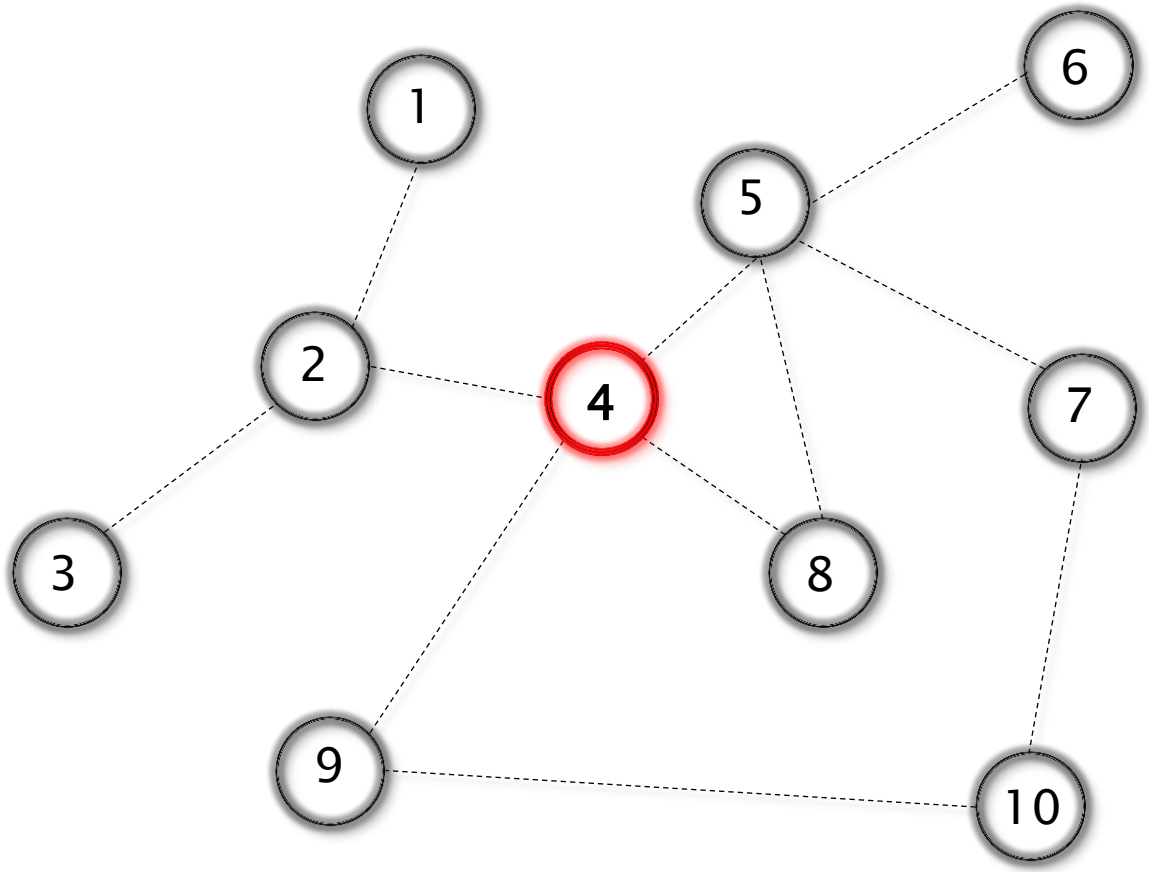
Реализация алгоритма поиска в ширину для кластеров с графическими ускорителями.

Колганов А.С.

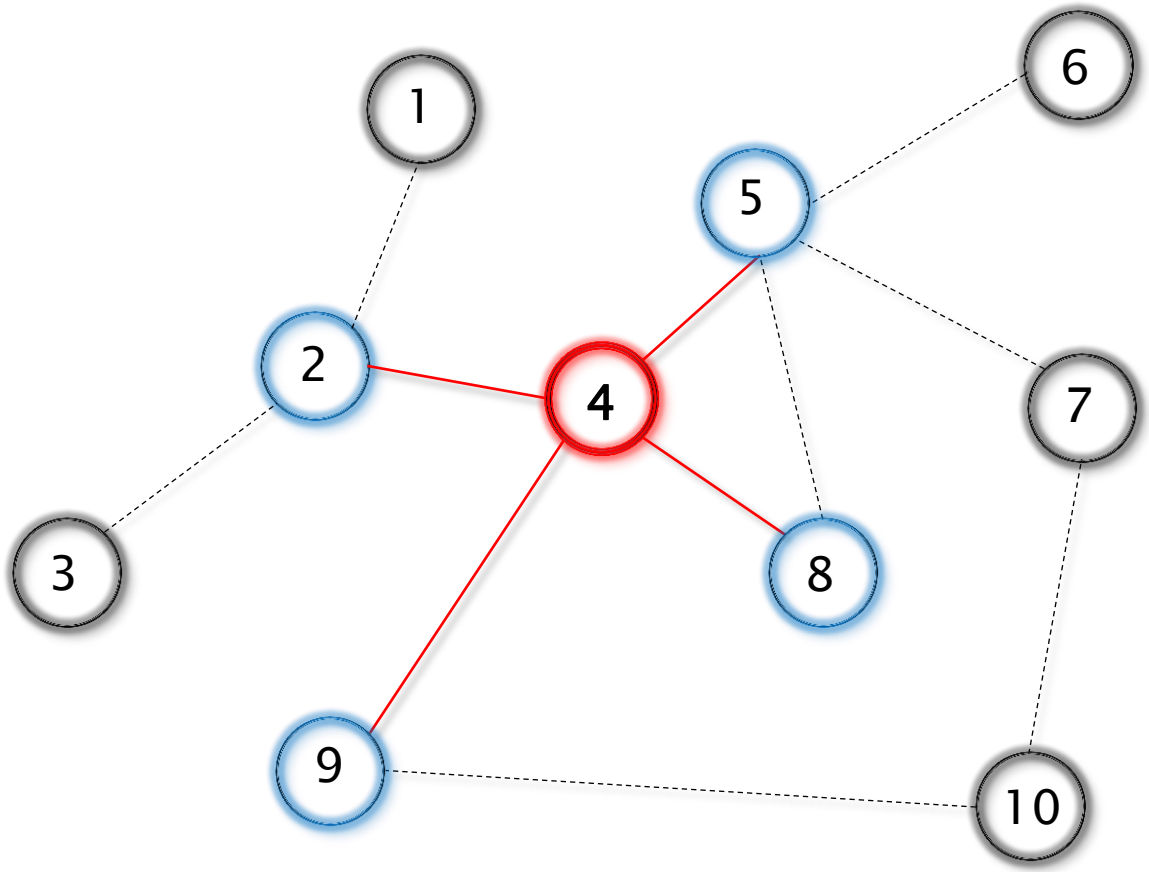
План доклада

- Задача поиска в ширину (Breadth First Search, BFS);
- Реализация алгоритма на одном GPU;
- Представление данных;
- Отображение на кластер;
- Оптимизации;
- Результаты.

