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

Параллельные высокопроизводительные вычисления

Отчёт

О выполненном задании №1
“Расписание сети сортировки”

Выполнил:
студент 523 группы
Латыпов Ш. И.
12.11.2023

Москва
2023

Содержание

Описание задачи

Разработать последовательную программу вычисления расписания сети сортировки, числа использованных компараторов и числа тактов, необходимых для её срабатывания при выполнении на n процессорах. Число тактов сортировки при параллельной обработке не должно превышать числа тактов, затрачиваемых четно-нечетной сортировкой Бетчера.

Параметр командной строки запуска: n.

$n \geq 1$ – количество элементов в упорядочиваемом массиве, элементы которого расположены на строках с номерами $[0, ..., n-1]$

Формат команды запуска: ./bsort n

Требуется:

1. Вывести в файл стандартного вывода расписание и его характеристики в представленном далее формате;
2. Обеспечить возможность вычисления сети сортировки для числа элементов $1 \leq n \leq 10000$;
3. Предусмотреть полную проверку правильности сети сортировки для значений числа сортируемых элементов $1 \leq n \leq 24$;
4. Представить краткий отчет удовлетворяющий указанным далее требованиям.

Формат файла результата:
Начало файла результата
n 0 0
cu0 cd0
cu1 cd1
...
cun_comp-1 cdn_comp-1
n_comp
n_tact
Конец файла результата

Здесь:

- n 0 0 – число сортируемых элементов, ноль, ноль.
- cui cdi – номера строк, соединяемых i-м компаратором сравнения перестановки.
- n_comp – число компараторов
- n_tact – число тактов сети сортировки

Метод решения

Функция main принимает на вход через параметры командной строки число n.

Для решения данной задачи выбран алгоритм построения сети обменной сортировки со слиянием Бэтчера. Основными функциями являются рекурсивные функции int B(int first, int step, int count) и int S(int first1, int first2, int step, int n, int m).

- B(first, step, count) — процедура рекурсивного построения сети сортировки группы линий (first, step, count).
- S(first1, first2, step, n, m), — рекурсивная процедура слияния двух групп линий (first1, step, n) и (first2, step, m).

В функции B(...) массив делится на две части, и далее для каждой из частей рекурсивно запускаются функции B(first, step, count1) и B(first + step $*$ count1, step, count $-$ count1), где count1 - количество элементов в первой половине массива (Вычисление: count1 = count / 2 + count % 2). После этого происходит запуск фукнции S(first, first + step $*$ count1, step, count1, count $-$ count1).

При этом есть дополнительные проверки на размеры массива: если он меньше 2, то функция завершает работу; если он равен двум, то в массив компаратора добавляется новый компаратор со значениями (first, first + step), после этого функция завершает работу.

В функции S(...) объединение элементов массивов с нечетными номерами и отдельно — с четными, после чего, с помощью заключительной группы компараторов, обрабатываются пары соседних элементов с номерами вида (2i, 2i + 1), где i — натуральные числа от 1 до $\frac{p}{2} - 1$.

Рекурсивный запуск фукнций S(first1, first2, 2 $*$ step, n1, m1) и (first1 + step, first2 + step, 2 $*$ step, n $-$ n1, m $-$ m1), где n1 = n $-$ n / 2, m1 = m $-$ m / 2;

В этой функции также присутствует проверка размера массивов, при котором, если размер одного из массивов равен 0, то фукнция завершает работу, а если оба массива имеют размер 1, то в вектор компараторов добавляется новый компаратор.

Хранение компараторов реализовано через vector<pair<int, int>> comps - вектор пар чисел. Для добавления компараторов выполняется функция add_comp(int first, int second).

Подсчёт тактов реализовано через возвращаемые значения функций B и S. После каждого рекурсивного запуска функций в B запоминается максимальное значение возвращаемых чисел, а далее к этому прибавляется значение функции S. В функции S также запоминается максимальное значение возвращаемых чисел от рекурсивных запусков, и далее к этому значению прибавляется + 1.

Проверка программы

Для проверки работоспособности алгоритма в функции реализована отдельная секция. Если через параметры командной строки в качестве второго аргумента было вписано 'test', то запускается скрипт, который генерирует 2^n массивов длинны n, которые заполнены всеми возможными комбинациями чисел 0 и 1. Пример запуска тестирующего режима: ./bsort 4 test

При каждой итерации создается копия такого вектора, который сортируется стандартным средством сортировки std::sort(). Сам же вектор сортируется через сеть компараторов, поочередно пробегаясь по вектору comps. Далее происходит проверка значений поэлементно для вектора, которые сортировался компараторами, и копии этого вектора, который сортировался через std::sort().

Вывод программы при тестировании:

- TESTING - означает, что тестирование началось,
- NOT OK: <число> - означает, что на итерации <число> значения элементов вектора и копии вектора не совпали, то есть сеть компараторов неверно отсортировала последовательность чисел.
- Test end - означает, что тестирование закончилось.

При успешном прохождении тестирования, в конце программы будет вывод:
TESTING
Test end

В случае неуспешного:
TESTING
NOT OK <число>
...
NOT OK <число>
Test end

Приложение

К данному отчёту прилагается файл task1.cpp с оригинальным кодом описанной программы.