Параллельная реализация MST с использованием CPU и GPU

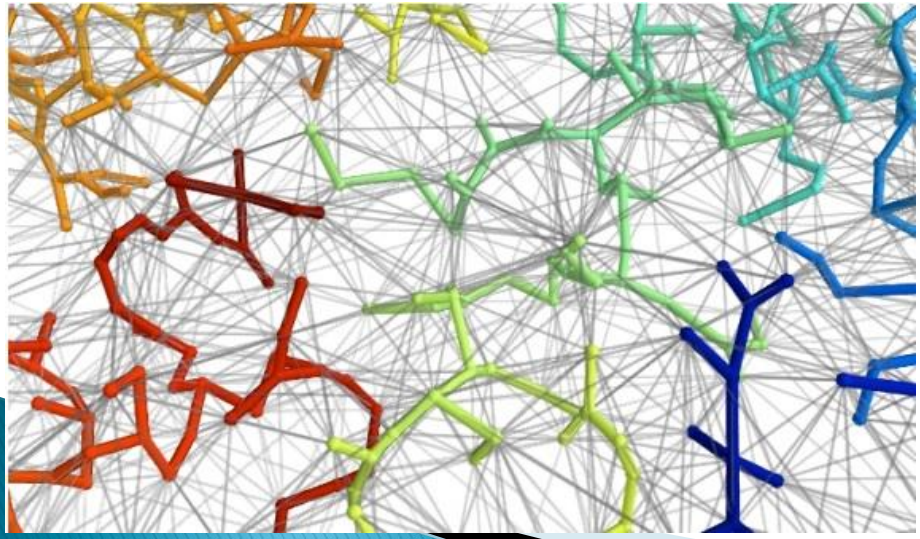
Колганов А.С.
ВМК МГУ

Минимальное остовное дерево

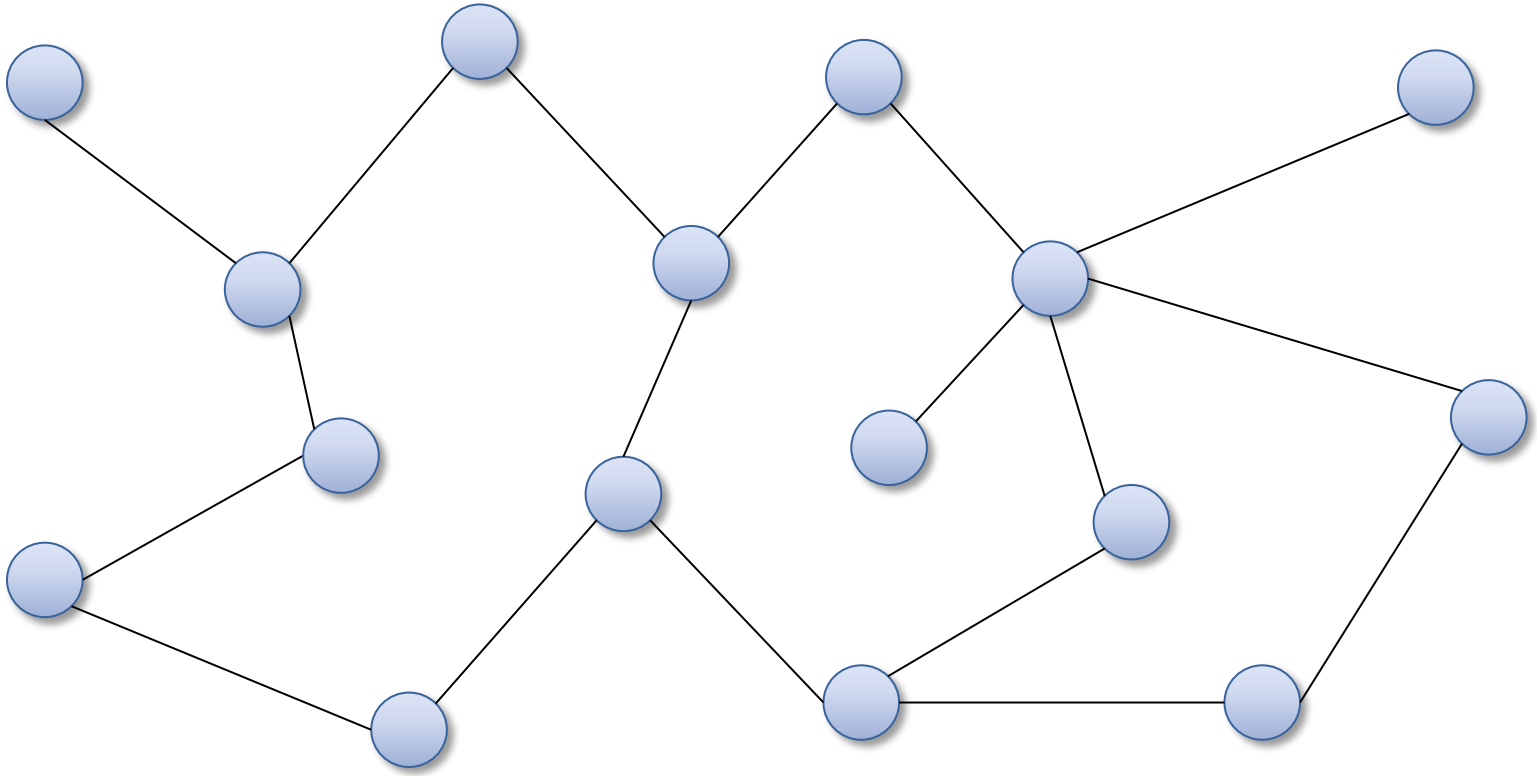
- Рассматривается граф G(V, E, W), требуется построить дерево с минимальным общим весом, покрывающее все вершины;
- Временная оценка сложности самой быстрой последовательной реализации O(E * a(E, V));
- Существуют параллельные реализации алгоритмов Прима, Крускала и Соллина;
- Алгоритм Борвуки, предложенный в 1926 и позже открытый Соллиным, обычно используется в параллельных реализациях.

Практическое применение MST

- Проектирование сетей;
- Поиск оптимальных маршрутов;
- Примерное решение задачи коммивояжёра;
- Химия и биология.

