Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных  
технологий математики и механики

Отчет по лабораторной работе

**Поразрядная сортировка для вещественных чисел (тип double) со слиянием «Разделяй и властвуй»**

**Выполнила**:студентка группы 1506-3

*Толоконцева Н.В.*

**Проверил:**

преподаватель кафедры МО ЭВМ

*Сысоев А.В.*

Нижний Новгород

2018

**Содержание**

[Введение 2](#_Toc510133761)

[Постановка задачи 4](#_Toc510133762)

[Пример запуска программ: 9](#_Toc510133763)

[Результаты вычислительных экспериментов 10](#_Toc510133764)

# Введение

Необходимость упорядочивания элементов в некотором множестве возникает довольно часто. Задача сортировки данных, решаемая для массивов и файлов, может быть сформулирована следующим образом:  
Пусть имеется массив (файл) записей R1, R2, …, RN с ключами K1, K2, …, KN . На множестве ключей Ki, i=1, N вводится такое отношение порядка типа “≤”, что для любых трех значений ключей a, b, c выполняются следующие условия:  
1) рефлексивность: a≤a;  
2) антисимметричность: если a≤b, b≤a, то a=b;  
3) транзитивность: если a≤b, b≤c, то a≤c;  
4) линейность: для произвольных a и b   
Любое множество записей, ключи которых удовлетворяют свойствам 1-4, называется полностью (линейно) упорядоченным (или совершенно упорядоченным).   
Задача упорядочения (сортировки) данных состоит в том, чтобы найти такую перестановку записей, т.е. такую комбинацию их взаимного расположения, при которой ключи Ki расположились бы в неубывающем порядке:  
K1 ≤ K2 ≤ … KN.

Существует множество алгоритмов сортировки. В данной работе рассматривается поразрядная сортировка по младшим разрядам. Для слияния используется стратегия «Разделяй и властвуй»

# Постановка задачи

Задача формулируется следующим образом: требуется разработать программу, которая сортирует массив состоящий из вещественных чисел (тип double), с использованием методов параллельного программирования.

Для этого разработать последовательный и параллельный алгоритм данной задачи. Затем сравнить время выполнения на нескольких процессах и потоках, определить ускорение.

Программа должна использовать многопоточные библиотеки OpenMP и TBB.

Выполнение задачи включает:

1. Освоение темы (постановка задачи).
2. Изучение метода решения.
3. Разработку схемы параллельных вычислений.
4. Реализацию программы.
5. Проведение вычислительных экспериментов с анализом масштабируемости.
6. Подготовку отчета с анализом результатов экспериментов.

Программа должна состоять из нескольких частей:

1. Генератор тестов, создающий файлы с неотсортированными массивами и с «эталонами» для сортировки
2. Сортировщик, производящий сортировку массива, созданного генератором, и записывающий время сортировки.
3. Checker, проверяющий соответствие файла, полученного в результате сортировки, и «эталона»

# Теоретические основы поразрядной сортировки

В поразрядная сортировке используется знание о внутреннем устройстве числа.

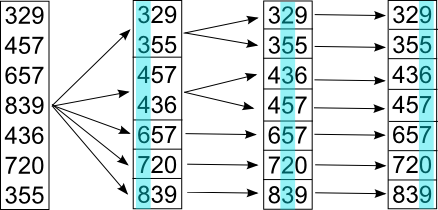
До сортировки необходимо установить два параметра: k и m, где

* k - количество разрядов в самом длинном ключе
* m - разрядность данных: количество возможных значений разряда ключа

В случае сортировки чисел типа double лучше всего использовать побайтовое деление, при этом k будет равняться 8, а m 256.

Алгоритм сортировки заключается в том, что на каждом шаге числа сортируются по i разряду начиная с самого младшего, каждого числа, при этом порядок записи внутри одного значения должен сохраняться.

Пример такой сортировки проиллюстрирован на картинке ниже:



Сортировка чисел типа double имеет некоторые отличия от сортировки целых чисел. Эти отличия продиктованы структурой хранения числа.

Если значение поля [Порядок] у одного числа больше соответствующего значения у другого, то первое число обязательно больше второго. Это верно для неотрицательных чисел. Если порядки равны, то сравниваются мантиссы. Это полностью соответствует обычной системе для целых чисел, когда сравниваются сначала старшие двоичные цифры, затем младшие.

Таким образом, можно интерпретировать числа стандарта IEEE-754 как целые соответствующей длины:

a > b равносильно (unsigned long long)a > (unsigned long long)b, если a, b > 0.

В случае отрицательных чисел, необходимо иначе обрабатывать первый бит, поскольку 1 обозначает отрицательное число, а 0 положительное.

# Описание последовательного алгоритма поразрядной сортировки

Функция radix\_sort(arr, res, size)принимает на вход указатель на исходный массив, указатель на массив для хранения результата сортировки, размер массива.

До начала вычислений создаётся массив со списками («карманы» для хранения результата распределения по i-ому разряду).

Далее сам алгоритм сортировки: в цикле от 0 до количества разрядов сортируем массив по i-ому разряду.

