



Министерство науки и высшего образования Российской
Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
“Московский государственный технический университет имени
Н.Э. Баумана
(Национальный исследовательский университет)”
МГТУ им. Н.Э. Баумана

ФАКУЛЬТЕТ “Информатика и системы управления”

КАФЕДРА “Программное обеспечение ЭВМ и информационные технологии”

О т ч ё т

п о л а б о р а т о р н о й р а б о т е 4

Дисциплина: Анализ Алгоритмов

Тема лабораторной работы работы: Параллельные вычисления

Студент гр. ИУ7-536 _____ **Челядинов И.Д.**

Преподаватель _____ **Волкова Л.Л.**

Москва, 2020г

Введение	3
1. Аналитическая часть	5
1.1. Описание алгоритма	5
2. Конструкторская часть	7
2.1. Разработка алгоритмов	7
2.2. Распараллеливание программы	10
3. Технологическая часть	11
3.1. Требования к программному обеспечению	11
3.2. Технические характеристики	11
3.3. Листинг кода	11
4. Экспериментальная часть	16
4.1. Постановка эксперимента по замеру времени	16
4.2 Выводы	17
Заключение	18
Список литературы	19

Введение

Параллельные вычислительные системы — это физические компьютерные, а также программные системы, реализующие тем или иным способом параллельную обработку данных на многих вычислительных узлах.

Писать программы для параллельных систем сложнее, чем для последовательных, так как конкуренция за ресурсы представляет новый класс потенциальных ошибок в программном обеспечении (багов), среди которых состояние гонки является самой распространённой. Взаимодействие и синхронизация между процессами представляют большой барьер для получения высокой производительности параллельных систем. В последние годы также стали рассматривать вопрос о потреблении электроэнергии параллельными компьютерами. Характер увеличения скорости программы в результате распараллеливания объясняется законами Амдала и Густавсона. [1]

Поток выполнения - в вычислительной технике, наименьшая последовательность запрограммированных команд, которые могут независимо управляться планировщиком, являющимся частью операционной системы. Реализация потоков и процессов различается в операционных системах, но в большинстве случаев поток является составной частью процесса. Несколько потоков могут существовать в одном процессе, одновременно выполняться и совместно использовать ресурсы, такие как память, в то время как разные процессы не делят эти ресурсы. В частности, потоки процесса совместно используют свой исполняемый код и значения переменных в любой момент времени. [2]

Целью данной лабораторной работы является исследование многопоточности на примере алгоритма Винограда.

Задачи лабораторной работы:

- изучить работу многопоточных приложений;
- реализовать алгоритм Винограда с использованием многопоточности;

- провести замеры времени многопоточной и однопоточной реализации.

1. Аналитическая часть

Произведение матриц АВ состоит из всех возможных комбинаций скалярных произведений вектор-строк матрицы А и вектор-столбцов матрицы В. Операция умножения двух матриц выполнима только в том случае, если число столбцов в первой матрицы равно числу строк во второй.

1.1. Описание алгоритма

Алгоритм Винограда умножения матриц основан на снижении доли умножений в алгоритме. Предполагается, что некоторые произведения можно вычислить заранее, а затем переиспользовать при вычислении произведений матриц.

Рассмотрим два вектора $V = (v_1, v_2, v_3, v_4)$ и $W = (w_1, w_2, w_3, w_4)$.