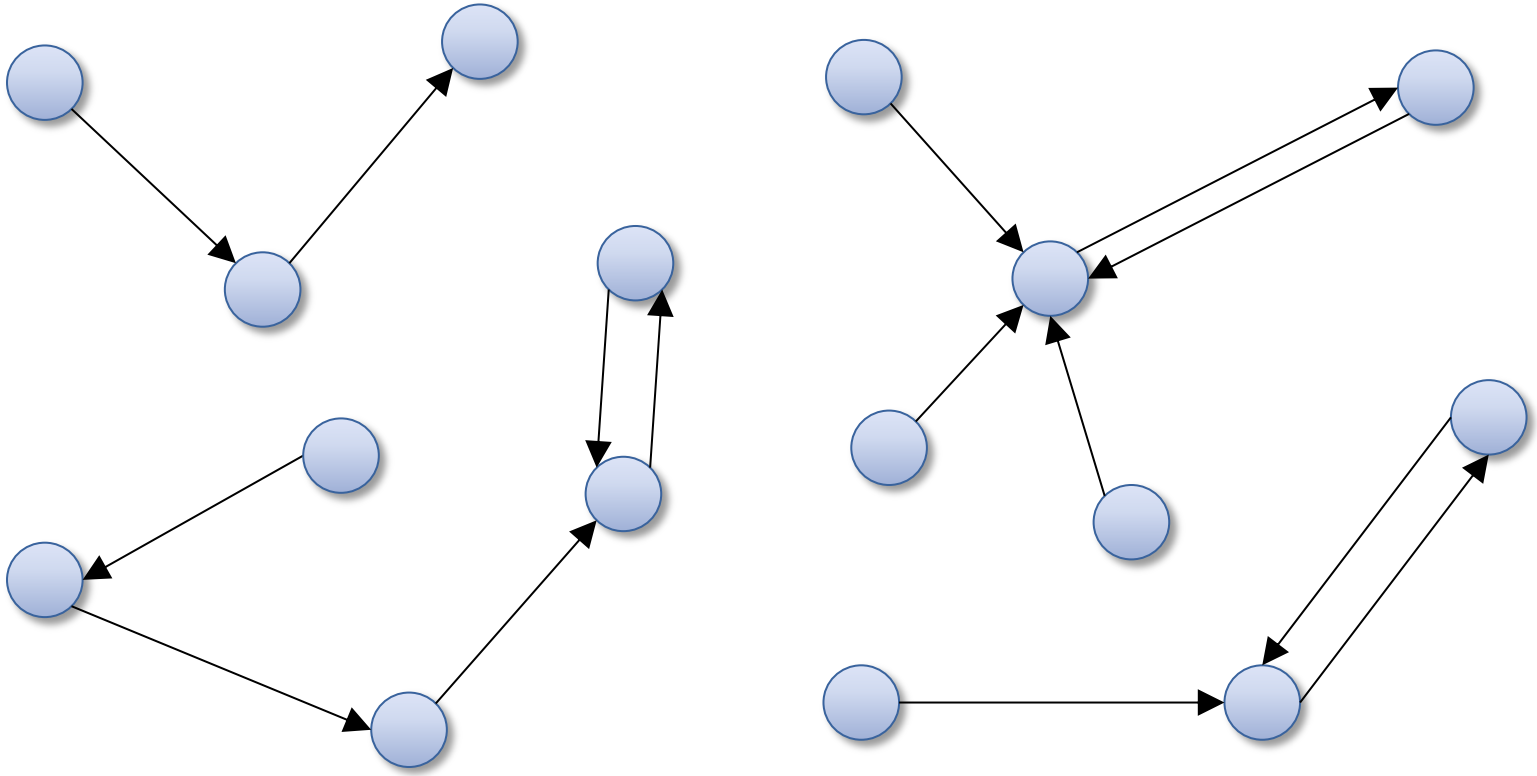


Алгоритм Борувки



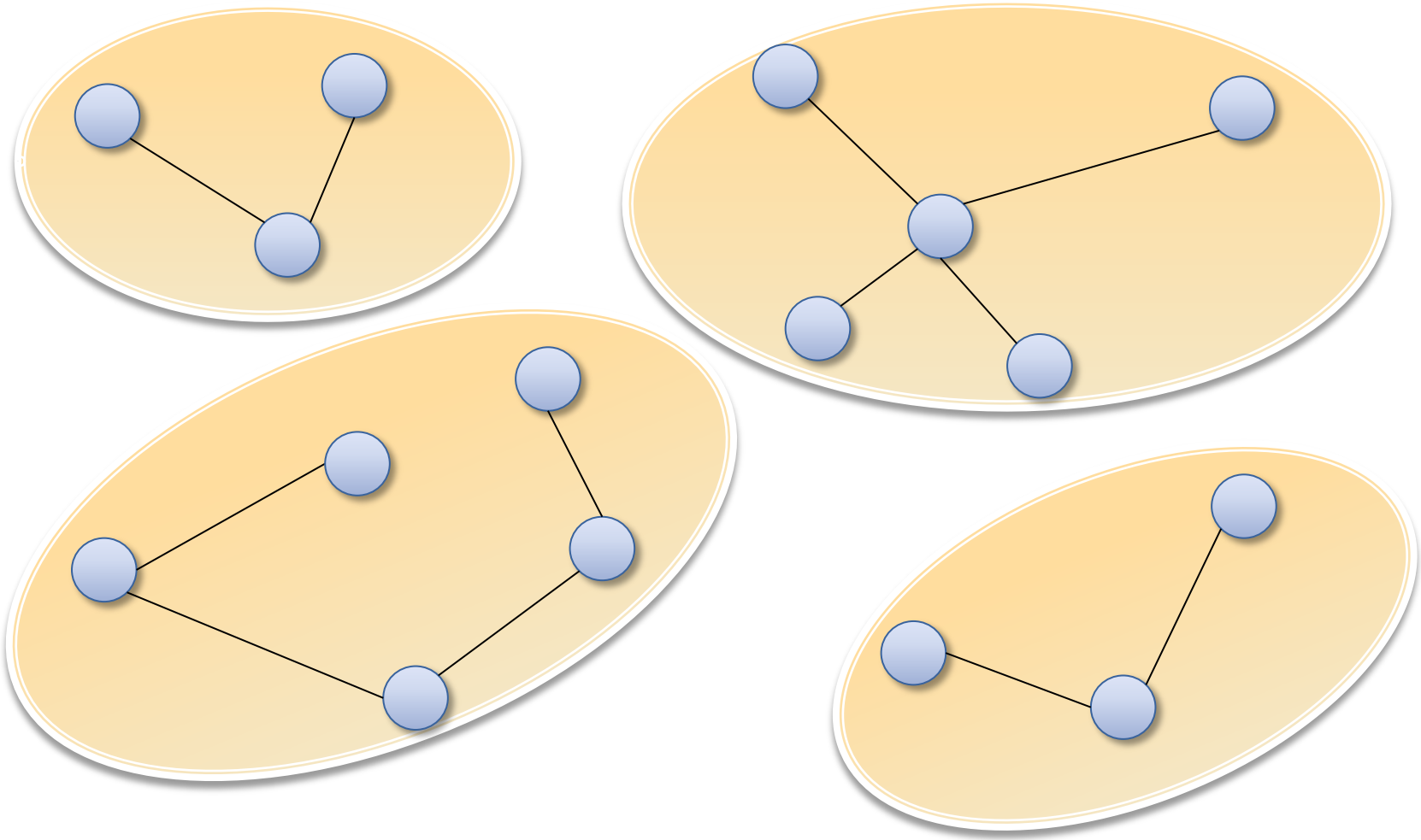
Рассматриваются ненаправленные графы с весами