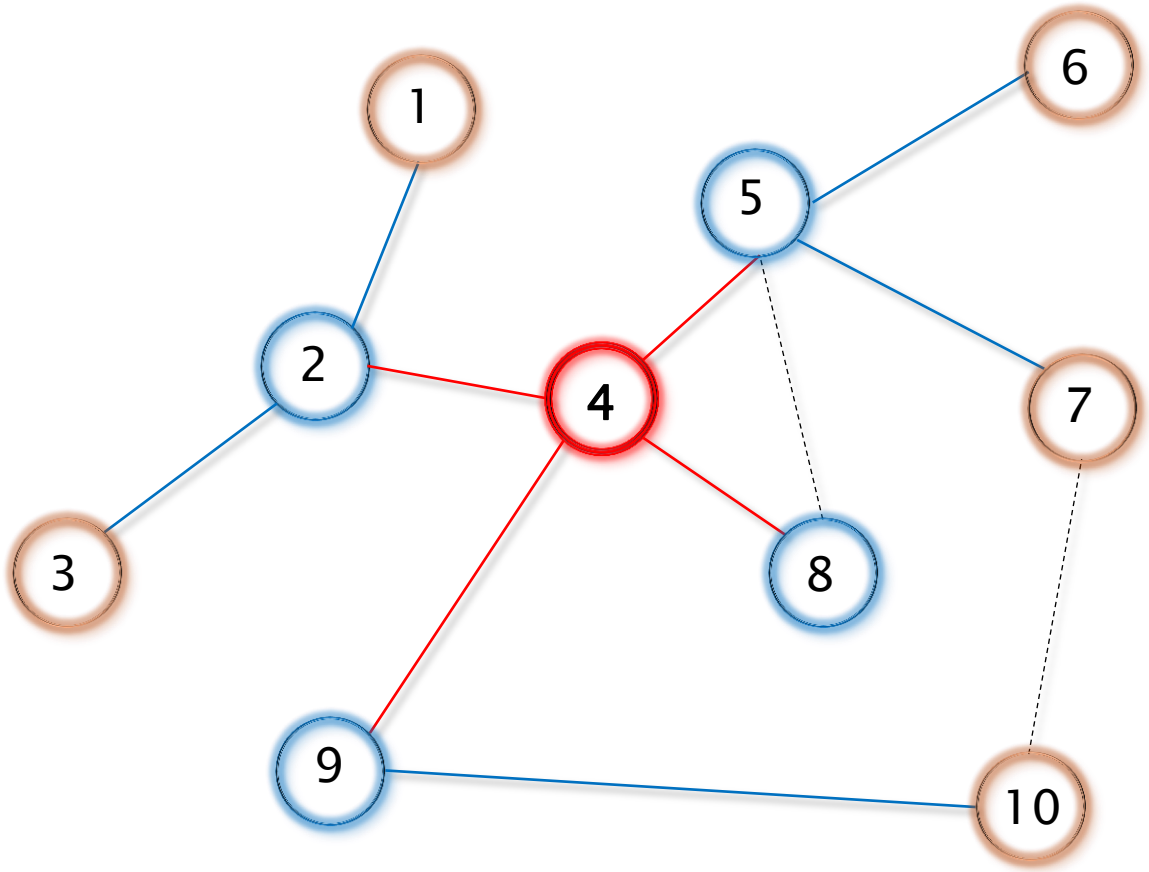
Breadth First Search



Breadth First Search STEP 1



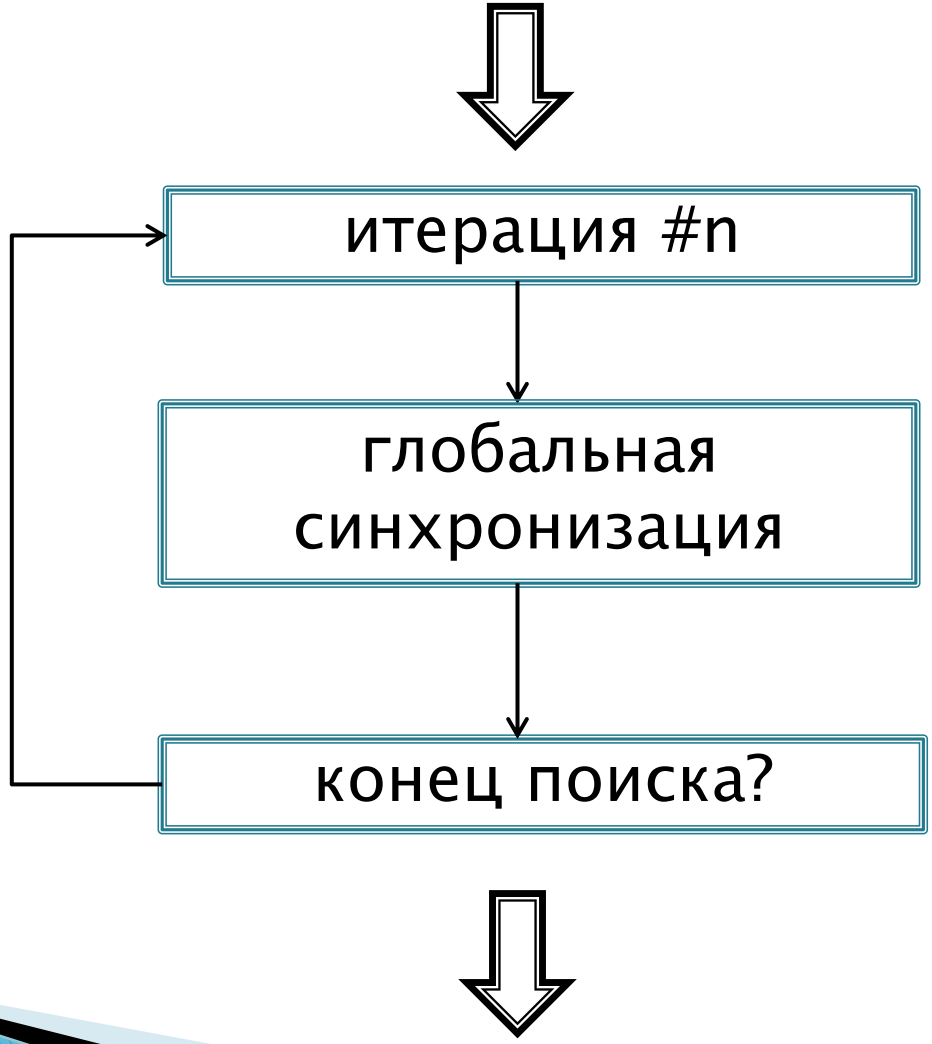
Breadth First Search STEP 2



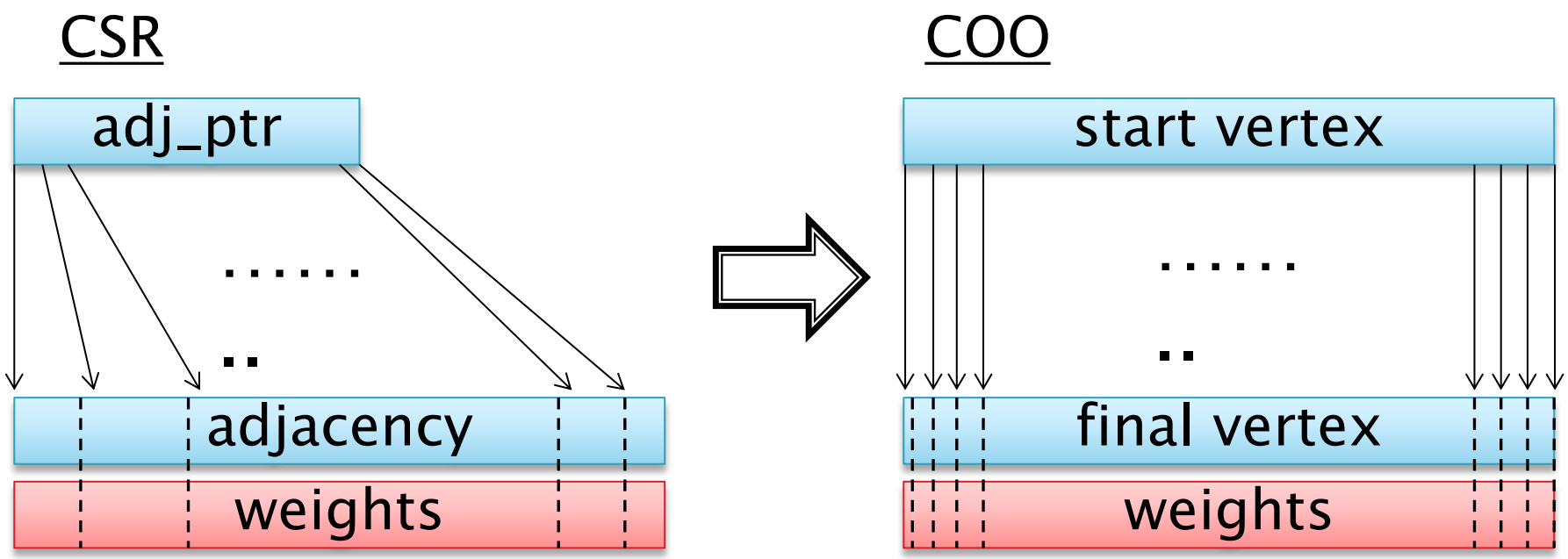
Идеи реализации алгоритма BFS

- Итерационная разметка уровней;
- Преобразование формата хранения данных;
- Оптимизации в работе с памятью GPU;
- Оптимизации межпроцессорных взаимодействий.

Алгоритм выполнения BFS

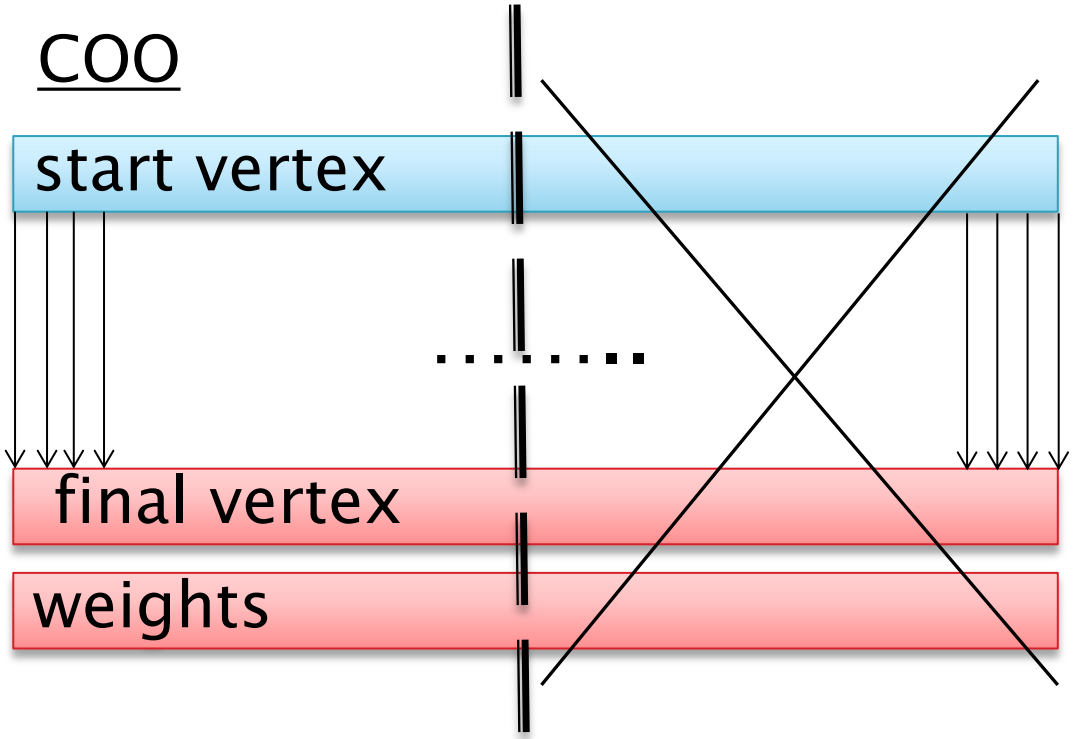


Представление данных



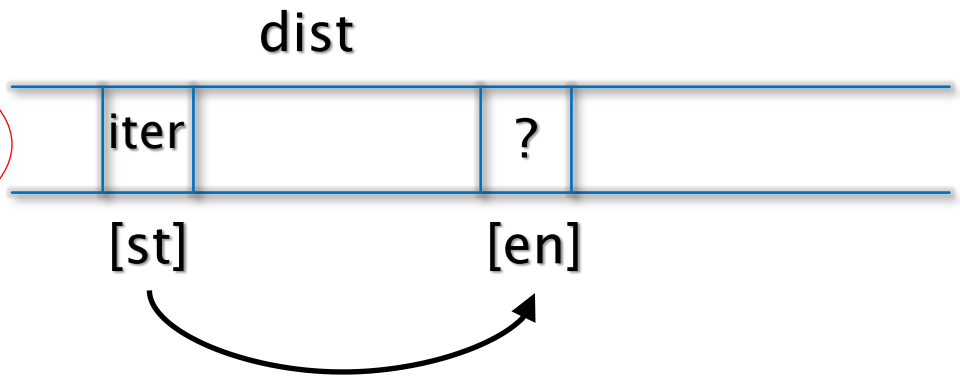
Удаление дубликатов

- Формат CSR содержит обе дуги (S, F) и (F, S), необходимо оставить одну – (S, F) или (F, S).



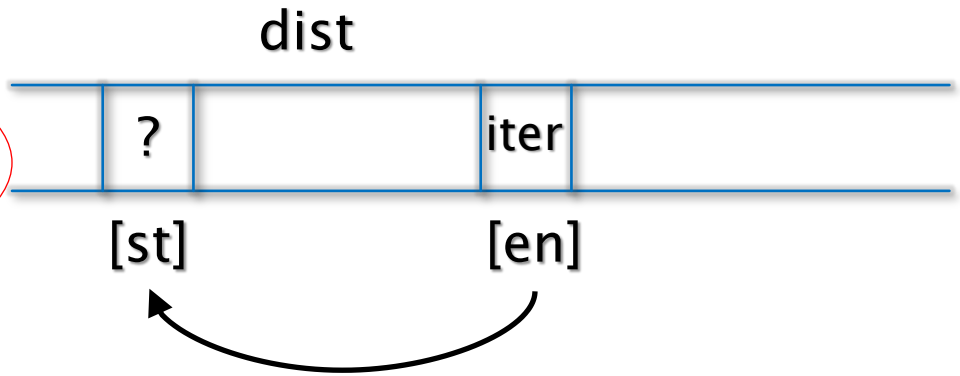
Основная итерация алгоритма

```
if(k < maxV)
{
    unsigned st = startV[k]
    unsigned en = endV[k];

    if(dist[st] == iter)
    {
        if(dist[en] > iter)
            dist[en] = iter + 1;
    }
    else if(dist[en] == iter)
    {
        if(dist[st] > iter)
            dist[st] = iter + 1;
    }
}
```



Основная итерация алгоритма

```
if(k < maxV)
{
    unsigned st = startV[k]
    unsigned en = endV[k];

    if(dist[st] == iter)
    {
        if(dist[en] > iter)
            dist[en] = iter + 1;
    }
    else if(dist[en] == iter)
    {
        if(dist[st] > iter)
            dist[st] = iter + 1;
    }
}
```



Основная итерация алгоритма

```
if(k < maxV)
{
    unsigned st = startV[k]
    unsigned en = endV[k];

    if(dist[st] == iter)
    {
        if(dist[en] > iter)
            dist[en] = iter + 1;
    }
    else if(dist[en] == iter)
    {
        if(dist[st] > iter)
            dist[st] = iter + 1;
    }
}
```