Их скалярное произведение равно:

$$V * W = v_1w_1 + v_2w_2 + v_3w_3 + v_4w_4 \quad (1)$$

Это равенство можно переписать в виде:

$$V * W = (v_1 + w_2)(v_2 + w_1) + (v_3 + w_4)(v_4 + w_3) - v_1v_2 - v_3v_4 - w_1w_2 - w_3w_4. \quad (2)$$

Несмотря на то, что второе выражение требует вычисления большего количества операций, чем первое: вместо четырех умножений - шесть, а вместо трех сложений - десять, выражение в правой части последнего равенства допускает предварительную обработку: его части можно вычислить заранее и запомнить для каждой строки первой матрицы и для каждого столбца второй, что позволяет выполнять для каждого элемента лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

Алгоритм Винограда состоит из следующих шагов:

1. Вычисление горизонтальных произведений $MulH$

$$MulH_i = \sum_{k=0}^{N/2} A_{i,2k} * A_{i,2k+1} \quad (3)$$

2. Вычисление вертикальных произведений MulV

$$MulM_j = \sum_{k=0}^{N/2} B_{2k,j} * B_{2k+1,j} \quad (4)$$

3. Вычисление матрицы результата

$$c_{ij} = -MulH_i - MulV_j + \sum_{k=0}^{N/2} (A_{i,2k} + B_{2k+1,j}) * (A_{i,2k+1} + B_{2k,j}) \quad (5)$$

4. Корректирование матрицы в случае нечетного N

$$C_{i,j} = C_{i,j} + A_{i,N-1} * B_{N-1,j} \quad (6)$$

2. Конструкторская часть

Требования к вводу: На вход подаются две матрицы

Требования к программе:

- Корректное умножение двух матриц;
- При матрицах неправильного размера программа должна завершаться;
- Организовать перемножение матриц с использованием потоков.

2.1. Разработка алгоритмов

На рисунках 1-2 представлена схема алгоритма Винограда.