Алгоритм Борувки



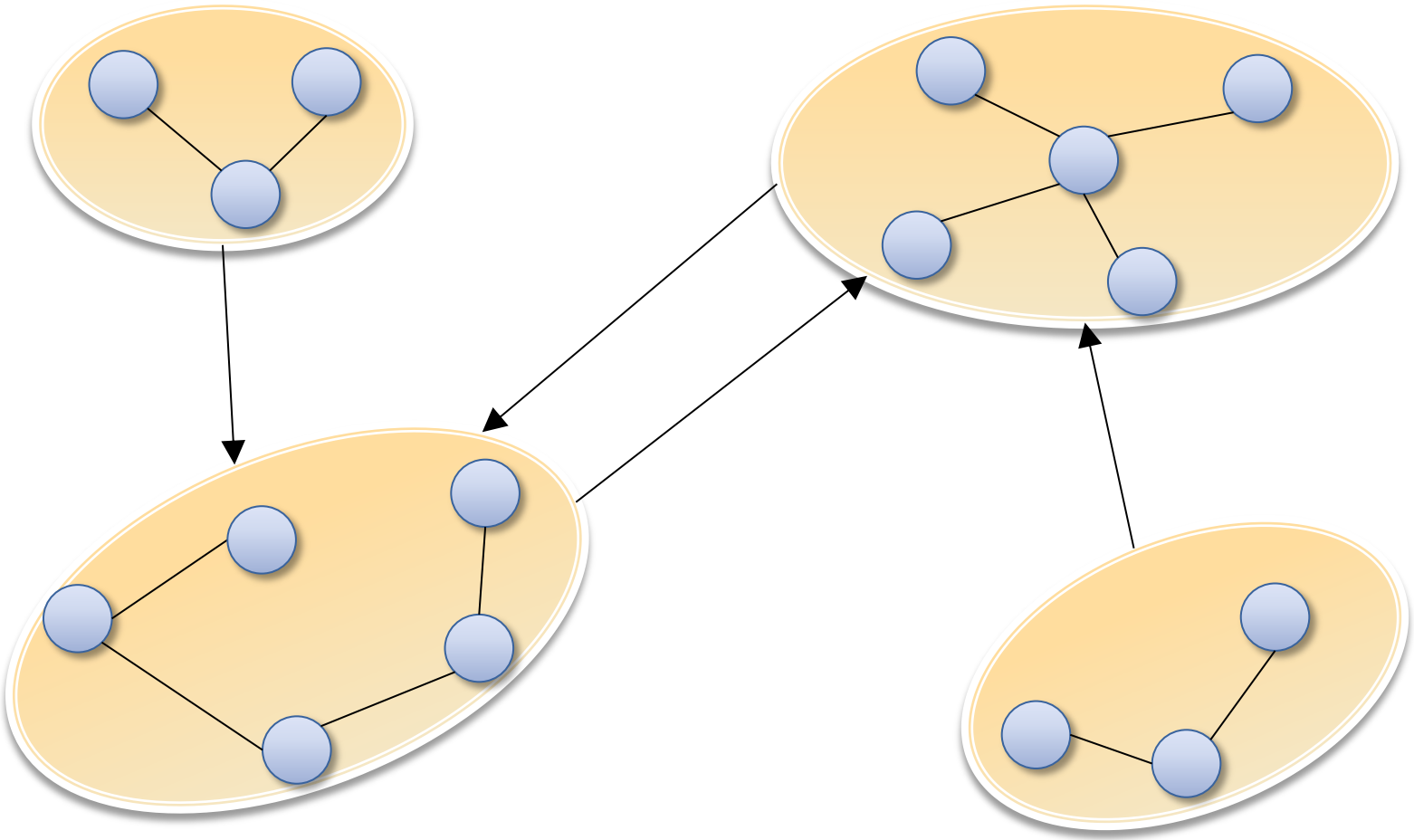
Каждая вершина ищет ребро минимального веса

Алгоритм Борувки



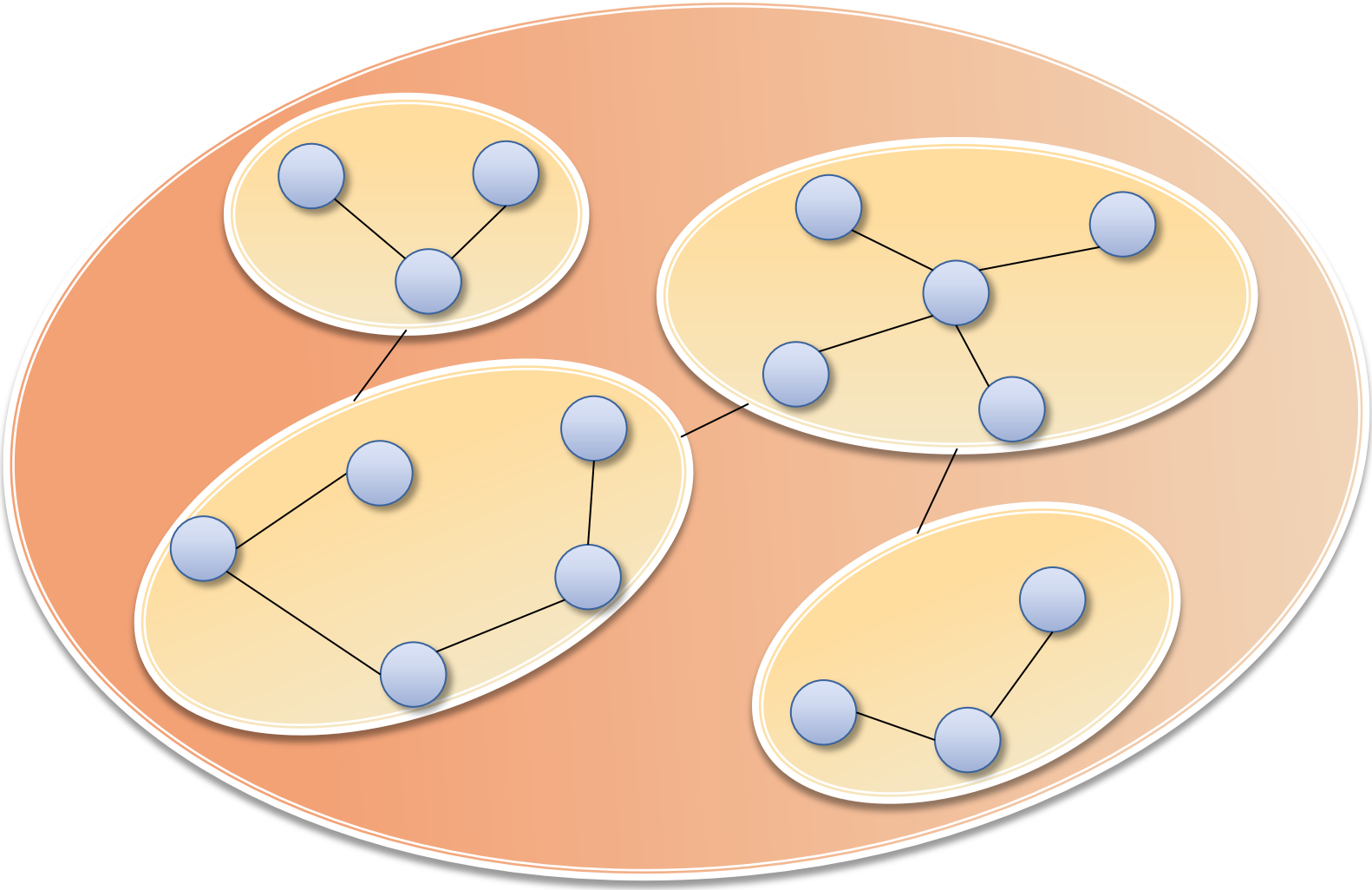
Вершины объединяются в непересекающиеся компоненты – **супервершины**

Алгоритм Борувки



Супервершины рассматриваются как обычные вершины и процесс повторяется ….

Алгоритм Борувки



…. до тех пор, пока не будет получена одна супервершина.

Общая схема алгоритма Борувки для несвязного графа

Существующие исследования

- [Зайцев] Построение MST на мультипроцессоре с ccNUMA архитектурой, **2014**;
- [HVN] Large Graph Algorithms for Massively Multithreaded Architectures, **2009**;
- [Bader and Madduri] GTgraph: A synthetic graph generator suite, **2006**;
- [Bader And Cong] A fast, parallel spanning tree algorithm for symmetric multiprocessors (SMPs), **2005**;
- [Chazelle] A minimum spanning tree algorithm with inverseAckermann type complexity, **2000**
- [Johnson And Metaxas] A parallel algorithm for computing minimum spanning trees, **1992**;
- [Blelloch] Scans as Primitive Parallel Operations, **1989**;
- [Boruvka] O Jistém Problému Minimálním (About a Certain Minimal Problem), **1926.**

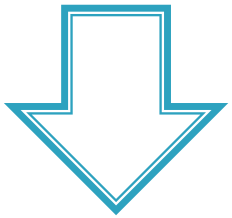
Проблемы обработки графов

- Не регулярный доступ к памяти;
- Низкая интенсивность вычислений (BFS, SSSP, MST, Community detection, и т.д.);

- Выбор формата хранения данных;
- Оптимизация алгоритмов под конкретную архитектуру (CPU, GPU, MIC, PowerPC).

