МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ГОУ НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ
ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА "Информатика и вычислительная техника"

Дисциплина: "Технология распределенной обработки данных"

Отчёт

по лабораторной работе № 3

Тема: "Распараллеливание алгоритма с помощью библиотеки CCR."

Вариант №9

Проверил:
Гай В. Е.
Выполнил:
Студент гр. 14-В-2
Балыкин Д. Е.

Нижний Новгород 2016

1. Цель и порядок выполнения работы

Цель работы: получить представления о возможности библиотеки Concurrent and Coordination Runtime для организации параллельных вычислений.

Порядок выполнения работы:

1. Разработка последовательного алгоритма, решающего одну из приведённых задач в соответствии с выданным вариантом задания;
2. Разработка параллельного алгоритма, соответствующий варианту последовательного алгоритма;
3. Выполнение сравнения времени выполнения последовательного и параллельного алгоритмов обработки данных при различных размерностях исходных данных.

2. Теоретические сведения

2.1. Библиотека Concurrent and Coordination Runtime

Разработать алгоритм умножения матрици a (m * n) элементов на вектор b(n элементов) по следующей формуле:

Библиотека Concurrent and Coordination Runtime (CCR) предназначена для организации обработки данных с помощью параллельно и асинхронно выполняющихся методов. Взаимодействие между такими методами организуется на основе сообщений. Рассылка сообщений основана на использовании портов. Основные понятия CCR:

1. Сообщение – экземпляр любого типа данных;
2. Порт – очередь сообщений типа FIFO (First-In-First-Out), сообщение остаётся в порте пока не будут извлечено из очереди порта получателем. Определение порта:

   Port<int> p = new Port<int>();

   Отправка сообщения в порт:

   p.Post(1);
3. получатель – структура, которая выполняет обработку сообщений. Данная структура объединяет:
   - один или несколько портов, в которые отправляются сообщения;
   - метод (или методы), которые используются для обработки сообщений (такой метод называется задачей);
   - логическое условие, определяющее ситуации, в которых активизируется тот или иной получатель.

     Делегат, входящий в получатель, выполнится, когда в порт intPort придёт сообщение. Получатели сообщений бывают двух типов: временные и постоянные (в примере получатель – временный). Временный получатель, обработав сообщение (или несколько сообщений), удаляется из списка получателей сообщений данного порта.
4. процессом запуска задач управляет диспетчер. После выполнения условий активации задачи (одним из условий активации может быть получение портом сообщения) диспетчер назначает задаче поток из пула потоков, в котором она будет выполняться. Описание диспетчера с двумя потоками в пуле:

   Dispatcher d = new Dispatcher(2, "MyPool");

Описание очереди диспетчера, в которую задачи ставятся на выполнение:

DispatcherQueue dq = new DispatcherQueue("MyQueue d);

2.1.1. Создание проекта

Нужно выполнить следующие действия:

1. Установить библиотеку CCR (CCR входит в состав Microsoft Robotics Developer Studio);
2. Создать проект консольного приложения и добавьте к проекту библиотеку Sl.Microsoft.Ccr.Core.dll.

2.1.2. Оценка времени выполнения

Время выполнения вычислений будем определять с помощью класса

```
Stopwatch:
Stopwatch sWatch = new Stopwatch();
sWatch.Start();
<выполняемый код>
sWatch.Stop();
Console.WriteLine(sWatch.ElapsedMilliseconds.ToString());
```

3. Выполнение лабораторной работы

3.1. Задание

Разработать алгоритм умножения матрицы a (m * n элементов) на вектор b (n 'элементов) по следующей формуле:

$$\sum_{j=1}^{n} a_{ij} b_j, 1 \leq i \leq m \tag{3.1}$$

3.2. Листинг программы

```
1 using System;
2 using System.Threading;
3
4 using Microsoft.Ccr.Core;
5
6 namespace lab_003
7 {
8     public class InputData {
9         public int start; // начало диапазона
10         public int stop;  // конец диапазона
11         //public int i;
12     }
13
14     class lab_003 {
15         static int[,] A; //хранение матрицы
16         static int[] B;  //хранение вектор-столбца для умножения
17         static int[] C;  //хранение результата
18         static int m;    //количество строк матрицы
19         static int n;    //количество столбцов матрицы
20         static int nc;   //количество ядер
21
22         static void Test() {
23             nc = 2;
24             ConsoleKey press;
25
26             Console.WriteLine("Желаете задать размер
```

```
 матрицы самостоятельно? (Y/N)\n");
27
28              press = Console.ReadKey(true).Key;
29              if (press != ConsoleKey.N & press != ConsoleKey.Y) {
30                  Console.WriteLine("Некорректная клавиша");
31                  press = ConsoleKey.N;
32              }
33
34              if (press == ConsoleKey.Y) {
35                  Console.Write("Введите количество строк матрицы
: ");
36                  m = Convert.ToInt32(Console.ReadLine());
37                  Console.Write("Введите количество столбцов матр
ицы: ");
38                  n = Convert.ToInt32(Console.ReadLine());
39              } else if (press == ConsoleKey.N) {
40                  m = 10000;
41                  n = 10000;
42                  Console.WriteLine("\nРазмеры матрицы заданы автома
тически и составляют {0} x {1}\n", m, n);
43              }
44
45              A = new int[m, n];
46              B = new int[n];
47              C = new int[m];
48
49              Console.WriteLine("\n\nЖелаете заполнить матрицу и вектор-столбец
 самостоятельно? (Y/N)\n");
50              press = Console.ReadKey(true).Key;
51
52              if (press != ConsoleKey.N & press != ConsoleKey.Y) {
53                  Console.WriteLine("Некорректная клавиша");
54                  press = ConsoleKey.N;
55              }
56
57              if (press == ConsoleKey.Y) {
58                  Console.WriteLine("Ввод матрицы\n");
59                  for (int i = 0; i < m; i++) {
60                      for (int j = 0; j < n; j++) {
61                          Console.Write("A[{0},{1}] = ", i + 1, j + 1);
62                          A[i, j] = Convert.ToInt32(Console.ReadLine());
63                      }
```

```
64
65                  }
66                  Console.WriteLine("\n\nВвод вектор-столбца\n");
67                  for (int j = 0; j < n; j++) {
68                      Console.Write("B[{0}] = ", j + 1);
69                      B[j] = Convert.ToInt32(Console.ReadLine());
70                  }
71              } else if (press == ConsoleKey.N) {
72                  Console.WriteLine("Заполнение случайными значениями...\n");
73                  Random r = new Random();
74                  for (int i = 0; i < m; i++) {
75                      for (int j = 0; j < n; j++) {
76                          A[i, j] = r.Next(100);
77                      }
78                  }
79                  for (int j = 0; j < n; j++) {
80                      B[j] = r.Next(100);
81                  }
82
83                  Console.WriteLine("Исходная матрица и вектор-столбец успешно
 заполнены случайными значениями!\n");
84              }
85          }
86
87          static void SequentialMul() {
88              System.Diagnostics.Stopwatch sWatch = new System.Diagnostics.
Stopwatch();
89              sWatch.Start();
90              for (int i = 0; i < m; i++) {
91                  C[i] = 0;
92                  //Console.WriteLine("\n");  //эта строка тоже относится к
блоку проверки правильности
93                  for (int j = 0; j < n; j++) {
94                      C[i] += A[i, j] * B[j];
95                      //Внимание!
96                      //ниже закоментирован блок отображения
проверки правильности умножения
97                      //Если включить данный блок в исполняемый код,
98                      //то для корректного вывода проверки
99                      //рекомендуется брать матрицы небольшого размера
100                      //например, 3x3
101                      //
```

```
102                     //Console.Write(" {0} * {1} ", A[i,j], B[j]);
103                     //if (j+1 != n)
104                     //{
105                     //    Console.Write("+");
106                     //}
107                     //else
108                     //{
109                     //    Console.Write("= {0}", C[i]);
110                     //}
111                 }
112             }
113             //Console.WriteLine("\n");   //эта строка тоже относится к
блоку проверки правильности
114             sWatch.Stop();
115             Console.WriteLine("Последовательный алгоритм = {0} мс.
",
116             sWatch.ElapsedMilliseconds.ToString());
117
118         }
119
120         static void ParallelMul() {
121             // создание массива объектов для хранения па
раметров
122             InputData[] ClArr = new InputData[nc];
123             for (int i = 0; i < nc; i++) {
124                 ClArr[i] = new InputData();
125             }
126             //Далее, задаются исходные данные для каждог
о экземпляра
127             //вычислительного метода:
128             // делим количество строк в матрице на nc част
ей
129             int step = (Int32)(m / nc);
130             // заполняем массив параметров
131             int c = -1;
132             for (int i = 0; i < nc; i++) {
133                 ClArr[i].start = c + 1;
134                 ClArr[i].stop = c + step;
135                 c = c + step;
136             }
137             //Создаётся диспетчер с пулом из двух потоко
в:
```

```
138             Dispatcher d = new Dispatcher(nc, "Test Pool");
139             DispatcherQueue dq = new DispatcherQueue("Test Queue", d);
140             //Описывается порт, в который каждый экземпл
яр метода Mul()
141             //отправляет сообщение после завершения выч
ислений:
142             Port<int> p = new Port<int>();
143             //Метод Arbiter.Activate помещает в очередь диспетчера две
задачи(два
144             экземпляра метода Mul):
145             for (int i = 0; i < nc; i++) {
146                 Arbiter.Activate(dq, new Task<InputData, Port<int>>
(ClArr[i], p, Mul));
147             }
148             //Первый параметр метода Arbiter.Activate - очередь
диспетчера,
149             //который будет управлять выполнением задач
и, второй параметр -
150             //запускаемая задача.
151
152             //С помощью метода Arbiter.MultipleItemReceive запускается задача
153             //(приёмник), которая обрабатывает получение
двух сообщений портом p:
154             Arbiter.Activate(dq, Arbiter.MultipleItemReceive
(true, p, nc, delegate (int[] ar
ray) {
155                 dispResult();
156                 Console.WriteLine("Вычисления завершены");
157                 Console.ReadKey(true);
158                 Environment.Exit(0);
159             }));
160         }
161
162         static void Mul(InputData data, Port<int> resp) {
163             System.Diagnostics.Stopwatch sWatch = new System.Diagnostics.
Stopwatch();
164             sWatch.Start();
165
166             for (int i = data.start; i < data.stop; i++) {
167                 C[i] = 0;
168                 for (int j = 0; j < n; j++)
169                     C[i] += A[i, j] * B[j];
```

```
170             }
171             sWatch.Stop();
172             Console.WriteLine("Поток № {0}: Паралл. алгоритм = {1} мс
."),
173            Thread.CurrentThread.ManagedThreadId,
174            sWatch.ElapsedMilliseconds.ToString());
175             resp.Post(1);
176         }
177
178         static void dispResult() {
179             int i;
180             Console.WriteLine("Показать результат умножения? (Y/N)
\n");
181
182             var press = Console.ReadKey(true).Key;
183
184             if (press != ConsoleKey.N & press != ConsoleKey.Y) {
185                 Console.WriteLine("Некорректная клавиша");
186                 press = ConsoleKey.N;
187             }
188
189             if (press == ConsoleKey.Y) {
190                 for (i = 0; i < m; i++)
191                     Console.WriteLine("C[{0}]: {1}", i + 1, C[i].ToString());
192             }
193
194             if (press == ConsoleKey.N) {
195                 Console.WriteLine("Вывод результата отклонен.");
196             }
197         }
198
199         static void Main(string[] args) {
200             Test();
201             SequentialMul();
202             dispResult();
203             ParallelMul();
204         }
205     }
206
207 }
```

