Московский государственный университет имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Параллельные (высокопроизводительные) вычисления

Отчет по заданию №2 «Параллельная сортировка Бэтчера»

Богатенкова Анастасия Олеговна
528 группа

14 ноября 2021 г.

Содержание

1 Постановка задачи

Дана структура (или класс)

$$Point\ \{float\ coord[2];\ int\ index;\}P[n1 * n2];$$
$$n_1 * n_2 \leq 2^{30}$$

Данная структура будет использоваться для работы с регулярной сеткой. Точки данной сетки имеют координаты

$$P[i * n2 + j].coord[0] = x(i, j)$$
$$P[i * n2 + j].coord[1] = y(i, j)$$

где $i = 0, ..., n_1 - 1, j = 0, ..., n_2 - 1$.

Индекс определяется соотношением

$$P[i * n2 + j].index = i * n2 + j$$

- **Особенности:** для инициализации координат можно использовать функции, принимающие на вход параметры (i, j), то есть фактически каждая точка этой сетки однозначно определяется (i, j).

- **На входе:** на каждом процессе одинаковое количество элементов структуры $Point$. (Если на некоторых процессах элементов структуры $Point$ меньше чем во всех остальных, тогда необходимо ввести фиктивные элементы, например, с отрицательным значением индекса.)

- **Цель:** реализовать параллельную сортировку Бэтчера для структур Point вдоль одной из координат x (или y). То есть с начала необходимо реализовать сортировку на каждом отдельном процессе, а потом реализовать сеть слияния Бэтчера.

- **На выходе:** на каждом процессе одинаковое количество элементов структуры $Point$. Каждый элемент структуры $Point$ одного процесса находиться левее по координате x (или y) по сравнению с элементом структуры Point любого процесса с большим рангом, за исключением фиктивных элементов.

2 Описание метода решения

Алгоритм сортировки реализован следующим образом:

1. на каждом из процессоров с помощью стандартного алгоритма std::sort сортируется фрагмент массива длиной $\frac{n}{p}$. При этом будем полагать, что $n = rp$, где r – целое число.

2. отсортированные фрагменты объединяются с помощью функции слияния, причем последовательность слияний определяется алгоритмом «обменной сортировки со слиянием» Бэтчера.

Сортировка Бэтчера реализована в соответствии с алгоритмом, указанным в книге Д. Э. Кнута «Искусство программирования» (т. 3). Вместо элементов в этом алгоритме сортируются упорядоченные массивы, каждый из процессов в соответствии с алгоритмом получает от другого процесса массив, с помощью функции слияния получает либо массив с наименьшими элементами, либо с наибольшими в зависимости от номера процесса.

Реализованы следующие функции:

- *runSort* – запуск последовательной сортировки и подсчет времени;
- *runSortParallel* – запуск параллельной сортировки и подсчет времени;
- *bSortParallel* – реализация сортировки Бэтчера;
- *compareExchangeParallel* – функция для получения массива из другого процесса и запуска функции слияния массивов;
- *merge* – функция слияния массивов.

3 Описание используемой вычислительной системы

Программа тестировалась на суперкомпьютере «IBM BlueGene/P». Характеристики вычислительной системы представлены в таблице 1.

4 Результаты численных экспериментов

Сортировка проводилась на сетках размера 1024×1024, 2048×2048, 4096×4096, 8192×8192. Массив структур *Point* сортировался по координате x типа

Пиковая производительность	27.9 TFlop/s
Число стоек	2
Вычислительных узлов	1024
Процессоры IBM PowerPC 450	2048
Число процессорных ядер	4
Общее число ядер	8192
Оперативная память на узле	2 Гбайт
Коммуникационная сеть	трехмерный тор
Система хранения данных	GPFS
Операционная система	Linux

Таблица 1: Характеристики «IBM BlueGene/P»

$float$. Для получения более точных результатов, сортировка проводилась 5 раз и временные результаты усреднялись. Результаты экспериментов приведены в таблицах 2, 3, 4 и 5.

5 Анализ полученных результатов

Время сортировки массива оценивается следующим выражением:

$$T(n,p) = k\frac{n}{p}\left(\log_2\left(\frac{n}{p}\right) + b \cdot s_p\right),$$

где n – число элементов массива, p – число процессов, b_1 $(b \sim 1)$ – константа, определяющая время слияния двух фрагментов массива, s_p – число шагов слияния (число тактов сортировки Бэтчера) вычисляется следующим образом:

$$s_p = \frac{\lceil \log_2 p \rceil \left(\lceil \log_2 p \rceil + 1\right)}{2}$$

.