Исследуемые проблемы реализации MST на GPU

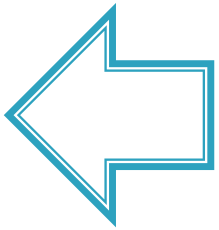
- Эффективный доступ к памяти GPU;
- Эффективная параллельная реализация на GPU;
- Попытка создания гибридной версии GPU + CPU;

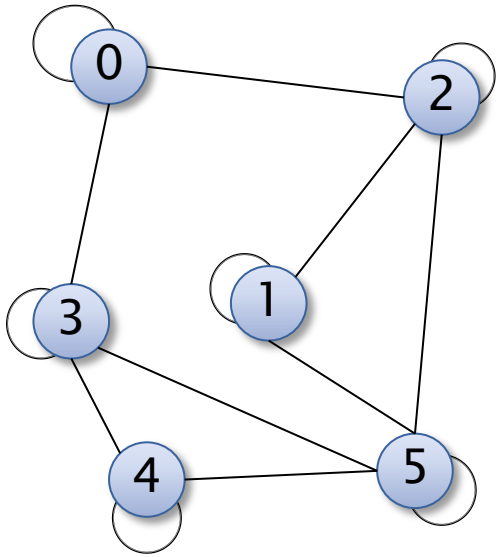


- Формат хранения данных + преобразования;
- Адаптация вычислительной части под формат данных и архитектуру.

Формат хранения графа: CSR

- CSR (Compressed Sparse Row)

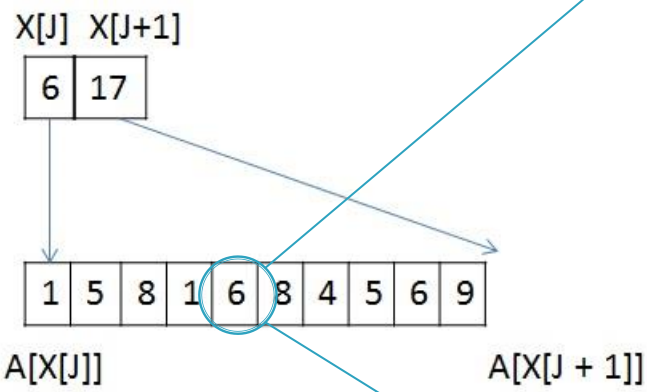




Преобразования данных CSR: свойства графа

- Локальная сортировка списка соседей по весу в порядке возрастания;
- Перенумерация вершин – наиболее связные вершины имеют наиболее близкие номера;
- Веса графа имеют 32 битный размер;
- Сжатие информации для слабо связных графов (например, транспортные сети, социальные сети).

Преобразования данных: сжатие



Без сжатия (unsigned int32):

0000 0000 0000 0000 0000 0100 1011 1000

Сжатие: base_J + 256 * k + short_endV

~~0000 0000 0000 0000 0000 0011 1011 1000~~

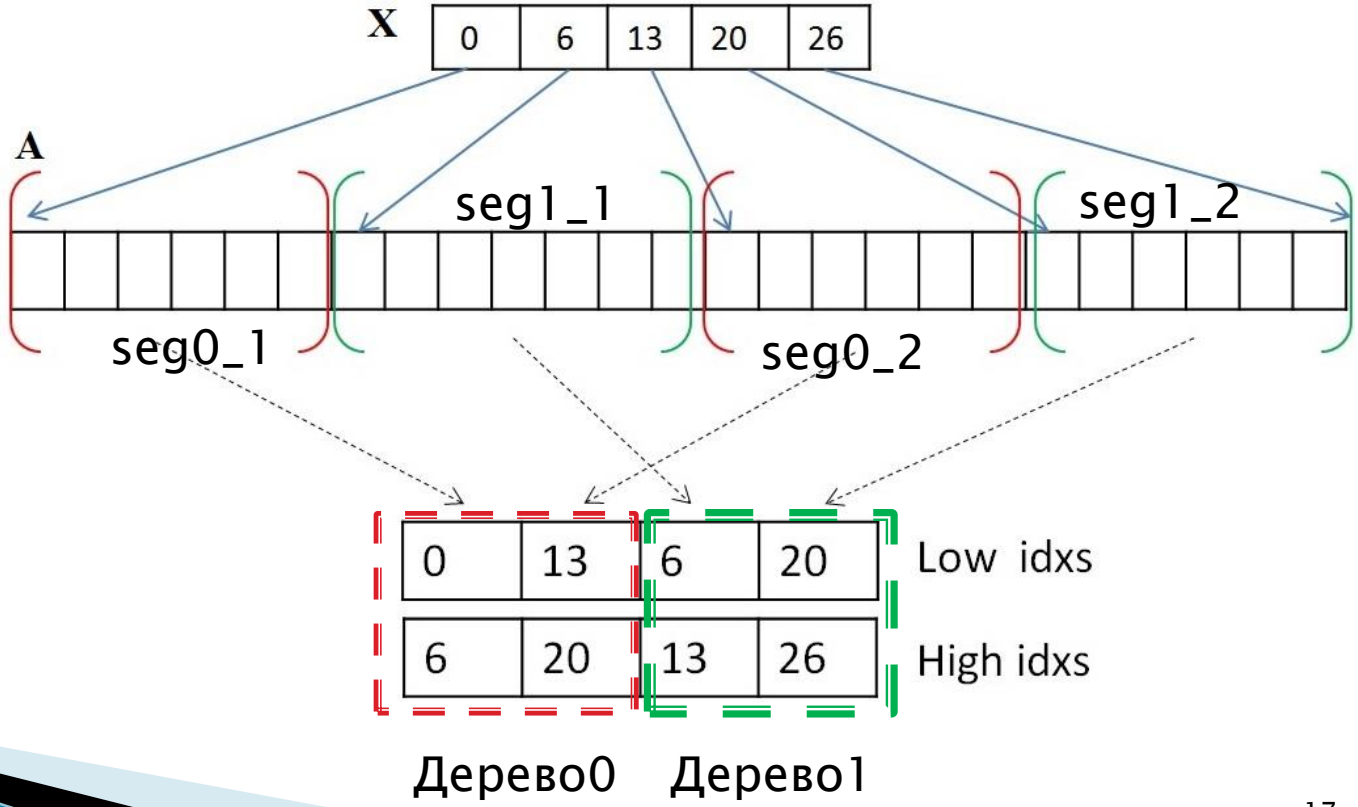
Параллельный алгоритм Борувки

- Шаг1. Найти минимальные ребра между всеми деревьями для их последующего объединения;
- Шаг2. Выполнить объединение деревьев: <u>удаление циклов</u> + <u>объединение;</u>
- Шаг3. Выполнить перенумерацию полученных деревьев (номера от 0 до k);

