

2. Команда 1

Участники: Рожков Александр, Перевалов Ефим,
Нурмухаметов Рафик

BFS Single-source Parent

BFS Single-source Parent - это алгоритм поиска в ширину, который обходит граф от заданной начальной вершины, сохраняя информацию о "родителях" каждой вершины

BFS Multiple-source Parent

Multiple-source Parent BFS - это алгоритм обхода графа, который начинается с нескольких исходных вершин одновременно, а не с одной исходной вершины, как в Single-source Parent BFS

Эксперимент

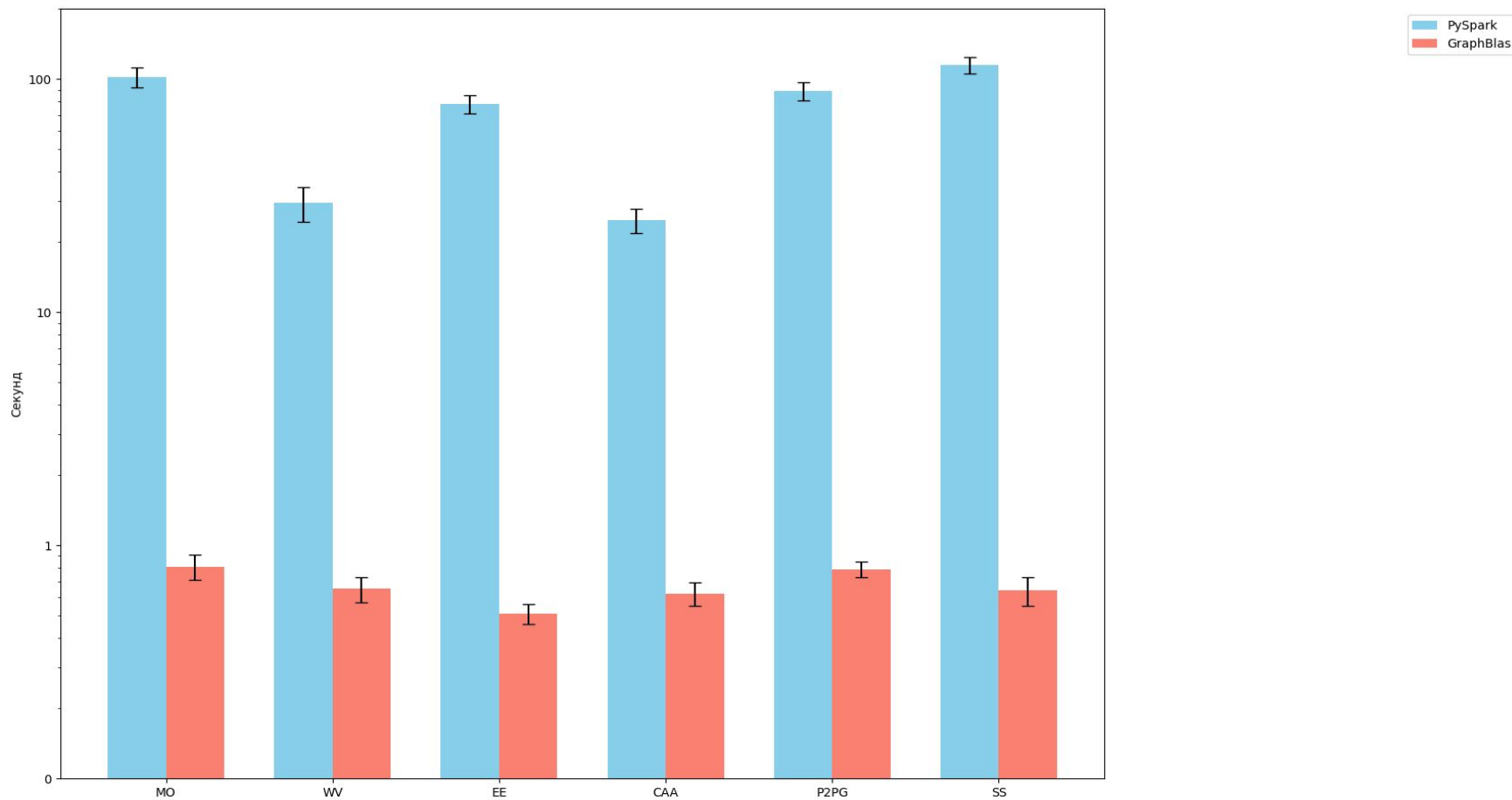
На какой из 2 библиотек (PySpark, GraphBLAS) быстрее будет реализация для алгоритмов BFS Single-source Parent, BFS Multiple-source Parent. Для MS BFS количество стартовые вершины: 2, 4, 8, 16, 32. На каждый вариант по 30 запусков

Граф	Вершин	Ребер
Math_overflow	55 863	858 490
wiki-Voute	7 115	103 689
Email-Enron	36 692	183 831
CA-AstroPh	18 772	396 160
p2p-Gnutella31	62 586	147 892
soc-sign-Slashdot090216	81 871	545 671