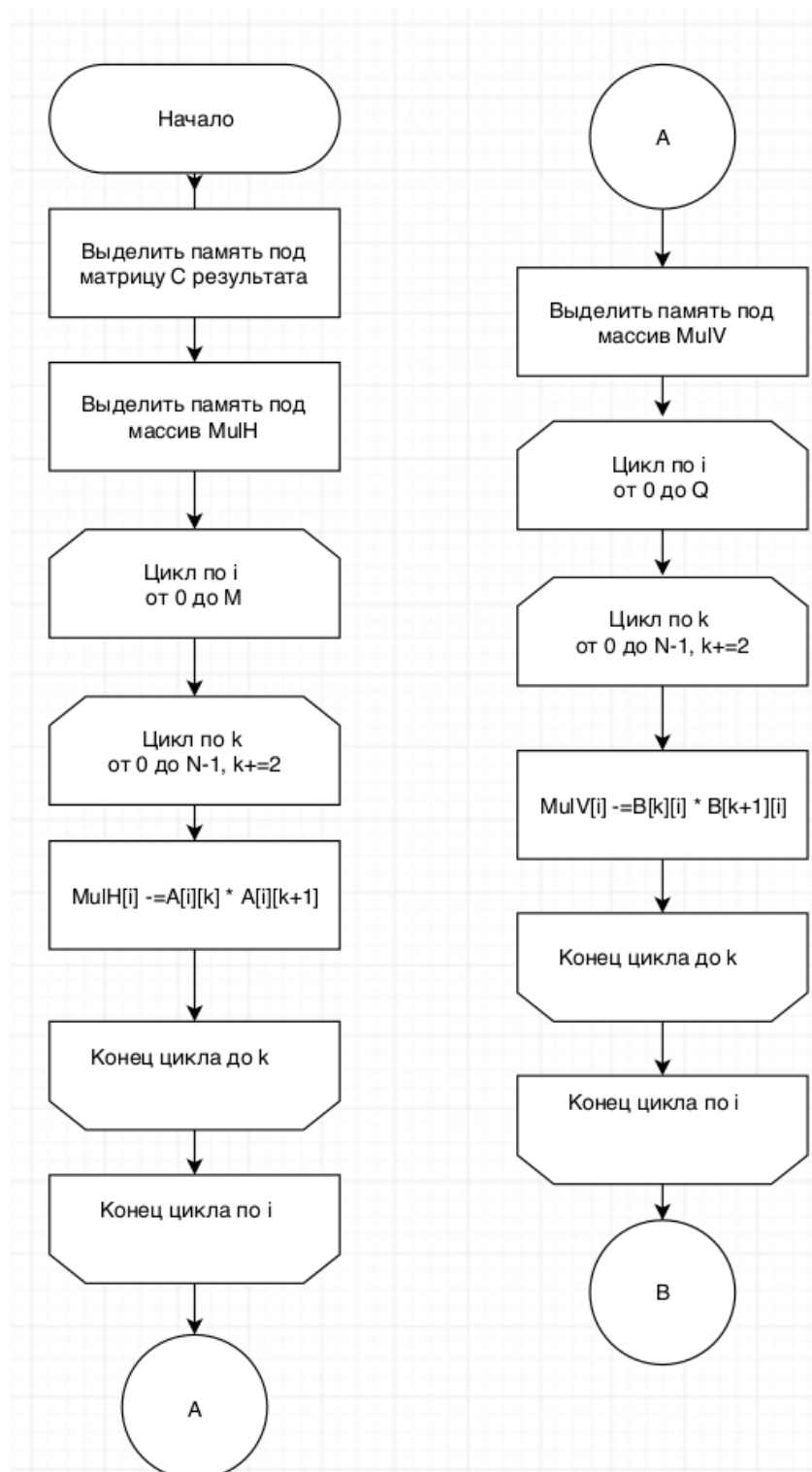


Рис 1. Схема алгоритма Винограда (часть 1)

