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.	
	ЕДЕЛЁННОЙ ОБРАБОТКИ ДАННЫХ абораторной работе №3
0 2 20 2 20 10	recording to the contract of t
	НИЕ АЛГОРИТМА С ПОМОЩЬЮ БЛИОТЕКИ ССК
	Студент гр. 13-В-1
	Смирнов А. В.
	<u>"</u> "

2015

подл.

СОДЕРЖАНИЕ

1	Цел	ь и порядок выполнения работы	3
2	Teo	ретические сведения	4
	2.1	Библиотека Concurrent and Coordination Runtime	4
	2.2	Создание проекта	5
	2.3	Оценка времени выполнения	5
3	Выг	толнение лабораторной работы	6
	3.1	Вариант задания	6
	3.2	Листинг программы	6
	3.3	Результат работы программы	12
4	Выв	зо д	13

Подп. и да						
Инв. дубл.						
Взам. инв.						
Подп. и дата					<i>Распараллеливание</i> .	
	Изм. Лист	докум.	Подп.	Дата	помощью библиот	геки CCR
Инв. подл.	Разраб. Пров. Н. контр. Утв.	Смирнов А. В Гай В. Е.			распределённой обработки данных Отчет к лабораторной	Лит. Лист Листов 2 13
-					работе №3 Копировал	Формат А4

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.

Подп. и дата		
Инв. дубл.		
Взам. инв.		
Подп. и дата		
Инв. подл.	Распараллеливание алгоритм изм Лист докум. Подп. Дата помощью библиотеки ССК	Лист 3
Ш	Копировал	омат А4

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 ССR:

- 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.

Инв. подл. Подп. и дата Взам. инв. Инв. дубл. Подп. и дата

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:

- a) Установить библиотеку ССR (ССR 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 sWatch = new Stopwatch(); sWatch.Start();
```

Stopwatch:

<выполняемый код>

sWatch.Stop();

Console. WriteLine(sWatch. Elapsed Milliseconds. To String());

Изм Лист докум. Подп. Дата

Подп.

дубл.

 N_{HB} .

инв.

Взам.

Подп.

подл.

Распараллеливание алгоритма с помощью библиотеки *CCR*

выполнение лабораторной 3 РАБОТЫ

3.1 Вариант задания

```
Вариант 9:
```

дубл.

инв.

Взам.

Подп.

подл.

Лист

докум.

Разработать алгоритм умножения матрицы a ($m \times n$ элементов) на вектор b(n элементов) по следующей формуле: $c_i = \sum a_{ij} b_j , 1 \leq i \leq m$. Рисунок 1

3.2 Листинг программы

Подп.

Дата

```
System;
        using
Подп. и дата
      using Microsoft. Ccr. Core;
      using System. Threading;
     namespace ConsoleApplication3
          public class InputData
              public int start; // начало диапазона
              public int stop; // конец диапазона
              //public int i;
          }
          class Program
              static int[,] A; //хранение матрицы
              static int[] В; //хранение вектор-столбца для умножения
              static int[] С; //хранение результата
              static int m;
                               //количество строк матрицы
```

Распараллеливание алгоритма с

помощью библиотеки ССР

Лист

```
static int n; //количество столбцов матрицы
static int nc; //количество ядер
static void Test()
    nc = 2;
    ConsoleKey press;
    Console. WriteLine ("Желаете задать размер матрицы самостоят
       ельно? (Y/N) \ n");
        press = Console.ReadKey(true).Key;
    if (press != ConsoleKey.N & press != ConsoleKey.Y)
        Console. WriteLine ("Некорректная клавиша");
        press = ConsoleKey.N;
    if (press == ConsoleKey.Y)
        Console. Write ("Введите количество строк матрицы: ");
        m = Convert. ToInt32 (Console. ReadLine());
        Console. Write ("Введите количество столбцов матрицы: ")
        n = Convert.ToInt32(Console.ReadLine());
    else if (press == ConsoleKey.N)
       m = 15000;
        n = 15000;
        Console. WriteLine ("\nРазмеры матрицы заданы автоматиче
           ски и составляют \{0\} х \{1\}\n", m, n);
    }
   A = new int[m, n];
    B = new int[n];
                  Распараллеливание алгоритма с
                                                              Лист
```

дубл.

инв.

Взам.

Подп.

подл.

Лист

докум.

Подп.

Дата

помощью библиотеки ССР