3.3. Результат выполнения

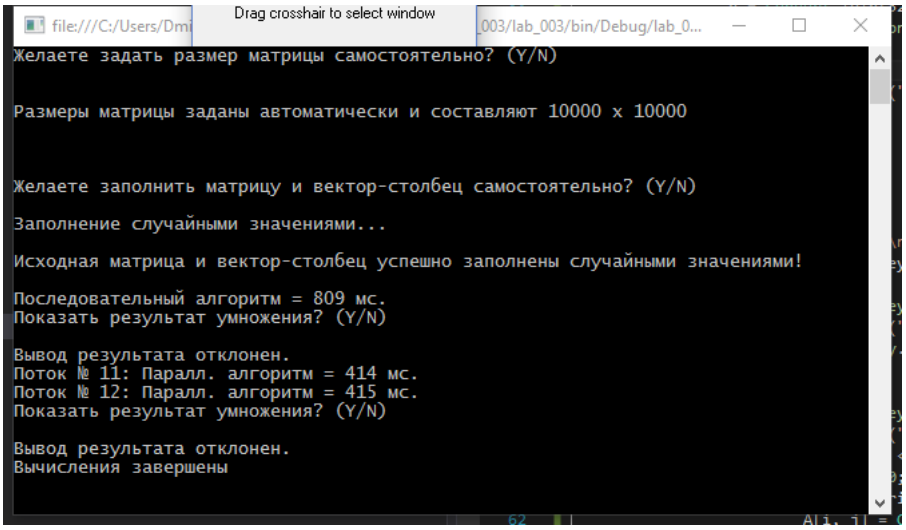

Рисунок 3.1. Пример работы программы.

Вывод

В результате выполнения лабораторной работы мы получили представление о возможности библиотеки Concurrent and Coordination Runtime для организации параллельных вычислений. Мы выяснили, что скорость работы параллельного алгоритма превосходит скорость работы последовательного алгоритма. Быстродействие параллельного алгоритма напрямую зависит от числа используемых ядер.