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»

Лабораторная работа №4

По предмету: «Анализ алгоритмов»

Тема: Параллельная реализация алгоритма Винограда

Студент: Гасанзаде М.А.,
Группа: ИУ7-56Б

Москва, 2019 г.

Оглавление

Введение

Параллельные вычисления используют для увеличения скорости исполнения программ. Ведь пока нет возможности сделать один очень быстрый процессор, который можно было бы сравнить с современными параллельными компьютерами.

Но приемы и алгоритмы, применяемые для однопоточных машин, для параллельных обычно не подходят.

Цель работы: изучение параллельных вычислений на материале алгоритма Винограда.

Задачи работы:

1) изучение основ параллельных вычислений;

2) применение метода динамического программирования для параллельной реализации алгоритма Винограда;

3) получение практических навыков параллельной реализации алгоритма Винограда;

4) сравнительный анализ параллельной и однопоточной реализаций алгоритма Винограда умножения матриц по затрачиваемым ресурсам;

5) экспериментальное подтверждение различий во временной эффективности реализаций алгоритма при помощи разработанного программного обеспечения на материале замеров процессорного времени выполнения реализаций на варьирующихся размерах матриц;

6) описание и обоснование полученных результатов в отчете о выполненной лабораторной работе, выполненного как расчётно-пояснительная записка к работе.

1. Аналитическая часть

В данной части дано теоретическое описание алгоритмов и указание области их применения.

1.1. Описание алгоритмов

Пусть A и B — матрицы, произведение которых C требуется найти: M * N и N * Q — их размер соответственно. Следующий псевдокод иллюстрирует стандартный алгоритм умножения матриц [1]:

Псевдокод 1. Стандартный алгоритм умножения матриц

```
for i = 1 to M do
    for j = 1 to Q do
        for k = 1 to N do
            C[i][j] = C[i][j] + A[i][k] * B[k][j]
        end for k
    end for j
end for i
```

Можно заметить, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнить заранее. Это реализовано в алгоритме Винограда [1].

Рассмотрим два вектора: $V = (v1, v2, v3, v4)$ и $W = (w1, w2, w3, w4)$. Их скалярное произведение равно: $v1w1 + v2w2 + v3w3 + v4w4$.

Это равенство можно переписать в виде: $(v1 + w2)(v2 + w1) + (v3 + w4)(v4 + w3) - v1v2 - v3v4 - w1w2 - w3w4$.

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. На практике это означает, что над предварительно обработанными элементами нам придется выполнять лишь первые два умножения и последующие пять сложений, а также дополнительно два сложения.

В случае нечетной длины перемножаемых векторов требуется провести расчёт по следующим правилам: c[i][j]=c[i][j] + a[i][n-1]*b[n-1][j]

Алгоритм Винограда можно оптимизировать. Если на каждой итерации увеличивать переменную цикла на два и объединить последние два цикла, то можно сэкономить ресурсы на их обслуживании. Следует также накапливать значение очередного элемента результирующей матрицы в буфер, что позволит выполнять меньше адресаций. А вычислив заранее значение N – 1, можно в дальнейшем использовать его без лишних вычислительных трат.

1.2. Применение алгоритмов

Алгоритмы активно применяются в [2, 3]:

1. расчётах трёхмерной графики;
2. математическом анализе при интегрировании систем дифференциальных уравнений, в теории вероятностей;
3. исследовании линейных отображений векторных пространств, линейных и квадратичных форм, систем линейных уравнений;
4. экономике для обработки балансово-нормативных моделей, отражающих соотношения затрат и результатов производства, нормативы затрат, производственные и экономические структуры.

2. Конструкторская часть

В данной части приведены схемы алгоритмов, а также их сравнительный анализ.

2.1 Разработка алгоритмов

Функциональная модель процесса умножения матриц в нотации IDEF0 представлена на рис. 1.

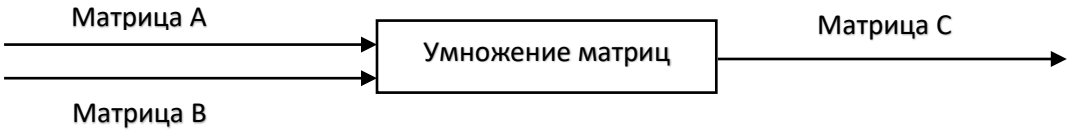

Рисунок 1 – Функциональная модель процесса умножения матриц

Схема алгоритма представлена на рис. 2, 3, 4.

