Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н. И. Лобачевского»

Институт информационных технологий, математики и механики Кафедра Математического обеспечения и суперкомпьютерных технологий

Отчёт по лабораторной работе:

**«Поразрядная сортировка MSD Radix Sort для целых чисел»**

**Выполнил:**  студент группы 381506-1 Левицкий Илья Олегович

Нижний Новгород, 2018 г.

# I.Постановка задачи

Дан неупорядоченный массив, состоящий из n положительных и отрицательных элементов типа int. Требуется реализовать последовательную и параллельную версию алгоритма восходящей поразрядной сортировки MSD Radix Sort для целых чисел и сравнить их время работы при различных значениях размера сортируемого массива.

Процесс выполнения задачи можно описать следующим образом:

1. Изучение метода решения задачи.
2. Разработка схемы параллельных вычислений.
3. Реализация программы.
4. Анализ результатов экспериментов.

Помимо прочего, в разрабатываемой программе необходимо:

1. Обеспечить генерацию данных для задач произвольной размерности. При этом параметры задачи должен задавать пользователь.

# II.Метод решения

Для представления числа в виде битовой строки осуществляется явное приведение к типу unsigned char. Таким образом получится строка из 8-ми символов (один символ равен одному байту).

Для демонстрации сути алгоритма сортировки рассмотрим пример. В качестве исходных данных возьмем битовые строки длины 4 (рис. 2). На первом шаге заводим две очереди “0” и “1”, в которые последовательно записываем числа из исходного массива, исходя из того, какая цифра стоит в первом разряде. Далее, каждую получившуюся очередь рассматриваем как новые исходные данные, для которых проделываем те же самые операции, что и на первом шаге. Продолжаем до тех пор, пока в очередных исходных данных не останется один элемент.



Рис. 2: Пример работы алгоритма.



Нетрудно заметить, что получившийся алгоритм является рекурсивным.

Поразрядная сортировка выполняется за линейное время, то есть алгоритм имеет сложность по времени *O*(*n*).

**III.Представление целых чисел в памяти компьютера**

Так как мы будем рассматривать побайтовую реализацию сортировки для элементов типа int, то покажем, как хранятся целые числа в памяти компьютера.

**Формат целочисленных типов**

Целочисленные типы без знака представляются в памяти таким образом (рассмотрим 4-байтовый тип ― «unsigned int»), что число x кодируется следующим образом:

Целочисленный тип со знаком кодируется таким образом, что при сложении равных по модулю отрицательного и положительного чисел получается нуль.

Кроме того, старший бит определяет знак числа:

* если старший бит равен нулю, то число положительное;
* если старший бит равен единице, то число отрицательное.

**Формат целых отрицательных чисел**

Целочисленные типы со знаком представляются в памяти таким образом (рассмотрим 4-байтовый тип ― «signed int»), что число x кодируется следующим образом:

При таком представлении отрицательные числа кодируются в так называемом дополнительном коде:

* записывается равное по модулю положительное число;
* выполняется инвертирование всех бит этого числа;
* к полученному значению добавляется ―1 по правилам сложения чисел без знака.

# IV.Схема распараллеливания

Суть параллельной реализации поразрядной сортировки MSD radix sort заключается в следующем: в начале выполняется разбиение массива исходных данных на количество, равное числу процессов, и передача созданных массивов всем процессам. Процессы параллельно сортируют свои массивы данных последовательным алгоритмом.

После этого получаются отсортированные массивы. Следующий этап заключается в том, что производится слияние этих массивов в один результирующий (рис. 3). Слияние проводится итерационно: на первом шаге все процессы делятся на пары, и правый процесс передаёт свои данные левому. Тот процесс сортирует их, используя операцию слияния merge. На втором шаге процессы снова объединяются в пары, но только те, которые были левыми на первом шаге. Далее повторяются те же действия: правый процесс передаёт свои данные левому, а тот их сортирует. Процесс продолжается до тех пор, пока не останется один процесс. Результат слияния последних двух массивов и будет результатом всей сортировки.



Рис. 3: Слияние отсортированных частей в один массив.