Каждая итерация представляет собой распределение всех чисел по карманам и последующее последовательное слияние всех корзин в исходный массив:

* Для распределения числа в корзину используется функция *process\_number (arr\_of\_lists, number, digit\_num),* которая на вход принимает указатель на массив «карманов», текущее число и номер разряда. «Цифра», соответствующая переданному разряду получается с помощью приведения числа к типу unsigned long long int, побитового сдвига и применению маски:

ulong n;

std::memcpy(&n, &d, sizeof d);

n >>= offset \* BITES\_IN\_DIGETS;

n &= MASK;

* Далее число записывается в карман, соответствующий полученной цифре.
* После распределения всех чисел по карманам, происходит их объединение. Для этого используется функция merge\_lists(res, arr\_list, size, digit\_num). В зависимости от разряда объединение карманов в массив происходит по разному. В случае, когда разряд не является старшим, элементы извлекаются последовательно из 0, 1, 2 и т.д. карманов. Если же разряд старший, то сначала извлекаются элементы (причем извлекаются в обратном порядке) из 255, 254, 253 и т.д. вплоть до 128 кармана, а потом из 0, 1, 2, … , 127 карманов (уже в обычном порядке). Так сделано из-за того, что старший разряд содержит в себе бит, отвечающий за знак.

# Описание параллельной реализации (Open MP)

Суть параллельной реализации поразрядной сортировки заключается в следующем:

1. Исходный массив разбивается на несколько частей. Число под-массивов подобрано опытным путём:

|  |  |
| --- | --- |
| Число потоков | Количество под-массивов |
| 2 | 4 |
| 4 | 16 |
| 8 | 8 |

Размер под-массивов фиксирован и равен

**(общий размер массива) / (число под-массивов)**

1. Каждый поток параллельно сортирует свои под-массивы. Распределение под-массивов по потокам выполнено с помощью #pragma omp for schedule(dynamic)
2. После того, как все потоки закончат сортировку, происходит слияние (стратегия «Разделяй и властвуй»). В начале каждой итерации слияния поток-мастер вычисляет индексы, нужные для слияний, и записывает их в общую память. Далее, используя эти индексы, все потоки параллельно сливают полученные отсортированные массивы. Для этого также используется #pragma omp for schedule(dynamic)

**Инструкция пользователя**

В программу входит 3 исполняемых файла: TestGenerator.exe, Solver.exe, Checker.exe.

* TestGenerator.exe

Принимает на вход три аргумента: имя результирующего файла, номер теста, имя файла с эталоном.

Случайные числа генерируются в диапазоне от -10^6 до 10^6

|  |  |  |
| --- | --- | --- |
| **Номер теста** | **Размер массива** | **Генерация чисел в массиве** |
| 0 | 1 | Случайно |
| 1 | 1000 | Все числа в массиве уже отсортированы |
| 2 | 1000 | Все числа в массиве отсортированы в обратном порядке |
| 3 | 1000 | Числа заданы случайно, гарантированно несколько чисел повторяются |
| 4 | 1000 | Все числа в массиве одинаковы |
| 5 | 10 | Случайно |
| 6 | 50 | Случайно |
| 7 | 100 | Случайно |
| 8 | 500 | Случайно |
| 9 | 500 | Случайно |
| 10 | 1000 | Случайно |
| 11 | 5000 | Случайно |
| 12 | 5000 | Случайно |
| 13 | 5000 | Случайно |
| 14 | 10000 | Случайно |
| 15 | 10000 | Случайно |
| 16 | 10000 | Случайно |
| 17 | 50000 | Случайно |
| 18 | 50000 | Случайно |
| 19 | 50000 | Случайно |
| 20 | 100000 | Случайно |
| 21 | 100000 | Случайно |
| 22 | 100000 | Случайно |
| 23 | 500000 | Случайно |
| 24 | 500000 | Случайно |

В результате создаются два бинарных файла с указанными именами.

Файл с неотсортированным массивом содержит размер массива и все его элементы.

Файл с эталоном (сортировка производится с помощью std::sort()) содержит размер массива, время сортировки и все его элементы (отсортированные).

* Solver.exe  
  Принимает на вход 3 параметра: имя файла с неотсортированным массивом, имя результирующего файла и число потоков.

Программа считывает массив из файла, сортирует его переданным числом потоков, и записывает результат сортировки.

Результирующий файл содержит размер массива, время сортировки и все элементы массива, полученного в результате сортировки.

* Checker.exe

Принимает на вход 3 параметра: имя файла с исходным массивом, имя файла с результатом сортировки, имя файла с эталоном.

Программа сравнивает результат сортировки с эталоном и пишет вердикт и время сортировки в файл «result.txt». Этот файл всегда перезаписывается. Кроме того, время сортировки дополнительно записывается в файл «times.txt», этот файл не перезаписывается.