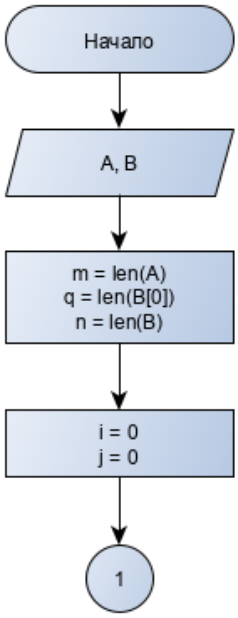

Рисунок 2 – Схема алгоритма Винограда с набором оптимизаций (начало)

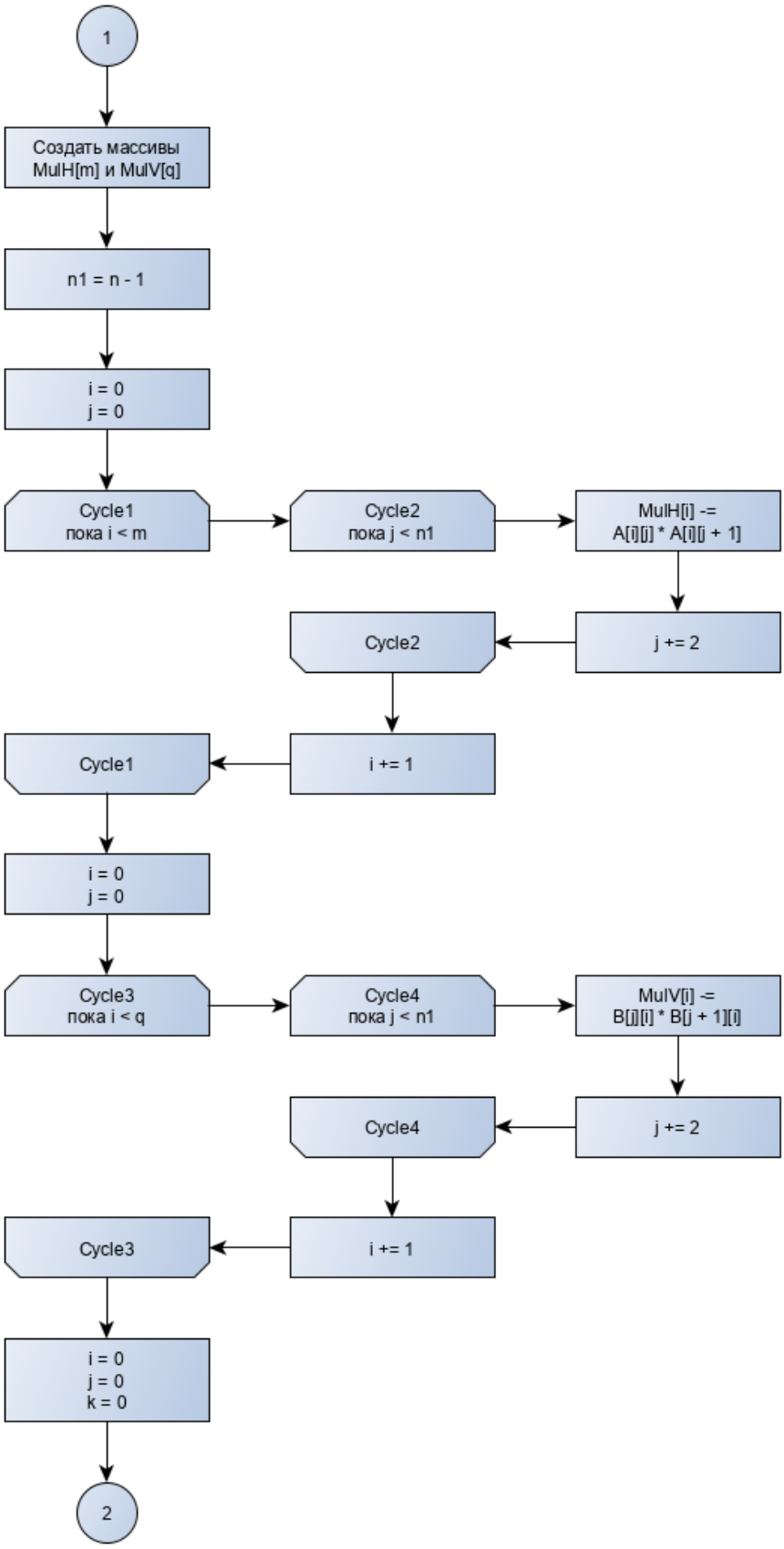

Рисунок 3 – Схема алгоритма Винограда с набором оптимизаций (продолжение)

