Команда №6

Сыресенков Илья, Нурмухаметов Рафик

Аппаратное обеспечение

- **12th Gen Intel i5-12500**
  - 16 ядер, 16 потоков
  - 2.5 Ггц (до 4.5 Ггц)
  - 18 Mb кэш-память 3 уровня
- **16 GB RAM**
  - DDR4
  - 3200 МГц

Программное обеспечение

- **ОС EndeavourOS Linux** (kernel 6.12.57-1-lts)
- **CMake** v4.1.2
- **gcc** v15.2.1
  - флаг оптимизации O3
- **intel-opencl-runtime** v2024.2.1-1
- **GraphBLAS** v10.1.0
- **LaGraph** v1.2.1
- **SPLA** fork (commit 74658a9, 09.04.2025)

Набор данных (BFS, Борувка)

Графы взяты из 9th DIMACS Implementation Challenge – Shortest Paths

Название	Кол-во вершин	Кол-во ребер
New York City	264 346	733 846
Florida	1 070 376	2 712 798
California and Nevada	1 890 815	4 657 742
Western USA	6 262 104	15 248 146
Full USA	23 947 347	58 333 344

Набор данных (PageRank)

Графы взяты из Stanford Large Network Dataset Collection

Название	Кол-во вершин	Кол-во ребер
Web-Stanford	281 903	2 312 497
Web-BerkStan	685 230	7 600 595
Web-Google	875 713	5 105 039
Wiki-Topcats	1 791 488	28 511 807
Sx-Stackoverflow	2 601 977	63 497 050

Ход экспериментов

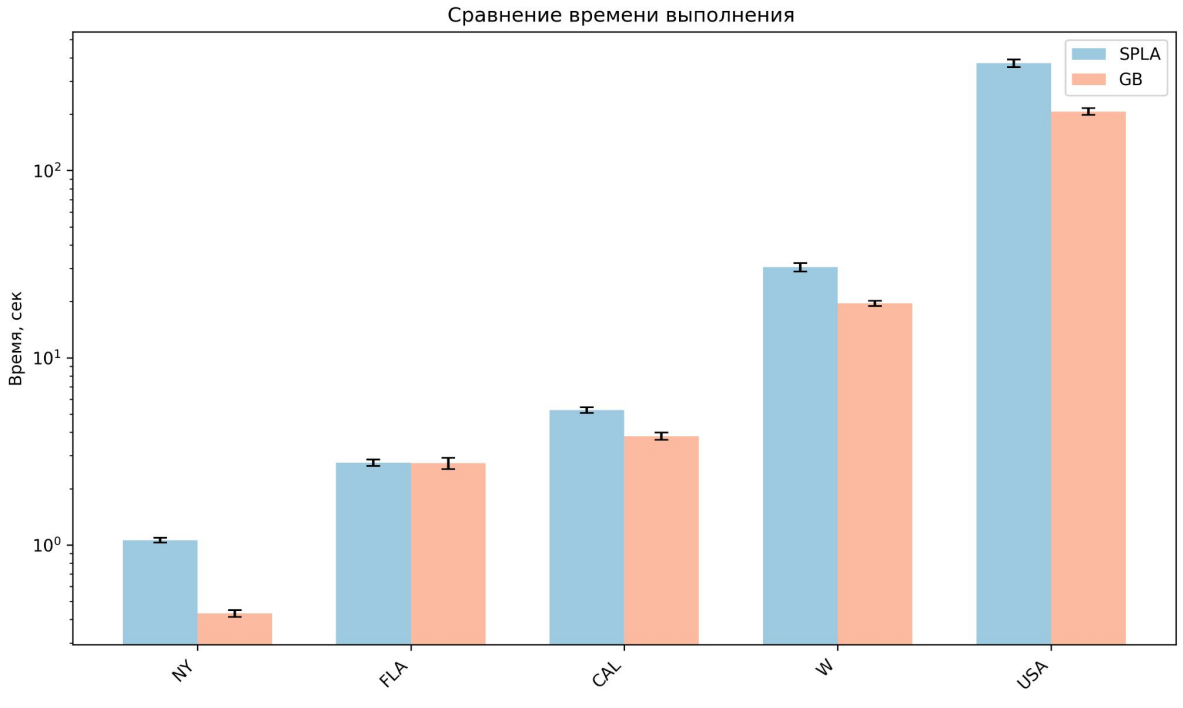
- Запуск алгоритмов в рамках каждого эксперимента 15 раз
- Сбор результатов каждого запуска
- Вычисление средних значений и доверительных интервалов
- Визуализация и анализ результатов

Эксперимент 1

Цель эксперимента: сравнить среднее время выполнения алгоритмов на **SuiteSparse::GraphBLAS** и **SPLA**

