Национальный технический университет Украины

«Киевский политехнический институт»

Лабораторная работа №2

По курсу

Параллельные и распределенные вычисления

Тема:

C#

Работу выполнил

Студент 3-го курса ФИВТ

Группы ИВ-73

Грубый Павел

Киев-2010

Техническое задание



a = max(MB + MC×MZ)

Этап 1. Построение параллельного алгоритма

**ai = max(MBH + MC×MZH)**

**a = max(a, ai); i=**

**ОР**: a, MC

Этап 2. Разработка алгоритмов работы каждого процесса

|  |  |
| --- | --- |
| **Задача Т1** | Точки синхронизации |
| 1. Ожидание ввода MB, MZ в задаче Т2. | W2-1 |
| 1. Ожидание ввода a, MC в задаче Т3. | W3-1 |
| 1. Вычисление a1 = max(MBH + MC1×MZH). |  |
| 1. Вычисление a = max(a, a1). | КУ |
| 1. Сигнал задаче Т3 о завершении счета. | S3-1 |

|  |  |
| --- | --- |
| **Задача Т2** | Точки синхронизации |
| 1. Ввод MB, MZ. |  |
| 1. Сигнал задачам Т1, T3, T4 о завершении ввода MB, MZ. | S1- 1, S3- 1, S4- 1 |
| 1. Ожидание ввода a, MC в задаче Т3. | W3-1 |
| 1. Вычисление a2 = max(MBH + MC2×MZH). |  |
| 1. Вычисление a = max(a, a2). | КУ |
| 1. Сигнал задачаe Т3 о завершении вычислений. | S3-2 |

|  |  |
| --- | --- |
| **Задача Т3** | Точки синхронизации |
| 1. Ввод a, MC. |  |
| 1. Сигнал задачам Т1, T2, T4 о завершении ввода a, MC. | S1- 1, S2- 1, S4- 1 |
| 1. Ожидание ввода MB, MZ в задаче Т2. | W2-1 |
| 1. Вычисление a3 = max(MBH + MC3×MZH). |  |
| 1. Вычисление a = max(a, a3). | КУ |
| 1. Ожидание сигнала от задач Т1, Т2, Т4 о завершении вычислений. | W1-1,W2-2, W4-1 |
| 1. Вывод а. |  |

|  |  |
| --- | --- |
| **Задача Т4** | Точки синхронизации |
| 1. Ожидание ввода MB, MZ в задаче Т2. | W2-1 |
| 1. Ожидание ввода a, MC в задаче Т3. | W3-1 |
| 1. Вычисление a1 = max(MBH + MC4×MZH). |  |
| 1. Вычисление a = max(a, a4). | КУ |
| 1. Сигнал задаче Т3 о завершении счета. | S3-1 |

Этап 3. Разработка структурной схемы взаимодействия задач

Семафоры:

**sem1** – для синхронизации окончания вычислений в задаче Т1.

**sem2** – для синхронизации окончания вычислений в задаче Т2.

**sem4** – для синхронизации окончания вычислений в задаче Т4.

События:

**E2** – для синхронизации окончания ввода в задаче Т2.

**E3** – для синхронизации окончания ввода в задаче Т3.

Мютекс **Mtx** – для управления доступом к ОР а.

Критическая секция **CS** – для управления доступом к ОР МС.



рис. 3.1. Схема взаимодействия задач.

Сравнение времени выполнения программы

|  |  |  |
| --- | --- | --- |
| Размерность | C++ и Win32 (мс) | C# (мс) |
| 100 | 0.3790680 | 2.99997505726436 |
| 500 | 34.5379 | 344.984021683552 |
| 1000 | 621.27 | 2843.23461594423 |
| 2000 | 5926.12 | 22823.5744706944 |
| 3000 | 19745.2 | 74866.2875552378 |

Этап 4. Разработка программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using Win32;

//--------------------------------------------

//-----ПРО. Лаб 3. C#

//-----Грубьй Павел ИВ-73

//-----a = max(MB + MC\*MZ)

