

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики, механики и компьютерных наук им. И.И. Воровича
Кафедра прикладной математики и программирования

ОТЧЕТ

НА ТЕМУ:

**БЛОЧНЫЕ ВЫЧИСЛЕНИЯ. МОДЕЛИ ВРЕМЕНИ ВЫПОЛНЕНИЯ
ПРОГРАММ. БЛОЧНЫЕ РАЗМЕЩЕНИЯ МАССИВОВ,
ДОПОЛНЯЮЩИЕ БЛОЧНЫЕ ВЫЧИСЛЕНИЯ**

Выполнила:
студентка 4 курса 2 группы
Чубинидзе Надежда
Романиевна

Ростов-на-Дону
2018

Содержание

Постановка задачи.....	3
Характеристики компьютера	3
Алгоритм выполнения программы.....	4
Проверка программы на корректность.....	5
Результаты работы программы.....	6
Таблица результатов.....	6
Графики зависимости.....	7
Выводы.....	7

Постановка задачи

Задание 48

Написать программу блочного умножения двух матриц $C = A * B$. Матрица A верхне-треугольная. Хранится в виде одномерного массива по блочным столбцам.

Матрица B симметричная, хранится как 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 каждого 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.

Характеристики компьютера

Процессор: Intel® Core™ i5-3330 CPU @ 3.00GHz 3.00GHz:

- кол-во ядер: 4
- размер кэша: 6 MB.

Память: DDR3 4 GB

Операционная система: Windows 7

Алгоритм выполнения программы

Для не параллельного, не блочного умножения матриц применен следующий алгоритм:

```
for (int i=0; i < size_array; ++i)
{
    if (i != 0 && i%size == 0)
    {
        c++;
        j = 0;
    }
    for (int k = c; k < size; ++k)
    {
        if (k < j)
            C[i] += A[k*(k + 1) / 2 + c] * B[j*(j + 1) / 2 + k];

        else
            C[i] += A[k*(k + 1) / 2 + c] * B[k*(k + 1) / 2 + j];

        ++j;
    }
}
```

Где `size_array` – размер массива, который хранит результат умножения матриц $A * B = C$, где блоки матрицы C хранятся построчно, `size` – порядок квадратных матриц A и B .

Рассмотрим алгоритм блочного умножения матриц

Алгоритм блочного умножения матриц организован относительно матрицы A :

- Если блок располагается на диагонали, то берется исходный блок
- Если блок располагается ниже главной диагонали, то он игнорируется, так как состоит из 0-матрицы
- Если блок выше главной диагонали, то берется исходный блок.

Для матрицы B :

- Если блок расположен ниже главной диагонали, то выбирается ему симметричный блок и транспонируется.
- Если блок расположен на диагонали, то элементы ниже главной диагонали берутся симметрично относительно ее.

Внутри блоков алгоритм перемножения организован аналогично не блочному алгоритму. Происходит только пересчет индексов и внутреннего цикла по k