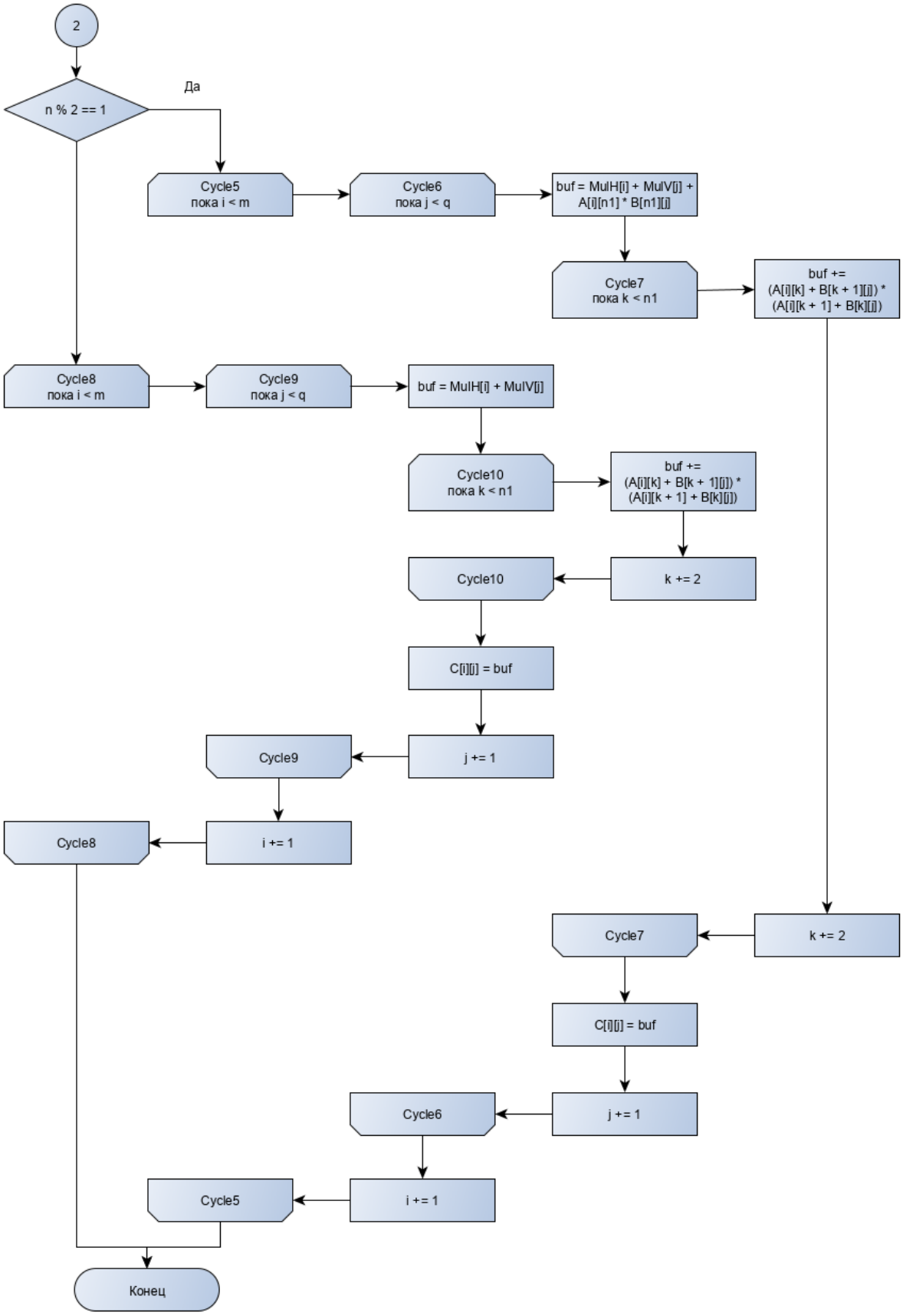

Рисунок 4 – Схема алгоритма Винограда с набором оптимизаций (конец)

2.2 Вывод

Алгоритм Винограда требует совсем немного дополнительной памяти для своей работы. При этом позволяет легко распараллелить свою работу, так как каждую итерацию каждого цикла может выполнять любой поток. При этом не будет происходить одновременной попытки доступа разных потоков к одному участку памяти.