//-----10.03.2010

namespace Lab3

{

class Program

{

public static int N = 1000;

public static int P = 4;

public static int H = N / P;

public static int[,] MB = new int[N, N];

public static int[,] MZ = new int[N, N];

public static int[,] MX = new int[N, N];

volatile public static int[,] MC = new int[N, N];

public static int a = int.MinValue;

public static Mutex Mtx;

public static EventWaitHandle E2, E3;

public static EventWaitHandle[] events = new EventWaitHandle[2];

public static Semaphore sem1, sem2, sem4;

public static Semaphore[] semaphores = new Semaphore[3];

static void Main(string[] args)

{

Mtx = new Mutex(false);

E2 = new ManualResetEvent(false);

E3 = new ManualResetEvent(false);

events[0] = E2;

events[1] = E3;

sem1 = new Semaphore(0, 1);

sem2 = new Semaphore(0, 1);

sem4 = new Semaphore(0, 1);

semaphores[0] = sem1;

semaphores[1] = sem2;

semaphores[2] = sem4;

Thread t1 = new Thread(T1);

Thread t2 = new Thread(T2);

Thread t3 = new Thread(T3);

Thread t4 = new Thread(T4);

t1.Start();

t2.Start();

t3.Start();

t4.Start();

Console.WriteLine("Main: End");

Console.ReadKey();

}

static void T1()

{

Console.Write("Task 1 started.\n");

Console.Write("Task 1 waits for E2, E3\n");

EventWaitHandle.WaitAll(events);

Console.Write("Task 1 started calculations.\n");

kernel(0,MC,MZ,MB);

// Console.Write("Task 1 ended calculations.\n");

sem1.Release();

//Console.Write("Task 1 finished.\n");

}

static void T2()

{

Console.Write("Task 2 started.\n");

Console.Write("Task 2 inputs data.\n");

matrixInput(MB);

matrixInput(MZ);

E2.Set();

Console.Write("Task 2 waits for E3\n");

E3.WaitOne();

Console.Write("Task 2 started calculations.\n");

kernel(1, MC, MZ, MB);

// Console.Write("Task 2 ended calculations.\n");

sem2.Release();

//Console.Write("Task 2 finished.\n");

}

static void T3()

{

Console.Write("Task 3 started.\n");

Console.Write("Task 3 inputs data.\n");

matrixInput(MC);

E3.Set();

Console.Write("Task 3 waits for E3\n");

E2.WaitOne();

Console.Write("Task 3 started calculations.\n");

kernel(2, MC, MZ, MB);

//Console.Write("Task 3 ended calculations.\n");

Semaphore.WaitAll(semaphores);

if (N <= 8) matrixOutput(MX);

Console.Write("a= ");

Console.WriteLine(a);

//Console.Write("Task 3 finished.\n");

}

static void T4()

{

Console.Write("Task 4 started.\n");

Console.Write("Task 4 waits for E2, E3\n");

EventWaitHandle.WaitAll(events);

Console.Write("Task 4 started calculations.\n");

kernel(3, MC, MZ, MB);

//Console.Write("Task 4 ended calculations.\n");

sem4.Release();

//Console.Write("Task 4 finished.\n");

}

static void matrixInput(int[,] matrix)

{

for (int i=0; i<N; i++){

for (int j=0; j<N; j++){

matrix[i,j] = 1;

}

}

}

static void matrixOutput(int[,] matrix)

{

String s = "";

for (int i=0; i<N; i++){

for (int j=0; j<N; j++){

s += matrix[i, j].ToString() + " ";

}

s += "\n";

}

Console.Write(s);

}

static void kernel(int id, int[,] M1, int[,] M2, int[,] M3)

{

HiPerfTimer pt = new HiPerfTimer(); // create a new PerfTimer object

pt.Start(); // start the timer

int max = 0;

for (int j = H \* id; j < H \* (id + 1); j++)

{

//M1\*M2

for (int i = 0; i < N; i++)

{

MX[i, j] = 0;

for (int k = 0; k < N; k++)

{

MX[i, j] += M1[i, k] \* M2[j, k];

}

}

//max(M3 + (M1\*M2))

for (int i = 0; i < N; i++)

{

MX[i, j] += M3[i, j];

if (max < MX[i, j])

max = MX[i, j];

}

}

Mtx.WaitOne();

if (a < max) a = max;

Mtx.ReleaseMutex();

pt.Stop(); // stop the timer

Console.Write("Task ");

Console.Write(id+1);

Console.Write(" time: {0} (ms)\n", pt.Duration); // print the duration of the timed code

}

}

}