Санкт-Петербургский государственный университет
Кафедра системного программирования

Теория графов

Хельмянов Артем, 22-Б11мм

Санкт-Петербург
2025

Постановка задачи: Подсчёт треугольников

Дан неориентированный граф $G = (V, E)$ без петель и кратных рёбер.
Задача: Найти количество треугольников в графе, то есть таких троек вершин (u, v, w), что между каждой парой вершин существует ребро: $(u, v) \in E$, $(v, w) \in E$, $(w, u) \in E$. Порядок вершин в тройке не важен.

Применения:

- Анализ социальных сетей
- Оценка кластеризации графа
- Поиск сообществ

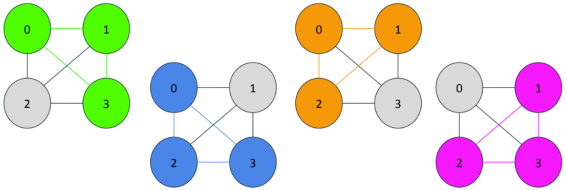
Базовый алгоритм

Пусть A — булева матрица смежности графа, тогда A^3 — все пути длины 3.
Тогда количество треугольников можно подсчитать по формуле:

$$T = \frac{1}{6} \cdot \text{trace}(A^3)$$

где $\text{trace}(X)$ — след матрицы (сумма всех элементов главной диагонали).
Откуда $\frac{1}{6}$? Каждый треугольник будет посчитан 6 раз: по одному разу для каждой из 3 вершин в качестве начальной, по и против часовой стрелки (граф неориентированный).

Пример



$$A^3 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}^3 = \begin{pmatrix} 6 & 7 & 7 & 7 \\ 7 & 6 & 7 & 7 \\ 7 & 7 & 6 & 7 \\ 7 & 7 & 7 & 6 \end{pmatrix}$$

Количество треугольников:

$$T = \frac{4 \cdot 6}{6} = 4$$

Алгоритм Burkhardt

Последнее умножение матриц можно заменить менее затратным поэлементным умножением (находим пути длины 2 и провеяем наличие ребра между началом и концом пути):

$$T = \frac{1}{6} \cdot \sum (A^2 \otimes A)$$

где $\otimes$ — поэлементное умножение матриц, а $\sum(X)$ — сумма всех элементов матрицы X.

$$A^2 \otimes A = \begin{pmatrix} 3 & 2 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 \\ 2 & 2 & 2 & 3 \end{pmatrix} \otimes \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 2 & 2 \\ 2 & 0 & 2 & 2 \\ 2 & 2 & 0 & 2 \\ 2 & 2 & 2 & 0 \end{pmatrix}$$

Алгоритм Sandia

Так как граф неориентированный, можно заменить матрицу смежности на её верхнюю (или нижнюю) треугольную часть U:

$$T = \sum (U^2 \otimes U)$$

Работаем с более разреженной матрицей, меньше вычислений.

Реализация

GraphBLAS:

```
GrB_mxm(squared, // output
        A, // mask
        nullptr,
        GxB_PLUS_TIMES_UINT64, // semiring
        A, A, // input
        nullptr);
GrB_Matrix_reduce_UINT64(...);
```

SPLA:

```
exec_mxmT_masked(B, // result
                 A, // mask
                 A, // left
                 A, // right
                 MULT_UINT, PLUS_UINT, GTZERO_UINT, zero);
exec_m_reduce(result, zero, B, PLUS_UINT);
```

Эксперимент

В качестве тестовых данных для эксперимента выбраны графы из набора Stanford Large Network Dataset Collection (`https://snap.stanford.edu/data/`).

Name	**Description**	**# nodes**	**# arcs**
musae-facebook	Facebook page-page network	22,470	171,002
com-Amazon	Amazon product network	334,863	925,872
amazon0601	Amazon product co-purchasing	403,394	3,387,388
roadNet-CA	Road network of California	1,965,206	2,766,607
com-LiveJournal	LiveJournal online social network	3,997,962	34,681,189

Основания выбора:

- Популярный набор данных для графовых алгоритмов
- Разные типы графов (социальные сети, дорожная сеть, сеть продуктов)
- Графы различных размеров и плотности

Эксперимент

В рамках эксперимента планируется провести сравнение производительности реализаций алгоритма Burkhardt и Sandia на библиотеках GraphBLAS и SPLA.

