Оглавление

Постановка задачи

В задаче требовалось реализовать параллельную сортировку Бэтчера для структур, представляющих точки на регулярной сетке (`Point`), вдоль одной из координат (x или y).

Структура `Point` имеет следующий вид:

```
Point {
    float coord[2];
    int index;
};
```

Пусть сетка размерности $n_1 * n_2$ представляется массивом таких структур `P[n1 * n2]`, а для инициализации координат точек используются функции `float x(int i, int j)` и `float y(int i, int j)`. Тогда `P[i*n2+j].coord[0] = x(i, j)`, `P[i*n2+j].coord[1] = y(i,j)`, `P[i*n2+j].index = i*n2+j`, где $i = \overline{0, n_1 - 1}, j = \overline{0, n_2 - 1}$.

На каждом процессоре должно обрабатываться одинаковое количество элементов. Каждый процессор выполняет упорядочивание элементов независимо от других. Слияние каждого отсортированного массива должно происходить в соответствии с расписанием, задаваемым сетью сортировки Бэтчера.

После окончания работы программы на каждом процессе должно находиться одинаковое количество элементов структуры `Point`. Каждый элемент структуры `Point` одного процесса должен находиться левее по координате по сравнению с элементом структуры `Point` любого другого процесса с бóльшим рангом.

Программа должна демонстрировать эффективность не менее 80% от максимально возможной на числе вычислительных ядер не менее 128.

Метод решения

Распределение элементов по процессам

Одним из требований к программе является одинаковое количество элементов на процессах. Чтобы выполнить это требование, при необходимости исходный массив дополняется фиктивными элементами типа `Point` с отрицательным значением индекса. Распределение фиктивных элементов происходит при инициализации массивов на каждом из процессоров.

Наличие фиктивного эелемента на процессоре зависит от его номера (`rank`). Количество процессоров, на которых нет фиктивных элементов, равно остатку от деления количества элементов в исходном массиве на количество процессоров. Если номер процессора меньше этого значения, то инициализируются все элементы соответствующего фрагмента массива. В противном случае, последний элемент массива считается фиктивным, и его поле `index` инициализируется отрицательным значением. Такие элементы будут игнорироваться при выводе конечного результата.

Инициализация фрагмента массива реализована в функции `init_points` в файле `point.cpp`.

Сортировка элементов

Для сортировки фрагментов массива на каждом процессе используется оптимизированный нерекурсивный алгоритм сортировки слиянием.

Массив разбивается на ещё меньшие фрагменты, к каждому из которых применяется алгоритм пирамидальной сортировки.

Слияние отсортированных фрагментов

Сеть слияния представляет собой сеть сортировки Бэтчера для массива, имеющего длину, равную количеству процессов. Каждый элемент массива равен своему индексу, то есть номеру процесса.

Для построения этой сети сортировки используются две рекурсивные функции, `sort` и `join`.

В функции `sort` массив рекурсивно разбивается на два подмассива, которые рекурсивно сортируются и передаются в функцию `join`, которая производит их слияние. База рекурсиик — массив из одного элемента, который уже очевидным образом отсортирован.

Элементы с чётными и нечётными номерами объединяются отдельно рекурсивными вызовами функции `join`. Если в подмассивах по одному элементу, эти элементы соединяются компаратором. Если суммарное число элементов в подмассивах больше двух, то после объединения чётных и нечётных элементов, пары соседних элементов обрабатываются с помощью заключительной группы компараторов. Массивы чётных и нечётных индексов хранятся в явном виде и строятся при каждом рекурсивном вызове функции `join`.

Таким образом, после обработки всего массива индексов процессов имеем массив компараторов, который определяет, какие процессы должны взаимодействовать при слиянии отсортированных фрагментов. Каждый компаратор представляет собой пару целых чисел — номеров процессов.

Компараторы обрабатываются следующим образом: два процесса с соответствующими номерами отправляют друг другу свои отсортированные фрагменты массивов (их длина одинакова), а затем перераспределяют элементы так, что на первом из них содержатся элементы с меньшими значениями, а на втором — с большими.

После того, как вся сеть компараторов будет обработана, на процессорах будут находится фрагменты упорядоченного массива, причём будет выполнено свойство: $a_{ir} \leq a_{jr}$ при $i < j$, и $a_{kp} \leq a_{lt}$, при $p < t$ и любых допустимых значениях k, l, где i, j, k, l — номера элементов в массиве на процессе, а r, p, t — номера процессов.

Структура программы

Используемая вычислительная система

Анализ полученных результатов