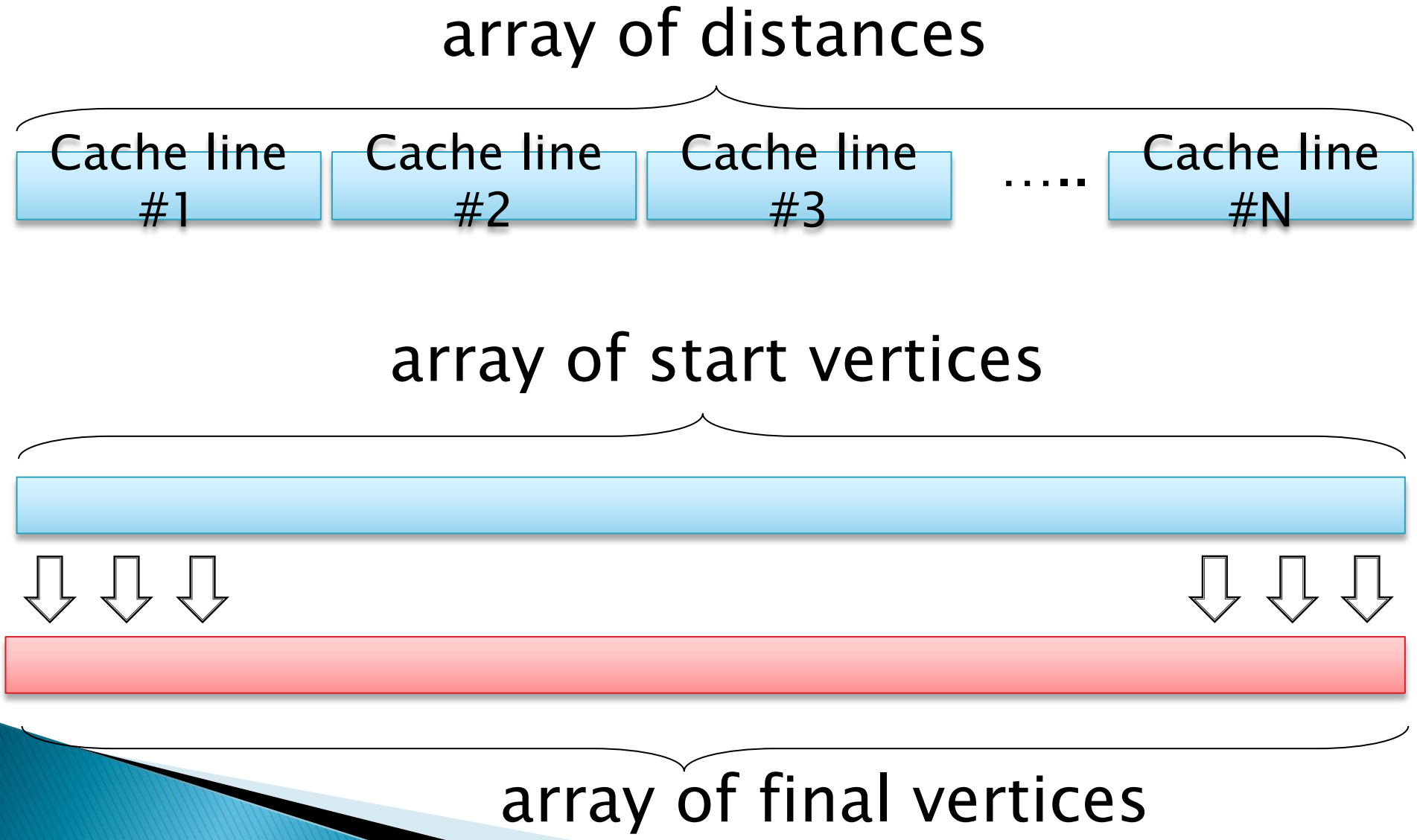
Косвенная адресация !

Оптимизации алгоритма

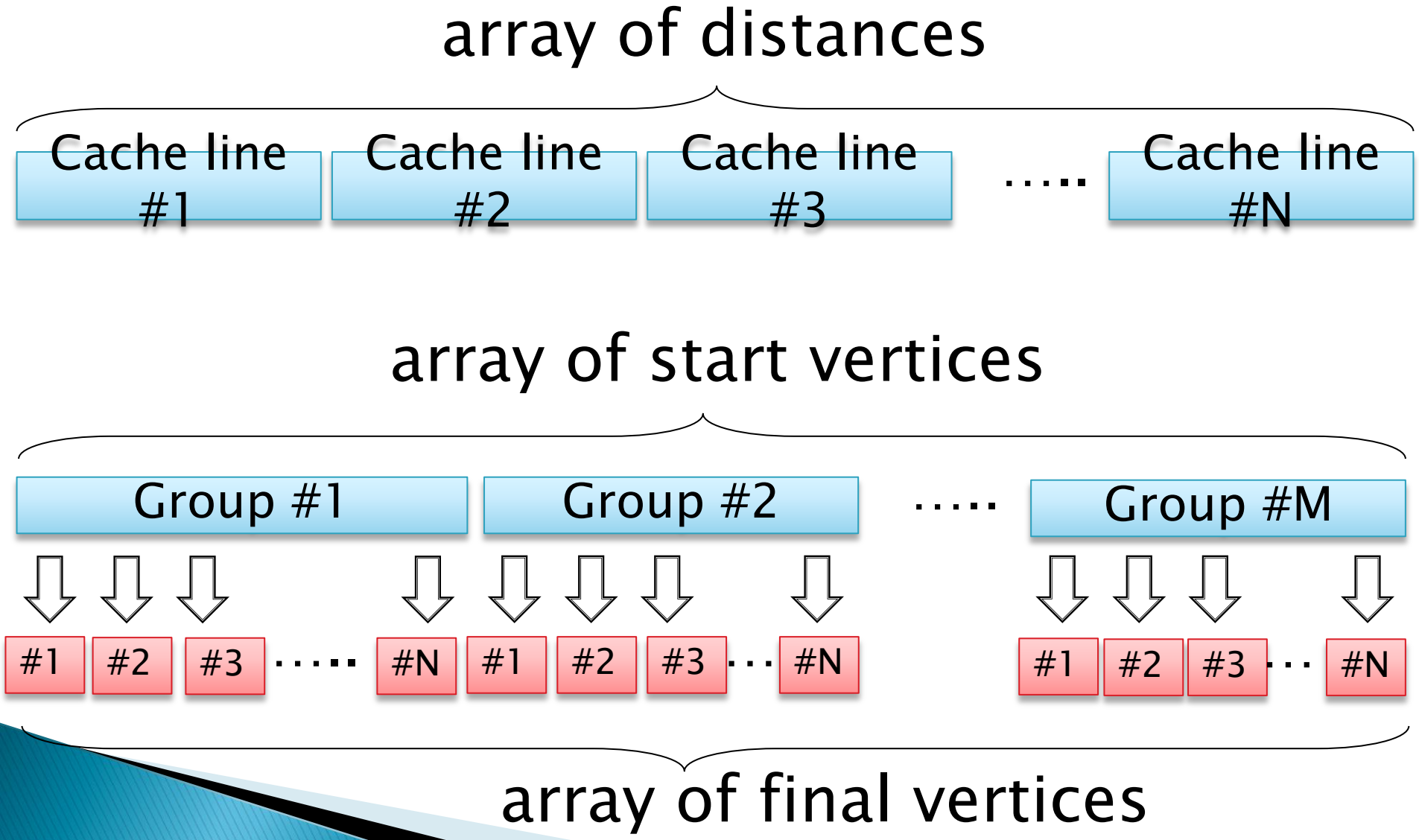
- Переупорядочивание данных для эффективного использования кэша и пропускной способности памяти GPU;

- Кэширование массива расстояний dist через текстурный кэш с помощью const __restrict.
  (поддерживается только для архитектуры Kepler и выше)