План эксперимента:

- Провести 10 запусков
- Собрать информацию о средних значениях
- Провести анализ полученных результатов

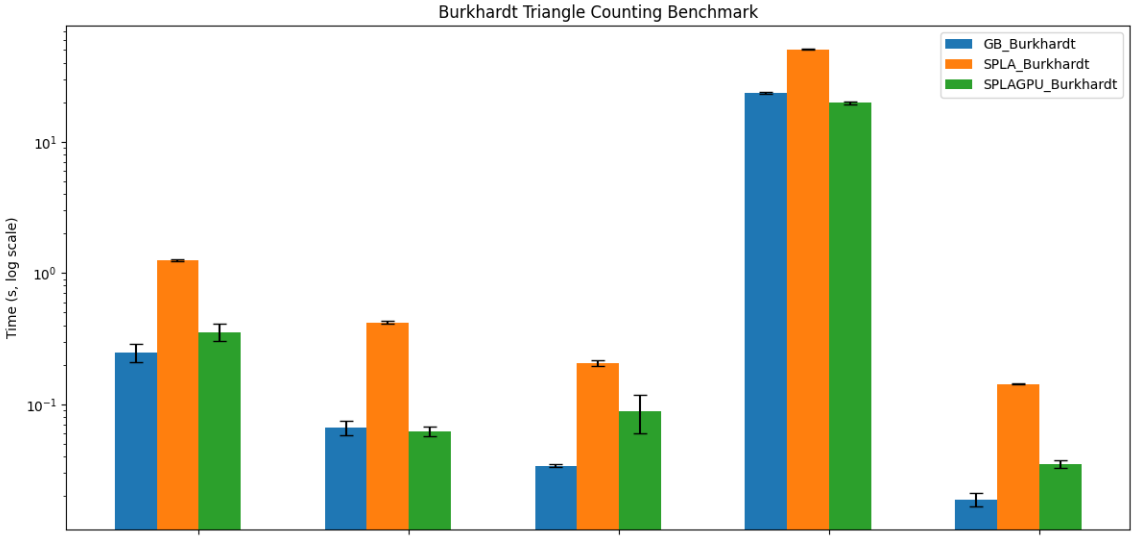
Гипотезы:

- Sandia будет быстрее Burkhardt.
- SPLA и GraphBLAS покажут сопоставимые результаты.