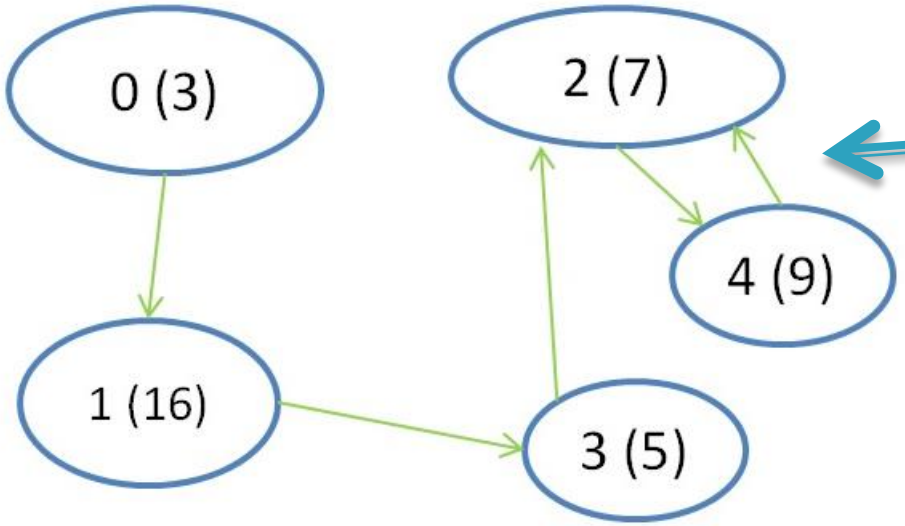
Параллельный алгоритм: Шаг1

- Выбор минимального ребра исходящего из каждого сегмента **супервершины**;
- Выбор ребра минимального веса **супервершины**.

CSR:

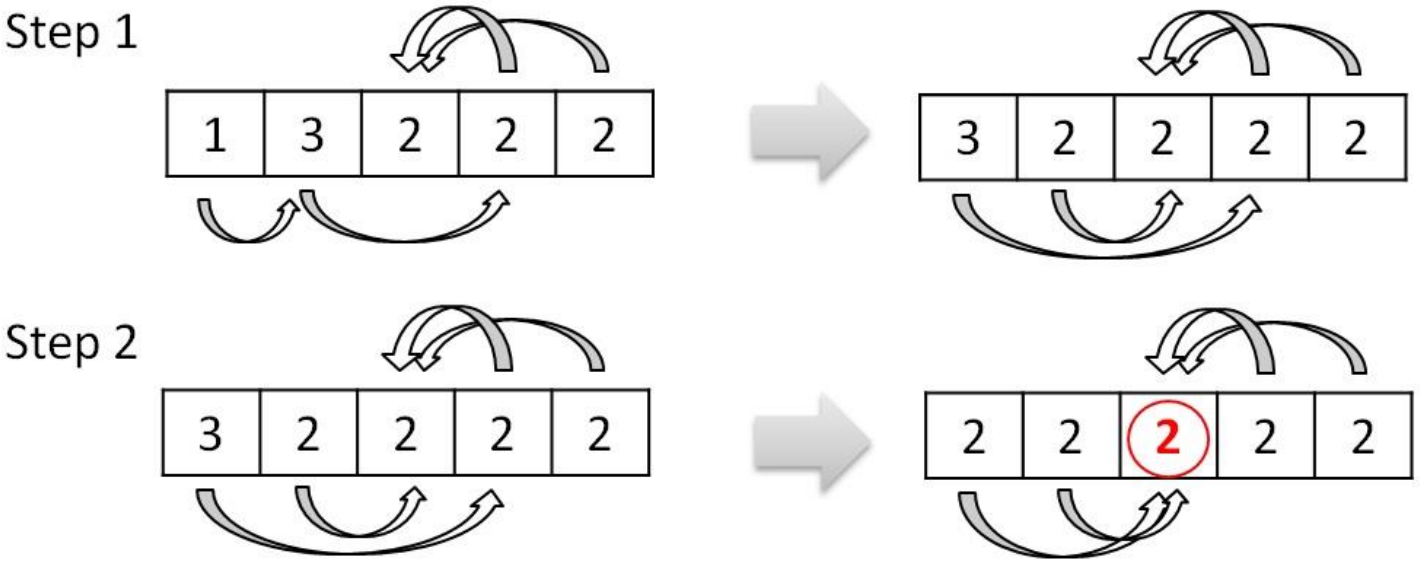


Параллельный алгоритм: Шаг2



1. Удаление циклов

2. Объединение деревьев



Параллельный алгоритм: Шаг3

- Перенумерация вершин после объединения, обновление всех структур данных.

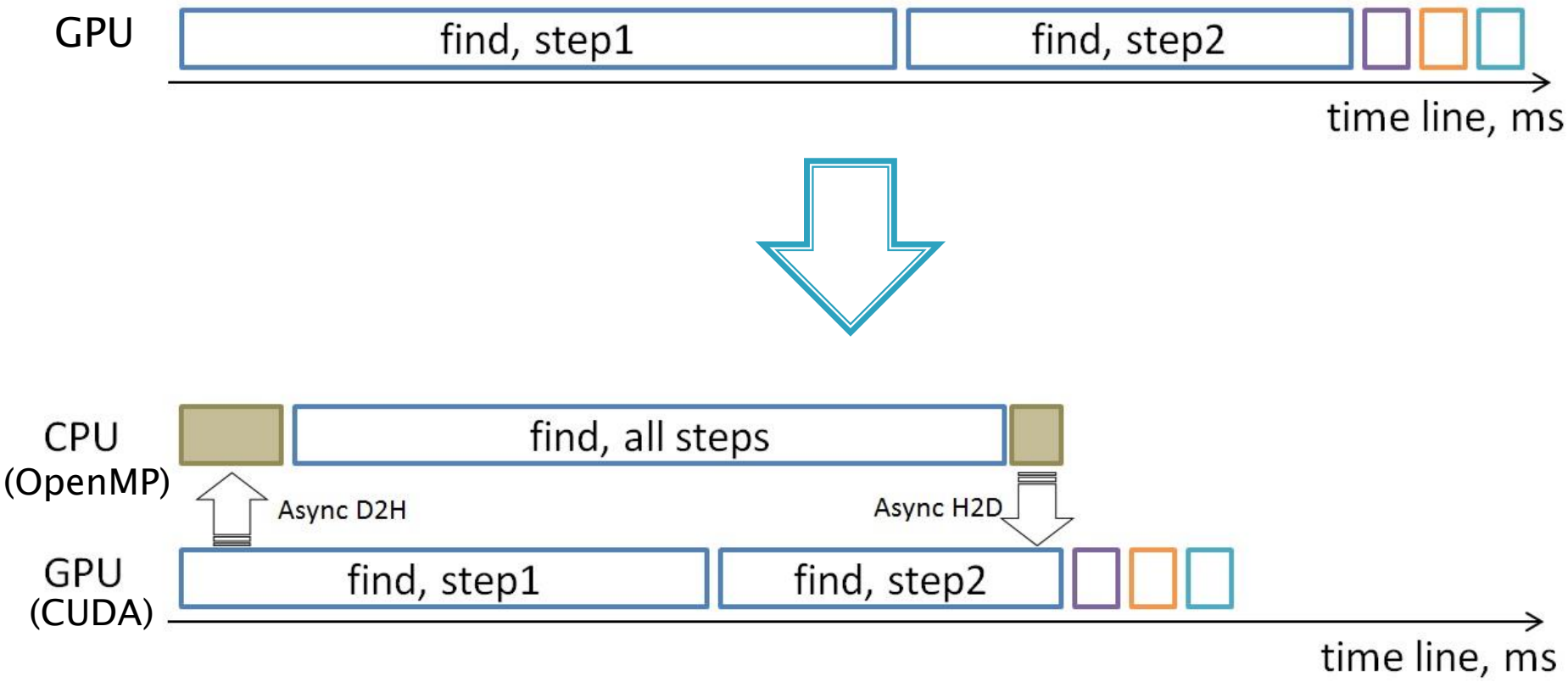
2	2	**2**	2	2	8	8	8	**8**	8	8	12	**12**	12	12	12	18	18	**18**	18

