МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Южный федеральный университет»

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

Направление подготовки 02.04.02 – Фундаментальная информатика и информационные технологии

ОТЧЕТ ПО ТЕМЕ  
«Блочное перемножение матриц»

Выполнила:  
магистр 1 года 6 группы  
Шкуро Т.А.

Содержание

[Постановка задачи 3](#_Toc24320469)

[Архитектура компьютера 3](#_Toc24320470)

[Характеристики операционной системы, языка программирования и компилятора 3](#_Toc24320471)

[Результаты для компилятора MSVC 4](#_Toc24320472)

[Результаты для компилятора g++ 6](#_Toc24320473)

[Сравнительный анализ 8](#_Toc24320474)

[Заключение 8](#_Toc24320475)

# Постановка задачи

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

Матрица A нижне-треугольная, хранится в виде одномерного массива по столбцам.

Матрица B симметричная, хранится как верхне-треугольная, в виде одномерного массива по столбцам.

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

* последовательное выполнение – при перемножении блоков не используется параллельности;
* параллельное 1 – параллельно выполняется умножение различных пар блоков;
* параллельное 2 – параллельно выполняется умножение каждого отдельного блока;

Провести численные эксперименты и построить таблицу сравнений времени различных программных реализаций решения задачи. Определить лучшие реализации.

# Архитектура компьютера

Processor: Intel® Core™ i7-4700MQ CPU @ 2.40GHz; 4 cores;

L3: 6 MB;

L2: 1 MB;

L1: 256 KB;

RAM: DDR3, 16 GB.

# Характеристики операционной системы, языка программирования и компилятора

OS: Windows 8 x64;

Programming language: C++;

Compilers:

* MSVC v. 19.16.27024.1, compiler options: /O2 /openmp
* GСС v. 7.4.0, compiler option: -fopenmp -Ofast;

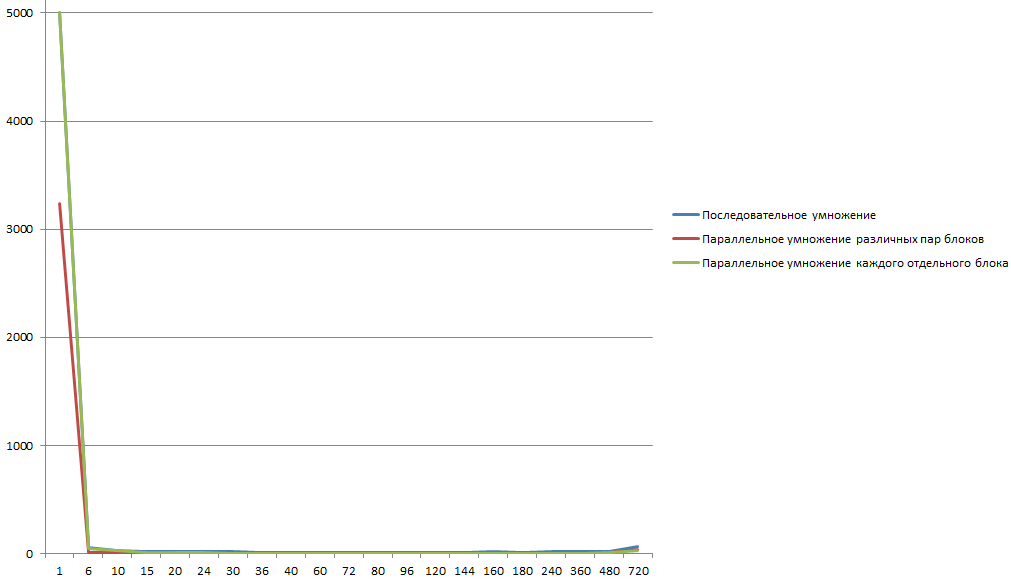
# Результаты для компилятора MSVC

Время выполнения трёх вариантов программных реализаций для типа данных double и компилятора MSVC представлено в таблице 1.

Таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер блока** | **Последовательное умножение** | **Параллельное умножение различных пар блоков** | **Параллельное умножение каждого отдельного блока** |
| 1 | >5000 | 3236.21 | >5000 |
| 6 | 55.053 | 10.337 | 50.539 |
| 10 | 30.636 | 5.803 | 30.58 |
| 15 | 22.308 | 4.487 | 13.59 |
| 20 | 17.74 | 3.561 | 8.496 |
| 24 | 16.38 | 3.257 | 6.622 |
| 30 | 15.25 | 3.241 | 5.335 |
| 36 | 13.8 | 2.914 | 4.517 |
| 40 | 13.4 | 2.9 | 4.293 |
| 60 | 12.9 | 2.62 | 3.626 |
| 72 | 12.853 | **2.56** | **3.394** |
| 80 | 12.95 | 2.68 | 3.65 |
| 96 | 13.641 | 3.166 | 4.217 |
| 120 | **12.672** | 2.938 | 3.66 |
| 144 | 14.233 | 3.687 | 4.237 |
| 160 | 16.263 | 4.015 | 4.427 |
| 180 | 13.168 | 3.42 | 3.8 |
| 240 | 15.282 | 4.36 | 4.75 |
| 360 | 16.764 | 5.21 | 5.001 |
| 480 | 19.183 | 8.543 | 13.791 |
| 720 | 65.148 | 37.045 | 30.531 |

На рисунке 1 представлен график зависимости времени счета от размерности блоков для трех вариантов вычислений.

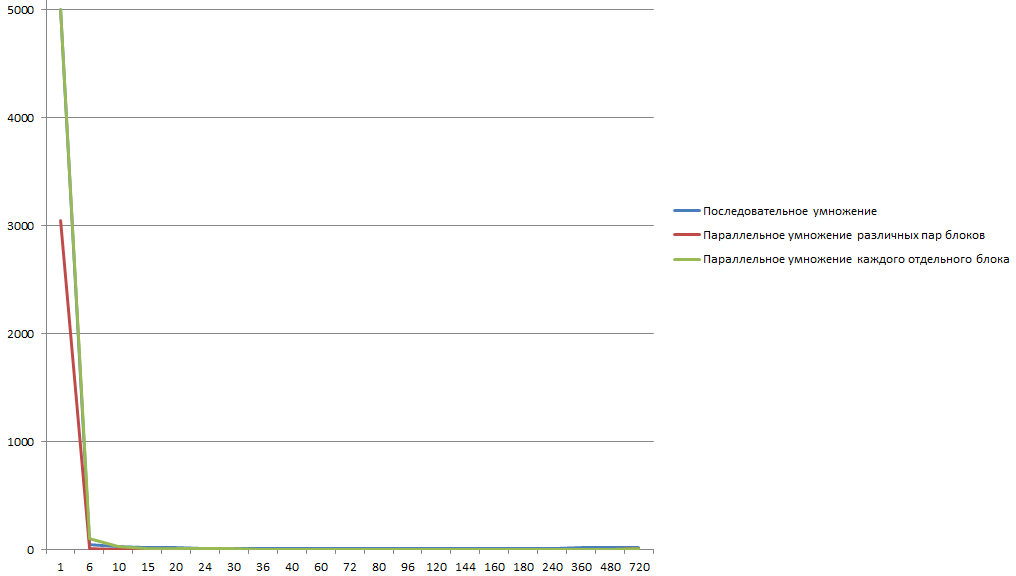
**Рисунок 1

Время выполнения трёх вариантов программных реализаций для типа данных float и компилятора MSVC представлено в таблице 2.

Таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер блока** | **Последовательное умножение** | **Параллельное умножение различных пар блоков** | **Параллельное умножение каждого отдельного блока** |
| 1 | >5000 | 3046.44 | >5000 |
| 6 | 49.341 | 10.122 | 101.402 |
| 10 | 27.82 | 5.375 | 26.603 |
| 15 | 19.72 | 3.967 | 11.198 |
| 20 | 16.016 | 3.203 | 6.669 |
| 24 | 14.622 | 2.975 | 7.862 |
| 30 | 14.042 | 2.892 | 6.67 |
| 36 | 12.746 | 2.58 | 5.508 |
| 40 | 12.418 | 2.545 | 5.31 |
| 60 | 11.932 | 2.393 | 4.57 |
| 72 | **11.903** | 2.263 | 4.239 |
| 80 | 11.928 | 2.3 | 4.129 |
| 96 | 11.961 | 2.329 | 4.018 |
| 120 | 11.987 | **2.218** | **3.72** |
| 144 | 12.165 | 2.757 | 3.97 |
| 160 | 13.426 | 3.229 | 4.352 |
| 180 | 12.554 | 3.186 | 3.79 |
| 240 | 13.202 | 3.843 | 4.19 |
| 360 | 15.363 | 4.198 | 4.412 |
| 480 | 17.493 | 4.476 | 4.81 |
| 720 | 18.8 | 6.991 | 10.826 |

На рисунке 2 представлен график зависимости времени счета от размерности блоков для трех вариантов вычислений.