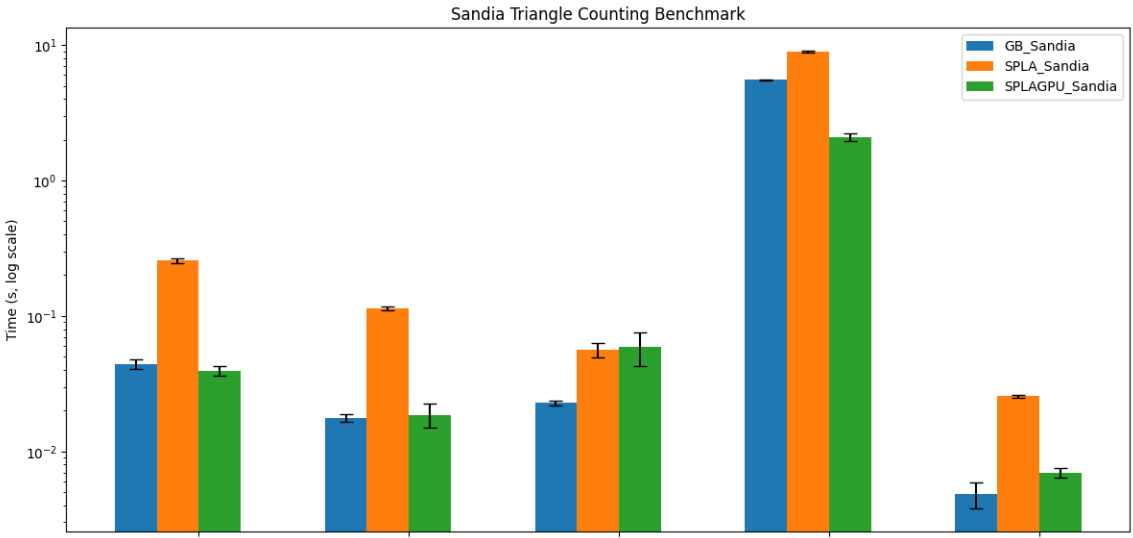
Характеристики машины для эксперимента

- CPU: AMD Ryzen 5 5500U, 6 cores, 12 threads, 4.0 GHz
- iGPU: Radeon RX Vega 7, 7 CUs, 1.8 GHz.
- RAM: 16 GB DDR4 3200 MHz
- OS: NixOS 25.11
- gcc 14.3.0, GraphBLAS 10.3.0, SPLA master @ b81d1e2

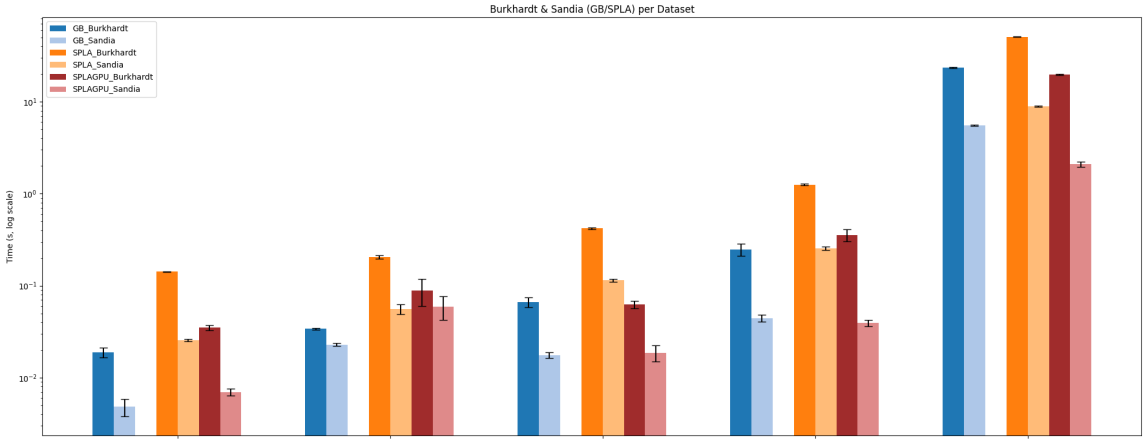
Результаты эксперимента (Burkhardt)



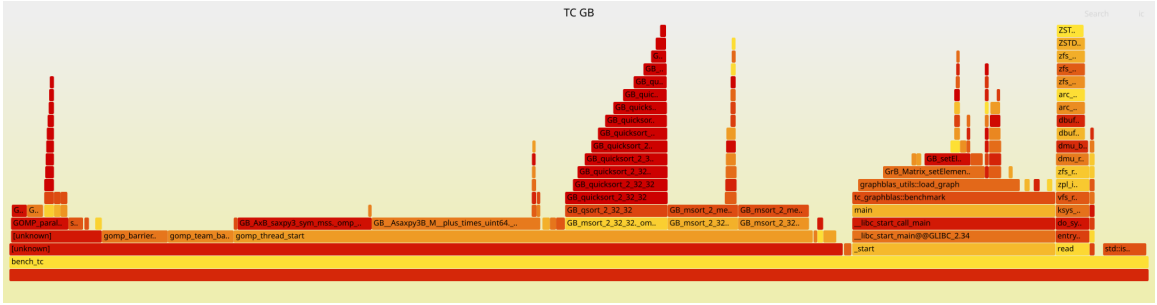
Результаты эксперимента (Sandia)