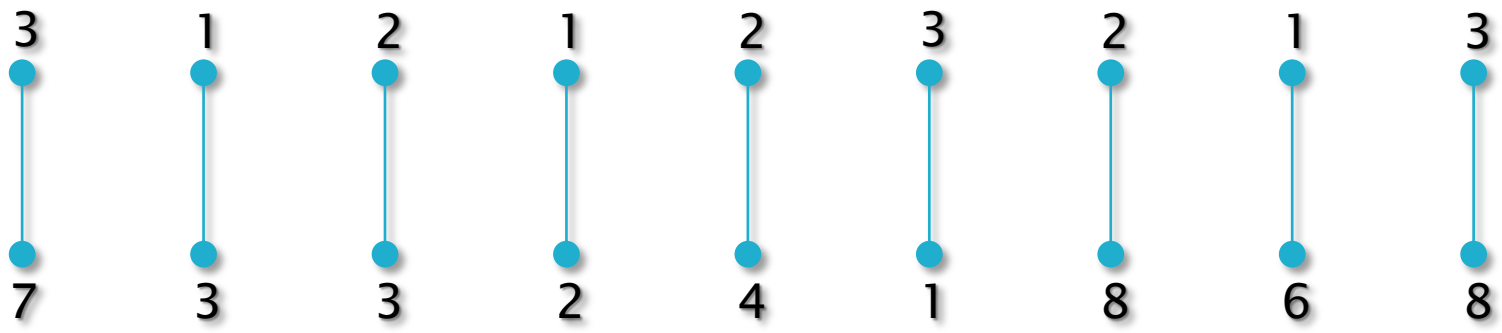
Переупорядочивание данных



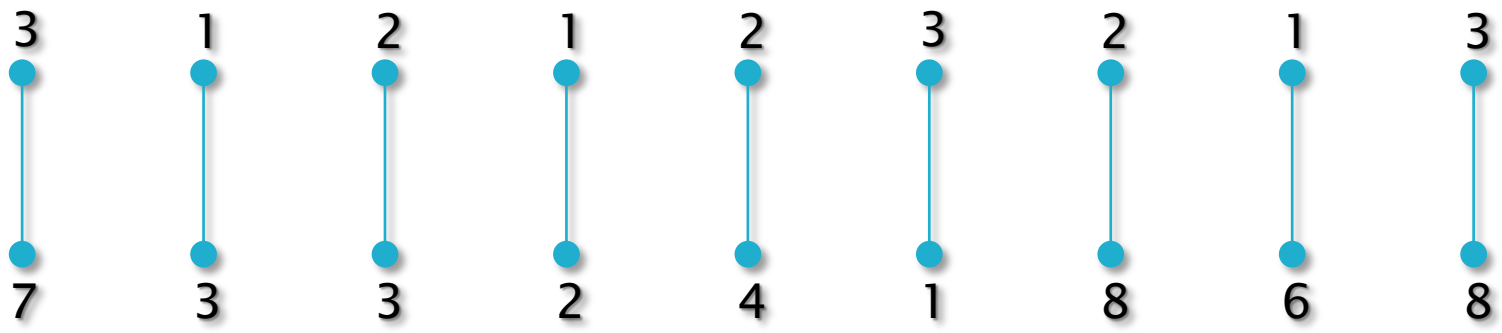
Переупорядочивание данных



Пример переупорядочивания



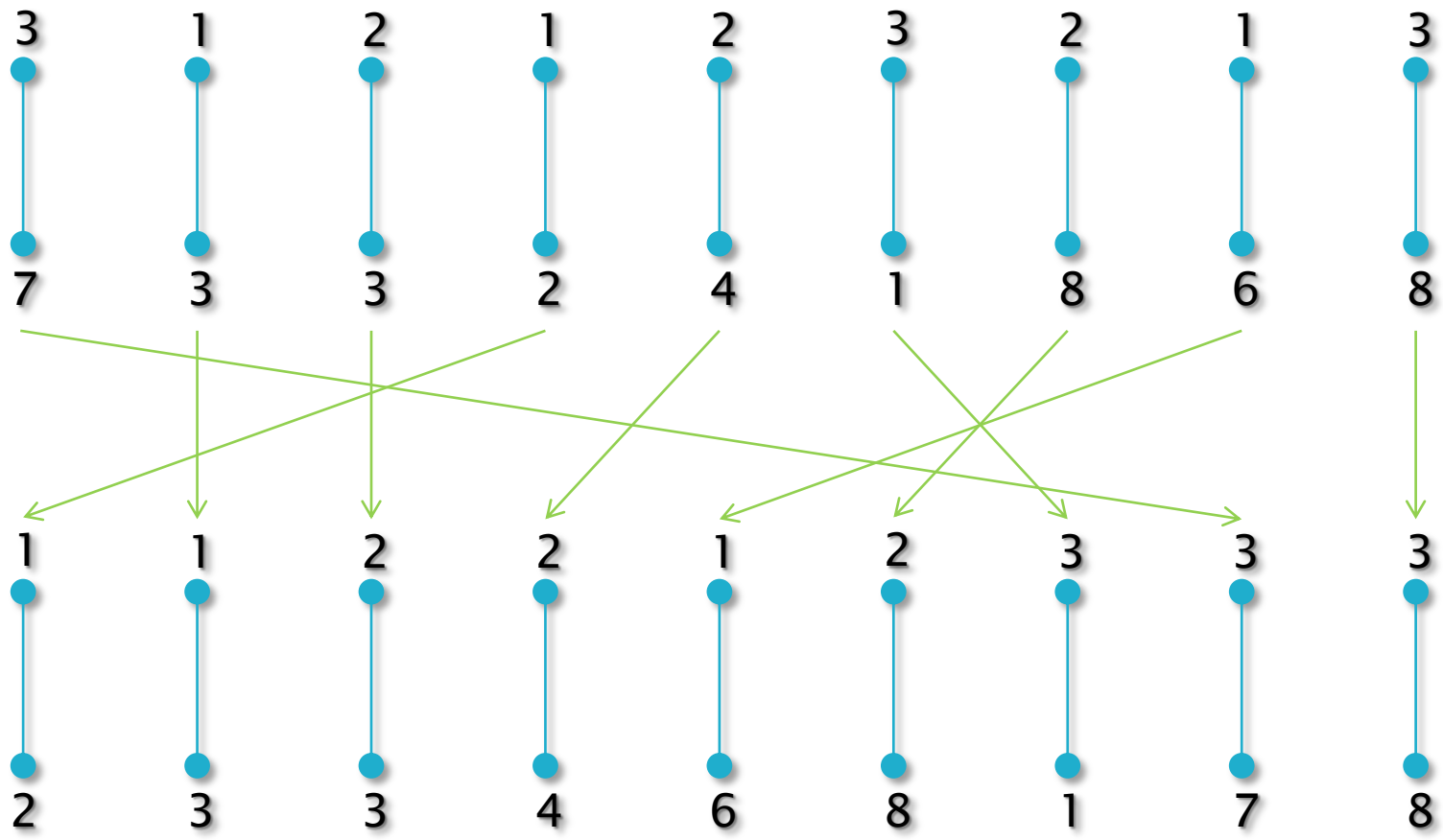
Пример переупорядочивания



Cache Line = 2

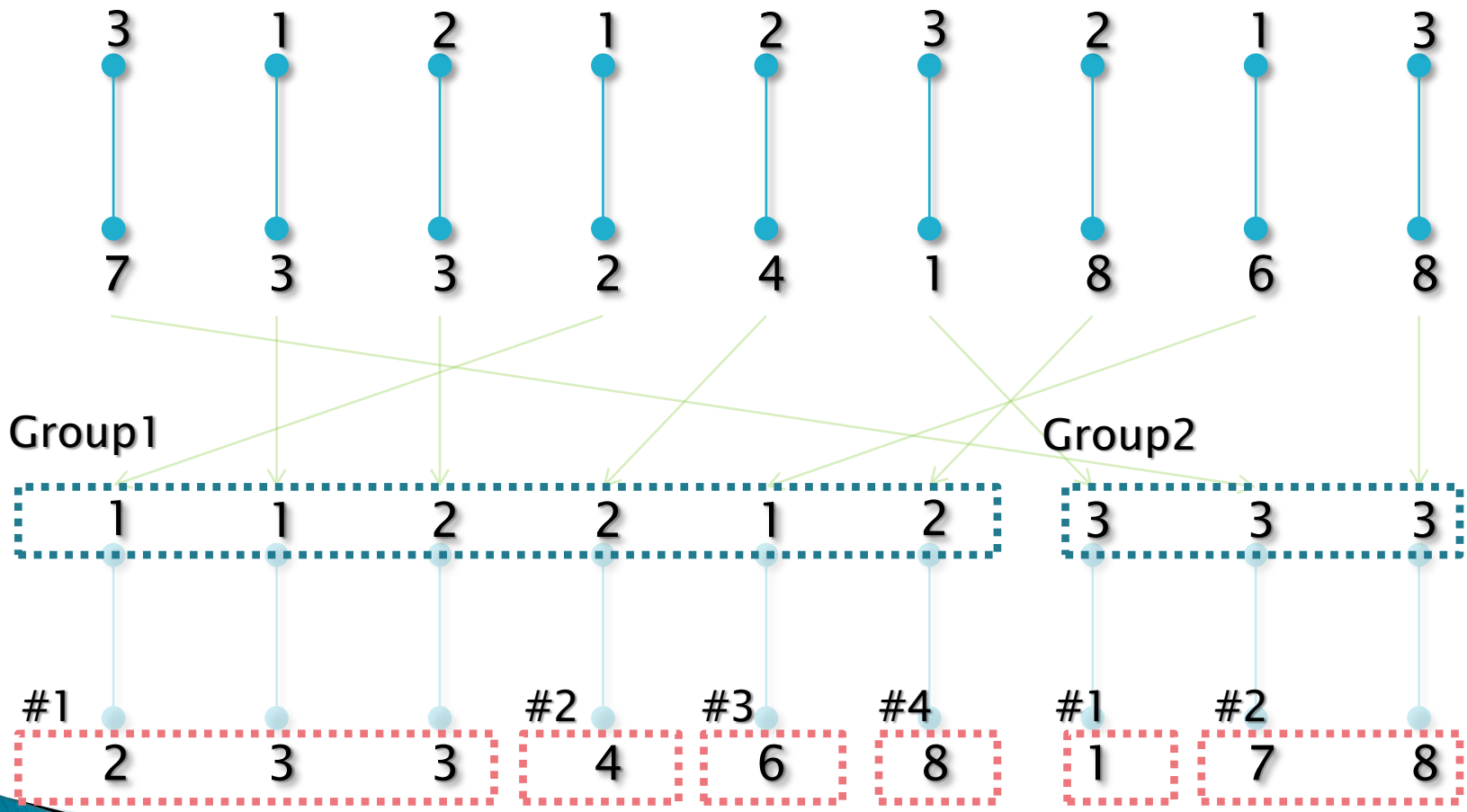
Пример переупорядочивания

Cache Line = 2

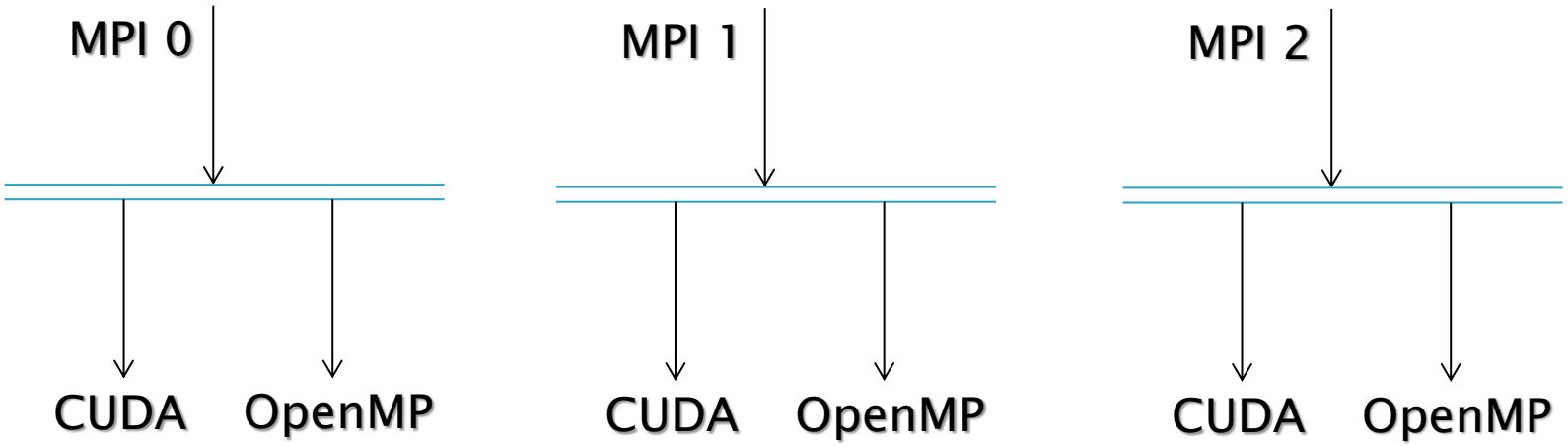


Пример переупорядочивания

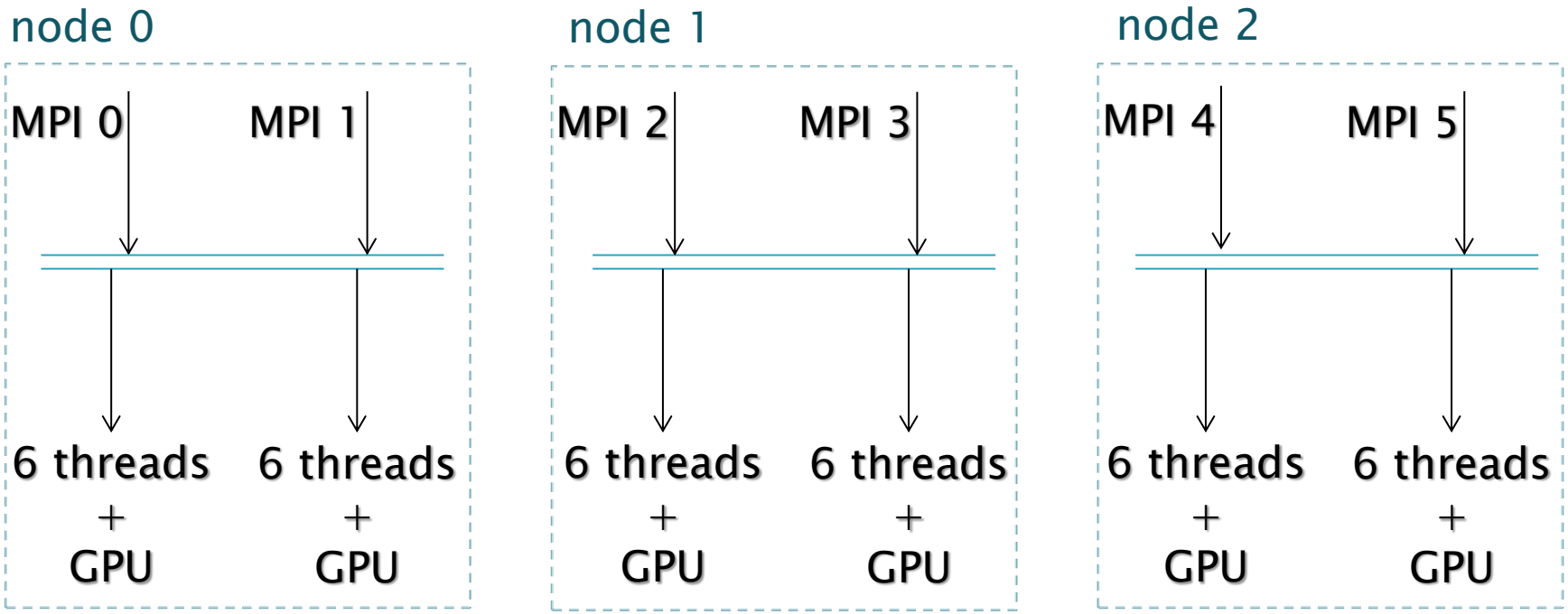
Cache Line = 2



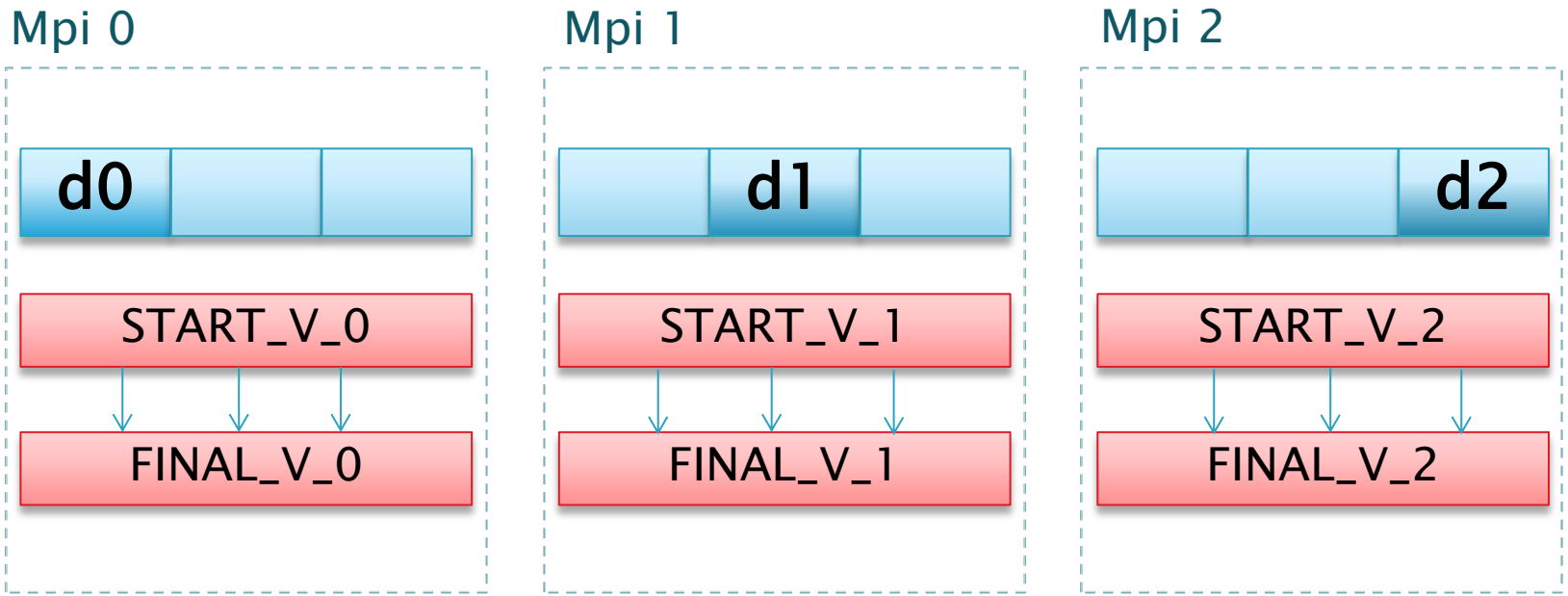
Отображение на гетерогенный кластер