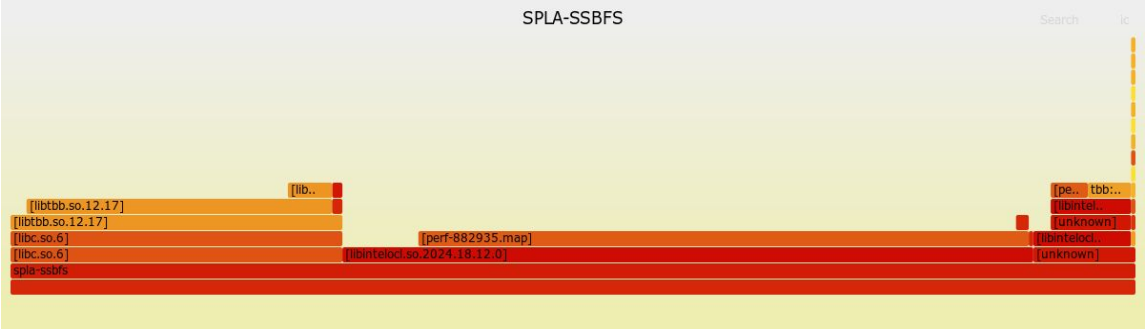
- Алгоритмы
  - **SSBFS**
  - **MSBFS** (количество стартовых вершин – 16)
  - **Boruvka**
  - **PageRank**

Эксперимент 1 (SSBFS)



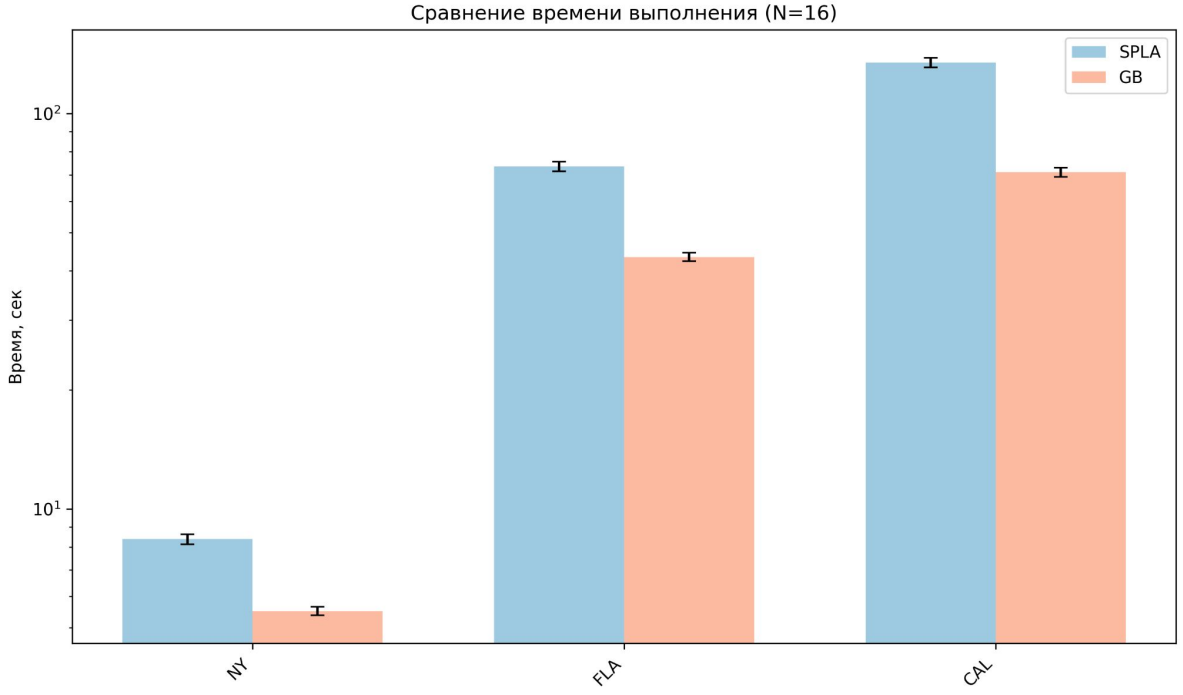
Эксперимент 1 (SSBFS)

- Оба алгоритма в хороших для себя условиях
- **SPLA** всё еще медленнее
  - **Intel OpenCL Runtime**