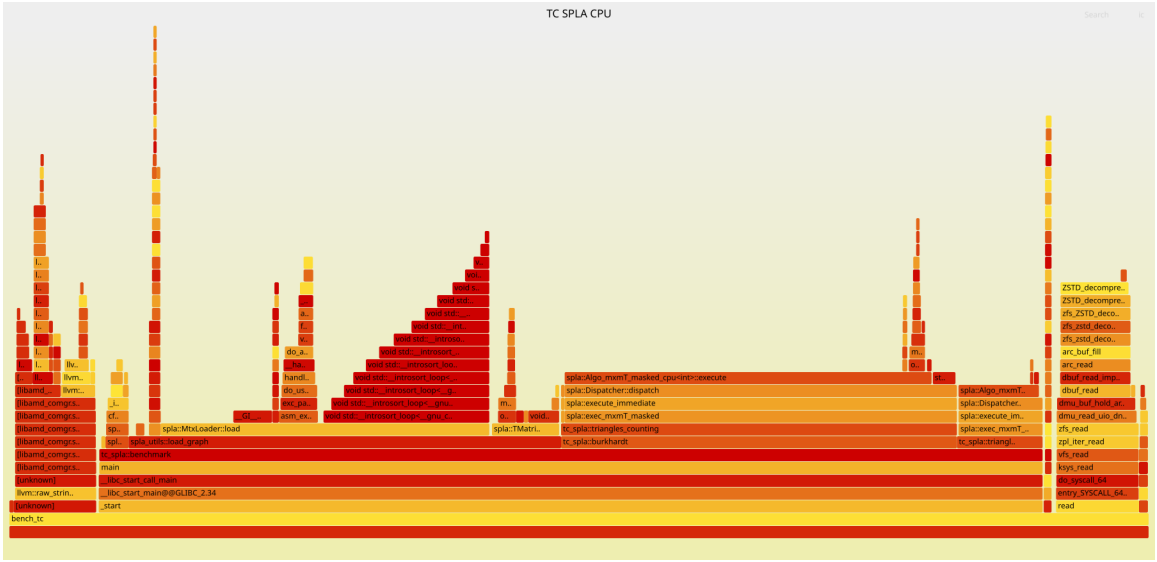
Результаты эксперимента



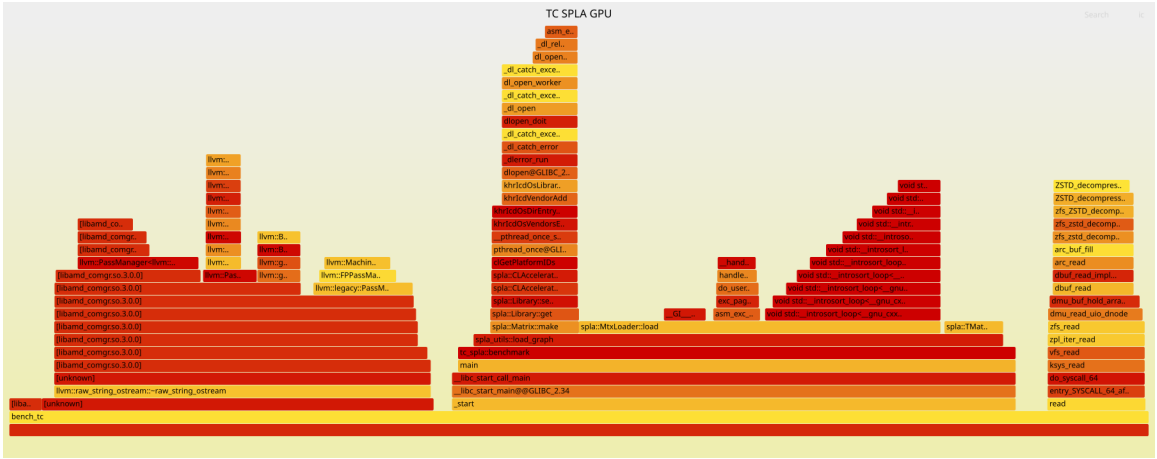
Результаты эксперимента



Результаты эксперимента



Результаты эксперимента



Выводы

- Алгоритм Sandia показывает лучшую производительность по сравнению с Burkhardt на всех наборах данных для каждой библиотеки.
- На небольших графах GraphBLAS показывает лучшую производительность по сравнению с SPLA GPU.
- С ростом размеров графа SPLA GPU начинает обгонять GraphBLAS

Multiple source parent BFS

Дан (не)ориентированный граф $G = (V, E)$ и множество стартовых вершин $S \subseteq V$.

