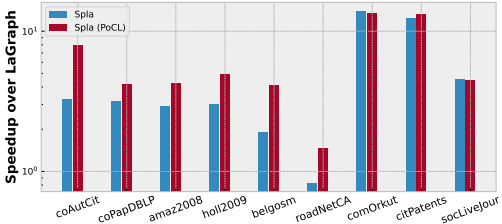
Triangle Count



Single Source Shortest Path



PageRank



Пара слов про Vortex: RISC-V GPGPU

- Набор инструкций, основанный на RISC-V ISA
- Поддержка OpenCL через POCL
  - Spla запускается (и даже что-то работает)

Пара слов про Vortex: RISC-V GPGPU

- Набор инструкций, основанный на RISC-V ISA
- Поддержка OpenCL через POCL
  - Spla запускается (и даже что-то работает)
- Проблемы со сбросом регистров
  - Типичные оптимизации не работают
  - Issue 1
  - Issue 2
- В целом, есть подозрение, что мало регистров
- Проблемы с поддержкой OpenCL
  - Атомарные операции
  - Некоторые функции работы с буферами (`clBuffer`)
  - ...
- Для ПЛИС с HBM
  - **?** Бонус для обработки граф-структурированных данных

Перспективы: RISC-V

- Идёт работа над расширениями[15]
  - IndexMAC: A Custom RISC-V Vector Instruction to Accelerate Structured-Sparse Matrix Multiplications, 2024 год
  - Optimizations for Very Long and Sparse Vector Operations on a RISC-V VPU: A Work-in-Progress, 2023 год
  - Optimizing Structured-Sparse Matrix Multiplication in RISC-V Vector Processors, 2025 год
  - Sparse Stream Semantic Registers: A Lightweight ISA Extension Accelerating General Sparse Linear Algebra, 2023 год
  - Hardware/Software Co-Design of RISC-V Extensions for Accelerating Sparse DNNs on FPGAs, 2024 год
- В основном для машинного обучения: малая разрядность, относительно большая плотность, фиксированный набор типов и операций (часто для инференса)
- Vortex: GPGPUs on FPGAs: A competitive approach for scientific computing?, 2025 год

[15] Оставим в покое RVV, Integrated Matrix Extension, XuanTie Matrix Extension, ...

Заключение

- Высокопроизводительная разреженная линейная алгебра ⇒ высокопроизводительные приложения
  - Машинное обучение
  - Графовые базы данных
  - Анализ социальных, банковских и других сетей
  - Анализ кода
  - ...
- Сделать **разреженную** линейную алгебру высокопроизводительной **сложно**
  - Нерегулярный доступ к данным
  - Хорошая алгебра — **обобщённая** алгебра
  - Сложности с балансировкой нагрузки
  - ...
- Есть попытки решить проблему
  - Вряд ли на основе RISC-V появится хоть сколько-то универсальное решение[16]

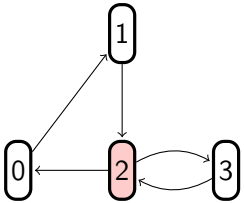
[16] И дело не в RISC-V

Семён Григорьев

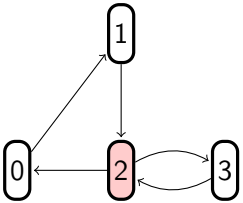
- Доцент кафедры системного программирования Санкт-Петербургского Государственного Университета
- Научный сотрудник лаборатории YADRO
- Руководитель исследовательской группы
- Области интересов
  - **Высокопроизводительная линейная алгебра** для анализа графов
    - **Обобщённая**: матрицы и вектора 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

Пример: обход в ширину



Пример: обход в ширину



Текущий фронт

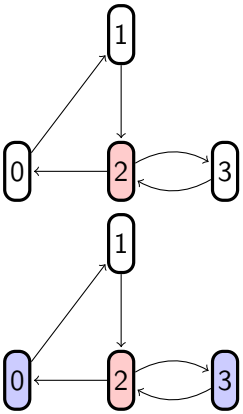
Матрица смежности

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

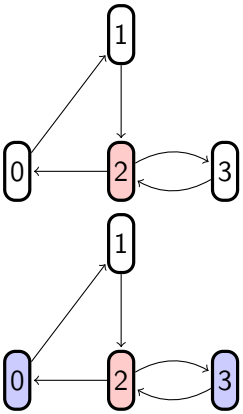
Новый фронт

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

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

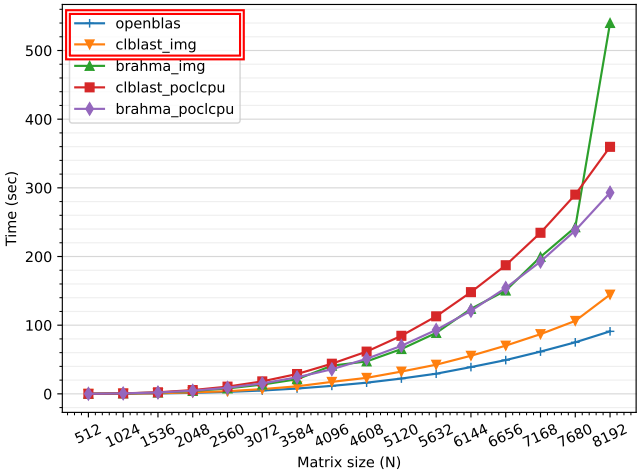
Новый фронт

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

Полукольцо

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

Результаты умножения плотных матриц на SpacemiT M1 с IMG GPU



(Некоторые) GPGPU от Imagination Technologies (пока) не совсем для вычислений

Специализированные решения для разреженной линейной алгебры

- Dedicated Hardware Accelerators for Processing of Sparse Matrices and Vectors: A Survey, 2024 год
- A Survey of Accelerating Parallel Sparse Linear Algebra, 2023 год
- A Systematic Literature Survey of Sparse Matrix-Vector Multiplication, 2024 год