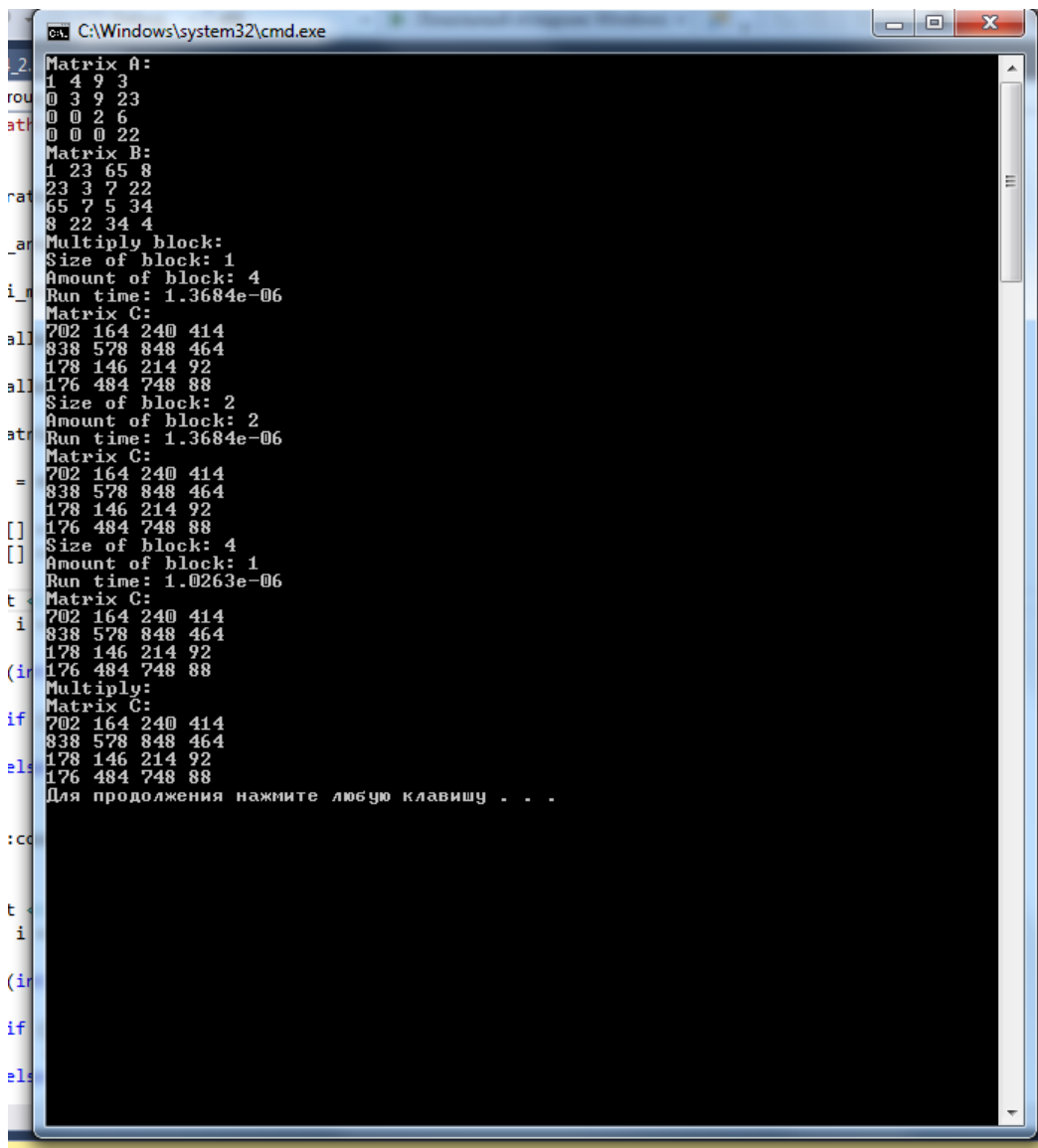
Результирующая матрица C хранится в виде вектора блочными строками.

Проверка программы на корректность

```
A := Matrix([[1, 4, 9, 3], [0, 3, 9, 23], [0, 0, 2, 6], [0, 0, 0, 22]]);  
B := Matrix([[1, 23, 65, 8], [23, 3, 7, 22], [65, 7, 5, 34], [8, 22, 34, 4]]);
```

$$\begin{bmatrix} 1 & 4 & 9 & 3 \\ 0 & 3 & 9 & 23 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 0 & 22 \end{bmatrix}$$
$$\begin{bmatrix} 1 & 23 & 65 & 8 \\ 23 & 3 & 7 & 22 \\ 65 & 7 & 5 & 34 \\ 8 & 22 & 34 & 4 \end{bmatrix}$$

```
C := evalm(A*B);
```