Задача:
Для каждой вершины $v \in V$ и стартовой вершин $s \in S$ найти верщину u, такую что u является родителем v на кратчайшем пути от s до v.

В результате получаем матрицу родителей P $(n_{start} \times n_{rows})$, где $P_{i,j}$ — родитель вершины j на пути от стартовой вершины i.

Multiple source parent BFS

```
def msbfs(A, src):
    nsrc = len(src)
    nvert = A.nrows()

    front = Matrix(nsrc, nvert)
    visited = Matrix(nsrc, nvert)

    for i in range(nsrc):
        cursrc = src[i]
        front[i, cursrc] = 1
        visited[i, cursrc] = cursrc

    nfront = nsrc
    nvisited = nsrc
    while nvisited < nvert * nsrc:
        # Expand frontier:
        # mxm_secondI(front, A) performs frontier x A
        # !visited masks out nodes already visited
        front = mxm_secondI(front, A).mask(!visited)

        nfront = front.count() # number of newly discovered nodes
        if nfront == 0:
            break

        visited = eadd(visited, front)
        nvisited += nfront

    return visited
```

Эксперимент

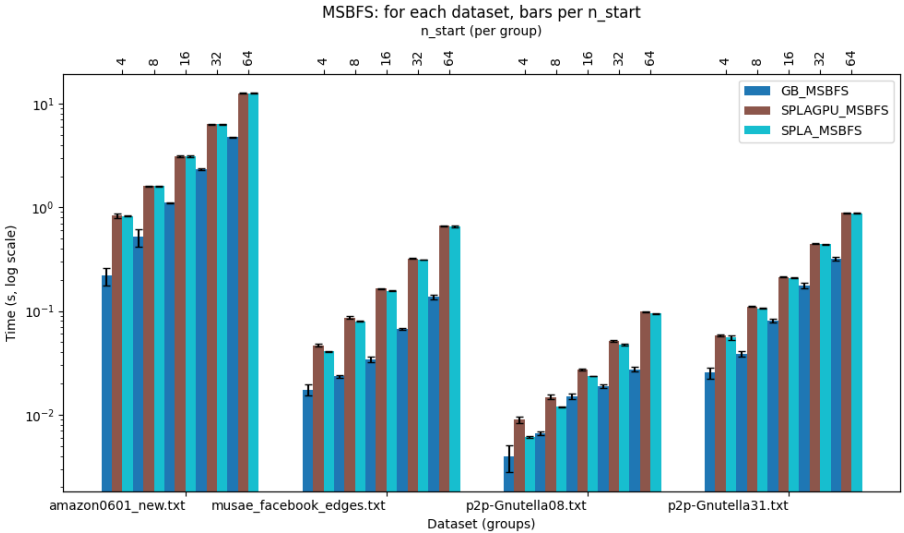
Stanford Large Network Dataset Collection (`https://snap.stanford.edu/data/`).

Name	**Description**	**# nodes**	**# arcs**
p2p-Gnutella08	Gnutella p2p network from 2002.08.08	6,301	20,777
p2p-Gnutella31	Gnutella p2p network from 2002.08.31	62,586	147,892
musae-facebook	Facebook page-page network	22,470	171,002
amazon0601	Amazon product co-purchasing	403,394	3,387,388