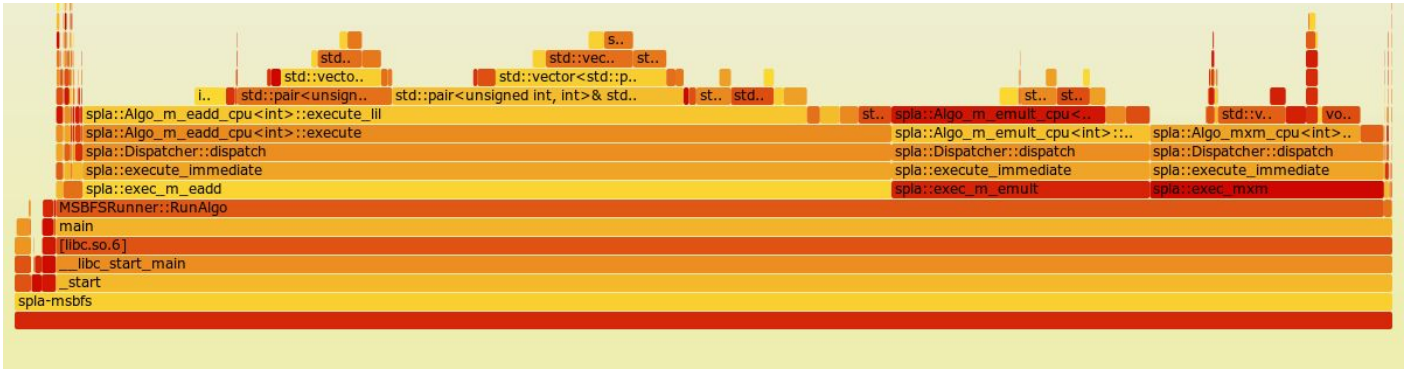
SPLA SSBFS
NY

Эксперимент 1 (MSBFS)



Эксперимент 1 (MSBFS)

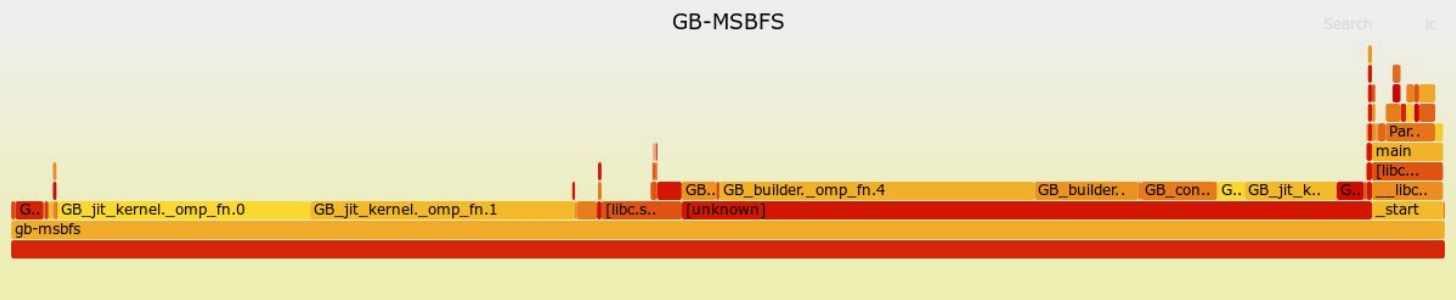
- На больших графах **SPLA** не справляется
  - 1 запуск на графе «Westen USA» занимает более 30 минут
- **SPLA** не умеет работать с матрицами на **OpenCL**
  - Вычисления на 1 ядре



SPLA MSBFS (N = 16)
NY

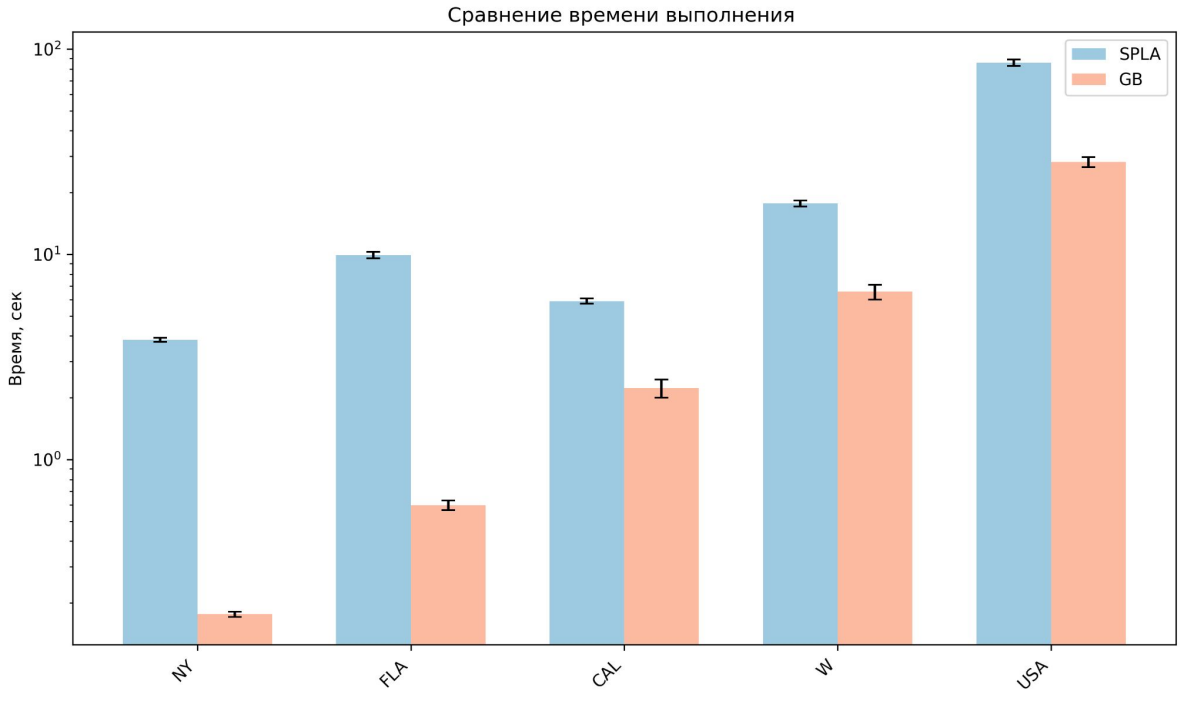
Эксперимент 1 (MSBFS)