Отображение на гетерогенный кластер K100

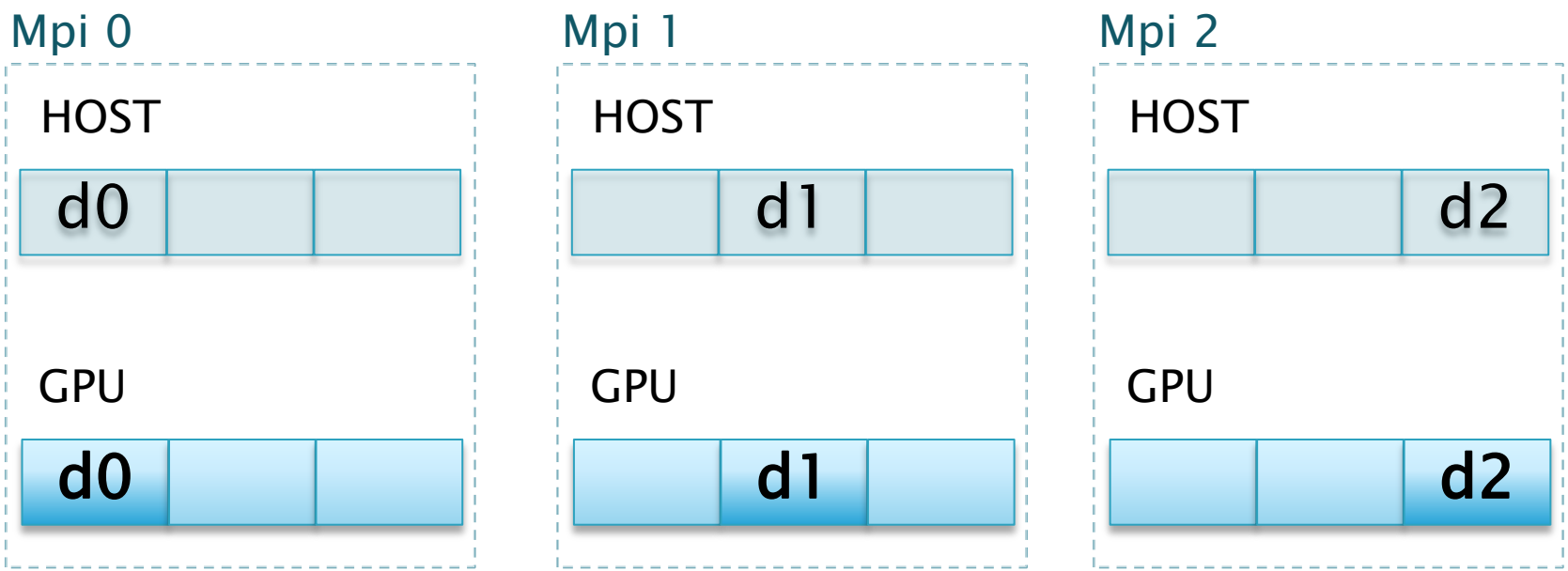


Разделение данных между процессами



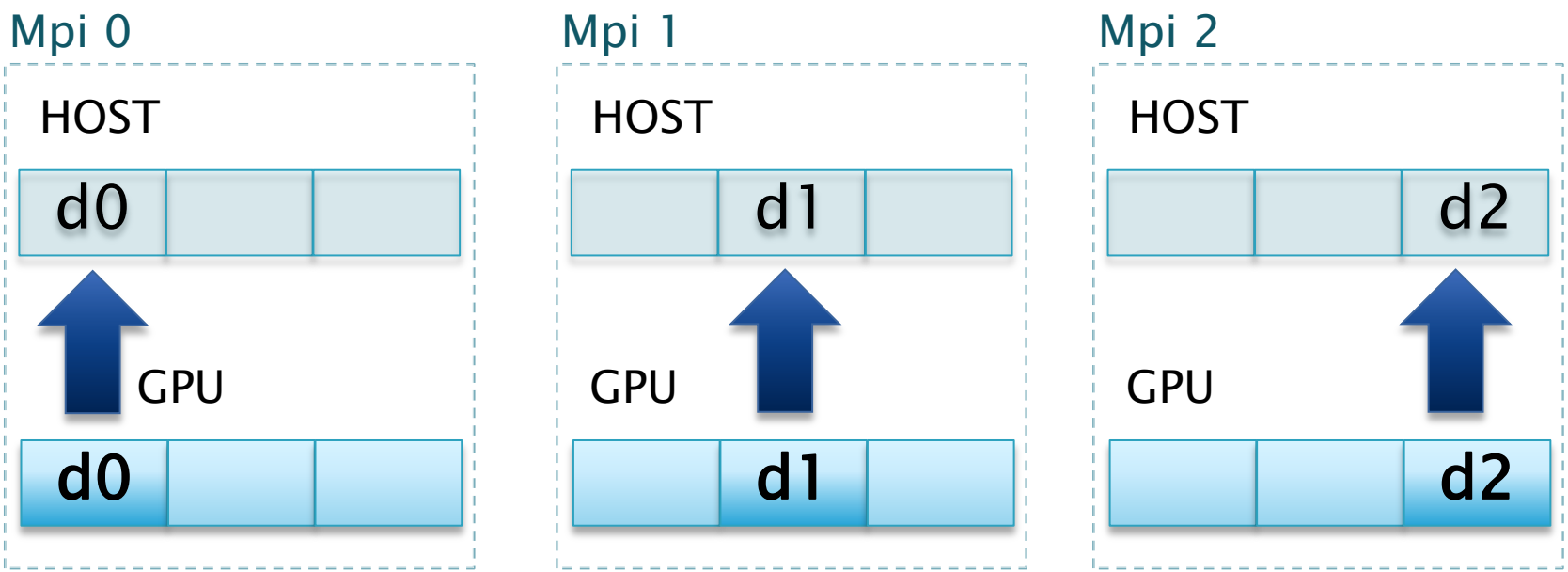
Организация итерации алгоритма

- Использование top-down подхода для разметки части массива расстояний (d_i);



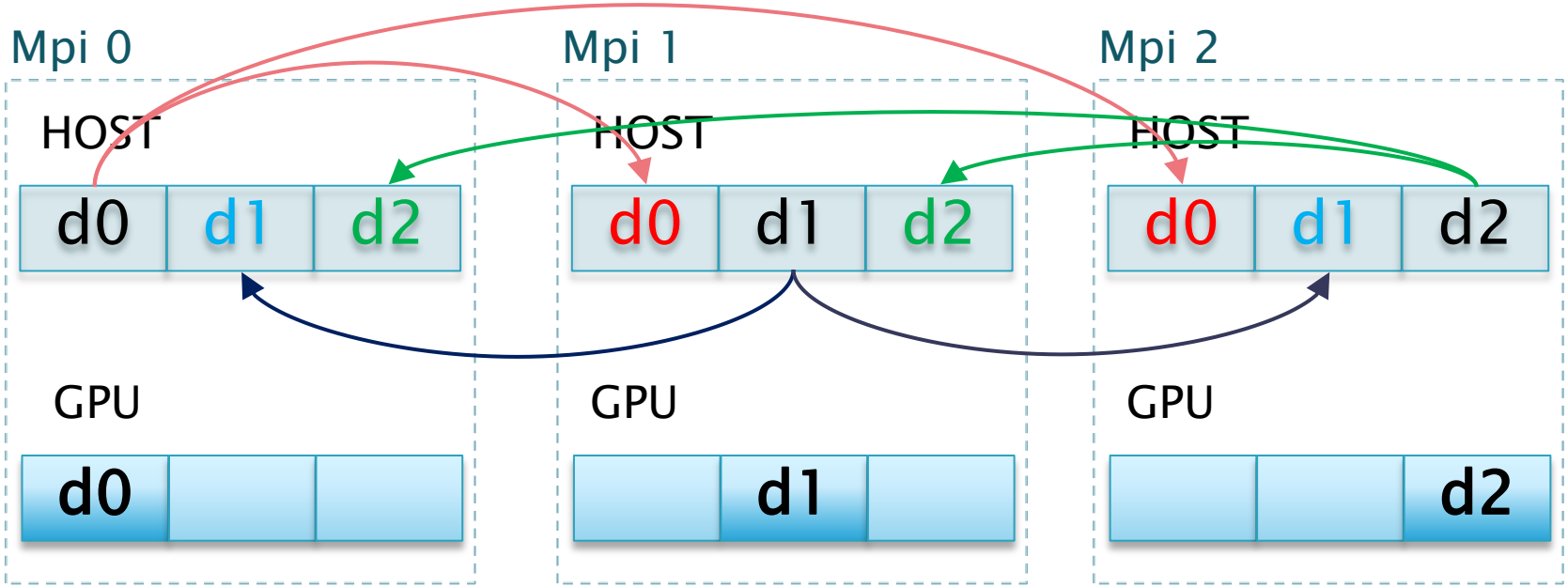
Организация итерации алгоритма

- Копирование частичных результатов с GPU на ЦПУ;



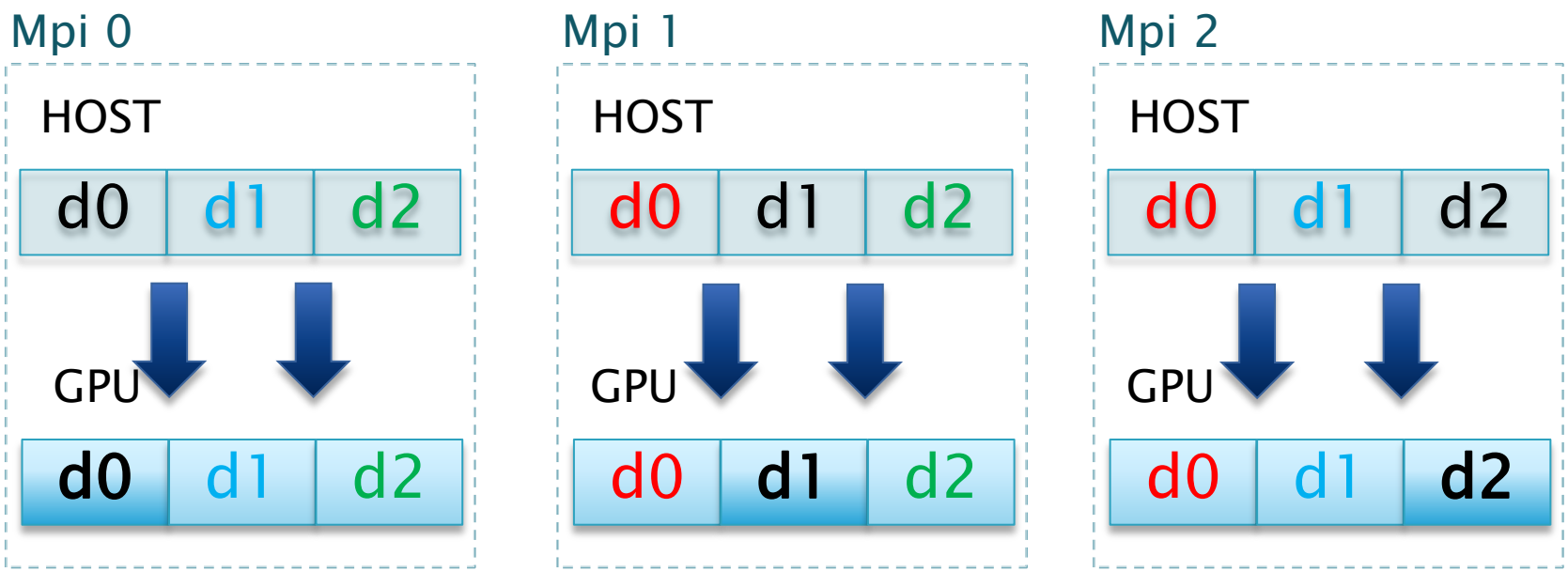
Организация итерации алгоритма

- Коллективный обмен между процессами для регистрации изменений другими узлами (MPI_Allgather);

