Оглавление

Постановка задачи **3**

Метод решения **4**

Построение сети слияния 4

Распределение элементов по процессам 5

Сортировка элементов 5

Слияние отсортированных фрагментов 5

Используемая вычислительная система **6**

Анализ полученных результатов **7**

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

Построение сети слияния

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.

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. Они записываются в конец исходного массива.

Если распределять элементы по процессам последовательно, то фрагмент массива на последнем процессе может целиком состоять из фиктивных элементов. Такое произойдёт, если количество фиктивных элементов равно количеству элементов на каждом процессе: например, при $n = 6, p = 4$, необходимо добавить 2 фиктивных элемента, и они оба будут находиться на последнем процессе.

Чтобы избежать этого, каждый процесс формирует свой фрагмент массива следующим образом: на процессе с номером $rank$ обрабатываются элементы с номерами $rank, rank + p, rank + 2 * p...$, где p — количество процессов. В итоге на каждом процессе обрабатывается максимум один фиктивный элемент.

Сортировка элементов

Слияние отсортированных фрагментов

Используемая вычислительная система

Анализ полученных результатов