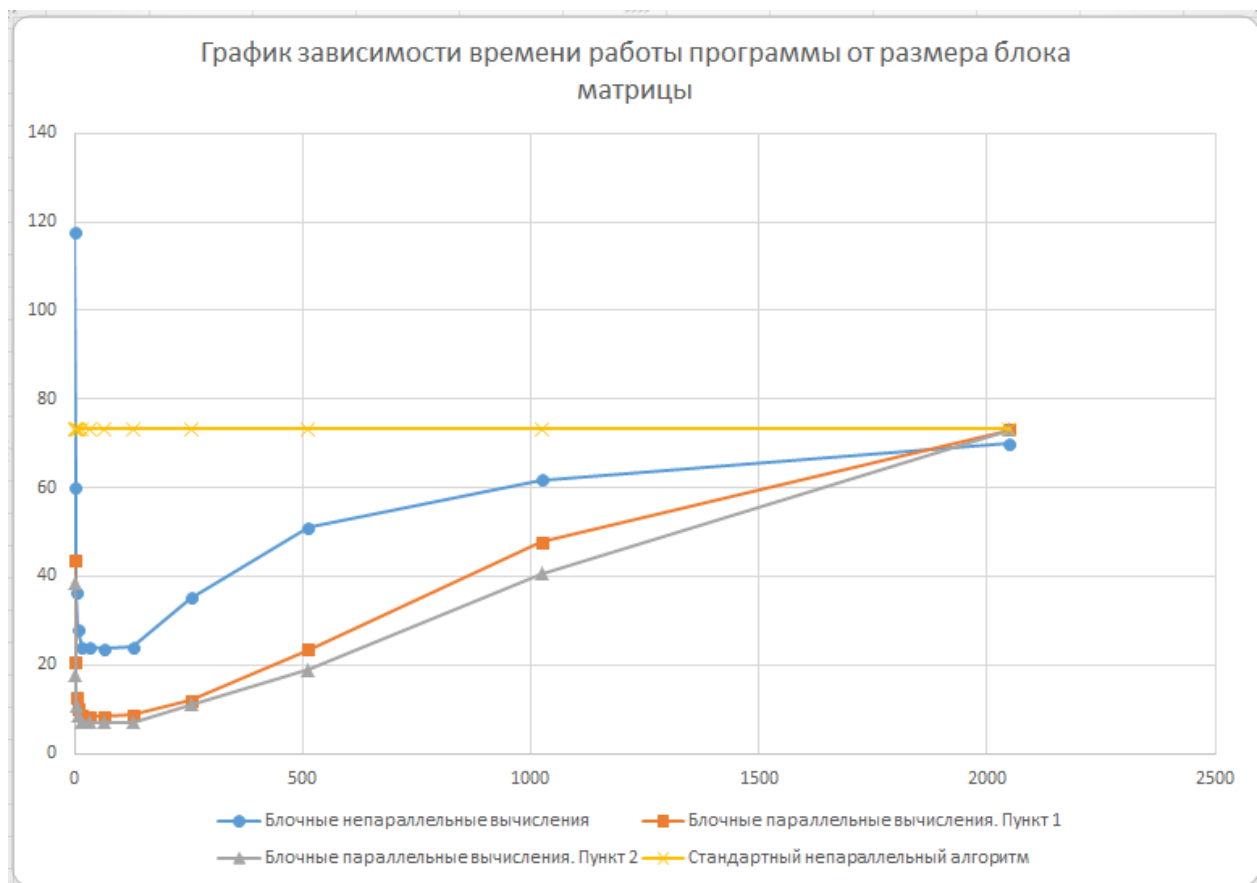
$$\begin{bmatrix} 702 & 164 & 240 & 414 \\ 838 & 578 & 848 & 464 \\ 178 & 146 & 214 & 92 \\ 176 & 484 & 748 & 88 \end{bmatrix}$$


```
C:\Windows\system32\cmd.exe  
_2. Matrix A:  
rou 1 4 9 3  
ath 0 3 9 23  
    0 0 2 6  
    0 0 0 22  
Matrix B:  
rat 1 23 65 8  
    23 3 7 22  
    65 7 5 34  
    8 22 34 4  
_an Multiply block:  
    Size of block: 1  
i_n Amount of block: 4  
    Run time: 1.3684e-06  
Matrix C:  
all 702 164 240 414  
    838 578 848 464  
    178 146 214 92  
all 176 484 748 88  
    Size of block: 2  
atr Amount of block: 2  
    Run time: 1.3684e-06  
Matrix C:  
= 702 164 240 414  
   838 578 848 464  
   178 146 214 92  
[] 176 484 748 88  
[] Size of block: 4  
   Amount of block: 1  
   Run time: 1.0263e-06  
t < Matrix C:  
i 702 164 240 414  
  838 578 848 464  
(ir 178 146 214 92  
   176 484 748 88  
Multiply:  
if 702 164 240 414  
  838 578 848 464  
els 178 146 214 92  
   176 484 748 88  
Для продолжения нажмите любую клавишу . . .  
:cc  
t <  
i  
(ir  
if  
els
```

Результаты работы программы

Вычисления проводились на матрице порядка 2048. Количество потоков -8.
При стандартном способе перемножения матриц время выполнения: 73,1571

Размер блоков	Блочный алгоритм (непараллельный)	Блочный алгоритм (распараллеливание перемножения пары блоков)	Блочный алгоритм (распараллеливание перемножения разных пар блоков)
1	117,485	43,5936	38,5417
2	59,9458	20,5662	17,5778
4	36,4558	12,5548	10,9188
8	28,0791	9,90128	8,58412
16	23,8937	8,56171	7,24827
32	23,9782	8,37598	7,11661
64	23,6503	8,25666	7,04644
128	24,0541	8,67087	7,0015
256	35,2219	11,9836	11,1424
512	50,941	23,5405	18,9013
1024	61,6836	47,7845	40,5446
2048	69,9275	73,1447	73,0572



Вывод

Лучший вариант разбиения матрицы на блоки 64×64 или 128×128 . Неоптимальный вариант разбиения матрицы на блоки размера $1 \times 1, 1024 \times 1024, 2048 \times 2048$. Замедленные параллельные варианты объясняются затратой процессора на “забор” данных из памяти и нехваткой или неоптимальным использованием кэша. Если разбивать вычисления по пункту 2: в разных вычислительных ядрах одновременно перемножать разные пары блоков, можно заметить, что в таком случае программа работает оптимальнее.