# V.Описание программной реализации

Программа реализована с использованием средств Open MP и Intel Threading Building Blocks. Для запуска программы необходимо выполнить в командной строке следующую команду:

Radixsort.exe <количество\_процессов> <имя входного файла> <имя выходного выйла>

Выполнение программы начинается с вызова функции main. В начале определяется общее число запущенных процессов и номер выполняемого процесса. Далее, нулевой процесс считывает параметры, введенные в командной строке - размер массива. На основе второго параметра формируются исходные данные – массив для сортировки. Элементы массива имеют тип BinaryInt, который является объединением union и содержит два поля: число типа int и его битовое представление типа unsigned char:

union BinaryInt

{

int d;

unsigned char c[sizeof(int)];

}

Нулевой процесс создает размеры порций массива входных данных всем остальным процессам. Для того, чтобы нулевой процесс смог отправить части исходного массива данных другим процессам, необходимо посчитать количество элементов, которые достанутся каждому процессу.

После получения своей части исходного массива, процессы сортируют их с помощью функции RadixSort. После этого встаёт задача о сборе отсортированных частей в один отсортированный массив. Для этого выполняется алгоритм, описанный в предыдущей части настоящего отчёта. Слияние двух отсортированных массивов в один отсортированный массив происходит посредством функции merge.

В конце выводится сообщение о времени работы последовательного и параллельного алгоритмов, ускорение и проверка на совпадение результатов, полученных в обоих версиях.

# VI.Подтверждение корректности

Для задач большой размерности (количество элементов превышает несколько десятков) корректность подтверждается с помощью программы проверки checker.exe. На вход поступают два “отсортированных” массива, полученных из параллельной реализаций и эталонного результата, сгенерированного программой-генератором generator.exe, и возвращается значение “Arrays are equal”, если результаты совпали, и “The result is different”, если существуют любые расхождения. Проверка на совпадение производится поэлементно. Результат выводится в сообщении в конце работы программы.

# VII.Результаты экспериментов

Вычислительные эксперименты для оценки эффективности параллельного алгоритма поразрядной сортировки MSD radix sort проводились с использованием домашнего ПК. ПК обладет следующими характеристиками:

* + Процессор: Intel Core i5-4460 3,20 GHz
  + Установленная память (ОЗУ): 8гб

Операционная система: Microsogt Windows 7 Ultimate (64 разрядная)

*•*

* + пиковая производительность: 3 Тфлопс

Результаты экспериментов на массиве из 10 000, 100 000, 1 000 000 и 10 000 000 элементов приведены в таблице.

Для 10 000 элементов

|  |  |  |  |
| --- | --- | --- | --- |
| Использованные средства | Последовательная версия | Open MP | TBB |
| Время (с) | 0,025 | 0,011 | 0,007 |
| Ускорение | — | 2,27 | 3.5 |

Для 100 000 элементов

|  |  |  |  |
| --- | --- | --- | --- |
| Использованные средства | Последовательная версия | Open MP | TBB |
| Время (с) | 0,145 | 0,020 | 0,015 |
| Ускорение | — | 7,25 | 9,7 |

Для 1 000 000 элементов

|  |  |  |  |
| --- | --- | --- | --- |
| Использованные средства | Последовательная версия | Open MP | TBB |
| Время (с) | 1,035 | 0,125 | 0,100 |
| Ускорение | — | 8,3 | 10,35 |

Для 10 000 000 элементов

|  |  |  |  |
| --- | --- | --- | --- |
| Использованные средства | Последовательная версия | Open MP | TBB |
| Время (с) | 10 | 1,050 | 0,8 |
| Ускорение | — | 9,5 | 12,5 |

На рисунке 4 приведён график зависимости ускорения от количества процессов. По горизонтальной оси указывается число процессов, по вертикальной – значение ускорения.

Рис. 4: График ускорения.

# VIII.Заключение

В ходе работы были реализованы три реализации нисходящей поразрядной сортировки MSD radix sort для целых чисел – последовательная, с использованием технологии Open MP и с использованием TBB. По результатам экспериментов можно сделать вывод о том, что параллельная реализация имеет смысл, реализация на Open MP и TBB имеют практически одинаковые ускорения

Из графика ускорения видно, что распараллеливание алгоритма поразрядной сортировки дало значительное увеличение ускорения.

## Список литературы

[1]А. А. Сиднев, А. В. Сысоев, И. Б. Мееров. Учебный курс «Технологии разработки параллельных программ»

[2]Дональд Кнут. Искусство программирования, том 3. Сортировка и поиск – 2-е изд. – М.:

«Вильямс», 2007.