ГБОУ ВПО Нижегородский государственный технический университет им. Р. Е. Алексеева
Институт радиоэлектроники и информационных технологий, кафедра "Вычислительные системы и технологии"

СОГЛАСОВАНО

Доцент каф. ВСТ

_____ *Гай В. Е.*

“____” _____ _____

ТЕХНОЛОГИИ РАСПРЕДЕЛЁННОЙ ОБРАБОТКИ ДАННЫХ
Отчет к лабораторной работе №3

РАСПАРАЛЛЕЛИВАНИЕ АЛГОРИТМА С ПОМОЩЬЮ БИБЛИОТЕКИ CCR

Студент гр. 13-В-1

_____ *Кононова И. В.*

“____” _____ _____

2015

СОДЕРЖАНИЕ

1 ЦЕЛЬ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

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

Порядок выполнения работы:

а) Разработка последовательного алгоритма, решающего одну из приведённых задач в соответствии с выданным вариантом задания;

б) Разработка параллельного алгоритма, соответствующий варианту последовательного алгоритма;

в) 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

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:

а) Сообщение – экземпляр любого типа данных;

б) 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);

в) 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.

г) 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);

2.2 Создание проекта

Нужно выполнить следующие действия:

а) Установить библиотеку CCR (CCR входит в состав Microsoft Robotics Developer Studio);

б) Создать проект консольного приложения и добавьте к проекту библиотеку Microsoft.Ccr.Core.dll.

2.3 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 Вариант задания

Вариант 1:

– Разработать алгоритм вычисления скалярного произведения векторов a и b.

3.2 Листинг программы

```
  using    System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using Microsoft.Ccr.Core;
using System.Threading;

namespace ConsoleApplication2
{
    class Program
    {
        public class InputData
        {
            public int start; // начало диапазона
            public int stop;  // начало диапазона
        }


        static void Main(string[] args)
        {
```

```
int[] A;//вектор a
int[] B;//вектор b
int[] C;//результат вычисления скалярного произведения век
   торов
int m;
int nc;

nc = 2;//количество ядер
m = 500000;//количество строк матрицы
A = new int[m];
B = new int[m];
C = new int[m];
Random r = new Random();
for (int i = 0; i < m; i++)
{
    A[i] = r.Next(100);

    B[i] = r.Next(100);
}

Stopwatch sWatch = new Stopwatch();//определение время вып
   олнения вычислений для последовательного алгоритма
sWatch.Start();
for (int i = 0; i < m; i++)
{
    C[i] = 0;
    C[i] += A[i] * B[i];
}
sWatch.Stop();
Console.WriteLine("Последовательный алгоритм = {0} мс.",
sWatch.ElapsedMilliseconds.ToString());
//Console.ReadKey();


// создание массива объектов для хранения параметров
InputData[] ClArr = new InputData[nc];
for (int i = 0; i < nc; i++)
    ClArr[i] = new InputData();

// делим количество строк в матрице на nc частей
int step = (Int32)(m / nc);
```

```
            // заполняем массив параметров
            int c = -1;
            for (int i = 0; i < nc; i++)
            {
                ClArr[i].start = c + 1;
                ClArr[i].stop = c + step;
                c = c + step;
            }

            Dispatcher d = new Dispatcher(nc, "Test Pool");//Создание
               диспетчера с пулом из двух потоков
            DispatcherQueue dq = new DispatcherQueue("Test Queue", d);
            Port<int> p = new Port<int>();

            for (int i = 0; i < nc; i++)
                Arbiter.Activate(dq, new Task<InputData, Port<int>>(
                   ClArr[i], p, Mul));

            Arbiter.Activate(dq,Arbiter.MultipleItemReceive(true, p,
               nc, delegate(int[] array) //запуск задачи, обрабатывающ
               ей получение двух сообщений портом p
{
    Console.WriteLine("Вычисления завершены");
}));

        }

        static void Mul(InputData data, Port<int> resp)
        {

            int[] A;
            int[] B;
            int[] C;
            int m;


            m = 500000;

            A = new int[m];
            B = new int[m];
            C = new int[m];
```

```
            Random r = new Random();
            for (int i = 0; i < m; i++)
            {
                A[i] = r.Next(100);

                B[i] = r.Next(100);
            }

            Stopwatch sWatch = new Stopwatch();//опредеделение время вып
                олнения вычислений для параллельного алгоритма
            sWatch.Start();

            for (int i = data.start; i < data.stop; i++)
            {

                C[i] = 0;
                C[i] += A[i] * B[i];
            }
            sWatch.Stop();
            Console.WriteLine("Поток      {0}: Паралл. алгоритм = {1} м
                с.",
        Thread.CurrentThread.ManagedThreadId,
          sWatch.ElapsedMilliseconds.ToString());
            resp.Post(1);
        }

    }
}
```

3.3 Результат работы программы

Скриншот работы программы представлен на Рис.1.

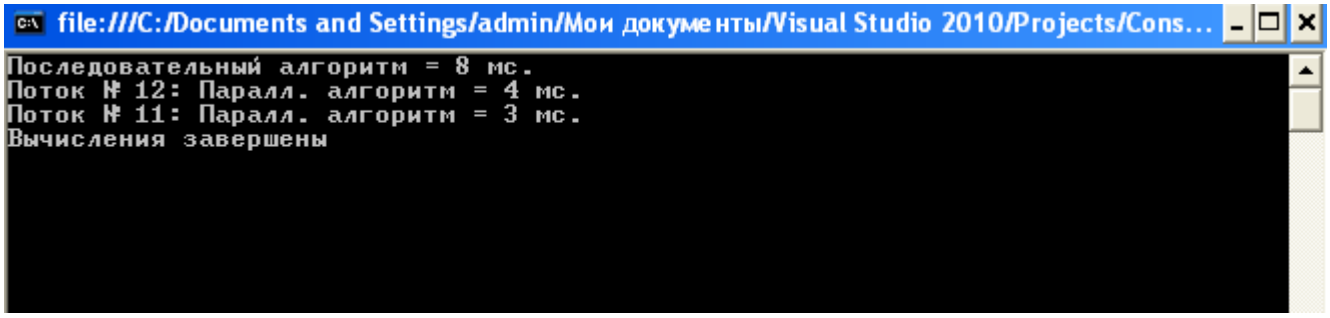

Рисунок 1

4 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 более чем в 2 раза. Быстродействие параллельного алгоритма напрямую зависит от числа используемых ядер.