Министерство образования Российской Федерации
Московский Государственный Технический Университет
им. Н.Э. Баумана

Отчёт по лабораторной работе № 1
По курсу: "Анализ алгоритмов"
Тема:"Умножение матриц. Многопоточное умножение матриц"

Студент: Орехова Е.О. ИУ7-51

Преподаватель: Волкова Л.Л.

4 марта 2018 г.

Содержание

1 Постановка задачи

В ходе выполнения лабораторной работы необходимо реализовать умножение матриц n потоками. Сравнить с однопоточной реализацией.

2 Умножение матриц

Пусть даны две матрицы, A и B, размерности $a \times n$ и $n \times b$ соответственно, тогда результатом их умножения будет матрица C, размерности $a \times b$, в которой

$$C_{i,j} = \sum_{k=1}^{n} A_{i,k} * Bk, j \tag{1}$$

3 Алгоритм Винограда

Если посмотреть на результат умножения двух матриц, то видно, что каждый элемент в нем представляет собой скалярное произведение соответствующих строки и столбца исходных матриц. Можно заметить также, что такое умножение допускает предварительную обработку, позволяющую часть работы выполнить заранее. 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

$$V * W = v_1 w_1 + v_2 w_2 + v_3 w_3 + v_4 w_4 \tag{2}$$

Это равенство можно переписать в виде:

$$V*W = (v_1+w_2)(v_2+w_1)+(v_3+w_4)(v_4+w_3)-v_1v_2-v_3v_4-w_1w_2-w_3w_4 \tag{3}$$

Из этого следует, что произведение матриц можно выполнить эффективнее, произведя некоторые вычисления заранее.

4 Реализация однопоточного умножения матриц

Листинг 1: Стандартный алгоритм умножения матриц

```
public static void Simple_Multiplication(ref int[,]C,
   int[,] A, int[,] B, int size)
{
        for (int i = 0; i < size; i++)
```

```
                for (int j = 0; j < size; j++)
                {
                        C[i, j] = 0;
                        for (int k = 0; k < size; k++)
                                C[i, j] = C[i, j] +
                                   A[i,k] * B[k,j];
                }
}
```

Листинг 2: Алгоритм Винограда

```
public static void Vinograd(ref int[,] C,int[,] A,
   int[,] B, int size, int [] Rows, int [] Column)
{

        for (int i = 0; i < size; i++)
        {
                Rows[i] = A[i, 0] * A[i, 1];
                for (int j = 1; j < size/2; j++)
                        Rows[i] = Rows[i]+ A[i, 2 * j] *
                           A[i, 2 * j + 1];
        }


        for (int i = 0; i < size; i++)
        {
                Column[i] = B[0, i] * B[1, i];
                for (int j = 1; j < size/2; j++)
                        Column[i] = Column[i]+ B[2 * j,
                           i] * B[2 * j + 1, i];
        }


        for(int i = 0; i< size; i++)
                for (int j = 0; j<size;j++)
                {
                        C[i, j] = -Rows[i] - Column[j];
                        for (int k = 0; k < size/2; k++)
                                C[i, j] = C[i,
                                   j]+(A[i,2*k+1]+B[2*k,j])
                                   *
                                   (A[i,2*k]+B[2*k+1,j]);
                }
        if (size % 2 == 1)
```

```
        {
                for (int i = 0; i < size; i++)
                        for (int j = 0; j < size; j++)
                                C[i, j] = C[i, j]+
                                   A[i,size-1] *
                                   B[size-1,j];
        }
}
```

5 Реализация многопоточного умножения матриц

Листинг 3: Класс для описания многопоточного умножения

```
class Simple_Multiply
{
private int begin;
private int end;
private int c;
private int b;

public Simple_Multiply(int b, int c, int begin, int end)
{
        this.begin = begin;
        this.end = end;
        this.c = c;
        this.b = b;
}

public void Simple_Multiply_Thread()
{
        for (int i = begin; i < end; i++)
                for (int j = 0; j < c; j++)
                {
                        Program.D[i, j] = 0;
                        for (int k = 0; k < b; k++)
                                Program.D[i, j] +=
                                   Program.A[i, k]
                                   *Program.B[k, j];
                }
}
```

```
public void Vinograd_Multiply_Thread()
{
        for (int i = begin; i < end; i++)
                for (int j = 0; j < c; j++)
                {
                        Program.D[i, j] =
                           -Multiplication.Rows[i] -
                           Multiplication.Column[j];
                        for (int k = 0; k < b / 2; k++)
                                Program.D[i, j] +=
                                   (Program.A[i, 2 * k +
                                   1] + Program.B[2 * k,
                                   j]) * (Program.A[i, 2
                                   * k] + Program.B[2 *
                                   k + 1, j]);
                }
                if (b % 2 == 1)
                {
                for (int i = begin; i < end; i++)
                        for (int j = 0; j < c; j++)
                                Program.D[i, j] +=
                                   Program.A[i,b - 1] *
                                   Program.B[b - 1, j];
                }
}
}
```