- **GraphBLAS**, кажется, плохо умеет в операторы
  - **GB_jit_kernel._omp_fn**
  - Это же касается и **SSBFS**



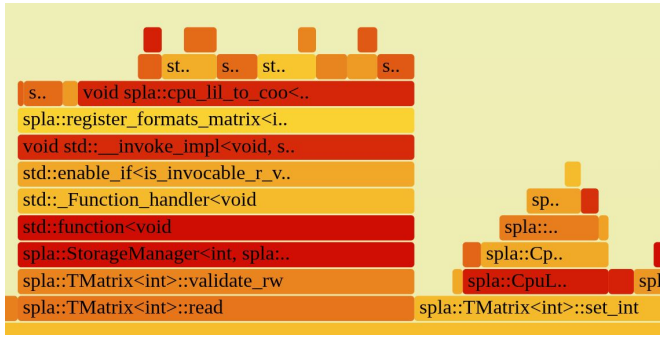
GB MSBFS (N = 16)
NY

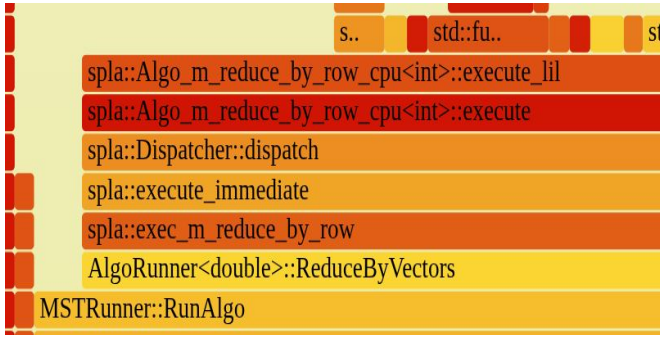
Эксперимент 1 (Boruvka)



Эксперимент 1 (Boruvka)