```
C = new int[m];
Console. WriteLine ("\n\nЖелаете заполнить матрицу и вектор—
   столбец самостоятельно? (Y/N) \setminus n");
press = Console.ReadKey(true).Key;
if (press != ConsoleKey.N & press != ConsoleKey.Y)
    Console. WriteLine ("Некорректная клавиша");
    press = ConsoleKey.N;
}
if (press == ConsoleKey.Y)
    Console. WriteLine ("Ввод матрицы\п");
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
             Console. Write ("A[\{0\},\{1\}] = ", i+1, j+1);
            A[i, j] = Convert. ToInt32 (Console. ReadLine());
    }
    Console. WriteLine ("\n Ввод вектор—столбца\n");
    for (int j = 0; j < n; j++)
        Console. Write ("B[\{0\}] = ", j+1);
        B[j] = Convert. ToInt32 (Console. ReadLine());
}
else if (press == ConsoleKey.N)
    Console. WriteLine ("Заполнение случайными значениями
       ...\n");
    Random r = new Random();
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
               Распараллеливание алгоритма с
                                                            Лист
```

дубл.

 N_{HB} .

инв.

Взам.

Подп.

подл.

Изм. Лист

Подп.

докум.

Дата

помощью библиотеки ССК

```
for (int j = 0; j < n; j++)
            B[i] = r.Next(100);
        Console. WriteLine ("Исходная матрица и вектор-столбец у
           спешно заполнены случайными значениями!\n");
    }
}
static void SequentialMul()
    System. Diagnostics. Stopwatch sWatch = new System.
       Diagnostics. Stopwatch ();
    sWatch. Start();
    for (int i = 0; i < m; i++)
        C[i] = 0;
        //Console. WriteLine("\n"); //эта строка тоже относитс
           я к блоку проверки правильности
        for (int j = 0; j < n; j++)
            C[i] += A[i, j] * B[j];
            //Внимание!
            //ниже закоментирован блок отображения проверки пр
               авильности умножения
            //Если включить данный блок в исполняемый код,
            //то для корректного вывода проверки
            //рекомендуется брать матрицы небольшого размера
            //например, 3x3
            //Console. Write(" \{0\} * \{1\} ", A[i,j], B[j]);
            //if (j+1 != n)
            //{
            //
                  Console. Write ("+");
            //}
            //else
            //{
            //
                  Console. Write (''=\{0\}'', C[i]);
            //}
        }
                   Распараллеливание алгоритма с
                                                               Лист
```

дубл.

 N_{HB} .

инв.

Взам.

Подп.

подл.

Изм. Лист

Подп.

докум.

Дата

A[i, j] = r.Next(100);

помощью библиотеки ССР

```
//Console. WriteLine("\n"); //эта строка тоже относится к
       блоку проверки правильности
   sWatch.Stop();
    Console. WriteLine ("Последовательный алгоритм = \{0\} мс.",
    sWatch. Elapsed Milliseconds. To String());
}
static void ParallelMul()
    // создание массива объектов для хранения параметров
    InputData[] ClArr = new InputData[nc];
    for (int i = 0; i < nc; i++)
        ClArr[i] = new InputData();
    //Далее, задаются исходные данные для каждого экземпляра
    //вычислительного метода:
    // делим количество строк в матрице на пс 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;
    //Создаётся диспетчер с пулом из двух потоков:
    Dispatcher d = new Dispatcher(nc, "Test Pool");
    Dispatcher Queue dq = new Dispatcher Queue ("Test Queue", d);
    //Oписывается порт, в который каждый экземпляр метода Mul
       ()
    //отправляет сообщение после завершения вычислений:
    Port < int > p = new Port < int > ();
    //Memod Arbiter. Activate помещает в очередь диспетчера две
       задачи (два
    //экземпляра метода Mul):
    for (int i = 0; i < nc; i++)
        Arbiter. Activate (dq, new Task<InputData, Port<int>>(
           ClArr[i], p, Mul));
                  Распараллеливание алгоритма с
                                                             Лист
                      помощью библиотеки ССР
```

}

Подп.

дубл.

 N_{HB} .

инв.

Взам.

Подп.

подл.

Изм. Лист

Подп.

докум.

Дата

10