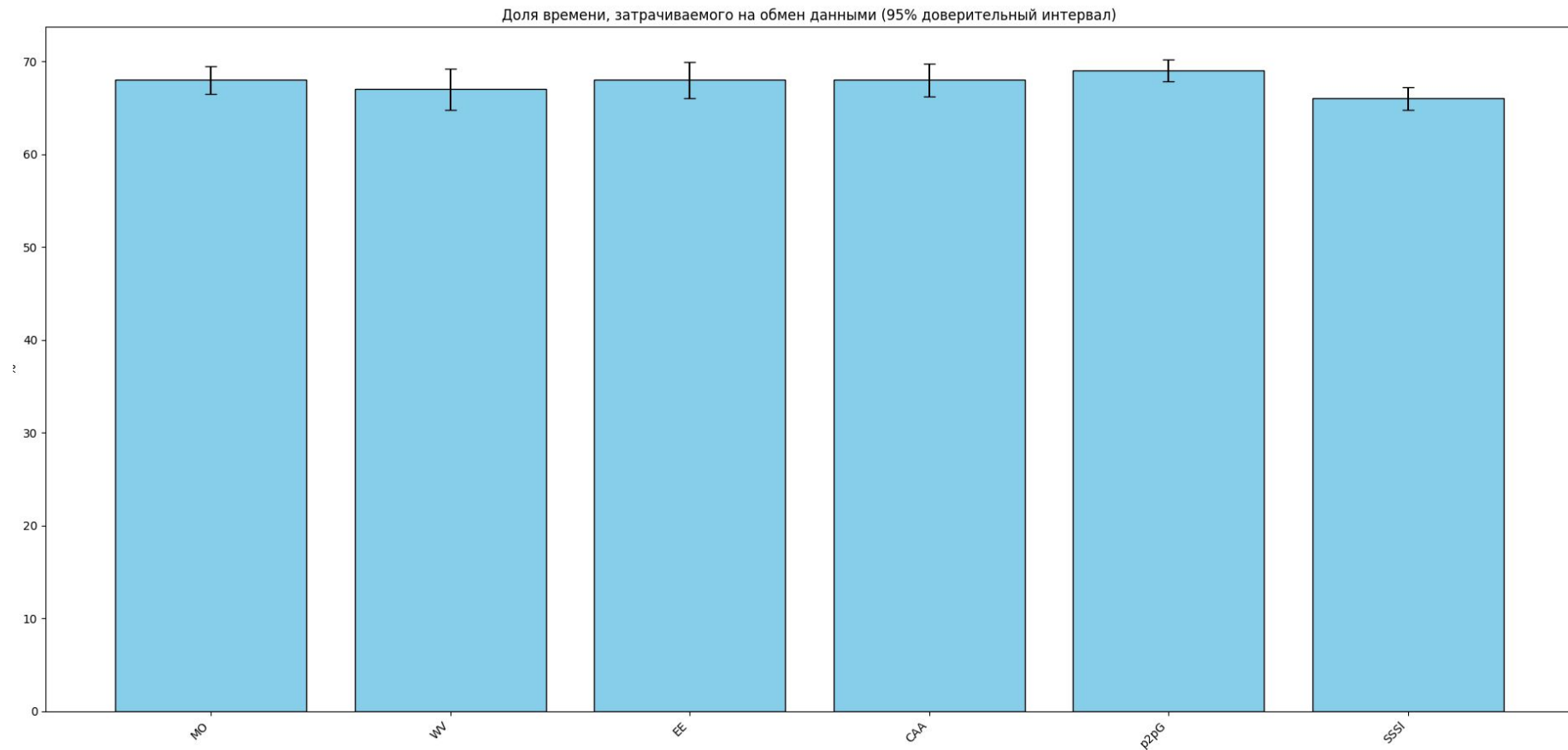
Характеристика машины

- ОС Ubuntu 20.04 LTS
- Intel Core i5-10210U
 - 1.6 GHz
 - 4 ядра
- 16 GB RAM

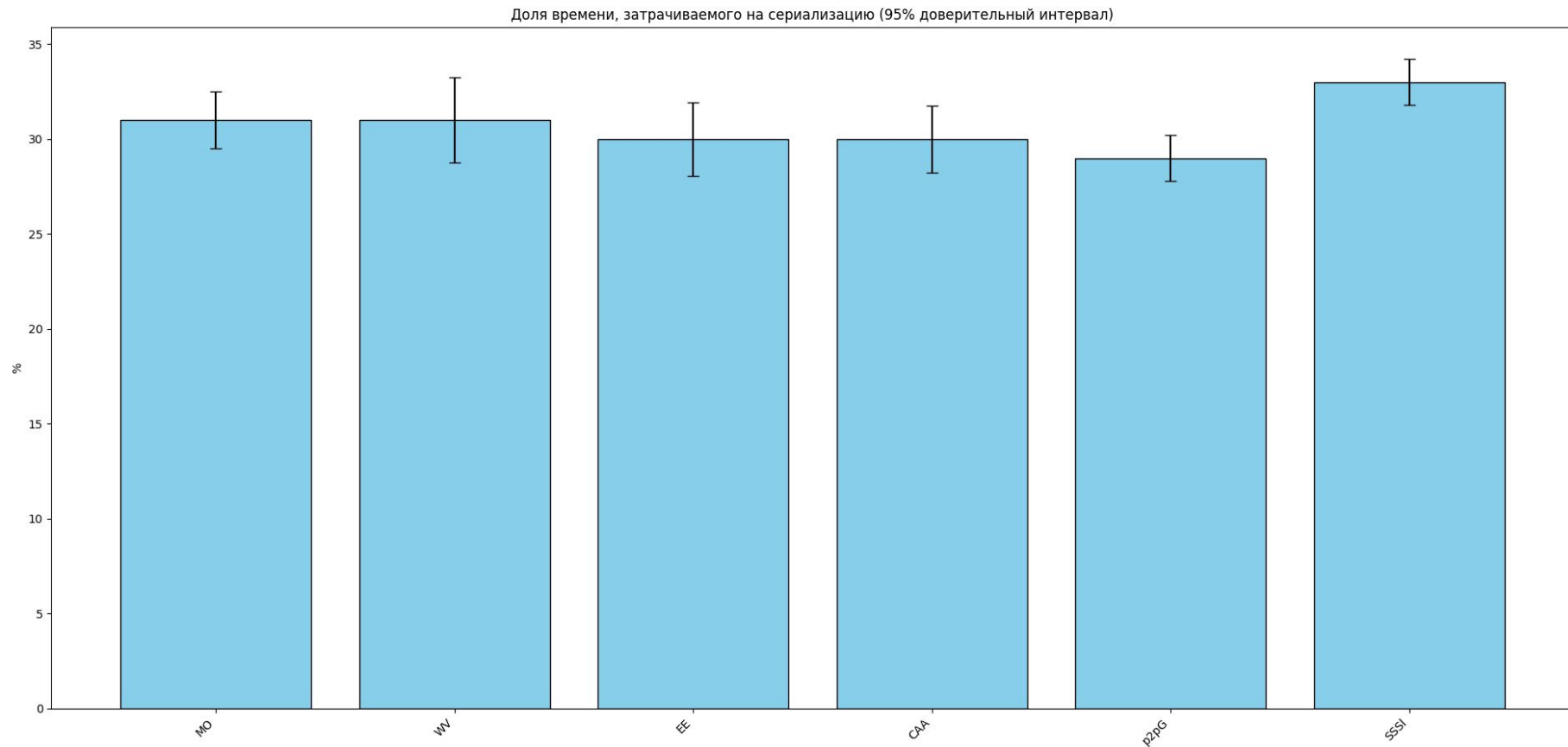
SS-BFS. Сравнение результатов эксперимента