Рисунок

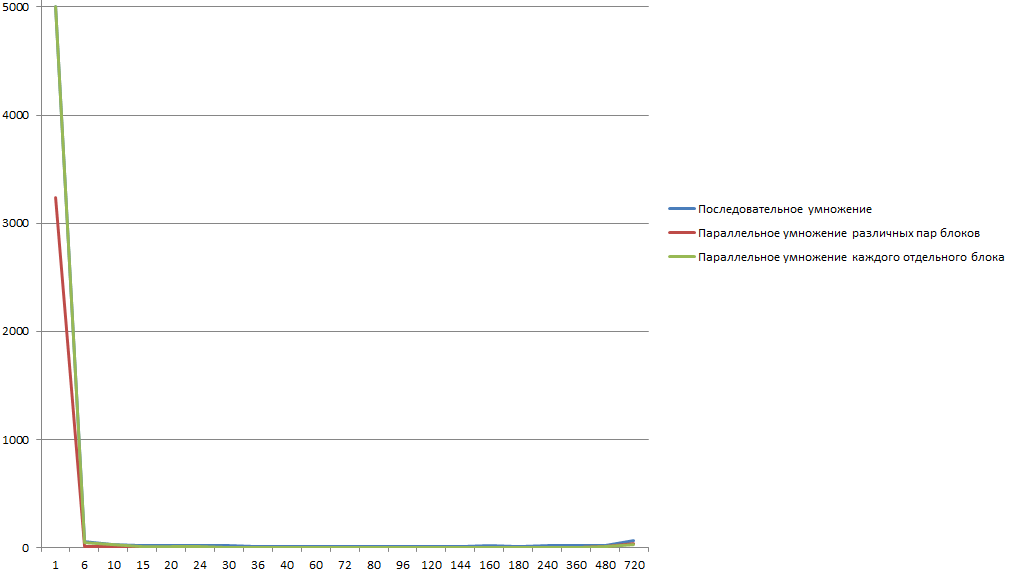
# Результаты для компилятора g++

Время выполнения трёх вариантов программных реализаций для типа данных double и компилятора GCC представлено в таблице 3.

Таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер блока** | **Последовательное умножение** | **Параллельное умножение различных пар блоков** | **Параллельное умножение каждого отдельного блока** |
| 1 | >5000 | 1319.21 | >5000 |
| 6 | 37.562 | 7.64 | 1970.32 |
| 10 | 21.95 | 4.132 | 442.496 |
| 15 | 16.687 | 3.095 | 148.925 |
| 20 | 13.418 | 2.693 | 59.511 |
| 24 | 12.124 | 2.535 | 36.027 |
| 30 | 11.034 | 2.396 | 21.065 |
| 36 | 10.793 | 2.449 | 12.238 |
| 40 | 10.258 | 2.4 | 9.725 |
| 60 | 9.149 | 2.177 | 4.434 |
| 72 | 8.982 | **2.101** | 3.627 |
| 80 | 9.228 | 2.139 | 3.344 |
| 96 | 9.696 | 2.69 | 3.33 |
| 120 | **8.848** | 2.572 | **2.734** |
| 144 | 9.369 | 2.813 | 3.246 |
| 160 | 14.038 | 3.55 | 4.21 |
| 180 | 8.904 | 2.564 | 3.191 |
| 240 | 10.989 | 3.479 | 3.57 |
| 360 | 11.462 | 4.597 | 3.991 |
| 480 | 15.212 | 6.808 | 14.615 |
| 720 | 61.9 | 36.408 | 28.549 |

На рисунке 3 представлен график зависимости времени счета от размерности блоков для трех вариантов вычислений.