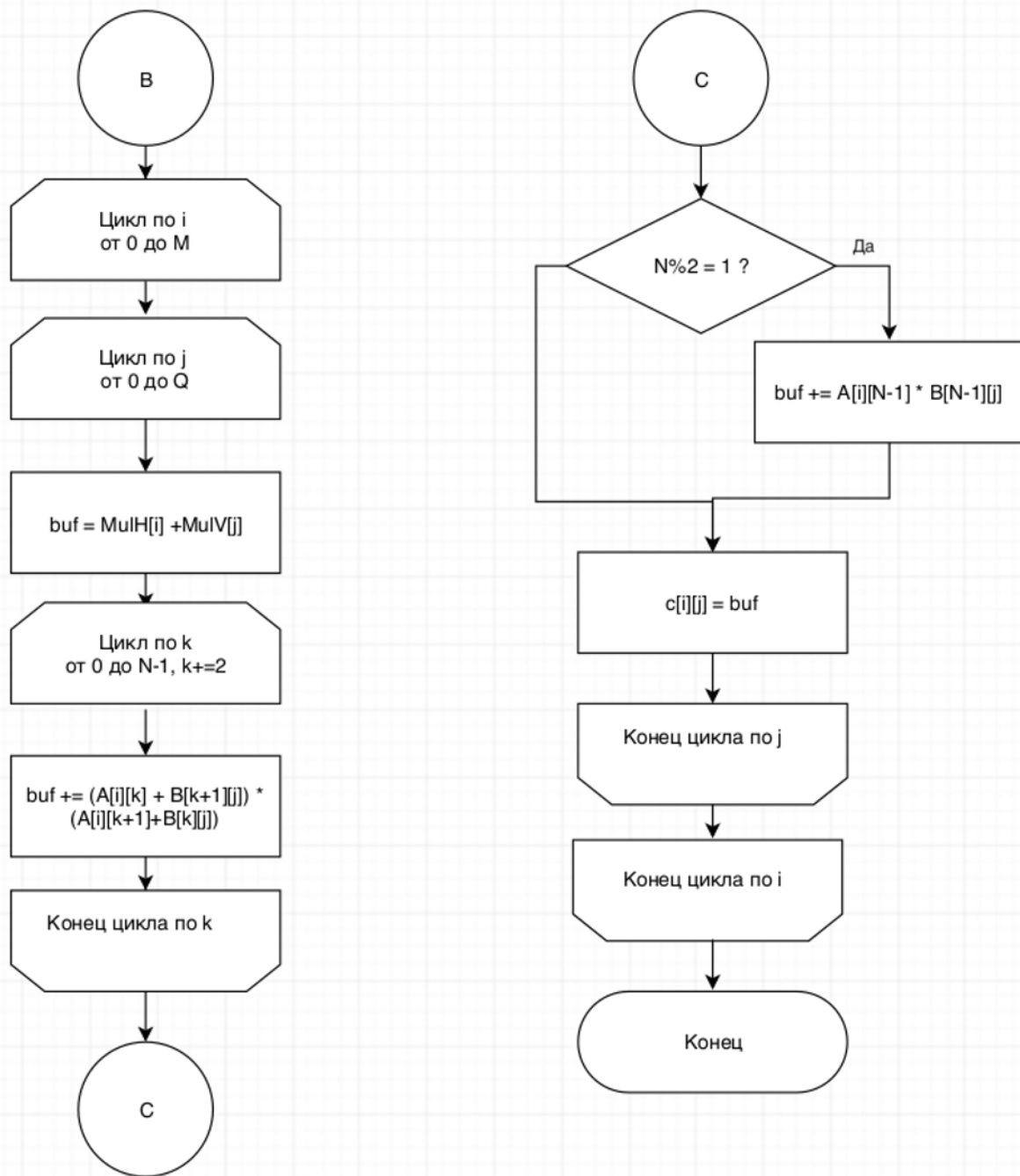


Рис 2. Схема алгоритма Винограда(часть 2)

2.2. Распараллеливание программы

Распараллеливание программы должно увеличить скорость работы.

Исходя из схемы, можно разделить программу на логические части, которые можно запускать в параллельных потоках. К примеру, массивы $MulH$ и $MulV$ можно вычислять параллельно. Также потоки могут считать разные элементы одного массива или одной матрицы, поэтому вычисление $MulH$ и $MulV$ можно разбить еще на несколько потоков, которые будут считать отдельные элементы. Тройной цикл вычисления результата можно распараллелить по строкам. Например, выделить вычисление с 1 по P строку в одном потоке, а с P по M в другом потоке.

В итоге при делении на P потоков:

- Цикл вычисления $MulH$ делим на $P/2$ потоков: каждый i -ый поток вычисляет элементы массива с $i * M/(p/2)$ по $(i+1) * M/(p/2)$. То есть каждый поток считает элементы с шагом $M/(p/2)$ до M .
- Цикл вычисления $MulV$ делим на $P-P/2$ потоков: каждый i -ый поток вычисляет элементы массива с $i * Q/(p - p/2)$ по $(i+1) * Q/(p - p/2)$. То есть каждый поток считает элементы с шагом $Q/(p-p/2)$ до Q .
- Цикл вычисления матрицы делим на P потоков: каждый i -ый поток вычисляет строки с $i * M/p$ до $(i+1) * M/p$.

3. Технологическая часть

Для реализации программы был выбран язык C++, так как этот язык позволяет работать с чистыми потоками, что удовлетворяет условиям лабораторной работы. В качестве среды разработки был выбран Clion.[3] Для работы с матрицами был использован класс `matrix`, реализованный в течение учебного процесса ранее. Замеры времени были проведены с помощью функции `clock()` из библиотеки `ctime`.

3.1. Требования к программному обеспечению

На вход программа получает две матрицы с размерами $M \times N$ и $N \times Q$. На выходе получается матрица размером $M \times Q$. Так же должна быть реализована функция для тестирования и функции замера времени.

3.2. Технические характеристики.

Замер времени производился на ноутбуке Apple с процессором Intel core i5, который обладает 1.4GHz тактовой частоты, четырехъядерным процессором, а также 8 гигабайт оперативной памяти.[4]

3.3. Листинг кода

На листингах 1-5 представлен код полученной программы.