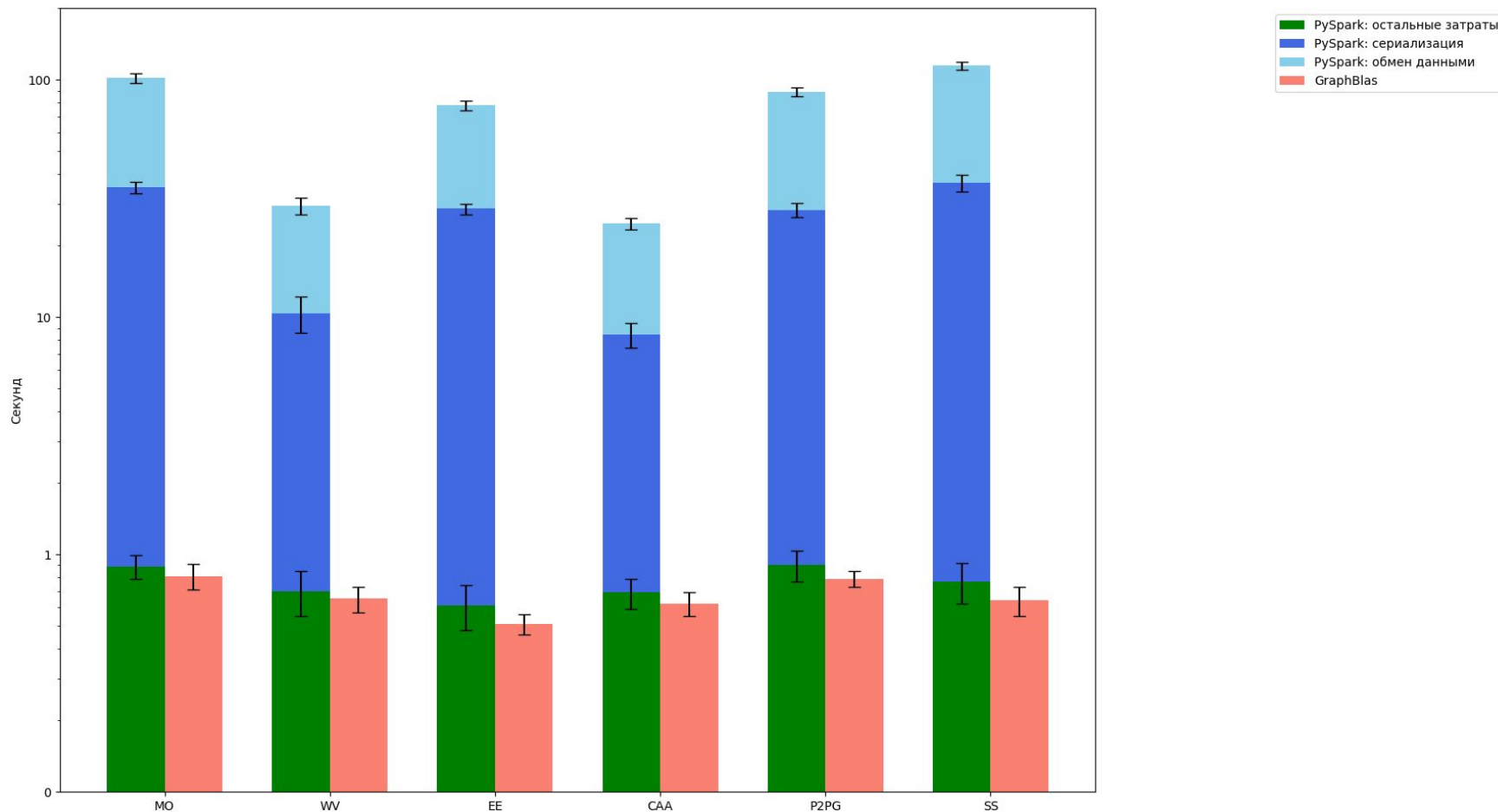
SS-BFS. Результаты профилирования PySpark 1