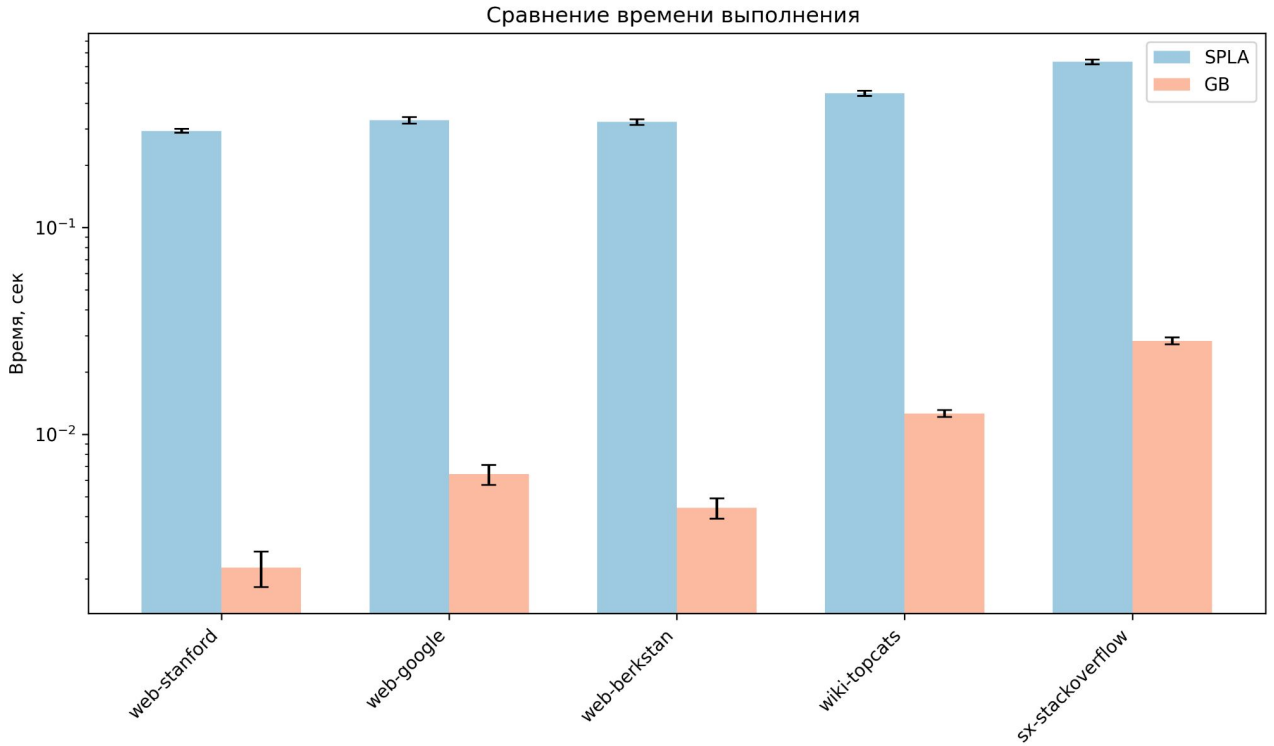
- В **SPLA** нет операции **select** и мало операций с масками
  - Построить остовное дерево дорого
- Не все **CPU** реализации работали



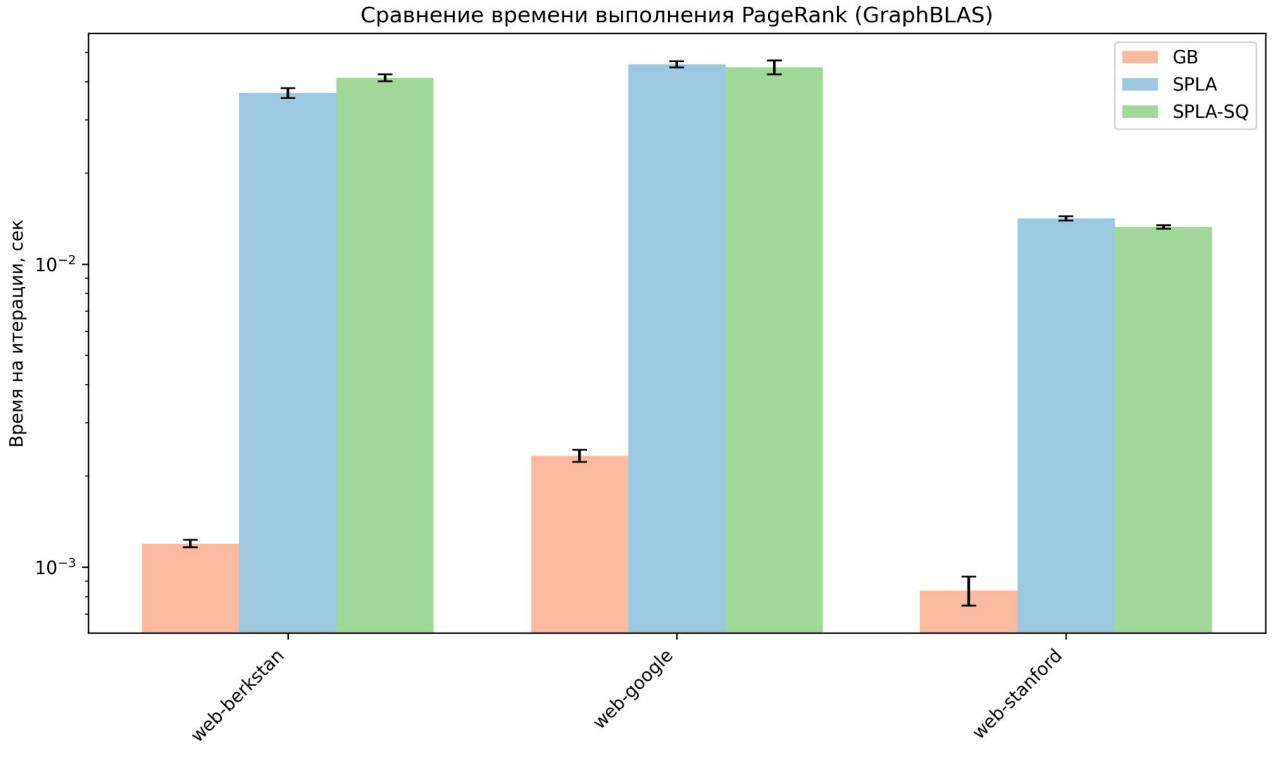


SPLA Boruvka
NY

Эксперимент 1 (PageRank)

