Сравнение производительности графовых библиотек на Prim’s MST и MS-BFS

Андрей Сухарев
Данил Слинчук

github.com/therain7/graphstuff

Prim’s MST

- Дан связный неориентированный граф.
  Для каждого ребра задана его стоимость

- Результатом работы алгоритма является
  остовное дерево минимальной стоимости

- Начинается с произвольной вершины.
  На каждой итерации добавляется
  самое “дешевое” ребро

Prim's MST

```
function Prim(vertices, edges) is
    for each vertex in vertices do
        cheapestCost[vertex] = INFINITY
    cheapestCost[START_NODE] = 0

    total_weight = 0

    unexplored ← set containing all vertices
    while unexplored is not empty do
        currentVertex ← vertex in unexplored with minimum cheapestCost[vertex]
        unexplored.remove(currentVertex)

        for each edge (currentVertex, neighbor) in edges do
            if neighbor in unexplored and weight(currentVertex, neighbor) < cheapestCost[neighbor] THEN
                cheapestCost[neighbor] ← weight(currentVertex, neighbor)
                total_weight += weight(currentVertex, neighbor)

    return total_weight
```

Prim’s MST на матрицах

```
function Prim(A) is
    Vector weights = extract_row(A, START_NODE)

    Vector visited(nrows)
    visited.set(START_NODE, true)

    visited_count = 0
    total_weight = 0
    while (visited_count < nrows) {
        Vector to_visit = weights . !visited

        Edge cheapest = find_min(to_visit)
        visited.set(cheapest.idx, true)
        total_weight += cheapest.val
        visited_count++

        Vector neighbors = extract_row(A, cheapest.idx)
        weights = eWiseAdd(weights, neighbors, OP_MIN)
    }

    return total_weight
```

Multi-Source BFS

- Дан связный (не/)ориентированный граф
- Результатом работы алгоритма является матрица найденных путей
- Начинается с нескольких заданных вершин-источников. На каждой итерации продвигается по всем путям одновременно

Multi-Source BFS

```
function msbfs(Matrix A, Vector src) is
    // front of BFS
    Matrix front(nsrc, nrows)
    // parent(i, j) is the parent id of node j in source i's BFS
    Matrix parent(nsrc, nrows)

    for (uint i = 0; i < nsrc; i++) {
        cursrc = src.get(i)
        front.set(i, cursrc, cursrc)
        parent.set(i, cursrc, cursrc)
    }

    nfront = nsrc
    for (uint nvisited = nsrc; nvisited < n * nsrc; nvisited += nfront) {
        front = mxm_secondI(front, A) . !parent
        nfront = front.count()
        if (!nfront) {
            break
        }

        parent = eadd(parent, front)
    }

    return parent
```

Эксперимент

- Реализовать алгоритмы на SuiteSparse:GraphBLAS и SPLA

- Провести 10 запусков обоих алгоритмов на выбранных библиотеках, замерить производительность

- Сравнить полученные значения

Параллельность

- Векторные и матричные операции параллельны
- SuiteSparse:GraphBLAS использует OpenMP

- SPLA — OpenCL
  - Все вычисления производить на ускорителе

  - Избегать копирования данных
    между хостом и ускорителем

  - Избегать конвертации между форматами хранения

SPLA. Патчи

Реализованы:

- OpenCL matrix extract_row
- OpenCL vector emult
- OpenCL mxmT_maskC_secondI
- OpenCL vector find_min (последовательный)

Оптимизирован set_float для OpenCL векторов

Выбранные графы

sparse.tamu.edu - SuiteSparse matrix collection

Граф	Количество вершин	Количество ребер
luxembourg_osm	114 599	239 332
la2010	204 447	980 634
cage12	130 228	2 032 536