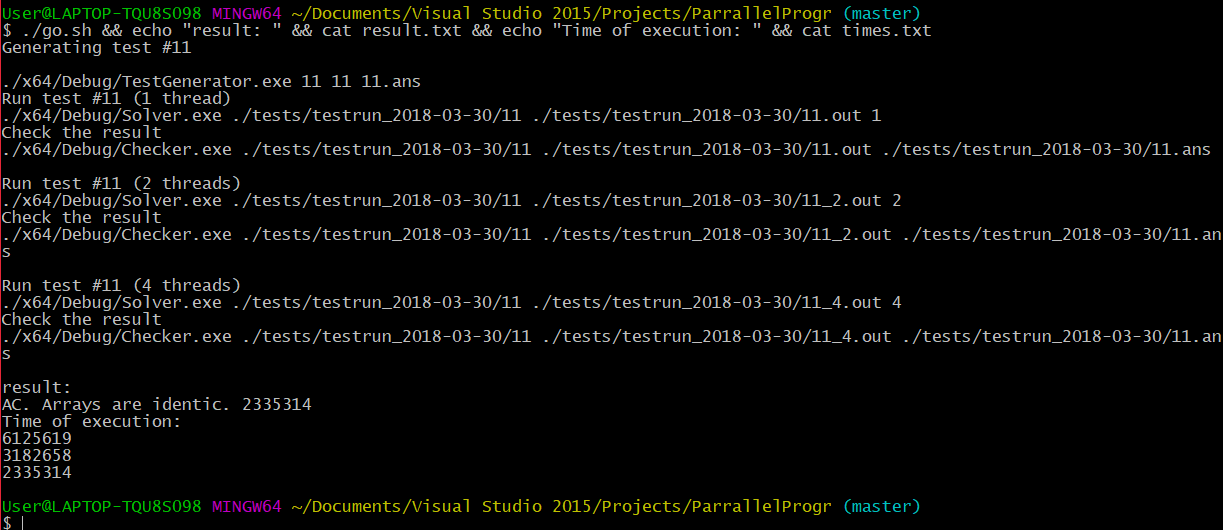
Программа может вынести 3 вердикта:

* + AC = Accepted = Решение выдаёт корректный результат на данном тесте
  + WA = Wrong Answer = Решение выдаёт некорректный результат на данном тесте
  + PE = Presentation Error = Ошибка формата выходных данных

В случае, если вердикт «WA» или «PE», exit code программы будет 1.

### Пример запуска программ:

Последовательно вызывается генерация теста №11, запуск сортировки в 1, 2 и 4 потока, с проверкой результата после каждой сортировки. В конце на консоль выводится вердикт последней сортировки (корректность результата автоматизировано проверяется по exit коду) и время работы каждой из сортировок. Для простоты все программы запускаются из bash-скрипта.



# Результаты вычислительных экспериментов

Вычислительные эксперименты для оценки эффективности параллельного алгоритма проводились при следующих условиях и оборудовании:

Компиляция кода проходила на ПК с параметрами:

* Процессор – AMD – A6-7310, Core 4 х 2.0 ГГц, Кэш L2 2Мб.
* Оперативная память - 6 GB.
* Операционная система - Microsoft Windows 10 Домашняя.
* Компилятор - MS Visual Studio 2015.
* Используемые библиотеки: OpenMP.

Tp – время работы в миллисекундах.

Sp – ускорение.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | Последователь  ный  алгоритм | Параллельный алгоритм | | | | | |
| 2 потока | | 4 потока | | 8 потоков | |
| Tp | Sp | Tp | Sp | Tp | Sp |
| 10000 | 1,2128149 | 0,6576024 | 1,8442982 | 0,433375 | 2,7985338 |  |  |
|
| 20000 | 2,3854087 | 1,3872 | 1,7196025 | 0,8973 | 2,6584291 |  |  |
|
| 30000 | 3,5386679 | 1,9305 | 1,8330658 | 1,276387 | 2,7724096 |  |  |
|
| 40000 | 4,7216758 | 2,5806 | 1,8297145 | 1,676401 | 2,8165552 |  |  |
|
| 50000 | 6,7390647 | 3,3639 | 2,0033348 | 2,080923 | 3,2384977 |  |  |
|
| 60000 | 7,056718 | 3,8588 | 1,8287336 | 2,469193 | 2,8579044 |  |  |
|
| 70000 | 8,2732648 | 4,4949 | 1,8405867 | 2,860793 | 2,8919477 |  |  |
|
| 80000 | 9,4848876 | 5,1361 | 1,8467161 | 3,296996 | 2,8768268 |  |  |
|
| 90000 | 10,610541 | 5,7797 | 1,8358197 | 3,757794 | 2,8236090 |  |  |
|
| 10000 | 11,790396 | 6,4099 | 1,8394167 | 4,210077 | 2,80051794 |  |  |
|

# Заключение

В результате работы была написана программа, позволяющая отсортировать массив элементов типа double с помощью поразрядной сортировки. Был реализован как последовательный, так и параллельный алгоритмом. Проведены серии экспериментов с анализом масштабируемости, показывающие ускорение параллельного алгоритма, достоинства и важность изучения методов параллельных вычислений.

**Список литературы**

1. Гергель В. П. Учебный курс «Параллельное программирование».
2. Балло Л.В., Барышева И.В., Гергель В.П., Гришагин В.А., Долгов Г.А., Кулакова А.П, Стронгин Р.Г. Учебное пособие «Программа общего курса и описание лабораторных работ».