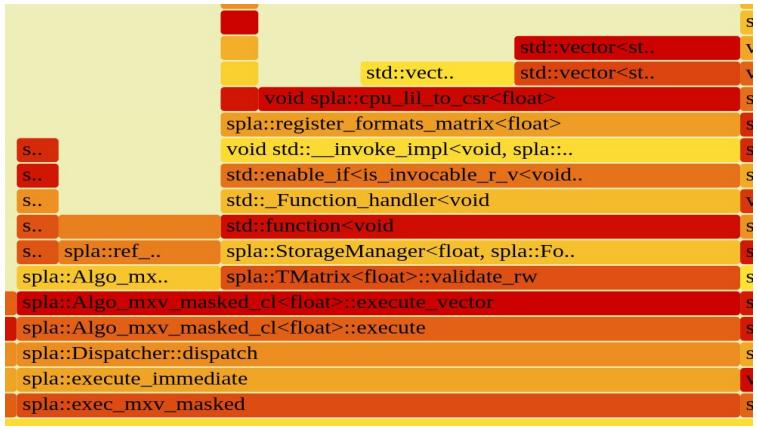


Эксперимент 1 (PageRank)



Эксперимент 1 (PageRank)

- Ключевая проблема **SPLA** — накладные расходы
  - **validate_rw, cl_compile_program**
  - запуск