F[i] = i;

atomic renumbering

1	1	**1**	1	1	2	2	2	**2**	2	2	3	**3**	3	3	3	4	4	**4**	4

Гибридная реализация: CPU + GPU одна итерация алгоритма



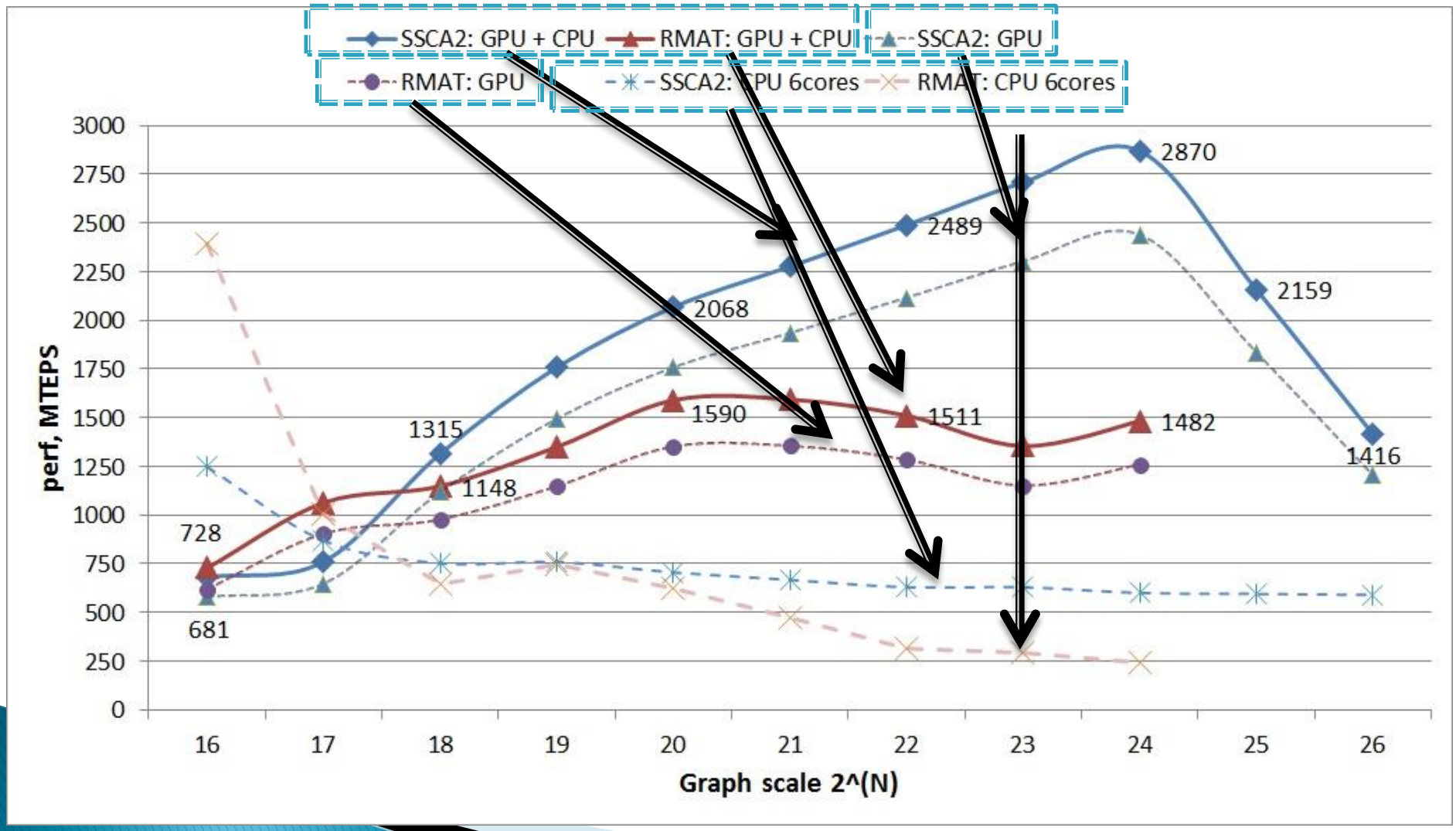
Тестирование алгоритма

- Платформа –
  CPU 6 core Intel Xeon E5 1660v2 3.7 ГГц,
  GPU GTX Titan (Kepler);

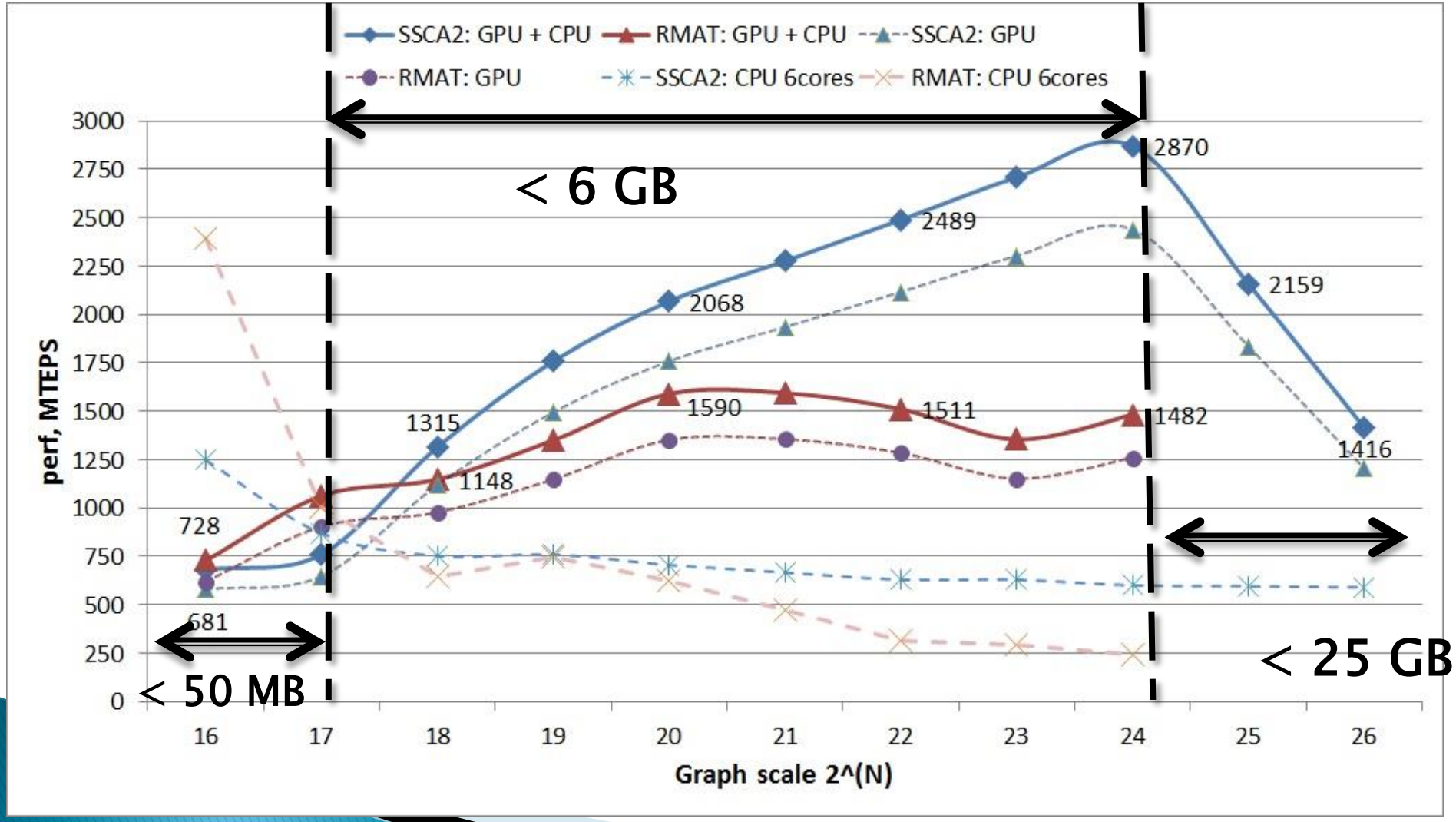
- Графы –
  RMAT (A Recursive Model for Graph Mining) и
  SSCA2 (Scalable Synthetic Compact Applications);