6 Эксперимент

В проводимом эксперименте задача разбивалась на 2 потока. Умножение матриц двумя потоками происходит быстрее в независимости от количества элементов.

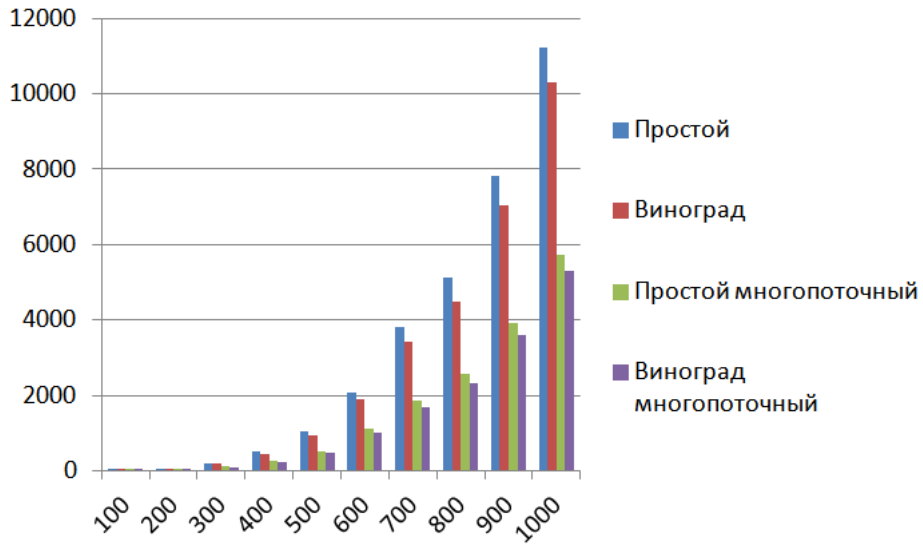

Рис. 1: Время умножения матриц в мс.

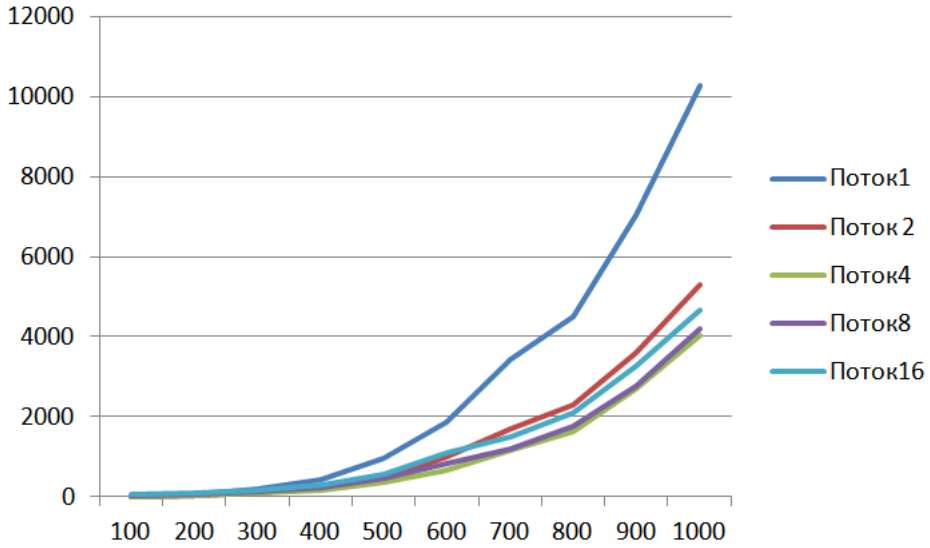

Рис. 2: Время умножения матриц в мс.

Оптимальное число потоков: 4-8.

7 Заключение

В ходе выполнения лабораторной работы были изучены и реализованы различные алгоритмы умножения матриц. Реализовано умножение матриц средствами n потоков. Экспериментально подтверждено, что большое число потоков не всегда работает быстрее.