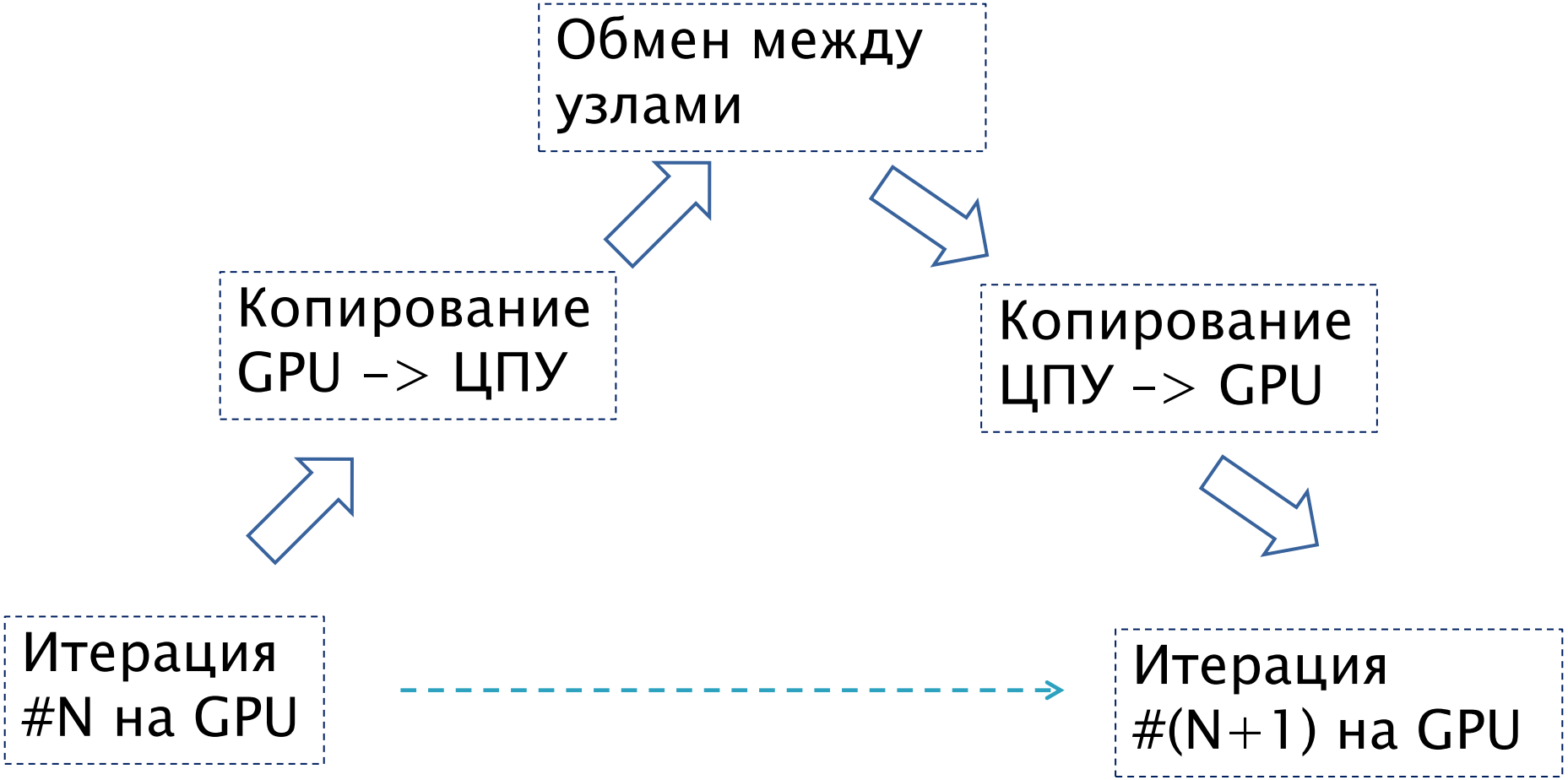


Организация итерации алгоритма

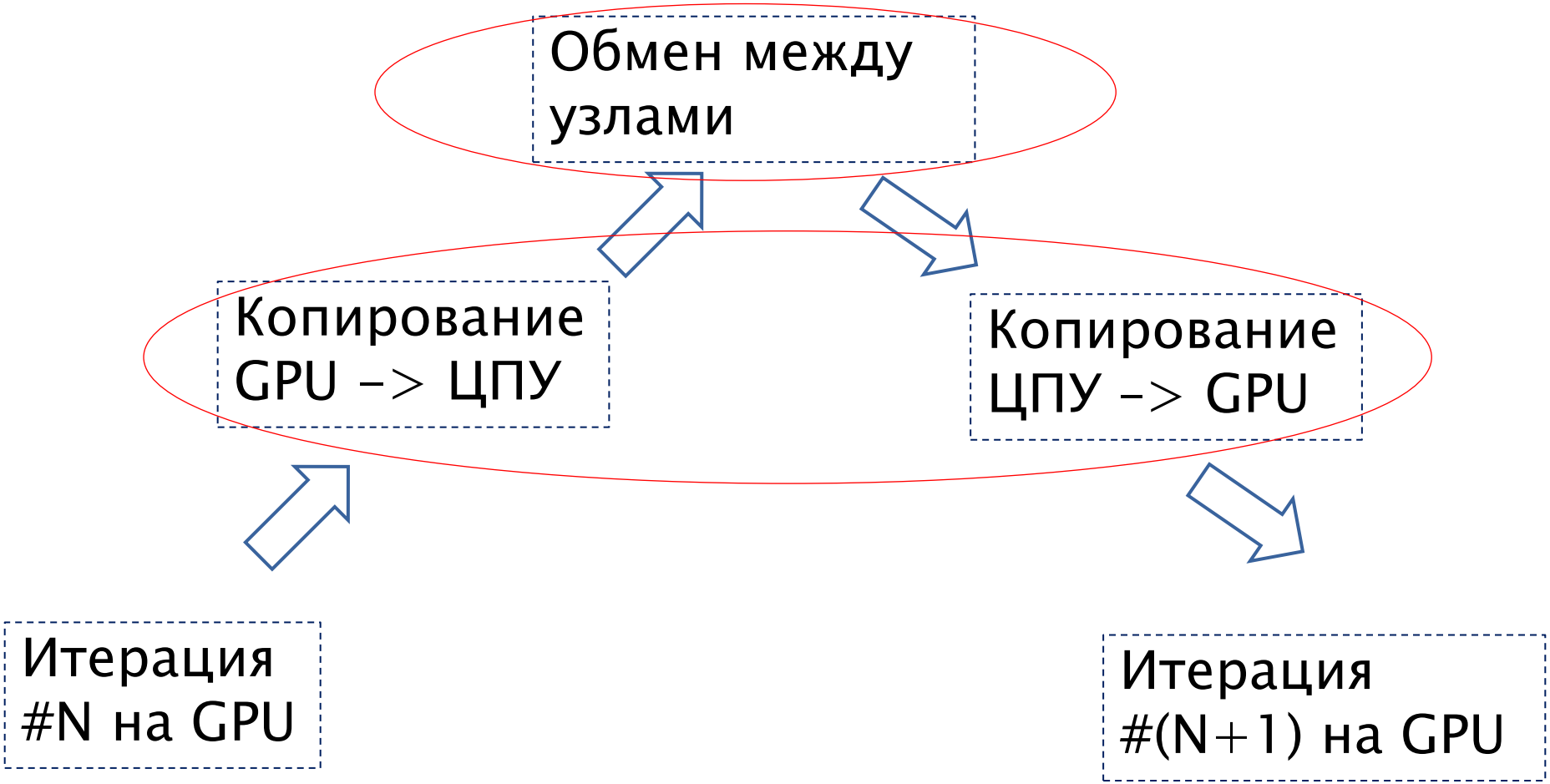
- Копирование полученных результатов с ЦПУ на GPU, переход на следующую итерацию;



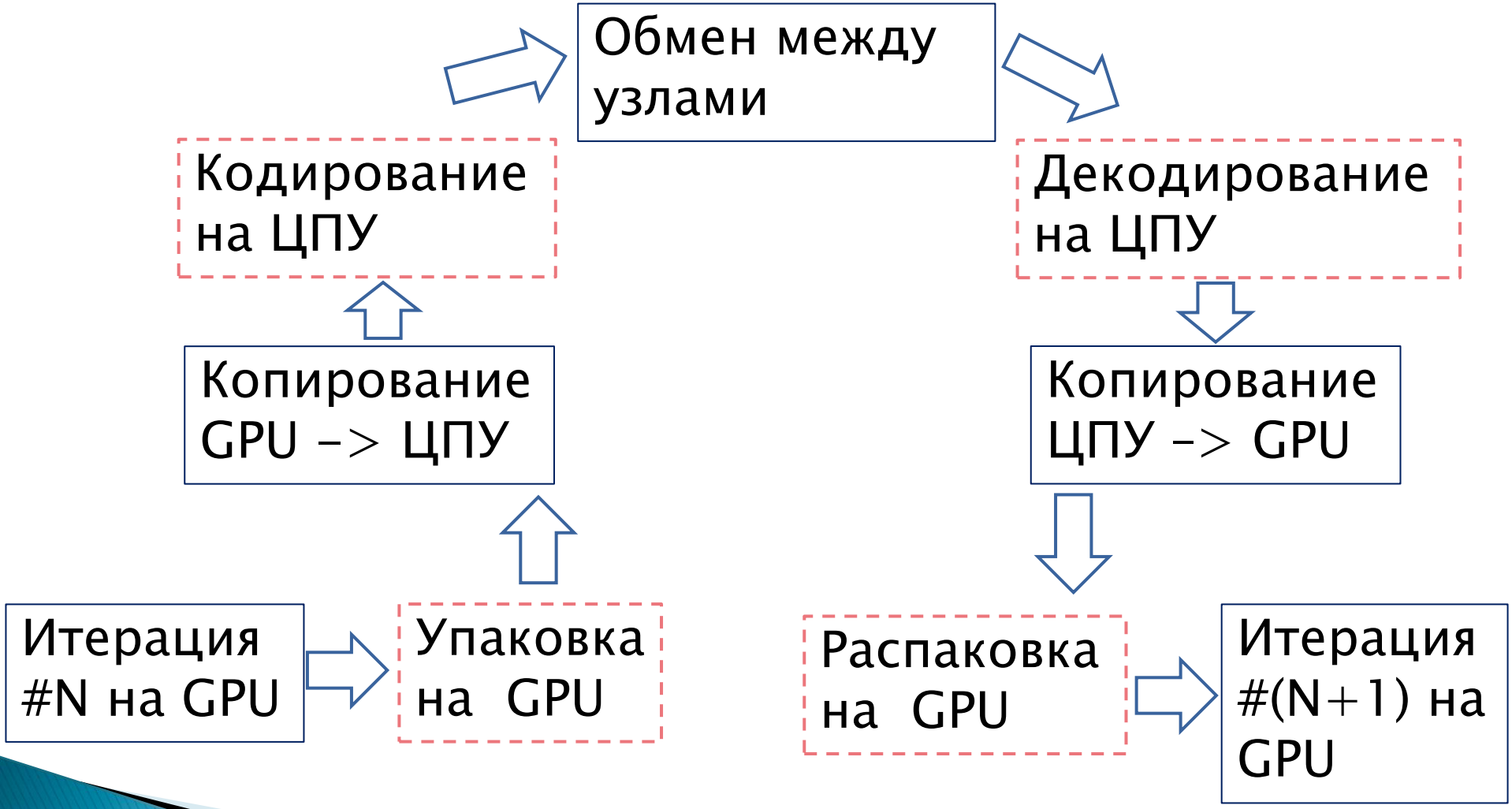
Общая схема


Обмен между узлами
Копирование ЦПУ -> GPU
Итерация #(N+1) на GPU
Копирование GPU -> ЦПУ
Итерация #N на GPU

Общая схема: накладные расходы



Общая схема: сжатие данных



Сжатие данных на GPU

- Необходимо передать информацию о помеченных вершинах только на текущей итерации;
- Для хранения этой информации достаточно 1 бита (вместо 8 бит);
- Позволяет сжать данные в 8 раз (при использовании типа uchar).