3. Технологическая часть

В данной части приведены используемые технические средства, а также примеры тестирования и листинг программы.

3.1 Требования к программному обеспечению

Программа должна корректно считывать две матрицы, верно вычислять их произведение как в однопоточном, так и в многопоточном режиме (с любым количеством потоков). Требуется также обеспечить возможность замера времени работы алгоритма на различных размерах квадратных матриц.

3.2 Средства реализации

Выбран язык программирования C# за высокую скорость работы программ, хорошую работу в Windows и опыт использования в параллельных вычислениях [4]. Работа с потоками осуществляется посредством функции `CreateThreads`. Замер времени проводился функцией, представленной на Листинге 1.

Листинг 1 – Замер времени

```
public int Multiply(int thr_count) //time with threads

        {
            t1 = DateTime.Now;
            CreateThreads(thr_count);
            t2 = DateTime.Now;
            TimeSpan result = t2 - t1;
            return result.Minutes * 60 + result.Seconds * 1000 +
result.Milliseconds;
        }

//time without threads
        public int Multiply()

        {
            t1 = DateTime.Now;
            MultiplyMatr(new int[] { 0, _size });
            t2 = DateTime.Now;
            TimeSpan result = t2 - t1;
            return result.Minutes * 60 + result.Seconds * 1000 +
result.Milliseconds;
        }
```

3.3 Листинг кода

Исходный код реализации программы приведен в листингах 2, 3.

Листинг 2 – Умножение матриц

```
public static int[][] ParallelMultVin(Object obj)
        {
            int n = _matr1.GetLength(0),

                q = _matr2.GetLength(1),
                m = _matr2.GetLength(0);


            int[] mulH = new int[n];
            int[] mulV = new int[m];

            int[][] res = new int[n][];
            for (int i = 0; i < n; i++)
                res[i] = new int[m];

            Thread[] t = new Thread[nThreads];
        }
```

Листинг 3 – Функция создания и запуска потоков

```
private void CreateThreads(int count)
        {
            _threadArray = new Thread[count];
            for (int i = 0; i < count; ++i)
            {
                _threadArray[i] = new Thread(this.MultiplyMatr);
            }
            int s = _size / count;

            for (int i = 0; i < count - 1; ++i)
            {
                _threadArray[i].Start(new int[] { i * s, s });
            }
            _threadArray[count - 1].Start(new int[] { (count - 1) * s,
_size - (count - 1) * s });

            for (int i = 0; i < count; ++i)
            {
                _threadArray[i].Join();
            }
        }
```

3.4 Описание тестирования

Тестирование проводится по методу чёрного ящика [5]. Требуется проверить корректность работы на квадратных матрицах с четными и нечетными размерами сторон, на неквадратных матрицах, на умножении вектора на вектор, при однопоточной реализации, при многопоточной реализации с различным количеством потоков.

3.5 Вывод

Текущая реализация на языке C# позволяет корректно считывать матрицы, вычислять их произведение различным количеством потоков, а также производить замеры времени для определенного диапазона размеров входных матриц.

4. Экспериментальная часть

В этой части приведены примеры входных данных тестирования, результаты замера времени и их анализ.

4.1 Результаты тестирования

В Таблице 1 представлены результаты в следующем порядке: традиционная реализация, многопоточная реализация с одним потоком, многопоточная реализация с 10 потоками.

Таблица 1.

Результаты тестирования по методу черного ящика

Размер матр. A	Размер матр. B	Результат
1 * 6	6 * 8	Ответ верный
4 * 5	5 * 1	Ответ верный
4 * 4	4 * 4	Ответ верный
5 * 5	5 * 5	Ответ верный
3 * 4	4 * 2	Ответ верный

Все тесты пройдены успешно.

