Национальный технический университет Украины

«Киевский политехнический институт»

Лабораторная работа №2

По курсу

Параллельные и распределенные вычисления

Тема:

Win32

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. Копия MC1:= MC. | КУ |
| 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. Копия MC2:= MC. | КУ |
| 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. Копия MC3:= MC. | КУ |
| 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. Копия MC4:= MC. | КУ |
| 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.

Этап 4. Разработка программы

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

//-----ПРО. Лаб 2. Win32

//-----Грубьй Павел ИВ-73

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

//-----3.03.2010

#include "stdafx.h"

#include "windows.h"

#include <iostream>

using namespace std;

const int N = 3000;

const int P = 4;

const int H = N / P;

int MB[N][N], MC[N][N], MZ[N][N];

int MX[N][N];

int a = 0;

HANDLE Mtx;

HANDLE E2, E3;

HANDLE events[2];

HANDLE sem1, sem2, sem4;

HANDLE semaphores[3];

CRITICAL\_SECTION CS;

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

void matrixInput(int matrix[N][N]);

void matrixOutput(int matrix[N][N]);

void T1(){

cout << "Task 1 started.\n";

cout << "Task 1 waits for E2, E3\n";

WaitForMultipleObjects(2, events, true, INFINITE);

cout << "Copy data in task 1\n";

int MC1[N][N];

EnterCriticalSection(&CS);

memcpy(MC1, MC, sizeof(MC));

LeaveCriticalSection(&CS);

cout << "Task 1 started calculations.\n";

kernel(0,MC1,MZ,MB);

cout << "Task 1 ended calculations.\n";

ReleaseSemaphore(sem1,1,NULL);

cout << "Task 1 finished.\n";

ExitThread(0);

}

void T2(){

cout << "Task 2 started.\n";

cout << "Task 2 inputs data.\n";

matrixInput(MB);

matrixInput(MZ);

SetEvent(E2);

cout << "Task 2 waits for E2\n";

WaitForSingleObject(E3, INFINITE);

cout << "Copy data in task 2\n";

int MC2[N][N];

EnterCriticalSection(&CS);

memcpy(MC2, MC, sizeof(MC));

LeaveCriticalSection(&CS);

cout << "Task 2 started calculations.\n";

kernel(1,MC2,MZ,MB);

cout << "Task 2 ended calculations.\n";

ReleaseSemaphore(sem2,1,NULL);

cout << "Task 2 finished.\n";

ExitThread(0);

}

void T3(){

cout << "Task 3 started.\n";

cout << "Task 3 inputs data.\n";

matrixInput(MC);

SetEvent(E3);

cout << "Task 3 waits for E2\n";

WaitForSingleObject(E2, INFINITE);

cout << "Copy data in task 3\n";

int MC3[N][N];

EnterCriticalSection(&CS);

memcpy(MC3, MC, sizeof(MC));

LeaveCriticalSection(&CS);

cout << "Task 3 started calculations.\n";

kernel(2,MC3,MZ,MB);

cout << "Task 3 ended calculations.\n";

WaitForMultipleObjects(3, semaphores, true, INFINITE);

if (N<8) matrixOutput(MX);

cout << "a = " << a << "\n";

cout << "Task 3 finished.\n";

ExitThread(0);

}

void T4(){

cout << "Task 4 started.\n";

cout << "Task 4 waits for E2, E3\n";

WaitForMultipleObjects(2, events, true, INFINITE);

cout << "Copy data in task 4\n";

int MC4[N][N];

EnterCriticalSection(&CS);

memcpy(MC4, MC, sizeof(MC));

LeaveCriticalSection(&CS);

cout << "Task 4 started calculations.\n";

kernel(3,MC4,MZ,MB);

cout << "Task 4 ended calculations.\n";

ReleaseSemaphore(sem4,1,NULL);

cout << "Task 4 finished.\n";

ExitThread(0);

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

cout << "Main thread has started \n";

InitializeCriticalSection(&CS);

Mtx = CreateMutex(NULL, 0, NULL);

DWORD TidA, TidB, TidC, TidD;

HANDLE Thread1, Thread2, Thread3, Thread4;

sem1 = CreateSemaphore(NULL,0,1,NULL);

sem2 = CreateSemaphore(NULL,0,1,NULL);

sem4 = CreateSemaphore(NULL,0,1,NULL);

semaphores[0] = sem1;

semaphores[1] = sem2;

semaphores[2] = sem4;

E2 = CreateEvent(NULL, 1, 0, NULL);

E3 = CreateEvent(NULL, 1, 0, NULL);

events[0] = E2;

events[1] = E3;

Thread1 = CreateThread(NULL, sizeof(MC),

(LPTHREAD\_START\_ROUTINE) T1, NULL, STACK\_SIZE\_PARAM\_IS\_A\_RESERVATION, &TidA);

Thread2 = CreateThread(NULL, sizeof(MC),

(LPTHREAD\_START\_ROUTINE) T2, NULL, STACK\_SIZE\_PARAM\_IS\_A\_RESERVATION, &TidB);

Thread3 = CreateThread(NULL, sizeof(MC),

(LPTHREAD\_START\_ROUTINE) T3, NULL, STACK\_SIZE\_PARAM\_IS\_A\_RESERVATION, &TidC);

Thread4 = CreateThread(NULL, sizeof(MC),

(LPTHREAD\_START\_ROUTINE) T4, NULL, STACK\_SIZE\_PARAM\_IS\_A\_RESERVATION, &TidD);

cout << "Main thread has finished \n";

cin.get();

return 0;

}

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

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];

}

}

WaitForSingleObject(Mtx, INFINITE);

if (a < max) a = max;

ReleaseMutex(Mtx);

}

void matrixInput(int matrix[N][N]){

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

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

//cout << "M["; cout << i; cout << ",";

//cout << j; cout << "]=";

//cin >> matrix[i][j];

matrix[i][j] = 1;

}

}

}

void matrixOutput(int matrix[N][N]){

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

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

// cout << "M["; cout << i;

//cout << ","; cout << j; cout << "]=";

cout << matrix[i][j]; cout << " ";

}

cout << "\n";

}

}