Сжатие данных на ЦПУ

- Использование алгоритма RLE (*Run-length encoding)*;
- Позволяет сократить объем передаваемых данных на начальных и конечных итерациях;
- Позволяет однозначно восстановить сжатые данные.

Сжатие данных на ЦПУ

- Кодирование с помощью RLE (*Run-length encoding)*;

<u>Исходный массив:</u>

|00000|1|0000000000000000|1……

<u>Сжатый массив:</u>

|101|0|1|1|1000|0|1|1|……

Сжатие данных на ЦПУ

- Кодирование с помощью RLE (*Run-length encoding)*;

|00000|1|0000000000000000|1……

|101|0|1|1|1000|0|1|1|……

Сжатие данных на ЦПУ

- Кодирование с помощью RLE (*Run-length encoding)*;

|00000|1|0000000000000000|1……

|101|0|1|1|1000|0|1|1|……

Сжатие данных на ЦПУ

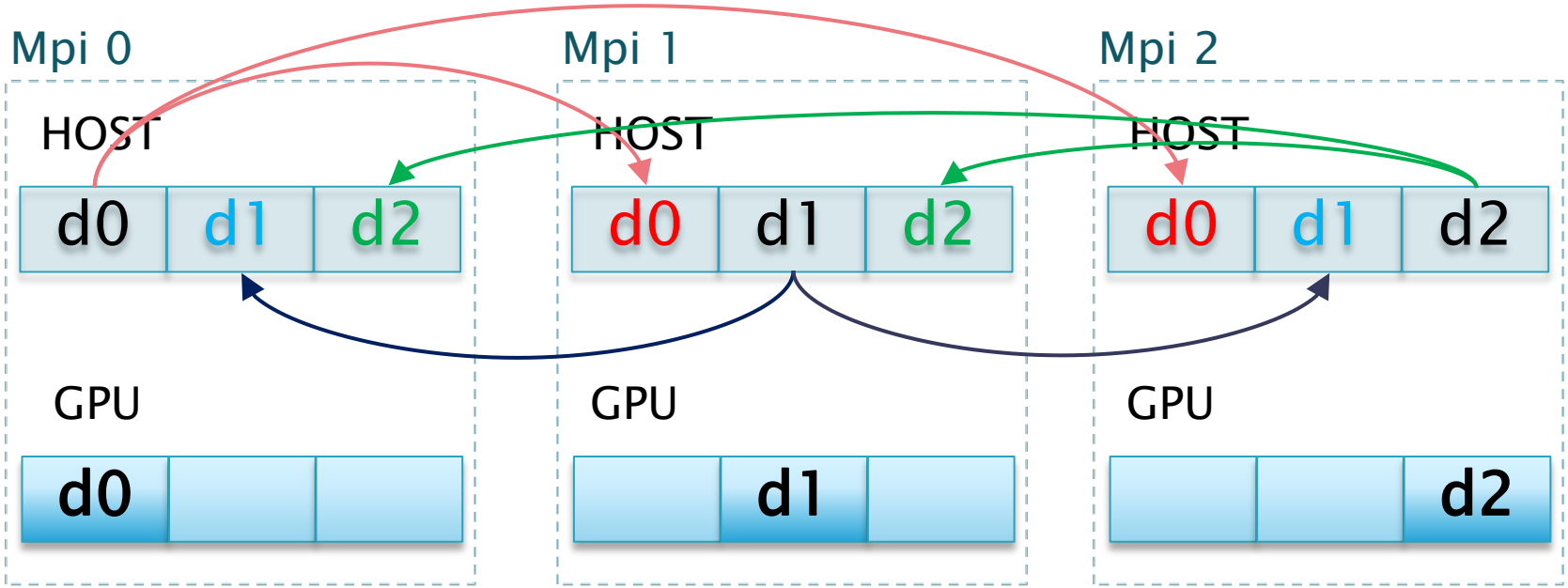
- Кодирование с помощью RLE (*Run-length encoding)*;

|00000|1|0000000000000000|1.....

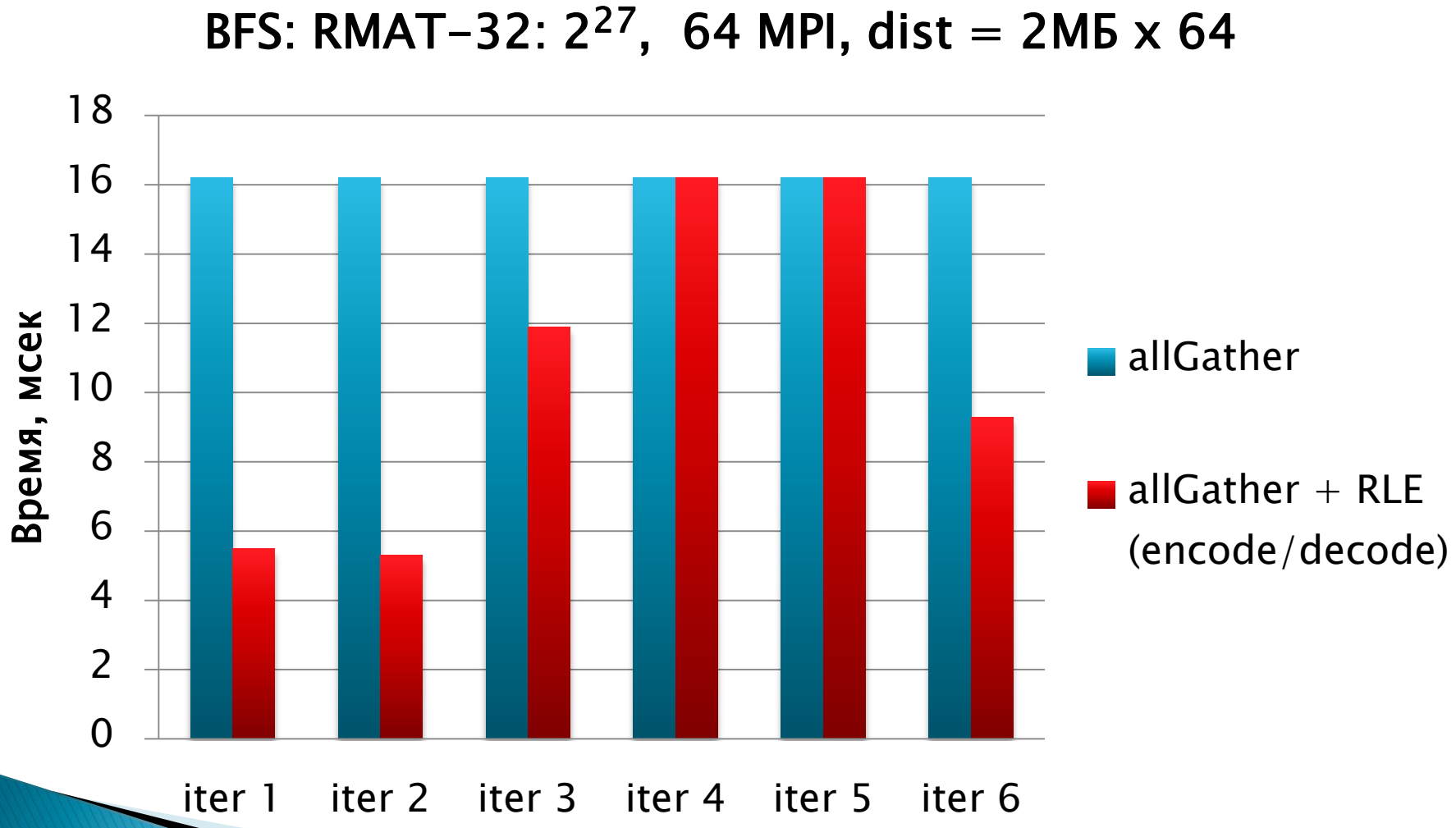
|101|0|1|1|1000|0|1|1|.....

Сжатие данных на ЦПУ

- Кодирование одним потоком;
- Декодирование – несколькими потоками;