Листинг 1. Реализация оптимизированного алгоритма Винограда

```
Matrix Vinograd(Matrix A, Matrix B) {  
    int N = A.cols();  
    int M = A.rows();  
    int Q = B.cols();  
    Matrix c(M, Q);  
  
    std::vector<int> MulH(M,0);
```

```

for (int i = 0; i < M; i++) {
    for (int k = 0; k < N-1; k += 2) {
        MulH[i] -= A[i][k] * A[i][k+1];
    }
}

std::vector<int> MulV(Q,0);
for (int i = 0; i < Q; i++) {
    for (int k = 0; k < N-1; k += 2) {
        MulV[i] -= B[k][i]*B[k+1][i];
    }
}

for (int i = 0; i < M; i++) {
    for (int j = 0; j < Q; j++) {
        int buf = MulH[i] + MulV[j];
        for (int k = 0; k < N-1; k += 2) {
            buf += (A[i][k] + B[k+1][j])*(A[i][k+1] + B[k][j]);
        }
        if (N%2 == 1) {
            buf += A[i][N-1]*B[N-1][j];
        }
        c[i][j] = buf;
    }
}

return c;

```

Листинг 2. Первый шаг алгоритма Винограда

```

void first (std::vector<int> &MulH, Matrix A, int N, int start, int end)
{

```

```

for (int i = start; i < end; i++) {
    for (int k = 0; k < N-1; k += 2) {
        MulH[i] -= A[i][k] * A[i][k+1];
    }
}
}

```

Листинг 3. Второй шаг алгоритма Винограда

```

void second (std::vector<int> &MulV, Matrix B, int N, int start, int end) {
    for (int i = start; i < end; i++) {
        for (int k = 0; k < N-1; k += 2) {
            MulV[i] -= B[k][i]*B[k+1][i];
        }
    }
}

```

Листинг 4. Третий шаг алгоритма Винограда

```

void third (Matrix &c, Matrix A, Matrix B, int N, std::vector<int> MulH,
std::vector<int> MulV, int starti, int endi, int startj, int endj){
    for (int i = starti; i < endi; i++) {
        for (int j = startj; j < endj; j++) {
            int buf = MulH[i] + MulV[j];
            for (int k = 0; k < N-1; k += 2) {
                buf += (A[i][k] + B[k+1][j])*(A[i][k+1] + B[k][j]);
            }
            if (N%2 == 1) {
                buf += A[i][N-1]*B[N-1][j];
            }
            c[i][j] = buf;
        }
    }
}

```

```
}
```

Листинг 5. Функция распараллеливания алгоритма Винограда

```
Matrix Mul(Matrix A, Matrix B, int tcount) {  
    int N = A.cols();  
    int M = A.rows();  
    int Q = B.cols();  
    Matrix c(M, Q);  
  
    std::vector<int> MulH(M,0);  
    std::vector<int> MulV(Q,0);  
  
    thread *th[tcount];  
    int n = tcount/2;  
    for (int i = 0; i < n; i++) {  
        th[i] = new thread(first, ref(MulH), A, N, i*M/n, (i+1)*M/n);  
    }  
    int m = tcount - n;  
    for (int i = 0; i < m; i++) {  
        th[n+i] = new thread(first, ref(MulV), B, N, i*M/m, (i+1)*M/m);  
    }  
  
    for (int i = 0; i < tcount; i++) {  
        th[i]->join();  
    }  
  
    thread *th3[tcount];  
    for (int i = 0; i < tcount; i++) {  
        th3[i] = new thread(third,  
                             ref(c),  
                             A, B, N,  
                             MulH, MulV,
```

```

        i*M/tcount, (i+1)*M/tcount,
        0, Q);
    }
    for (int i = 0; i < tcount; i++) {
        th3[i]->join();
    }
    for (int i = 0; i < tcount; i++) {
        delete th[i];
        delete th3[i];
    }
    return c;
}

```

4. Экспериментальная часть

В данной части будут представлены результаты тестирования программы, сравнение временных характеристик для перемножения матриц, используя разное количество потоков.