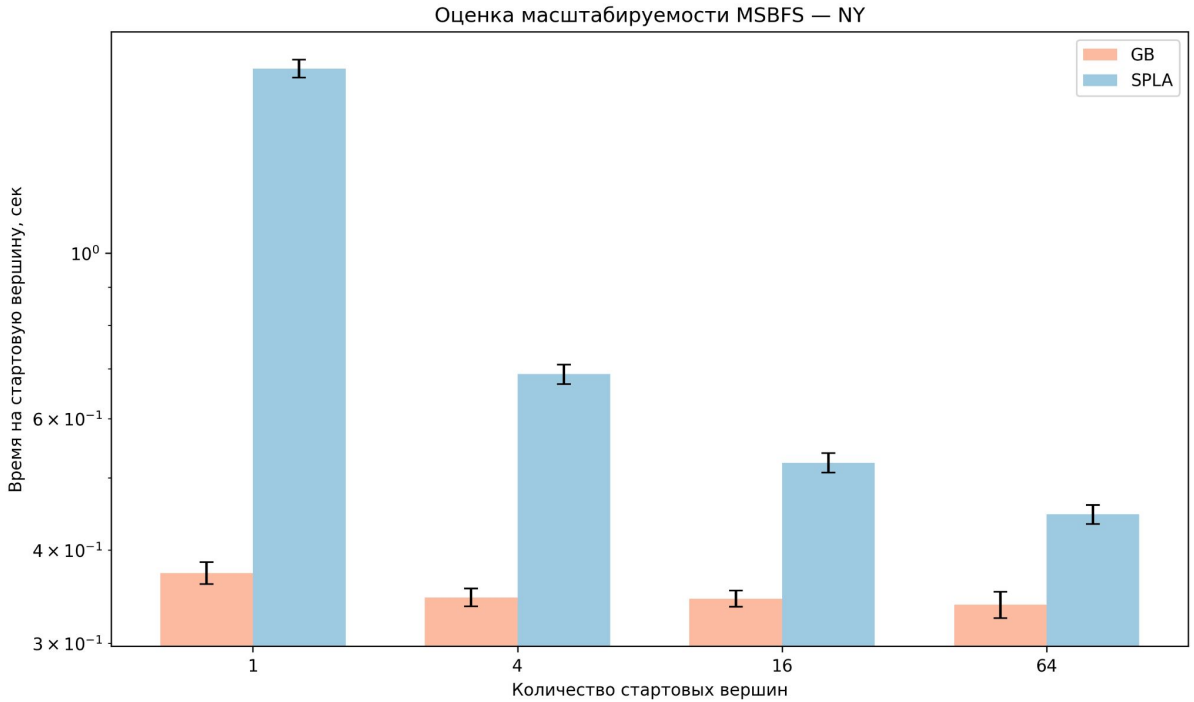
SPLA PR
web-Stanford

Эксперимент 2

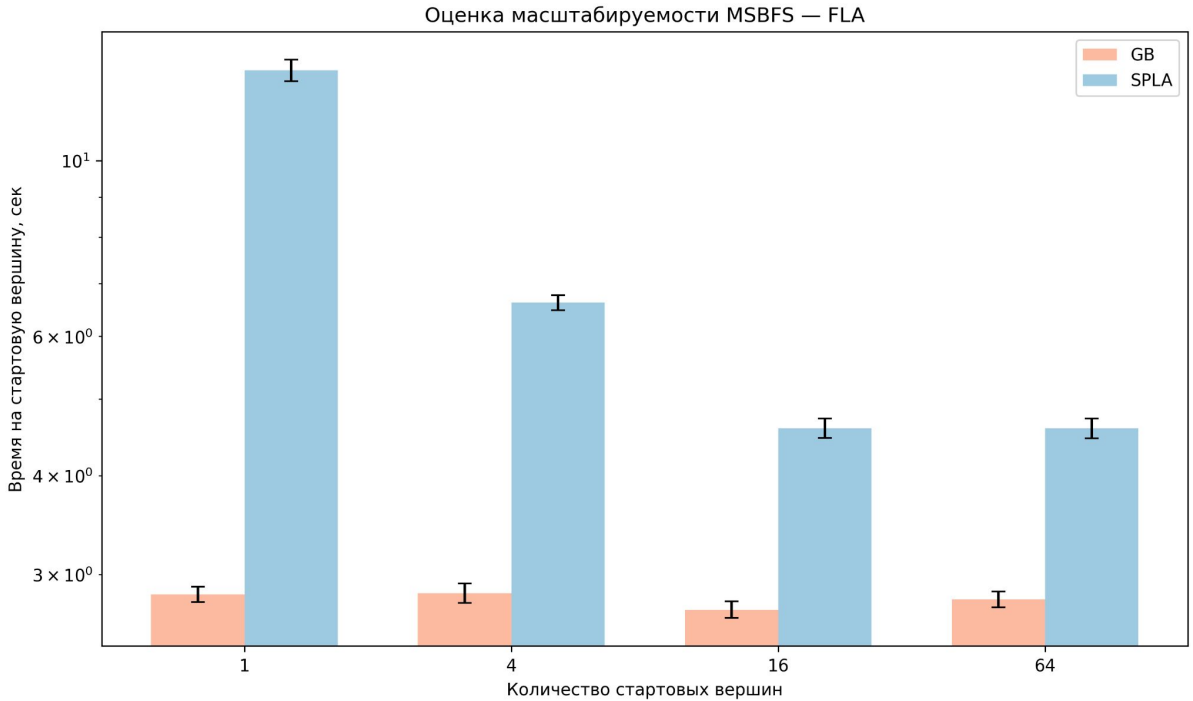
Цель эксперимента: оценить масштабируемость **MSBFS** при изменении числа стартовых вершин для библиотек **SPLA** и **SuitSparse::GraphBLAS**

- Оцениваем замедление при увеличении числа стартовых вершин
- 1, 4, 16, 64 вершин, выбираются случайно с фиксированным сидом