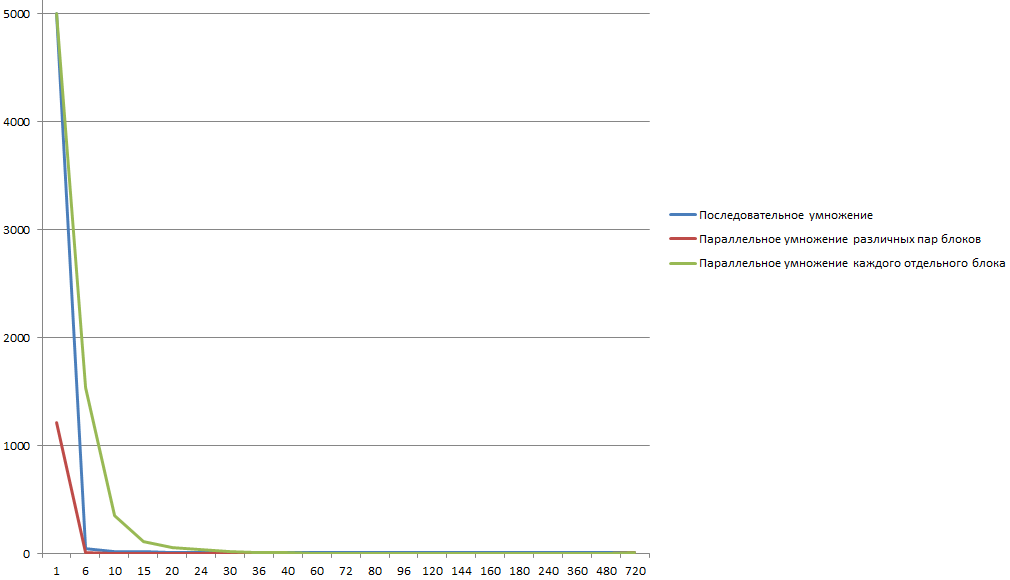
Рисунок

Время выполнения трёх вариантов программных реализаций для типа данных float и компилятора GCC представлено в таблице 4.

Таблица

|  |  |  |  |
| --- | --- | --- | --- |
| **Размер блока** | **Последовательное умножение** | **Параллельное умножение различных пар блоков** | **Параллельное умножение каждого отдельного блока** |
| 1 | >5000 | 1212.2 | >5000 |
| 6 | 46.552 | 7.46 | 1538.88 |
| 10 | 20.972 | 3.94 | 356.639 |
| 15 | 15.151 | 2.839 | 108.537 |
| 20 | 12.391 | 2.402 | 58.327 |
| 24 | 11.15 | 2.217 | 34.341 |
| 30 | 10.03 | 2.117 | 18.693 |
| 36 | 9.021 | 2.061 | 11.256 |
| 40 | 9.101 | 2.101 | 7.961 |
| 60 | 7.817 | 1.907 | 3.751 |
| 72 | 7.671 | 1.808 | 3.046 |
| 80 | 7.467 | **1.771** | 2.709 |
| 96 | 7.302 | 1.818 | 2.434 |
| 120 | **7.102** | 1.805 | **2.168** |
| 144 | 7.273 | 2.002 | 2.243 |
| 160 | 8.44 | 2.475 | 3.99 |
| 180 | 7.256 | 2.329 | 2.651 |
| 240 | 7.36 | 2.721 | 2.761 |
| 360 | 9.748 | 3.593 | 2.502 |
| 480 | 13.544 | 4.909 | 3.687 |
| 720 | 14.215 | 5.551 | 10.914 |

На рисунке 4 представлен график зависимости времени счета от размерности блоков для трех вариантов вычислений.

Рисунок

# Сравнительный анализ

Из полученных результатов можно заключить, что параллельное выполнение вычислений работает в 3-4 раза быстрее, чем последовательное, для обоих компиляторов, если учитывать оптимальные размеры блоков. Для малых же размеров блоков вариант с параллельным умножением для каждого отдельного блока менее эффективен, так как в кэше хранится мало элементов, и происходит частое обращение к памяти.

Также, для типа данных double размер для размера блока 720 параллельные вычисления работают значительно дольше, так как один блок не помещается в кэш, и происходит много обращений к памяти. Для типа float требуется в два раза меньше памяти, поэтому блоки помещаются в кэш даже при размере блока, равном 720.

Программы, скомпилированные компилятором GCC, в среднем немного быстрее, чем программы, скомпилированные MSVC.

Эффективнее всего оказалось параллельное умножение различных пар блоков. Программа, скомпилированная MSVC, для размера блока 72 выполнилась за 2,56 секунды для типа данных double, и для размера блока 120 за 2,218 секунды для типа данных float. Для вариантов программ, скомпилированных GCC, также быстрее всего выполнился вариант параллельного умножения различных блоков: для типа double за 2,101 секунды при размере блока 72, и для типа float за 1,771 секунды при размере блока 80.

Таким образом, наиболее оптимальным вариантом перемножения двух матриц является параллельное умножение различных пар блоков при размере блока 72-120.

# Заключение

В ходе проделанной работы были выполнены и проанализированы различные варианты программных реализаций блочного перемножения матриц. Для каждого из вариантов был определен оптимальный размер блока, позволяющий достичь наименьшего времени выполнения программы.

Эксперименты показали, что наилучшим вариантом реализации является параллельное умножение различных пар блоков с размером блока от 72 до 120.