SS-BFS. Результаты профилирования PySpark 2



SS-BFS. Анализ полученных результатов 1



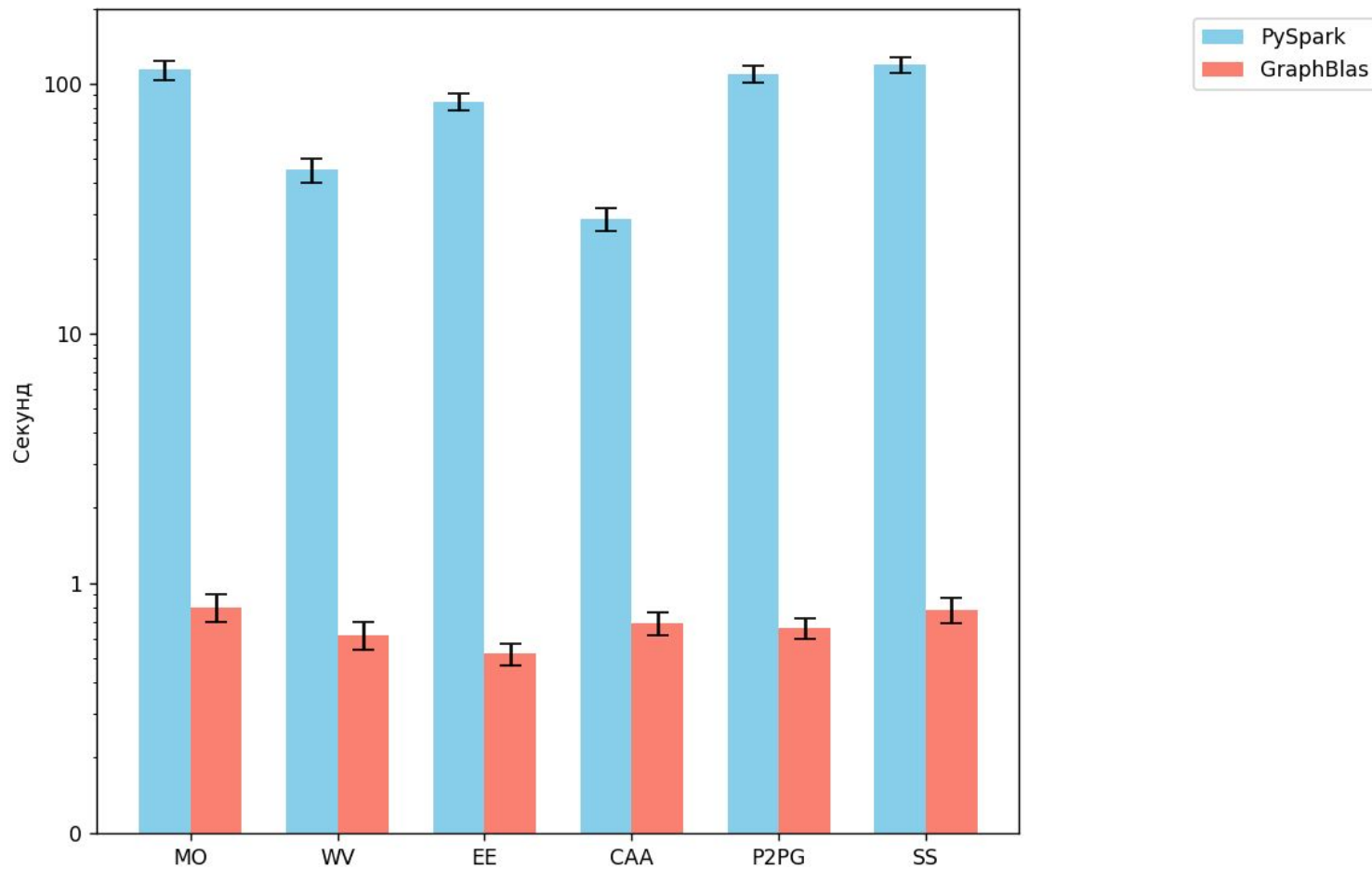
SS-BFS. Анализ полученных результатов 2

PySpark проигрывает реализации на GraphBlas в среднем в 100 раз

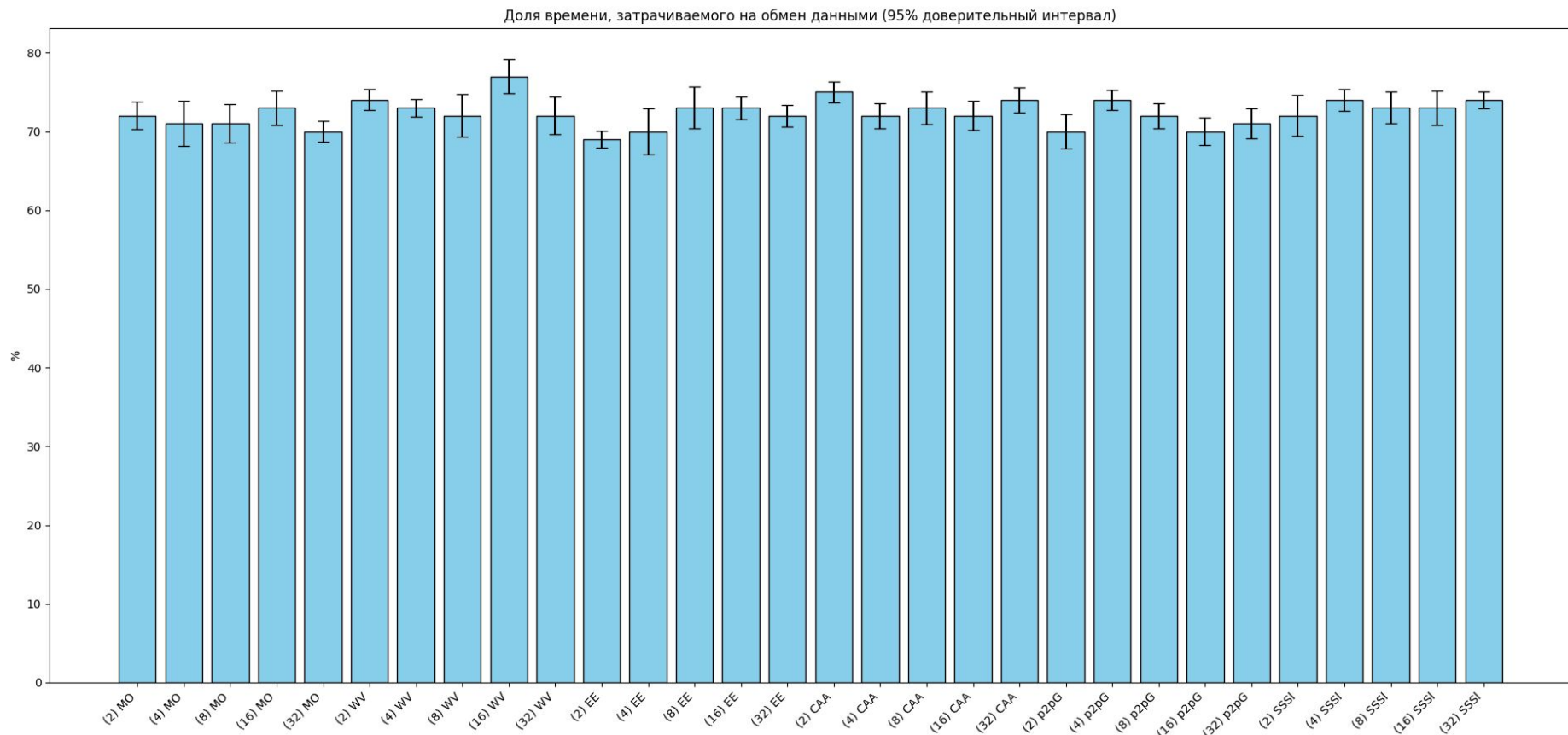
Причины:

- PySpark работает через несколько потоков с затратной передачей данных (обмен данными занимает около 68% рабочего времени)
- Сериализация при проверке новых непосещенных вершин требует пересылки данных между JVM и Python (занимает около 31% рабочего времени)

MS-BFS. Сравнение результатов эксперимента

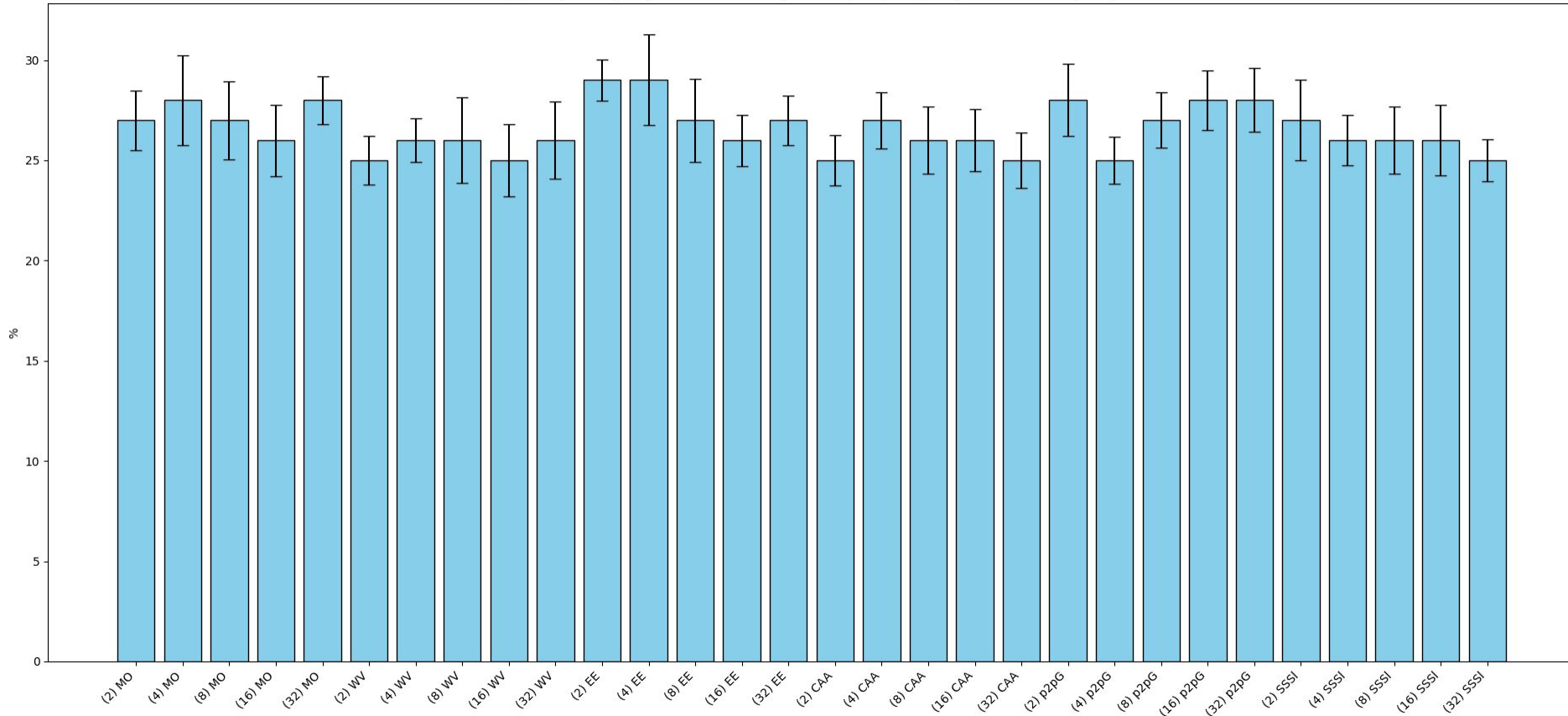


MS-BFS. Результаты профилирования PySpark 1

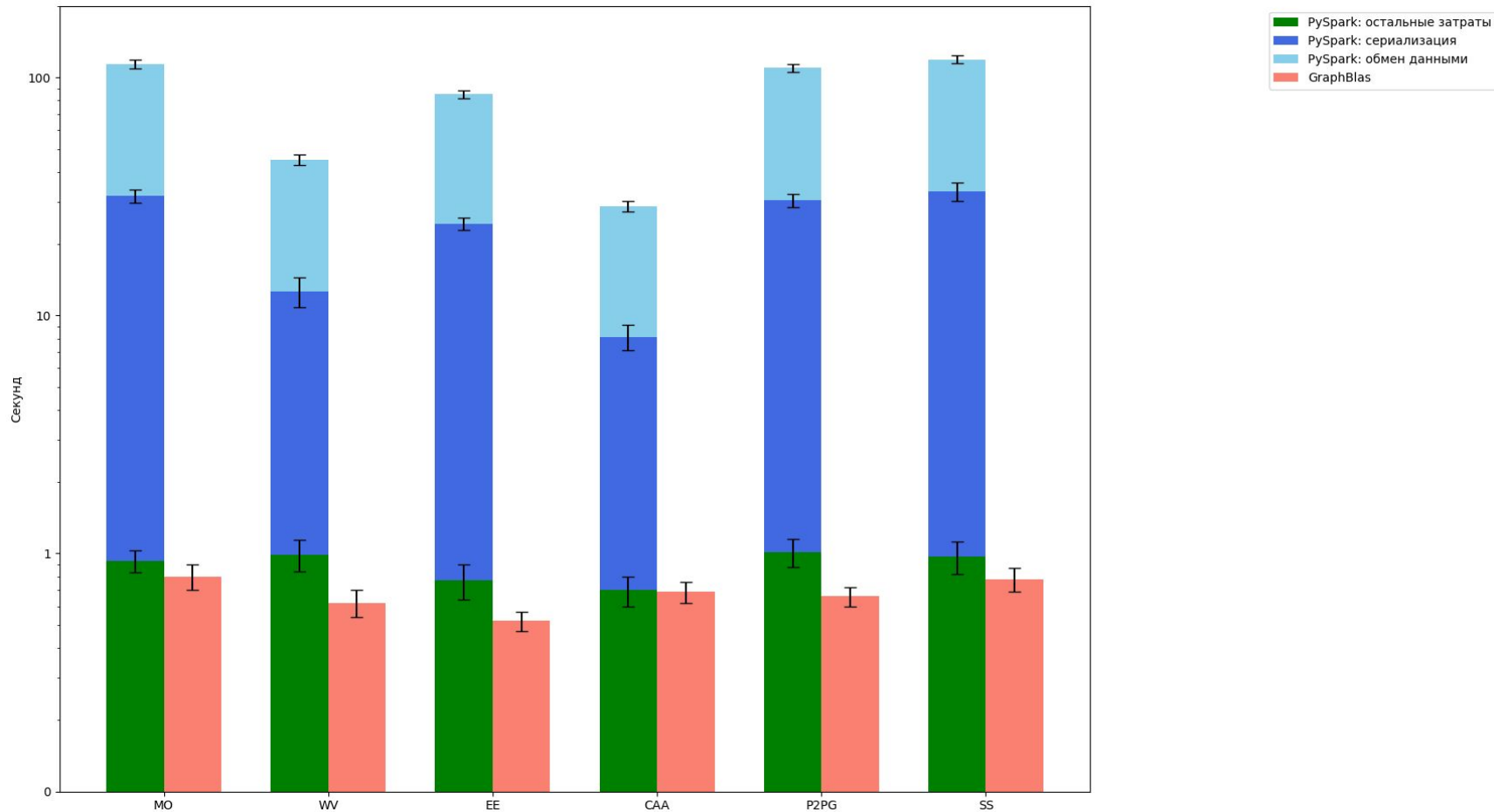


MS-BFS. Результаты профилирования PySpark 2

Доля времени, затрачиваемого на сериализацию (95% доверительный интервал)



MS-BFS. Анализ полученных результатов 1



MS-BFS. Анализ полученных результатов 2

PySpark проигрывает реализации на GraphBlas в среднем в 100 раз

Причины:

- PySpark работает через несколько потоков с затратной передачей данных (обмен данными занимает около 70% рабочего времени)
- Сериализация при проверке новых непосещенных вершин требует пересылки данных между JVM и Python (занимает около 28% рабочего времени)
- GraphBlas оперирует матричным умножением

Треугольники

Задача: найти количество уникальных треугольников в графе.

Алгоритмы:

- Sandia $\sum_j \sum_i (U^2 * U)$
- Burkhardt $\frac{1}{6} \sum_j \sum_i (A^2 * A)$

SuiteSparse:GraphBLAS - полная реализация стандарта GraphBLAS, использующаяся для работы с графовыми алгоритмами на языке линейной алгебры.

SPLA - открытая библиотека для работы с разреженными матрицами, предоставляющая поддержку GPU.

Эксперимент

Цель: Сравнить реализации на SuiteSparse:GraphBLAS(CPU) и SPLA(CPU/GPU).

Ход эксперимента:

- 10 запусков для каждого графа
- Вычисление среднего значения
- Анализ полученных результатов

Гипотезы:

- SuiteSparse:GraphBLAS быстрее SPLA на CPU
- SPLA с GPU превосходит SuiteSparse:GraphBLAS

Вычислительная машина и по

Вычислительная машина:

- Процессор: 11th Gen Intel(R) Core(TM) i3-1115G4
 - Количество ядер: 2
 - Количество логических ядер: 4
- RAM: 8 GB
- GPU: Mesa Intel® UHD Graphics (TGL GT2)

