Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра автоматизации систем вычислительных комплексов

Отчёт по заданию № 1

Измерение времени выполнения программ и её оптимизация

Задорожная Юлия

321 группа

Москва 2021

# Оглавление

Постановка задачи…………………………………………………………………………………… 1

Характеристика системы………………………………………………………………………….. 1

Замеры времени работы первоначальной программы с помощью утилиты time…………………………………...…………………………………………………………………... 1

# Способы ускорения выполнения программы и результаты их применения……... 2

# Замеры времени работы первоначальной программы с помощью утилиты time. 3

# Вывод……………………………………………………………………………………………………… 3

# Постановка задачи

Необходимо измерить время выполнения заданной программы с помощью утилиты time. Повторить замеры несколько и вывести среднее до и после редактирования программы. Далее требуется оптимизировать программу так, чтобы время ее выполнения уменьшилось, а концепция и вывод оставались теми же. Также нужно пояснить, почему выбранные изменения ускорили работу программы.

**Конфигурация вычислительной системы**

|  |  |
| --- | --- |
| Процессор | Intel® Core™ i5, 10210U, 1.60 GHz |
| Операционная система | Ubuntu 20.04 x64 |
| RAM | 16 Gb |

# Замеры времени работы первоначальной программы при помощи утилиты time

Обозначения:

real — реальное (общее) время выполнения;

|  |  |  |
| --- | --- | --- |
| **Исходный Файл** | **Оптимизация компилятора** | **AVG real time, сек (после 5 тестов)** |
| p\_var2.c (начальная программа) | O0 | 3,86 |
| O1 | 1.48 |
| O2 | 1.49 |
| O3 | 1.43 |

На данной машине с описанными выше характеристиками программа выполнялась чуть менее, чем за 4 секунды, а все оптимизации компилятора давали приблизительно одинаковый прирост в скорости в 2,5 раза.

# Способы ускорения выполнения программы и результаты их применения

Для ускорения работы программы были применены следующие методы:

* Второй тройной вложенный цикл и также два обращения в двумерный массив заметно тормозят программу. Избавимся от них, заведя новый одномерный массив C, каждый элемент которого будет суммой соответствующей строки матрицы B. Обнулим его элементы и заполним одновременно с заполнением матрицы B. Очевидно, вторым этапом просто приравниваем каждый элемент матрицы A к сумме двух элементов массива C. После такого “распараллеливания” сложность программы на O(n^3) заменена на O(n^2).
* В программе первым делом заполняются элементы матрицы B, затем элементы матрицы A обнуляются, и в каждый элемент циклично записываются элементы двух строк матрицы B. Следовательно, вместо того, чтобы множество раз считать сумму строки, модифицированная программа считает ее всего 1 раз.
* Более того, оптимизации компиляции была применена компиляция с ключами оптимизации

|  |  |
| --- | --- |
| Компиляторы | Ключи оптимизации |
| gcc | -O0, -O1, -O2, -O3 |
| Icc | -O0, -O1, -O2, -O3, -fast |
| Ccc | -O0, -O1, -O2, -O3, -O4, -fast |

# Замеры времени работы оптимизированной программы при помощи утилиты time (без автоматической оптимизации)

Обозначения:

real — реальное (общее) время выполнения;

|  |  |  |
| --- | --- | --- |
| **Исходный Файл** | **Оптимизация компилятора** | **AVG real time, мс (после 5 тестов)** |
| Task1.c | O0 | 34 |
| O1 | 20 |
| O2 | 22 |
| O3 | 14 |

Вновь проведены замеры времени выполнения программы и в таблицу вынесены

средние арифметические значения Real time. Модифицированная программа работает заметно быстрее (при одинаковом виде оптимизации компилятора). Если же посмотреть на

AVG real time, то модифицированная программа с оптимизацией O3 быстрее, чем

оригинальная программа в ~240 раз.

**Вывод**

Основной эффект был достигнут с помощью оптимизации алгоритма. Нам удалось улучшить асимптотику, уменьшив ее с кубической до квадратичной. Это также заметно из сравнения результатов последнего столбца двух таблиц: улучшение времени работы изменилась (уменьшилось!) заметно.

Основной эффект дала оптимизация самого алгоритма. Нам удалось улучшить асимптотику с кубической до квадратичной, а применяя опцтимизацию компилятора удалось добиться увеличения скорости еще в несколько раз.