Максимальное значение коэффициента эффективности использования вычислительной мощности задается выражением:

$$E^{max}(n,p) = \frac{t(n,1)}{p \cdot t(n,p)} = \frac{\log_2 n}{\log_2 n + s_p - \log_2 p} \approx \frac{1}{1 + \log_n p(\log_2 p - 1)/2}.$$

Максимальное значение ускорения:

$$S^{max}(n,p) = p \cdot E^{max}(n,p)$$

В таблицах 2, 3, 4 и 5 для сеток разных размеров были посчитаны максимальные значения ускорения и эффективности по формулам, написанным выше. Также приведены результаты численных экспериментов.

Обозначения, используемые в таблицах:

- p – число процессов;
- T – время сортировки (сек);
- S – ускорение, полученное при численном расчете;
- s_p – число тактов сортировки Бэтчера;
- S^{max} – максимально возможное ускорение;
- E – эффективность, полученное при численном расчете;
- E^{max} – максимально возможная эффективность.

p	T	S	s_p	S^{max}	E	E^{max}	E/E^{max}
1	1.687	1.000	0	1.000	1.000	1.000	1.000
2	0.874	1.930	1	2.000	0.965	1.000	0.965
4	0.496	3.400	3	3.810	0.850	0.952	0.893
8	0.298	5.660	6	6.957	0.708	0.870	0.814
16	0.178	9.467	10	12.308	0.592	0.769	0.769
32	0.111	15.255	15	21.333	0.477	0.667	0.715
64	0.068	24.961	21	36.571	0.390	0.571	0.683
128	0.043	38.964	28	62.439	0.304	0.488	0.624
256	0.028	59.418	36	106.667	0.232	0.417	0.557

Таблица 2: Сортировка 1024×1024 элементов

6 Приложение

Алгоритм сортировки реализован на языке C++ в файле *task2.cpp*. Для расчёта ускорения и эффективности написан скрипт *compute_statistics.py* на языке Python. Для запуска программы на суперкомпьютере написан $Makefile$.

p	T	S	s_p	S^{max}	E	E^{max}	E/E^{max}
1	7.438	1.000	0	1.000	1.000	1.000	1.000
2	3.914	1.900	1	2.000	0.950	1.000	0.950
4	2.187	3.401	3	3.826	0.850	0.957	0.889
8	1.273	5.841	6	7.040	0.730	0.880	0.830
16	0.773	9.623	10	12.571	0.601	0.786	0.765
32	0.468	15.879	15	22.000	0.496	0.688	0.722
64	0.282	26.373	21	38.054	0.412	0.595	0.693
128	0.177	42.119	28	65.488	0.329	0.512	0.643
256	0.112	66.369	36	112.640	0.259	0.440	0.589

Таблица 3: Сортировка 2048×2048 элементов

p	T	S	s_p	S^{max}	E	E^{max}	E/E^{max}
1	33.330	1.000	0	1.000	1.000	1.000	1.000
2	17.231	1.934	1	2.000	0.967	1.000	0.967
4	9.283	3.591	3	3.840	0.898	0.960	0.935
8	5.541	6.015	6	7.111	0.752	0.889	0.846
16	3.262	10.218	10	12.800	0.639	0.800	0.798
32	1.987	16.772	15	22.588	0.524	0.706	0.742
64	1.193	27.947	21	39.385	0.437	0.615	0.710
128	0.751	44.388	28	68.267	0.347	0.533	0.650
256	0.454	73.378	36	118.154	0.287	0.462	0.621

Таблица 4: Сортировка 4096×4096 элементов

p	T	S	s_p	S^{max}	E	E^{max}	E/E^{max}
1	143.480	1.000	0	1.000	1.000	1.000	1.000
2	74.295	1.931	1	2.000	0.966	1.000	0.966
4	40.846	3.513	3	3.852	0.878	0.963	0.912
8	23.388	6.135	6	7.172	0.767	0.897	0.855
16	13.895	10.326	10	13.000	0.645	0.812	0.794
32	8.325	17.235	15	23.111	0.539	0.722	0.746
64	5.032	28.513	21	40.585	0.446	0.634	0.703
128	3.111	46.124	28	70.809	0.360	0.553	0.651
256	1.922	74.668	36	123.259	0.292	0.481	0.606

Таблица 5: Сортировка 8192×8192 элементов