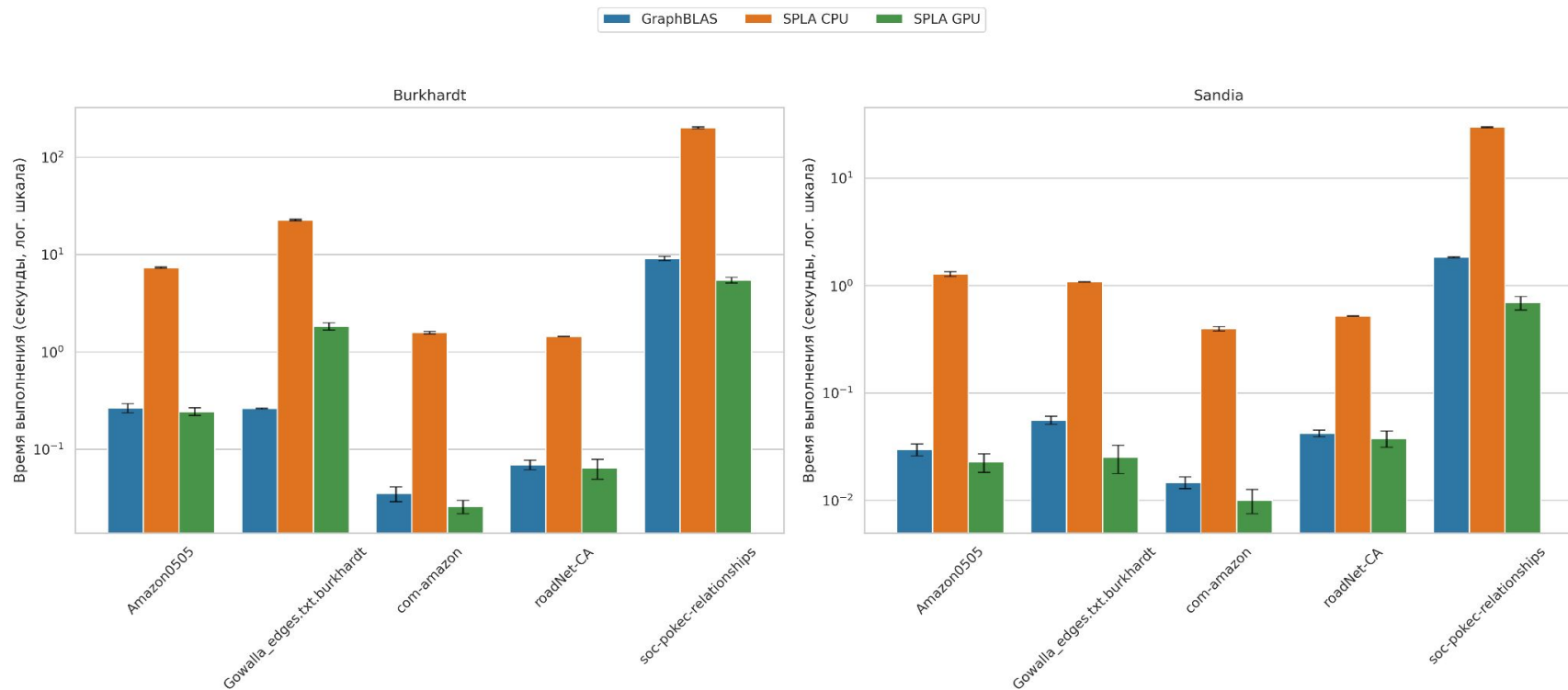
ПО:

- ОС: Ubuntu 22.04.5 LTS
- gcc 11.4.0
- intel openCL 22.14.22890

Набор данных

Название	Вершин	Ребер
roadNet-CA	1 965 206	2 776 607
loc-Gowalla	196 591	950 327
amazon0505	410 236	3 356 824
com-Amazon	334 863	925 872
soc-Pokec	1 632 803	30 622 463

Анализ полученных результатов



Анализ полученных результатов

- SuiteSparse:GraphBLAS лучше заточен для вычислений на CPU
- SPLA с GPU превосходит SuiteSparse:GraphBLAS
- С увеличением количества вычислений SPLA требует больше ресурсов

Кратчайшие пути

Алгоритм Форда-Беллмана — алгоритм, предназначенный для поиска кратчайших путей во взвешенном графе без петель от заданной вершины, до всех остальных.

Эксперимент

Цель: Сравнить реализацию на основе GraphBLAS и PySpark.

Ход эксперимента:

- 10 запусков для каждого графа
- Вычисление среднего значения
- Анализ полученных результатов

Гипотезы:

- Реализация на GraphBLAS будет быстрее, чем на PySpark

Вычислительная машина и по

Вычислительная машина:

- Процессор: 11th Gen Intel(R) Core(TM) i3-1115G4
 - Количество ядер: 2
 - Количество логических ядер: 4
- RAM: 8 GB
- GPU: Mesa Intel® UHD Graphics (TGL GT2)

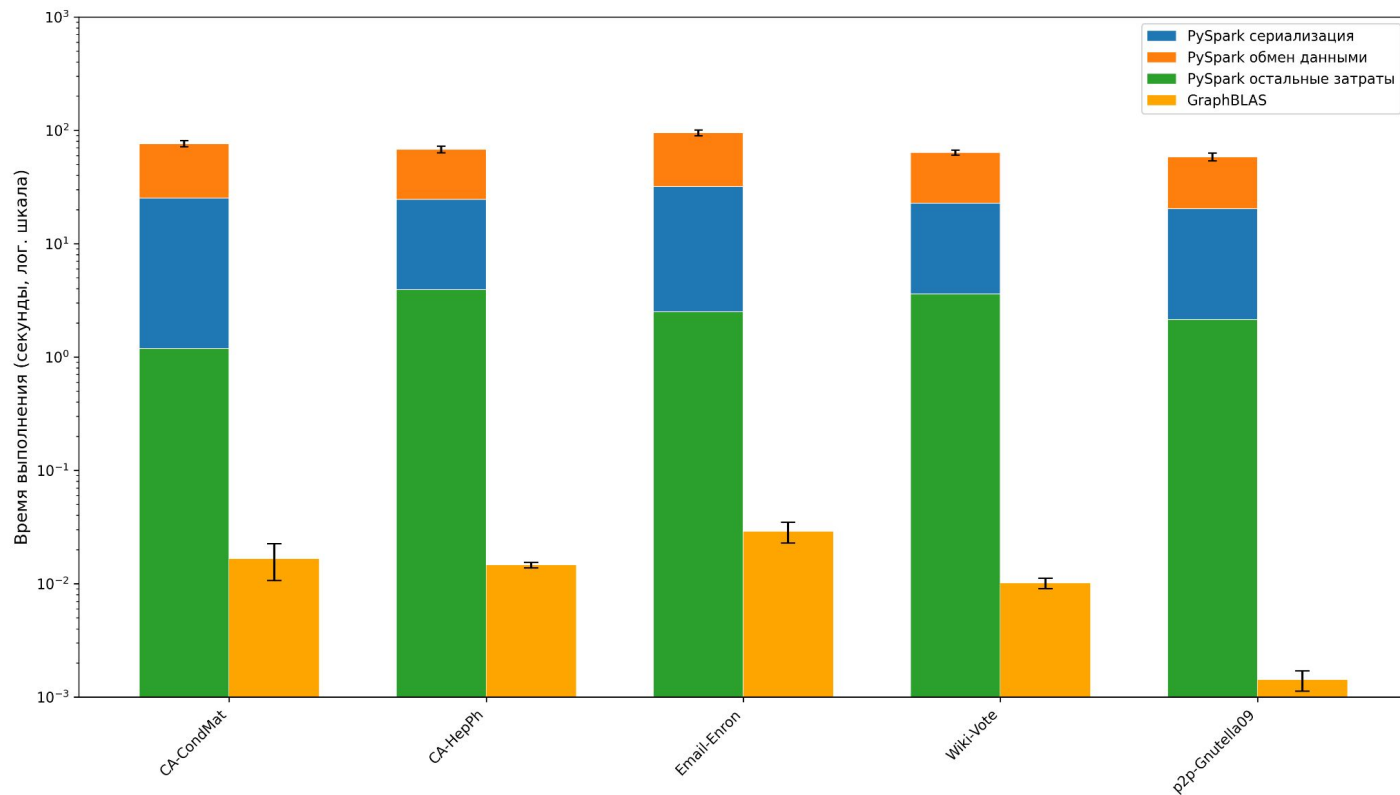
ПО:

- ОС: Ubuntu 22.04.5 LTS
- gcc 11.4.0
- intel openCL 22.14.22890

Набор данных

Название	Вершин	Ребер
CA-CondMat	23 133	93 497
CA-HepPh	12,008	118,521
Email-Enron	36 692	183 831
Wiki-Vote	7 115	103 689
p2p-Gnutella09	10 876	39 994

Анализ полученных результатов



Анализ полученных результатов

- PySpark затрачивает много времени на обмен сообщениями (около 65-70%)
- На сериализацию при обновлении вершин также уходит 25-30%
- GraphBLAS оптимизирован для матричных вычислений на cpu.

Эксперимент

Цель: Сравнить реализацию на SuiteSparse:GraphBLAS и встроенную на SPLA.

Ход эксперимента:

- 10 запусков для каждого графа
- Вычисление среднего значения
- Анализ полученных результатов

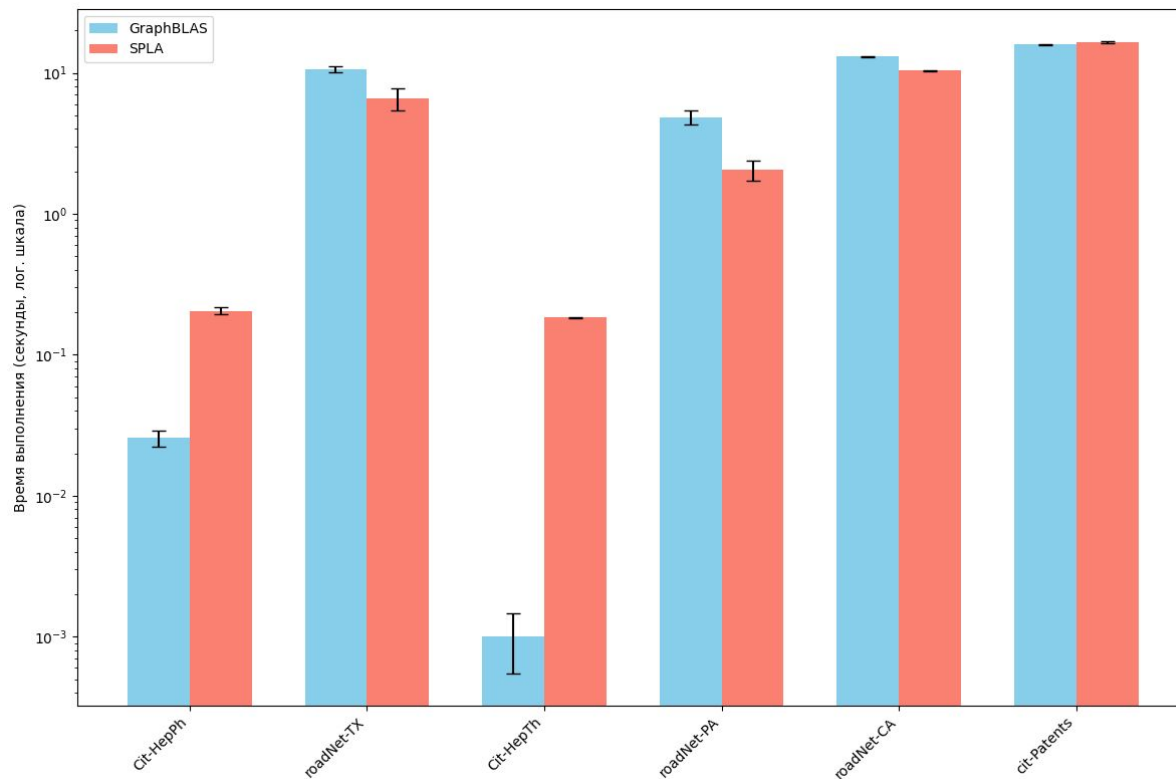
Гипотезы:

- SPLA будет превосходить по скорости реализацию на SuiteSparse:GraphBLAS

Набор данных

Название	Вершин	Ребер
Cit-HepPh	34 546	421 578
Cit-HepTh	27 770	352 807
cit-Patents	3 774 768	16 518 948
roadNet-CA	1 965 206	2 766 607
roadNet-PA	1 088 092	1 541 898
roadNet-TX	1 379 917	1 921 660

Анализ полученных результатов



Анализ полученных результатов

- SuiteSparse:GraphBLAS на маленьких графах работает быстрее
- В остальных случаях SPLA превосходит реализацию на SuiteSparse:GraphBLAS
- С увеличением количества вычислений SPLA требует больше ресурсов