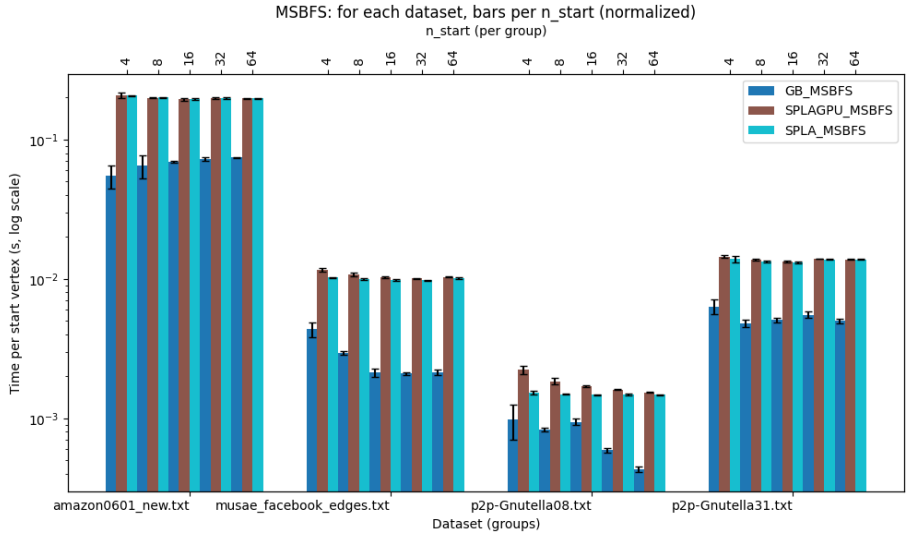
Эксперимент

- Реализовать алгоритмы на GraphBLAS и SPLA
- Провести 10 запусков с разным количеством стартовых вершин (2, 4, 8, ..., 64)
- Собрать информацию о средних значениях
- Провести анализ полученных результатов

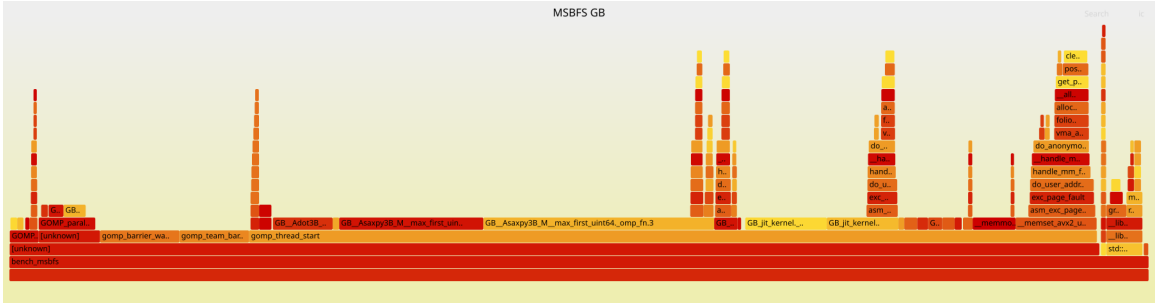
Результаты эксперимента



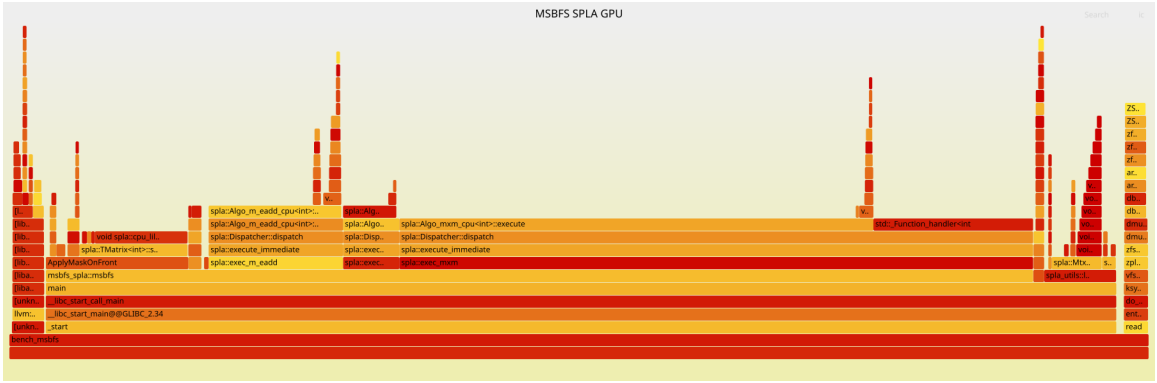
Результаты эксперимента



Флеймграф GraphBLAS



Флеймграф SPLA GPU



Результаты эксперимента

- GraphBLAS тратит половину времени на max_first_uint
- SPLA не поддерживает MxM и M_eadd на OpenCL
- При увеличении количества стартовых вершин время выполнения кратно растёт
- В некоторых случаях увеличение кол-ва стартовых вершин приводит к улучшению производительности