ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ бюджетное ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики, механики и компьютерных наук им. И.И. Воровича

Кафедра алгебры и дискретной математики

**ОТЧЕТ**

на тему:

### **Блочные вычисления. Модели времени выполнения программ. Блочные размещения массивов, дополняющие блочные вычисления**

**Выполнила:**

студентка 4 курса 2 группы

Жинкина А.А.

Ростов-на-Дону

2018

**Содержание**

[Постановка задачи 3](#_Toc487789905)

[Алгоритм решения 3](#_Toc487789908)

[Результаты работы программы 4](#_Toc487789912)

[Характеристики компьютера 5](#_Toc487789916)

[Сравнение результатов 5](#_Toc487789931)

[Выводы 7](#_Toc487789938)

**Постановка задачи**

Задание 32.

Написать программу блочного умножения двух матриц C = A\*B.

Матрица A симметричная, хранится как верхне-треугольная. Хранится в виде одномерного массива по блочным столбцам.

Матрица B верхне-треугольная. Хранится в виде одномерного массива по блочным столбцам.

Распараллелить блочную программу умножения двух матриц C = A\*B с использованием технологии OpenMP двумя способами

* Перемножение каждых двух блоков выполнить параллельно
* В разных вычислительных ядрах одновременно перемножать разные пары блоков.

Определить оптимальные размеры блоков в обоих случаях. Провести численные эксперименты и построить таблицу сравнений времени выполнения различных программных реализаций решения задачи. Определить лучшие реализации.

Проверить корректность (правильность) программ.

**Алгоритм решения**

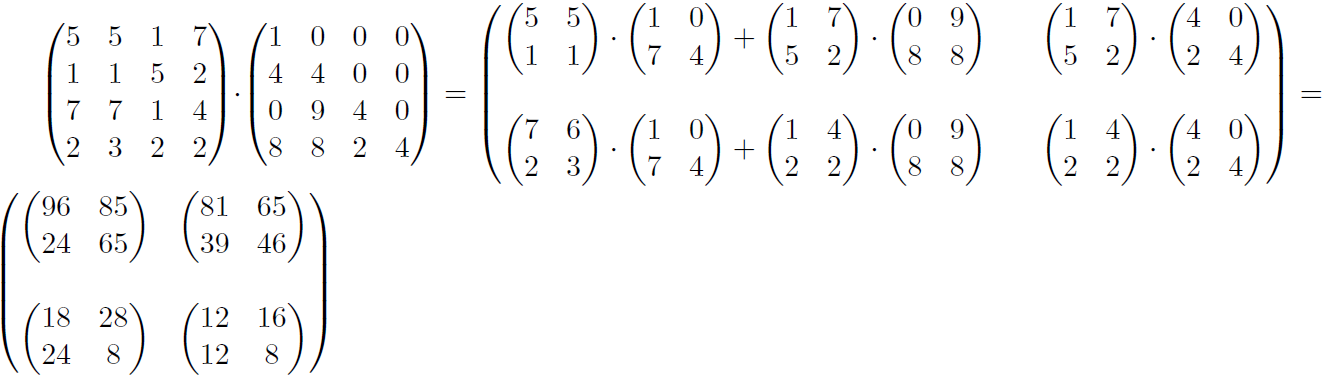


Рис. 1 Демонстрация блочного умножения матрицы 4x4 с разбиением на блоки размером 2x2

Алгоритм умножения будет выглядеть следующим образом:

1. Организуем стандартный алгоритм умножения для блоков:

for (i = 0; i<S; ++i)

for (j = 0; j<S; ++j)

for (k = j; k<S; ++k)

S=N/k, где N – количество строк/столбцов исходной матрицы, k – количество строк/столбцов в блоке.

1. Найдем индекс начала необходимого блока, записанного в векторе.

Размер массива, в котором хранится матрица А определяется по формуле арифметической прогрессии: Sn = (n + 1) \* n / 2 где n – размер матрицы А.

1. Организуем стандартный алгоритм умножения для элементов внутри блока.

Внутри всех блоков расположение элементов построчное, поэтому для перехода на следующую строку требуется умножение индекса строки на количество элементов. Изменение индексов элементов будет происходить одинаково во всех блоках независимо от представления матрицы в векторе.

**Проверка правильности работы программы**

Проверка правильности работы программы проверяется в процессе выполнения путем сравнения двух матриц: обычного умножения и блочного.

**Результаты работы программы**

При блочном умножении матриц размером 512x512 получили результаты, которые можно представить в виде графика:

**Характеристики компьютера**

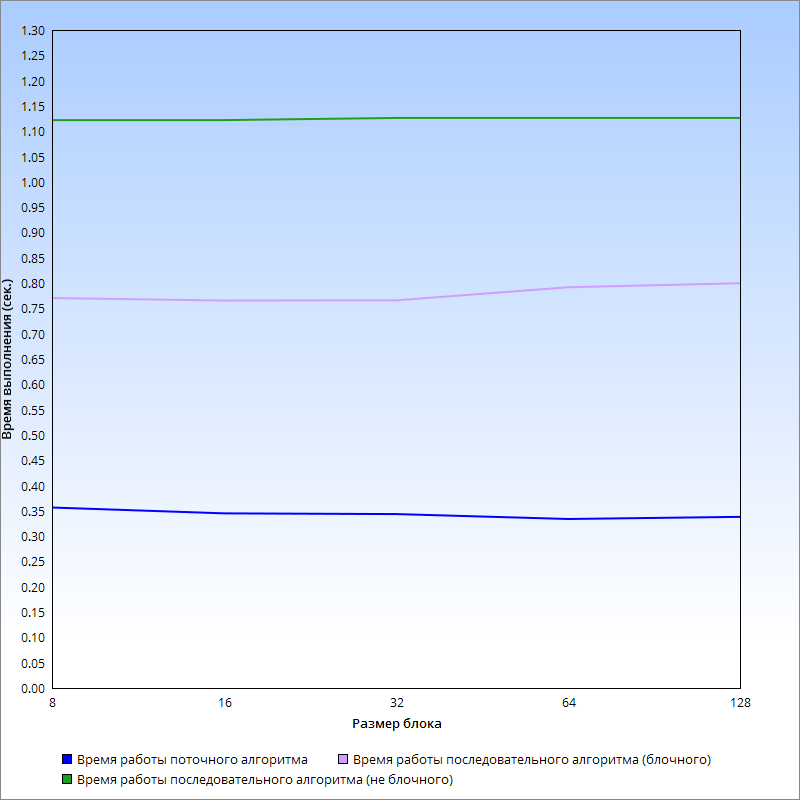
Процессор Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, 2712 МГц.

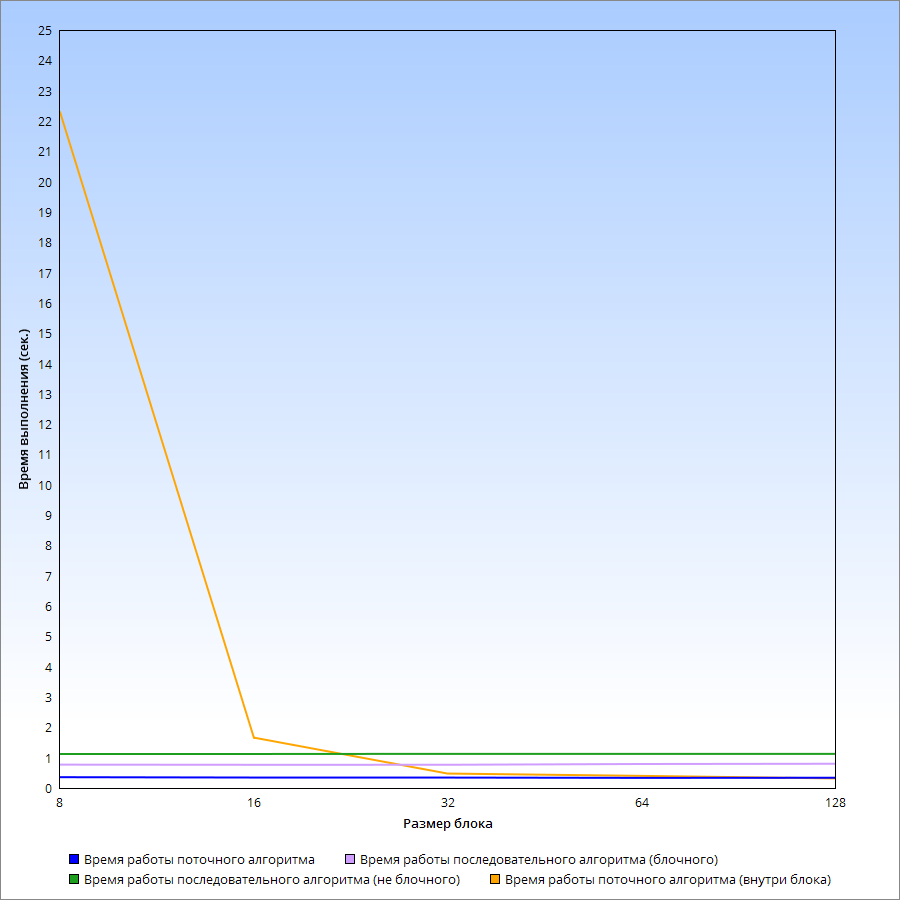
Количество ядер: 2.

Количество логических процессоров: 4.

Установленная оперативная память (RAM): 8,00 ГБ.

**Сравнение результатов**

****

****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Размеры матриц** | **Размер блока** | **Время работы поточного алгоритма** | **Время работы последовательного алгоритма (блочного)** | **Время работы последовательного алгоритма (не блочного)** | **Время работы поточного алгоритма (внутри блока)** |
| 512 | 8 | 0.356542 | 0.770405 | 1.121908 | 22.3124 |
| 512 | 16 | 0.345009 | 0.765482 | 1.121994 | 1.66404 |
| 512 | 32 | 0.343604 | 0.765981 | 1.126237 | 0.476617 |
| 512 | 64 | 0.333881 | 0.791766 | 1.126231 | 0.400331 |
| 512 | 128 | 0.338137 | 0.799721 | 1.126233 | 0.315599 |

Табл. 1. – Тестирование результатов работы программы

Поточный алгоритм: потоки разделяются по блочным строкам/столбцам. Было выяснено, что такое распределение по потокам работает быстрее всего.

**Выводы**

Неоптимальные варианты - разбиения матрицы на блоки размером 1x1 и 512x512. Такой результат обосновывается особенностью работы с кэш-памятью: при наилучшем разбиении, количество кэш-промахов уменьшается. Как следствие, получаем более быструю работу программы.