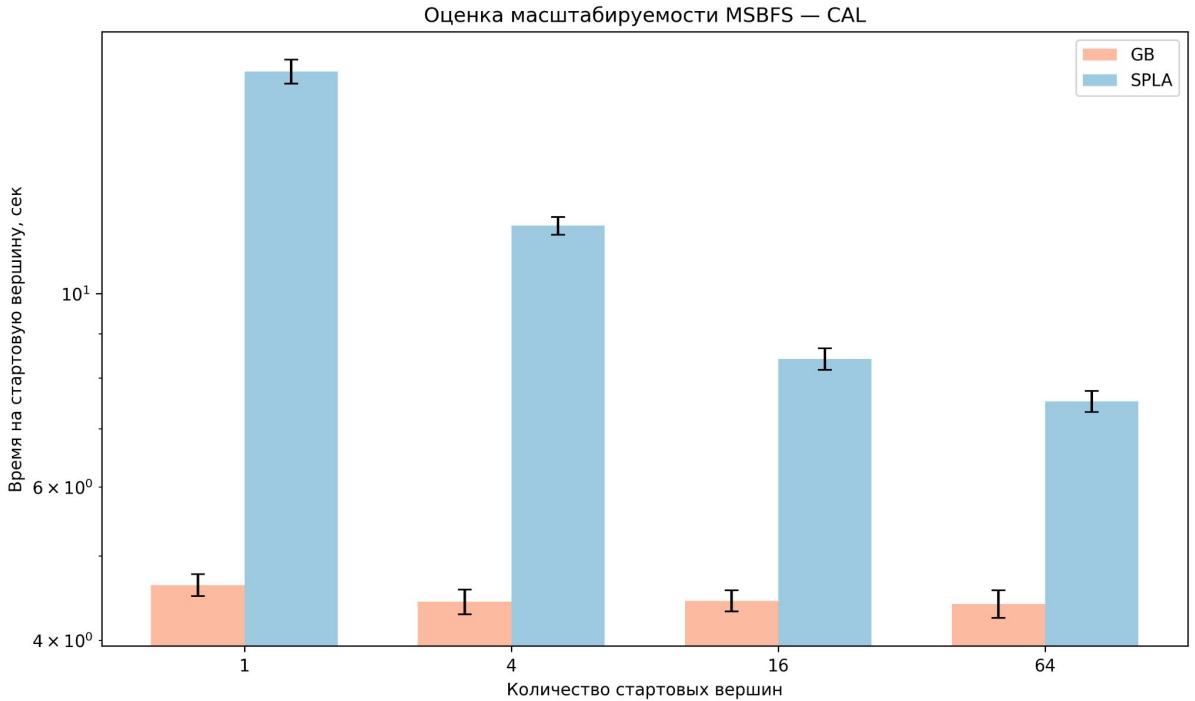
Эксперимент 2



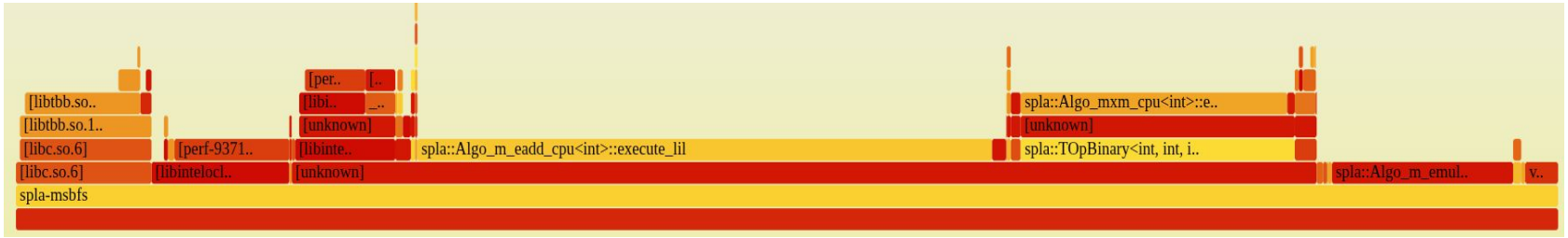
Эксперимент 2



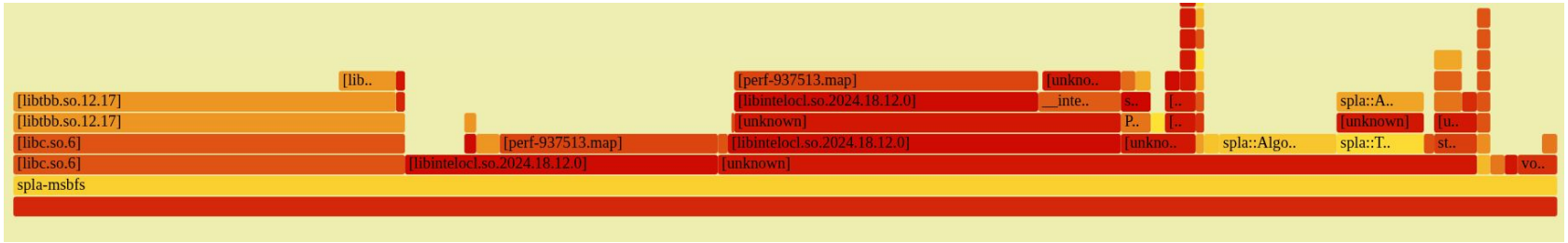
Эксперимент 2



Эксперимент 2



SPLA MSBFS (N = 1)
NY



SPLA MSBFS (N = 16)
NY

Эксперимент 2

- **SPLA** масштабируется лучше не «благодаря», а «вопреки»
  - Больше нод
  - Больше операций на **OpenCL**
  - Лучше производительность

Результаты

- **SPLA** работает медленнее, чем **SuiteSparse::GraphBLAS**
  - **intel_opencl_runtime**
  - Затраты на **JIT** и валидацию
  - Отсуствие важных операция на **OpenCL**

Исходный код и результаты представлены в репозитории.

Немного про SPLA

- Мало операций для работы с линейной алгеброй
- Мало операций на **OpenCL**
- Некоторые операции в принципе не работали
  - **exec_mxm**, **exec_m_reduce**
  - **clCreateBuffer, cpu_lil_to_coo**
- Проблема с **float** и **spla::T_FLOAT**
- Отсутствие пользовательских типов
  - **int64_t**