- Производительность алгоритма в TEPS
  (Traversed Edges Per Second).

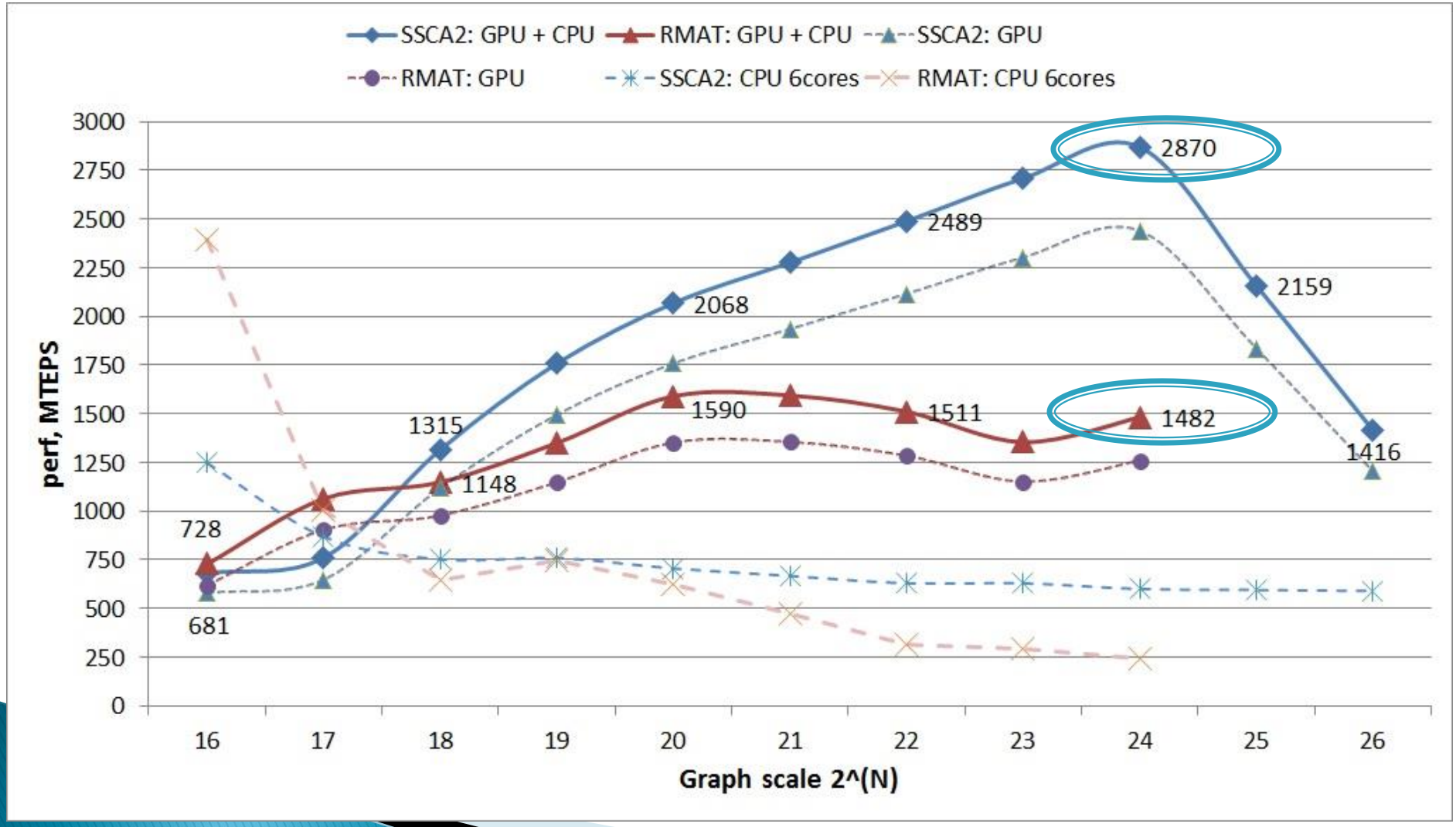
Результаты


SSCA2: GPU + CPU
RMAT: GPU + CPU
SSCA2: GPU
RMAT: GPU
SSCA2: CPU 6cores
RMAT: CPU 6cores
perf, MTEPS
3000
2750
2500
2250
2000
1750
1500
1250
1000
750
500
250
0
16
17
18
19
20
21
22
23
24
25
26
Graph scale 2^(N)
2870
2489
2159
2068
1590
1511
1482
1416
1315
1148
728
681