4.2 Постановка эксперимента по замеру времени

*З*амер времени проводился для двух квадратных матриц одинаковых размеров. Размер стороны матрицы составлял от 100 до 1000 с шагом 100 и от 101 до 1001 с шагом 100. Один эксперимент повторялся не менее 5 раз, результат одного эксперимента рассчитывался как среднее значение результатов проведенных испытаний с одинаковыми входными данными. Замерялись традиционная реализация и многопоточная реализация с 1, 2, 4, 10 потоками.

4.3 Сравнительный анализ на материале экспериментальных данных

На рис. 5, 6 представлены результаты замеров времени.

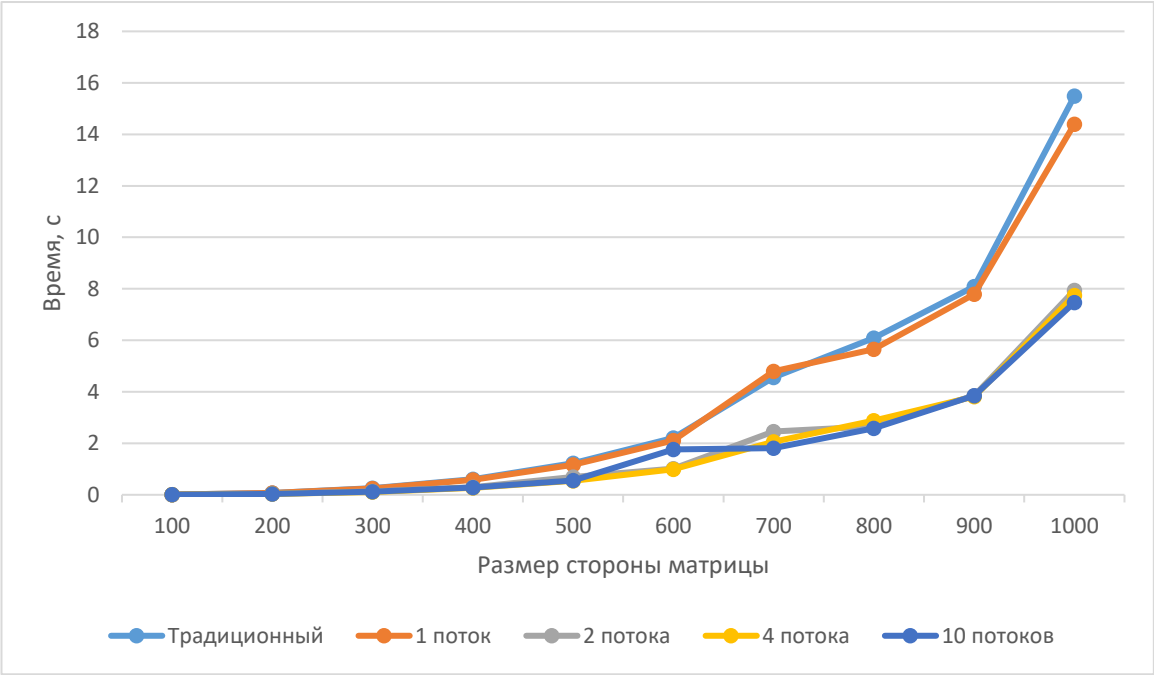

Рисунок 5 – Время работы реализаций алгоритма на четных размерах матриц

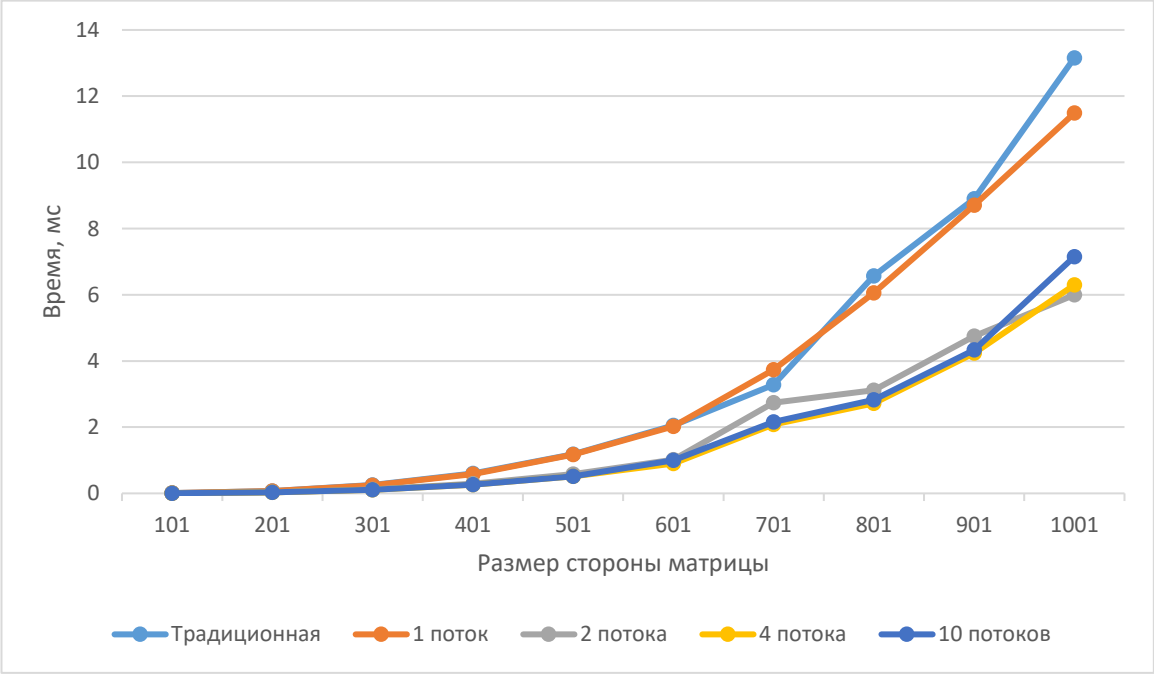

Рисунок 6 – Время работы реализаций алгоритма на нечетных размерах матриц

4.4 Вывод

По результатам эксперимента подтверждено, что время исполнения реализаций с 2, 4, 10 потоками почти совпадает и составляет половину от времени исполнения однопоточной реализации и многопоточной реализации с одним потоком. Использование нескольких потоков сильно ускоряет работу алгоритма, но усложняет отладку. Вычисление с использованием десяти потоков проигрывает вычислению с использованием четырёх, из-за в связи с архитектурой [6] потоки дольше стоят в очереди, ожидая своего времени, чем выполняют полезную работу.

Заключение