Конфигурация

Ноутбук

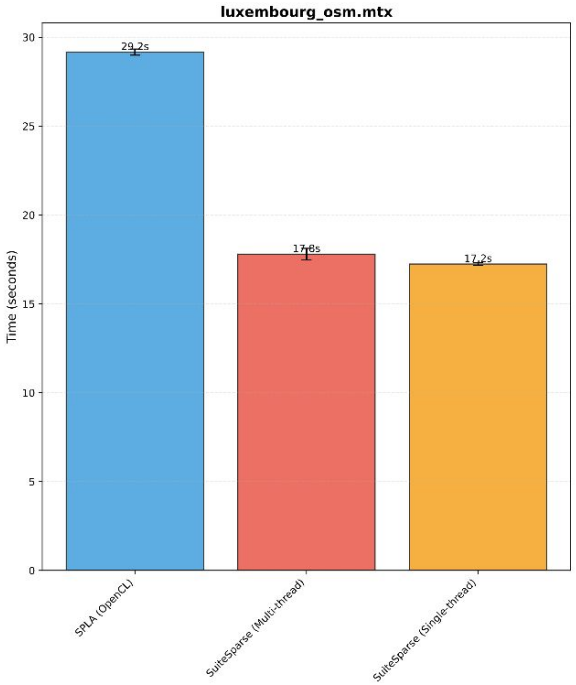
- Intel Core i5 1240P (16 threads)
- Linux 6.16.12-200.fc42

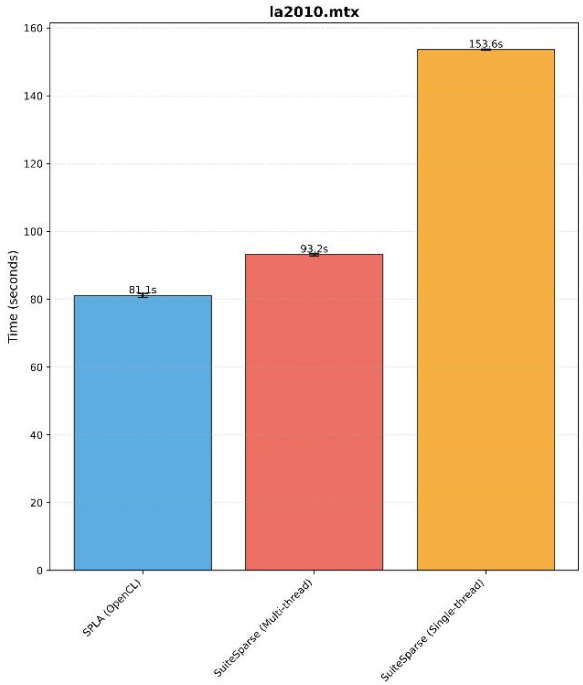
Сервер

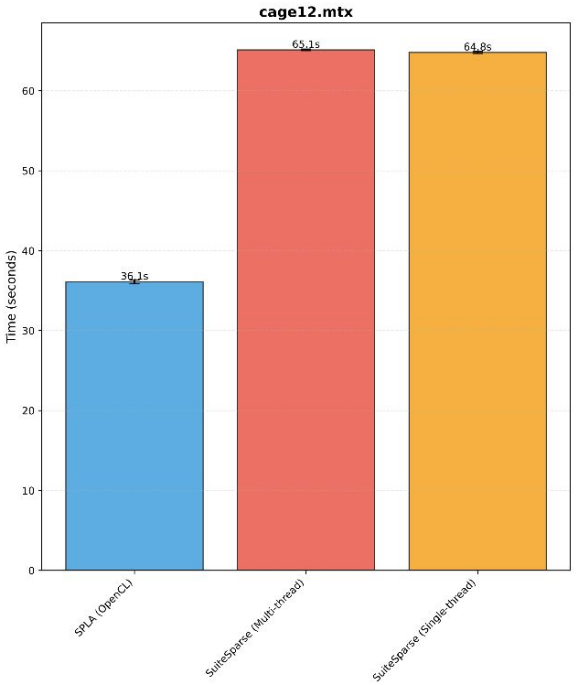
- 2 x Intel Xeon Silver 4214R (24 threads)
- 2 sockets, 1 NUMA node
- Linux 6.16.13