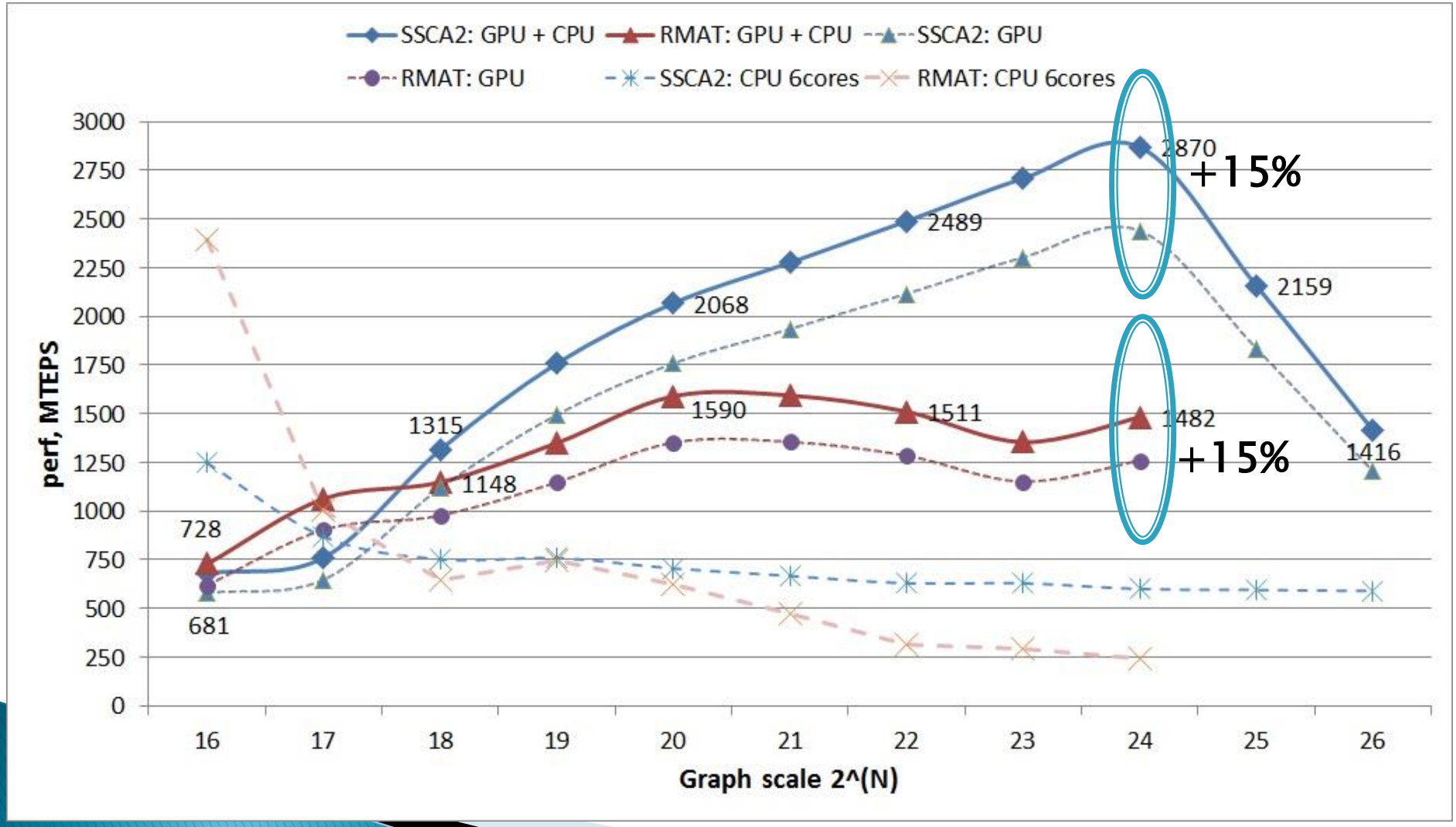
Результаты: размеры графов



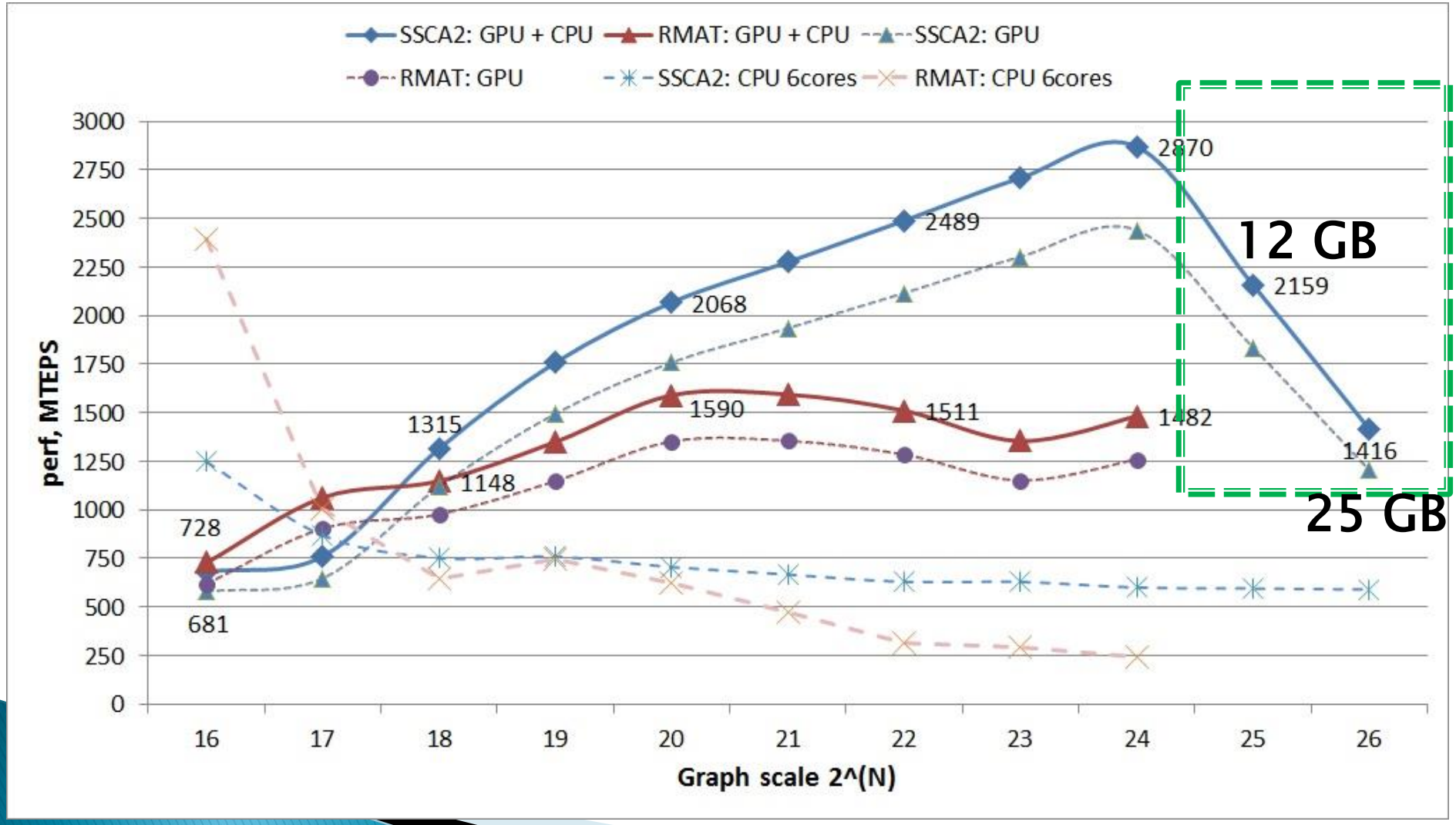
Результаты:


SSCA2: GPU + CPU
RMAT: GPU + CPU
SSCA2: GPU
RMAT: GPU
SSCA2: CPU 6cores
RMAT: CPU 6cores
perf, MTEPS
3000
2750
2500
2250
2000
1750
1500
1250
1000
750
500
250
0
16
17
18
19
20
21
22
23
24
25
26
Graph scale 2^(N)
2870
2489
2159
2068
1590
1511
1482
1416
1315
1148
728
681

Результаты:
+15% при использовании CPU+GPU



Результаты: графы SSCA2 до 25 ГБ (CUDA UMA)



Существующие решения на GPU

- **Полученные результаты: ~2000 MTEPS;**

- Конкурс GraphHPC 2015, GPU, **~1100 MTEPS;**
- Scalable Parallel Minimum Spanning Forest Computation, 2012, **~40 MTEPS;**
- GPU-Based Fast Minimum Spanning Tree Using Data Parallel Primitives, 2010;
- Fast Minimum Spanning Tree for Large Graphs on the GPU, 2009, **~36 MTEPS;**

Выводы

- Получены высокие результаты: **2800 MTEPS** на SSCA2 и **1500 MTEPS** на RMAT;
- Тестировались большие графы до 25 ГБ;
- Исследованы подходы к оптимизации работы с памятью GPU в графовых задачах;
- Разработано оптимальное представление графа в памяти GPU;
- Оптимизирован алгоритм Борувки для GPU;
- Реализована гибридная версия алгоритма CPU + GPU.

Спасибо за внимание !

Александр Колганов, ВМК МГУ
alexander.k.s@mail.ru