```
//который будет управлять выполнением задачи, второй парам
    //запускаемая задача.
    //C помощью метода Arbiter. MultipleItemReceive запускается
        задача
    //(приёмник), которая обрабатывает получение двух сообщени
       й портом р:
    Arbiter. Activate (dq, Arbiter. MultipleItemReceive (true, p,
       nc, delegate (int[] array)
        dispResult();
        Console. WriteLine ("Вычисления завершены");
        Console. ReadKey(true);
        Environment. Exit (0);
    }));
}
static void Mul(InputData data, Port<int> resp)
    System. Diagnostics. Stopwatch sWatch = new System.
       Diagnostics. Stopwatch ();
    sWatch.Start();
    for (int i = data.start; i < data.stop; i++)</pre>
        C[i] = 0;
        for (int j = 0; j < n; j++)
            C[i] += A[i, j] * B[j];
    sWatch.Stop();
    Console. WriteLine ("Поток
                              \{0\}: Паралл. алгоритм = \{1\} м
      c.",
   Thread. CurrentThread. ManagedThreadId,
  sWatch. Elapsed Milliseconds. To String());
    resp. Post(1);
}
                   Распараллеливание алгоритма с
                                                               Лист
```

//Первый параметр метода Arbiter. Activate

етчера,

Подп.

дубл.

 N_{HB} .

инв.

Взам.

Подп.

подл.

Инв.

Изм. Лист

Подп.

докум.

Дата

помощью библиотеки ССК

11

очередь дисп

```
static void dispResult()
            int i;
            Console. WriteLine ("Показать результат умножения? (Y/N) \ n")
            var press = Console.ReadKey(true).Key;
            if (press != ConsoleKey.N & press != ConsoleKey.Y)
                Console. WriteLine ("Некорректная клавиша");
                press = ConsoleKey.N;
            }
            if (press == ConsoleKey.Y)
               for (i = 0; i < m; i++)
                  Console. WriteLine("C[\{0\}]: \{1\}", i + 1, C[i].
                     ToString());
            }
            if (press == ConsoleKey.N)
                Console. WriteLine("Вывод результата отклонен.");
        }
        static void Main(string[] args)
            Test();
            Sequential Mul();
            dispResult();
            ParallelMul();
        }
}
      3.3
            Результат работы программы
     Скриншот работы программы представлен на Рис. 2.
                           Распараллеливание алгоритма с
                                                                      Лист
```

дубл.

 N_{HB} .

инв.

Взам.

Подп.

подл.

Изм. Лист

Подп.

докум.

Дата

помощью библиотеки ССР

12

```
■ М:\трод\Смирнов Александр\lab3\ConsoleApplication1\bin\Debug\ConsoleApplication1.exe
Желаете задать размер матрицы самостоятельно? (Y∕N)
Размеры матрицы заданы автоматически и составляют 15000 x 15000
Желаете заполнить матрицу и вектор-столбец самостоятельно? (Y/N)
Заполнение случайными значениями...
Исходная матрица и вектор-столбец успешно заполнены случайными значениями!
Последовательный алгоритм = 2981 мс.
Показать результат умножения? (Y/N)
Вывод результата отклонен.
Поток № 4: Паралл. алгоритм = 1547 мс.
Поток № 3: Паралл. алгоритм = 1557 мс.
Показать результат умножения? (Y/N)
Вывод результата отклонен.
Вычисления завершены
```

Рисунок 2

ВЫВОД

дата

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.

Подп. дубл. Инв. инв. Взам. Подп. подл. Изм. Лист Подп. Дата докум.