OpenCL Intel runtime 2025.20.10.0.10_160000

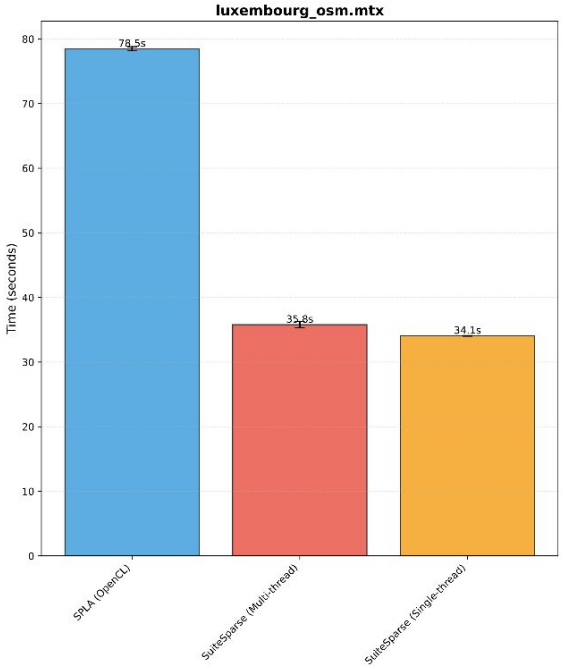
Prim. Intel Core. 95% д.и.

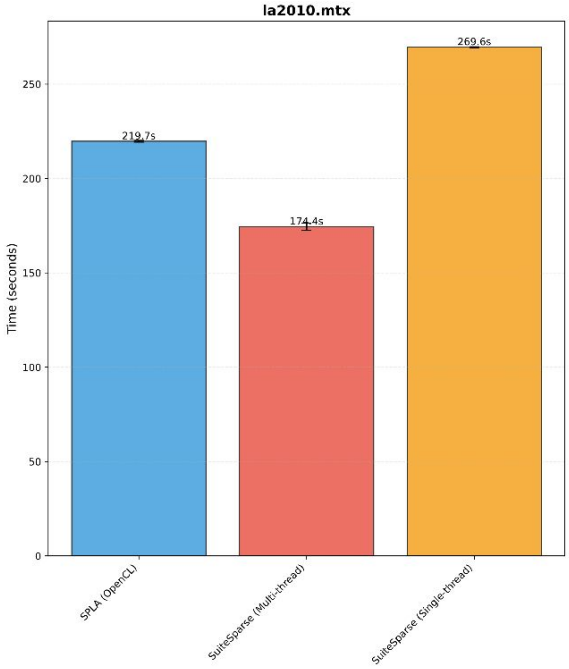


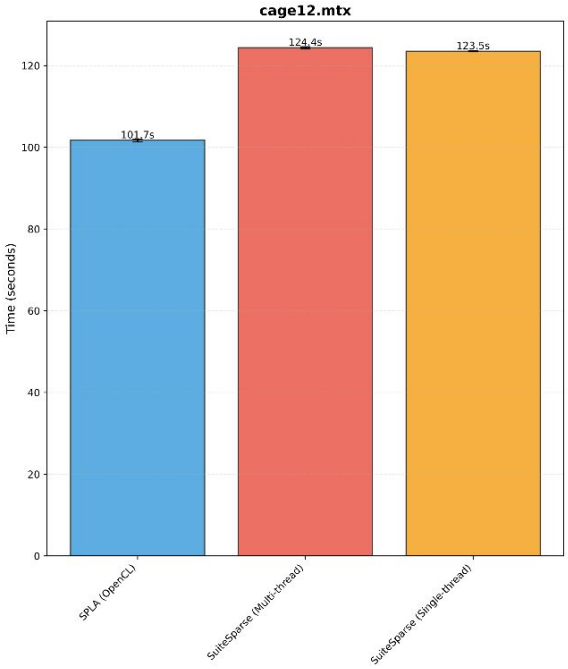




Prim. Intel Xeon. 95% д.и.







Prim. Выводы

- Высокий overhead на синхронизацию
  на маленьких графах

- Высокий overhead на синхронизацию
  на большом количестве ядер

- SuiteSparse:GraphBLAS не всегда
  уходит в полную параллельность.
  Multi-thread & single-thread
  производительность может быть сравнима

MS-BFS

SPLA DNF!

```
front = mxm_secondI(front, A) . !parent
```

Необходима обратная маска,
но все матрицы на SPLA OpenCL в формате CSR

Реализована маска по значению, а не структуре
→ взрыв количества элементов фронта
   после первой итерации