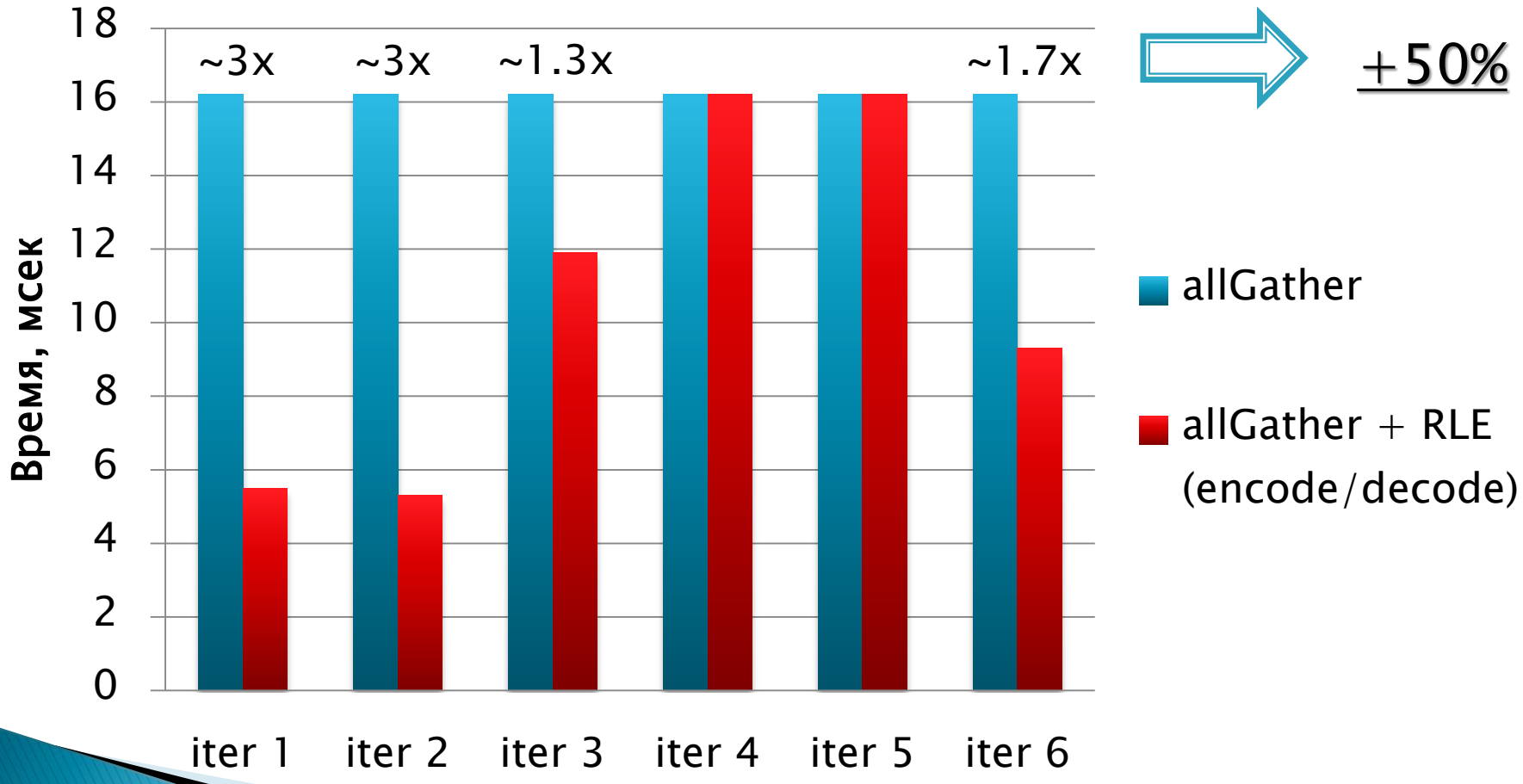


Результаты применения алгоритма RLE



Результаты применения алгоритма RLE, коммуникации

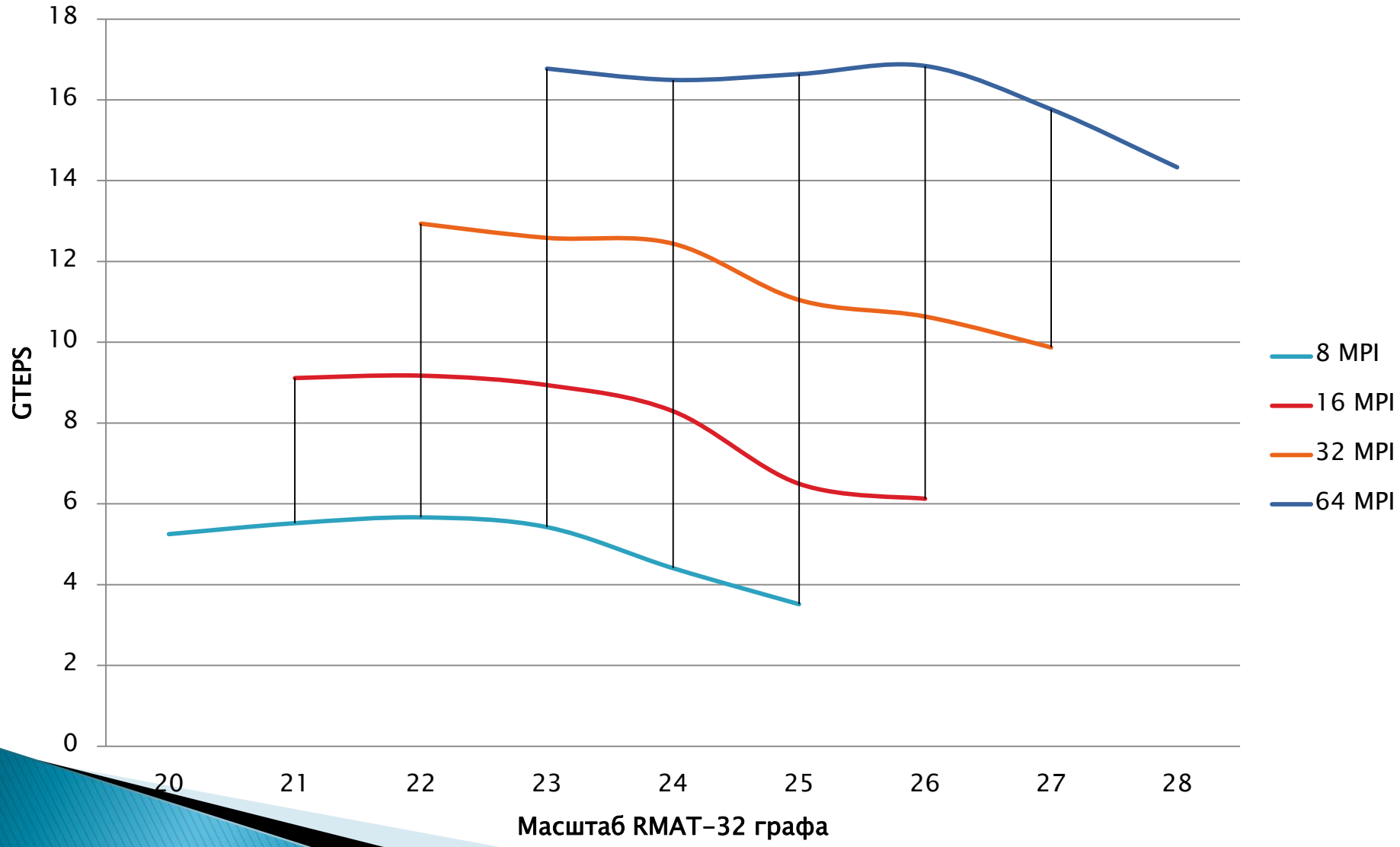
BFS: RMAT-32: 2^{27}, 64 MPI, dist = 2МБ x 64



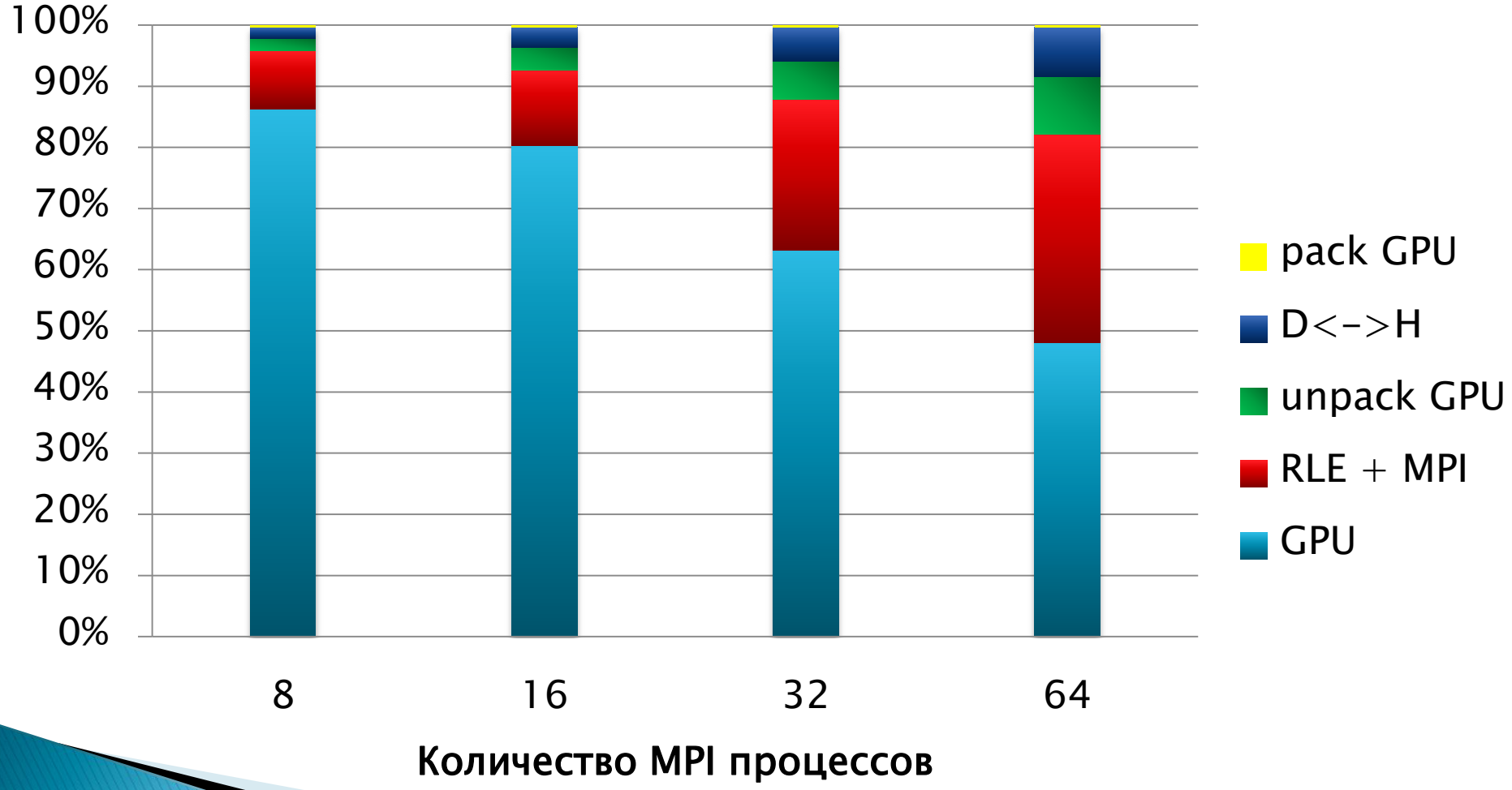
Результаты производительности

- Кластер k100:
  node: 2x Xeon 5670 + 3 GPU Tesla c2050;
- Графы RMAT-32;
- Масштабы 20-28 (392 МБ – 100 352 МБ);
- Использование до 64 MPI процессов.

Производительность

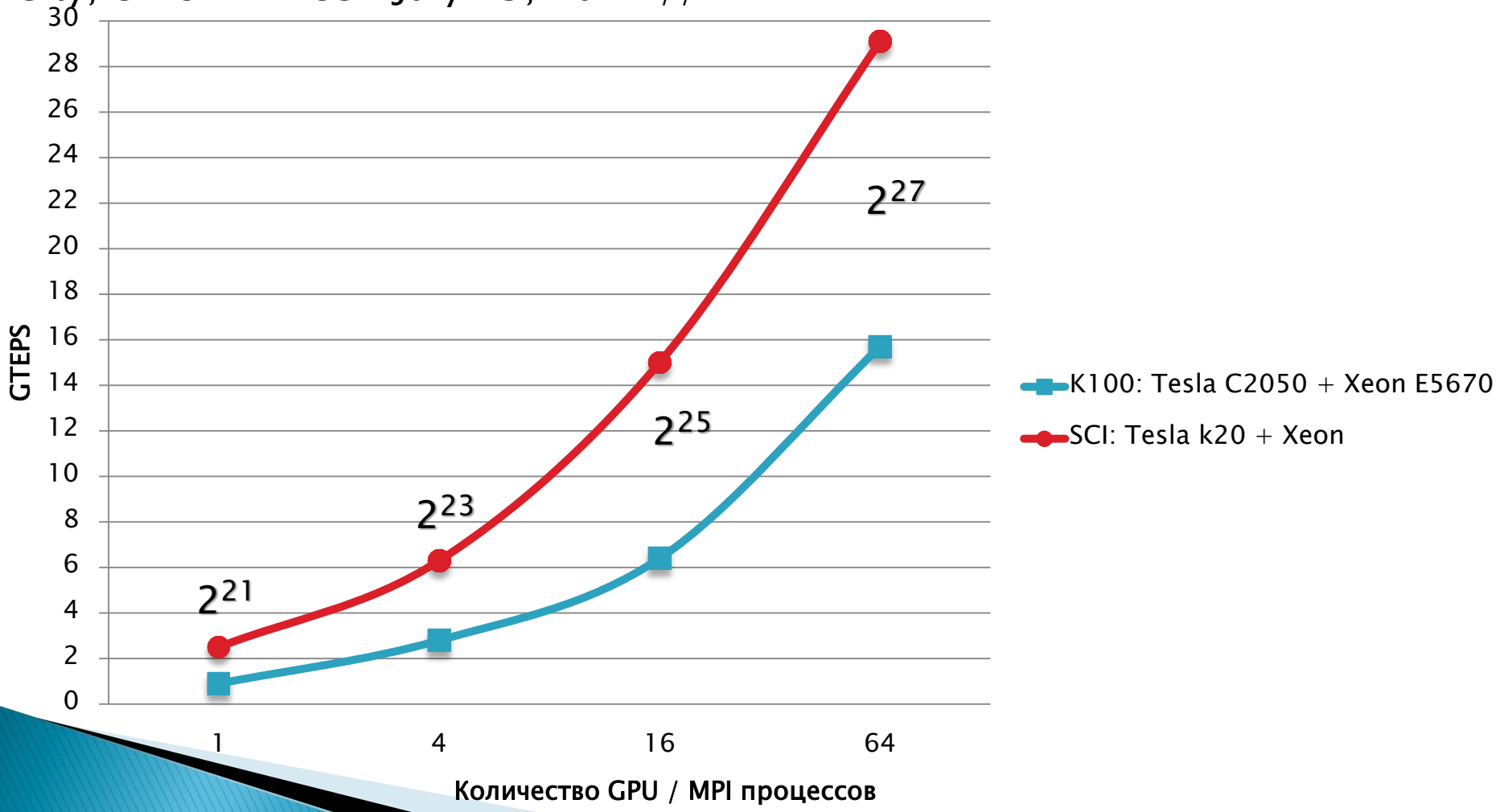


BFS: RMAT-32 2^{25}



Сравнение полученных результатов

“Parallel Breadth First Search on GPU Clusters”, Zhisong Fu, Harish Kumar Dasari, Martin Berzins and Bryan Thompson
// Scientific Computing and Imaging Institute University of Utah Salt Lake City, UT 84112 USA July 29, 2014 //



Спасибо за внимание!

Вопросы?

Колганов А.С.: alexander.k.s@mail.ru