В рамках данной работы успешно изучены основы параллельных вычислений. Применен метод динамического программирования для параллельной реализации алгоритма Винограда. Проведен сравнительный анализ параллельной и однопоточной реализаций алгоритма Винограда умножения матриц. Подтверждены экспериментально различия во временной эффективности реализаций алгоритма Винограда умножения матриц при помощи разработанного программного обеспечения на материале замеров времени выполнения на варьирующихся размерах матриц. Дано описание и обоснование полученных результатов.

Экспериментально получено, что 2 потока работают почти в 2 раза быстрее, чем 1 поток. При этом работа 4 потоков лишь немного быстрее 2 потоков, так как в используемом процессоре 2 ядра и 4 потока [6] это является физическим пределом потоков. Работа на более чем 4 потоках не приносит выигрыша во времени, и даже местами проигрывает так как является бессмысленной при данном процессоре.

Список литературы

1. Умножение матриц // AlgoLib URL: http://www.algolib.narod.ru/Math/Matrix.html (дата обращения: 08.10.2019).
2. Матричная алгебра в жизни человека // База знаний "Allbest" URL: https://otherreferats.allbest.ru/mathematics/00489400_0.html (дата обращения: 11.10.2019).
3. C# Tutorials for parallel programming: https://www.tutorialspoint.com/csharp/csharp_multithreading.htm
4. Praktikum ”Parallele Programmierung” URL: https://wr.informatik.uni-hamburg.de/teaching/sommersemester_2014/parallele_programmierung
5. Kara kutu test technologisi URL:https://www.mobilhanem.com/kara-kutu-test-teknigi-ve-uygulanmasi/ (дата обращения 1.11.2019)
6. Характеристики процессора Intel Core i3 4005U URL: https://ark.intel.com/content/www/us/en/ark/products/75105/intel-core-i3-4005u-processor-3m-cache-1-70-ghz.html