4.1. Постановка эксперимента по замеру времени

Были проведены временные эксперименты для матриц от 100×100 до 1000×1000 с шагом 100 и для матриц от 101×101 до 1001×1001 с шагом 100. Для каждого замера взят средний результат из 5 замеров. Результаты замеров представлены на рисунке 4.1.

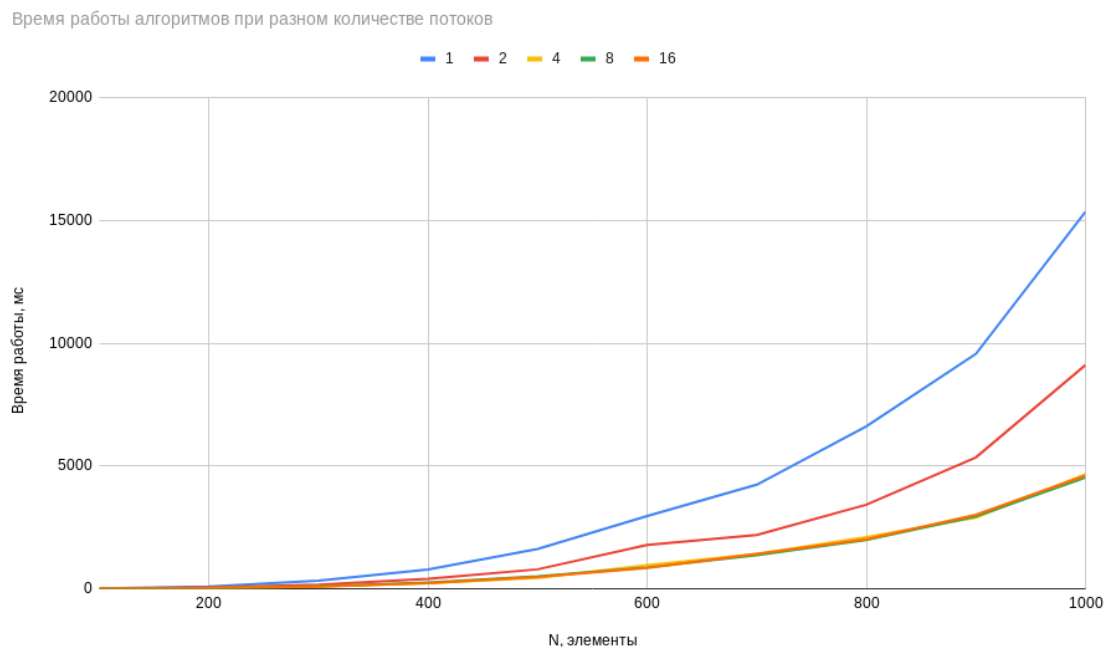


Рис 4.1. Диаграмма результатов замера времени(100-1000)

На рисунке 4.2 представлены результаты замеров для матриц с нечетным размером.

