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

ОТЧЕТ

По курсу «Параллельные высокопроизводительные вычисления»

Практическое задание №1.

«Расписание сети сортировки»

Работу выполнил:
Студент группы 528
Факультета вычислительной математики и кибернетики
Цирунов Леонид Александрович

9 ноября 2023 года

Описание условия

Разработать последовательную программу вычисления:

1. расписания сети сортировки, числа
2. использованных компараторов и числа тактов, необходимых для её срабатывания при
3. 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, где n>=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<=n<=10000;
3. предусмотреть полную проверку правильности сети сортировки для значений числа сортируемых элементов 1<=n<=24;
4. представить краткий отчет удовлетворяющий указанным далее требованиям.

Формат файла результата:

Начало файла результата

n 0 0

cu_0 cd_0

cu_1 cd_1

…

cu_{n_comp-1} cd_{n_comp-1}

n_comp

n_tact

Конец файла результата

Где:

n 0 0 – число сортируемых элементов, ноль, ноль.

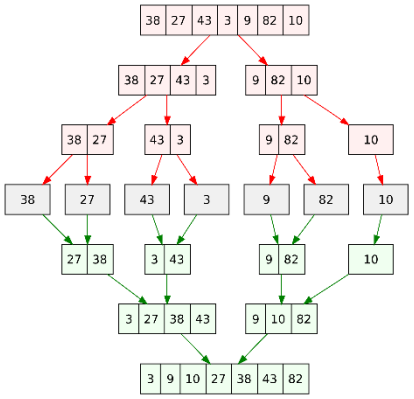
cu_i cd_i – номера строк, соединяемых i-м компаратором сравнения перестановки.

n_comp – число компараторов

n_tact – число тактов сети сортировки

Описание метода решения

Для решения поставленной задачи был реализован частичный алгоритм четно-нечетной сортировки Бэтчера, в котором отсутствует логика работы с массивом и логика слияния. Алгоритм используется только для составления расписания. Сортировка массивов в проверке производится непосредственно по построенному расписанию.
Для построения сети использовался рекурсивный алгоритм.
При сортировке массива из n элементов с номерами **[0, …, n-1]** следует разделить его на две части: в первой оставить **p=**$\left[\frac{n}{2}\right]$ элементов с номерами **[0, …, p]**, а во второй **m = n – p** элементов с номерами **[p+1, …, n]**.
Наглядно принцип деления исходного массива и слияния частей представлен на рисунке:



BatcherSort – функция рекурсивного делит массив на два подмассива из **p** и **m** элементов соответственно, после чего вызывает функцию слияния BatcherMerge для этих подмассивов. В реализации отсутствует работа с массивом, оперирование происходит только с индексами условного массива элементов.

BatcherMerge – функция рекурсивного слияния двух групп линий. В сети нечетно-четного слияния отдельно объединяются элементы массивов с нечетными номерами и отдельно с четными, после чего с помощью заключительной группы компараторов обрабатываются пары соседних элементов. Данные пары записываются в массив компараторов comparators для дальнейшего использования. В реализации опять же отсутствует работа с массивов, оперирование идет только с индексами, и происходит заполнение массива сравниваемых пар номеров строк, соединяемых i-м компаратором сравнения перестановки.

В результате работы программы получается только составленное расписание работы сети.

Описание метода проверки

Компиляция программы осуществляется командой:
g++ bsort.cpp -o bsort

Запуск программы составления расписания для определенного числа n:
./bsort n
Расписание выводится в заданном формате в файл "***schedule.txt***".

Запуск полной проверку правильности сети сортировки для количества сортируемых элементов ∈ [1, n]:
./bsort n -t

Тестирование проводилось при помощи 0–1 принципа.
0–1 принцип: если сеть сортирует все последовательности из нулей и единиц, то сеть является сортирующей. Необходимо перебрать все перестановки из n элементов, состоящие из нулей и единиц, пропустить их через сеть и проверить, что они корректно отсортированы.

Была реализована функция ***generateBinaryArrays***, которая генерирует все перестановки из 0 и 1, длины n. Также была реализована функция ***testSchedule***, которая вызывается при тестировании и итеративно производит генерацию перестановок для каждого i ∈ [1, n], после чего составляется расписание и производится сортировку, а также вывод в файл.

В результате работы тестовой программы создаются три файла:

- "***input_arrays.txt***" – вывод исходных массивов для каждого i ∈ [1, n].
- "***output_arrays.txt***" – вывод отсортированных массивов для каждого i ∈ [1, n].
- "***comparators.txt***" – вывод расписания, количества компараторов и количества тактов для каждого i ∈ [1, n].

Приложение1

Исходный текст программы в отдельном c++ файле.