Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

Московский государственный технический университет имени Н.Э.Баумана (МГТУ им. Н.Э.Баумана)

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**

**«Распараллеливание алгоритма вычисления произведения двух матриц»**

Выполнила: Митрошкин А.А.

(Фамилия И.О. студента)

ИУ9-51Б

(Индекс группы)

Преподаватель: Царев.А. С.

(Фамилия И.О. преподавателя)

(Подпись)

**Оглавление**

[**1.Цель**](#_heading=h.gjdgxs) **работы**

[**2.Условие**](#_heading=h.30j0zll) **задачи**3

[**2.1. Листинг кода программы 4**](#_heading=h.1fob9te)

# Задача

Сравнить время работы вычисления матрицы на одном потоке и нескольких.

# 2. Условие задачи

Две квадратные матрицы A и B размерности n сначала перемножить стандартным алгоритмом для получения матрицы С той же размености. Замерить время вычисления, сравнить с временем при вычислении элементов матрицы С не по строкам, а по столбцам. Размер матриц подобрать таким образом, чтобы время выполнения на вашей машине было не слишком непоказательно малым (меньше нескольких минут), но и не чересчур большим (несколько часов). Использовать библиотечные функции для вычисления произведений матриц нельзя. Затем конечную матрицу С условно разделить на примерно равные прямоугольные подматрицы и распараллелить программу таким образом, чтобы каждый поток занимался вычислением своей подматрицы. Матрицы А и В для этого разделить на примерно равные группы строк и столбцов соответственно. Сделать для разного количества потоков (разных разбиений), также замерить время вычисления, сравнить с вычислениями стандартным алгоритмом. Также по окончании вычислений сравнивать получившуюся матрицу с той, что была вычислена стандартным алгоритмом, для проверки правильности вычислений (проверка во время выполнения задачи не входит). Уделить внимание проблеме синхронизации потоков: основной поток, который создает потоки, занимающиеся вычислением подматриц, не должен закончиться до того, как они закончат свою работу, поэтому организовать ожидание и сигнализацию завершения рабочих потоков.

1. **Листинг кода программы**
2. using System;
3. using System.Diagnostics;
4. using System.Threading.Tasks;
5. class MatrixMultiplication
6. {
7. static int[,] MultiplyMatrices(int[,] matrixA, int[,] matrixB)
8. {
9. int n = matrixA.GetLength(0); // Количество строк в матрице A
10. int m = matrixB.GetLength(1); // Количество столбцов в матрице B
11. int[,] result = new int[n, m]; // Создание результирующей матрицы C размерности n x m
12. Parallel.For(0, n, i =>
13. {
14. for (int j = 0; j < m; j++)
15. {
16. for (int k = 0; k < matrixA.GetLength(1); k++)
17. {
18. result[i, j] += matrixA[i, k] \* matrixB[k, j];
19. }
20. }
21. });
22. return result;
23. }
24. static int[,] Fill\_Matrix(int[,] matrixA)
25. {
26. int n = matrixA.GetLength(0); // Количество строк в матрице A
27. int m = matrixA.GetLength(1); // Количество строк в матрице A
28. int[,] result = new int[n, m];
29. for (int i = 0; i < n; i++)
30. {
31. for (int j = 0; j < m; j++)
32. {
33. for (int k = 0; k < matrixA.GetLength(1); k++)
34. {
35. result[i, j] += new Random().Next(0, 10);
36. }
37. }
38. }
39. return result;
40. }
41. static void Print(int[,] Matrix)
42. {
43. int n = Matrix.GetLength(0); // Количество строк в матрице A
44. int m = Matrix.GetLength(1); // Количество строк в матрице A
45. for (int i = 0; i < n; i++)
46. {
47. for (int j = 0; j < m; j++)
48. {
49. Console.Write(Matrix[i, j].ToString() + " ");
50. }
51. Console.WriteLine();
52. }
53. }
54. static int[,] MultiplyMatrices\_non\_parallel(int[,] matrixA, int[,] matrixB)
55. {
56. int n = matrixA.GetLength(0); // Количество строк в матрице A
57. int m = matrixA.GetLength(1); // Количество столбцов в матрице A (и строк в матрице B)
58. int[,] result = new int[n, m]; // Результирующая матрица C размерности n x p
59. if (m != matrixB.GetLength(0))
60. {
61. throw new ArgumentException("Несовместимые размерности матриц для умножения.");
62. }
63. for (int i = 0; i < n; i++)
64. {
65. for (int j = 0; j < m; j++)
66. {
67. int sum = 0;
68. for (int k = 0; k < m; k++)
69. {
70. sum += matrixA[i, k] \* matrixB[k, j];
71. }
72. result[i, j] = sum;
73. }
74. }
75. return result;
76. }
77. static void Main()
78. {
79. int n = 500; // Количество строк в матрице A
80. int m = 500; // Количество столбцов в матрице B
81. int[,] matrixA = new int[n, m]; // Создание матрицы A размерности n x m
82. int[,] matrixB = new int[m, n]; // Создание матрицы B размерности m x n
83. matrixA = Fill\_Matrix(matrixA);
84. matrixB = Fill\_Matrix(matrixB);
85. // Заполнение матриц A и B данными (ваш код)
86. Console.WriteLine("start");
87. Stopwatch stopwatch = new Stopwatch();
88. stopwatch.Start();
89. int[,] result = MultiplyMatrices(matrixA, matrixB);
90. stopwatch.Stop();
91. Console.WriteLine($"Время выполнения параллельного алгоритма: {stopwatch.ElapsedMilliseconds} мс");
92. stopwatch.Start();
93. int[,] result\_non\_parallel = MultiplyMatrices\_non\_parallel(matrixA, matrixB);
94. stopwatch.Stop();
95. Console.WriteLine($"Время выполнения алгоритма: {stopwatch.ElapsedMilliseconds} мс");
97. }
98. }

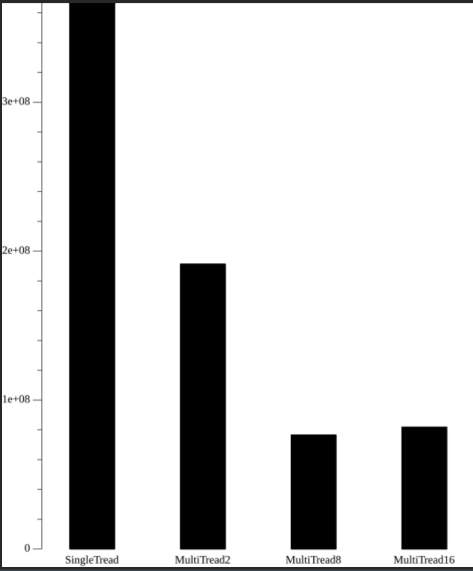
**Характеристики компьютера**

**12th Gen Intel(R) Core(TM) i5-12450H 2.50 GHz**

**16,0 ГБ (доступно: 15,7 ГБ)**

**64-разрядная операционная система, процессор x64**

**4. Результаты**

****

Можно заметить, что результат улучшается с увеличением потоков, но остается примерно равным при размере от 8. Именно столько физических ядер есть на компьютере, которые позволяют считать матрицу параллельно