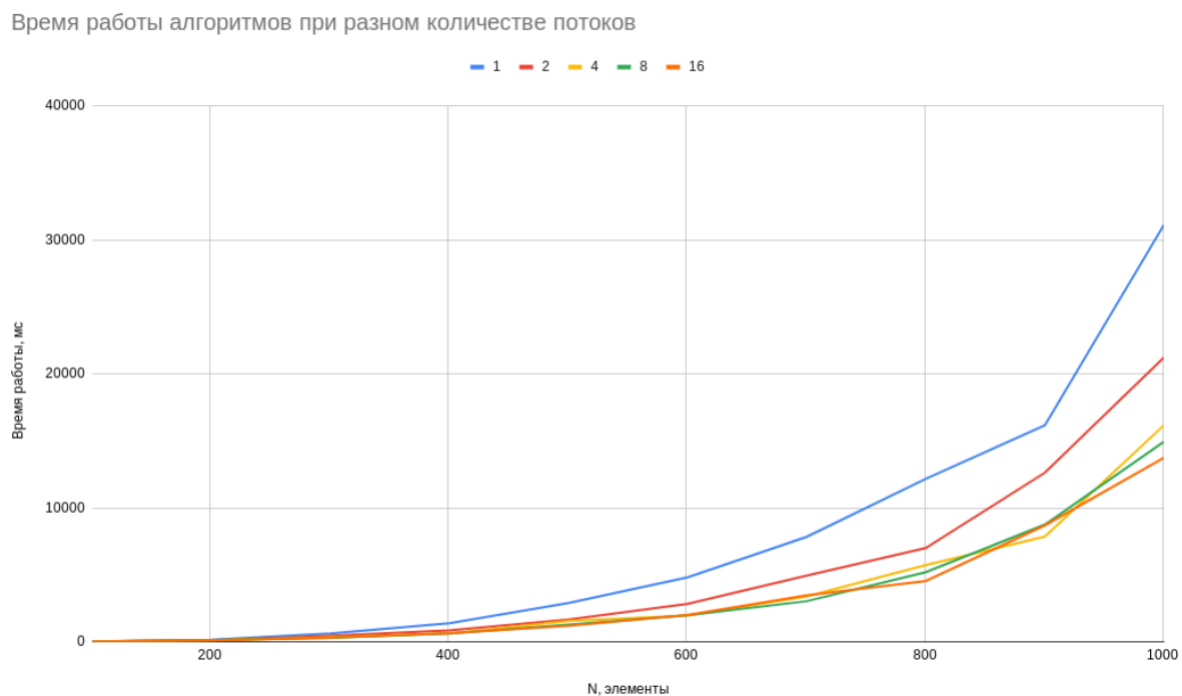


Рис 4.2. Диаграмма результатов замера времени(101-1001)

4.2 Выводы

По графикам видно, что многопоточные реализации работают гораздо быстрее однопоточной реализации. Реализация алгоритма на двух потоках работает почти в 2 раза быстрее, а реализация на 4 потоках — почти в 4 раза. Однако при дальнейшем разделении на потоки скорость работы не меняется. Из этого следует, что при количестве потоков, превышающем количество ядер процессора, увеличения эффективности работы алгоритма не наблюдается. Можно сделать вывод, что грамотное разделение функции на потоки увеличивает скорость работы алгоритма, а именно, максимальную эффективность дает количество потоков, равное числу логических процессоров.

Заключение

В ходе лабораторной работы были изучены возможности и способы распараллеливания трудоёмких программ, реализован алгоритм Винограда для нахождения произведения матриц с использованием многопоточности. Были проведены замеры времени работы алгоритма при различном количестве потоков. На основе этого эксперимента можно сказать, что максимальную эффективность даёт количество потоков, равное количеству логических процессоров.

Список литературы

1. Параллельные вычислительные системы. Электронный ресурс Wikipedia

URL: https://ru.wikipedia.org/wiki/параллельные_вычислительные_системы

2. Потоки выполнения. Электронный ресурс Национальная библиотека им. Н. Э. Баумана. Режим доступа:

https://ru.bmstu.wiki/index.php?title=%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F&mobileaction=toggle_view_mobile

3. Среда разработки Clion. Режим доступа:

https://www.jetbrains.com/clion/promo/?gclid=CjwKCAiAnvj9BRA4EiwAuUMDfyCKoJdi7mIAuEZwBNBIMV7-u4mZtXeYU_E08eFTKIWVeBg6javDQBoCKg4QAvD_BwE&gclsrc=aw.ds

4. Процессор Intel core i5 10gen. Режим доступа:

<https://www.intel.ru/content/www/ru/ru/products/docs/processors/core/10th-gen-processors.html>(дата обращения: 30.09.2020)