**ОТЧЁТ**

по индивидуальному заданию

Студент 4 курса:

*Демяненко Сергей Алексеевич*

Преподаватель:

*Ассистент Баглий Антон Павлович*

**Постановка задачи:**

Необходимо реализовать непараллельное и параллельное умножение матриц. Опытным путём получить оптимальный размер блока матрицы. Сравнить время расчёта в обоих случаях.

Программа будет реализована в трех исполнениях:  
**Последовательная  
Паралельная "Блоки"  
Паралельная "Вычисл. Ядра"**

**Реализация программы:**

Реализация будет проводиться на языке программирования C++ версии v141, в среде разработки Visual Studio 17 версии 15.9, на операционной системе Windows 10 версии 1803. Будем дробить матрицы на блоки, что бы в дальнейшем можно было проще распараллелить, а также для облегчения проверки программы на верность (отладки). И так, в программе реализована система которая подбирает размер блока относительно величины матрицы, единственное, что нам необходимо указать, это минимальный размер блока, от которого он будет расширяться. После определения оптимального размера блока для данной матрицы, мы дробим матрицу на блоки. И приступаем к перемножению. В дальнейшем эти блоки помогут нас с добавлением OpenMP. В данной программе, так как тестировалась она на одном моём ПК, было подобранно оптимальное количество потоков, для обработки всех типов матриц, которые участвовали в тестах.

**Параметры ЭВМ:**

Тесты проводились на машине:  
**Процессор: Intel core i3 2200m 2500 МГц  
 Кеш 1-го уровня 32 Кб  
 Кеш 2-го уровня 256 Кб  
 Кеш 3-го уровня 3 Мб  
 2 Ядра 4 Потока  
ОЗУ: 4 Гигабайта 1800 МГц  
Устройство памяти: SSD накопитель 256 Гб**

Опытным путём вычислим оптимальный размер блоков:

Зададим матрицы размеров и количество потоков равное . Размеры блоков будем изменять в цикле по степеням двойки, начиная с и заканчивая размеров в элементов. Проведём сравнение трех вариантов реализации:

* Непараллельный алгоритм, в случае, когда матрица записана в массив построчно
* Параллельный “Блоки”
* Параллельный “Вычислит. ядра”

|  |  |  |  |
| --- | --- | --- | --- |
| Block's size | Parallel Block time | Parallel Assum kernel time | NonParallel time |
| 8 | 3,71759 | 4,09536 | 21,5754 |
| 16 | 2,31249 | 2,32696 | 6,72188 |
| 32 | 2,59754 | 2,19268 | 5,86812 |
| 64 | 1,86317 | 1,876915 | 4,22951 |
| 128 | 1,96276 | 1,95346 | 4,98838 |
| 256 | 2,34879 | 2,90107 | 5,37195 |
| 512 | 3,52338 | 5,47921 | 8,15631 |

Оптимальное время работы достигается при размере блока – 64x64.

Наглядно представлена зависимость времени работы от величины блока. Как можем заметить, выбранный нами блок 64 – оптимальный по времени выполнения программы.

**Хранение матрицы**

Матрица разбита на блоки, после чего вытягиваем все в одномерный массив. Далее, индекс элемента высчитывается в цикле при обращении к данному блоку.

Отметим, что хранение массива с нулевыми элементами неоптимально, т.к. тратится память программы впустую. **В нашей программе реализовано ОПТИМАЛЬОЕ ХРАНЕНИЕ матрицы без нулевых элементов.**

**Выводы:**

При тестировании работы программы в 3 режимах работы – в параллельном “блоками”, параллельном “на ядрах” и последовательном мы заметили, что параллельная программа работает быстрее во всех случаях для достаточно большого размера матриц. Параллельные варианты на размере блоков 64x64 работают практически одинаково быстро, но “блочный” вариант несколько опережает.

Реализация содержала такие виды исполнения:

**Последовательная  
Паралельная "Блоки"  
Паралельная "Вычисл. Ядра"**

**Методом тестировки выявили, что совмещение двух методов дало значительный прирост в скорости, на которой и были проведены исследования.**

Оптимальное количество потоков для данного ЭВМ – 4, т.к. процессор неэффективно обслуживает больше.