МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

 



**ОТЧЕТ**

**по лабораторной работе**

|  |  |
| --- | --- |
| *Дисциплина* | Теория языков программирования и вычислительных процессов (ТЯПиВП) |
| *Номер работы* | 1 |
| *Тема работы* | Исследование влияния псевдопараллелизма на производительность |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Принял:** | |  | **Выполнил (-и):** | | | |
|  | В.В.Хашковский, доцент каф. МОП ЭВМ |  |  | *Фамилия И.О.* | *Группа* | *Оценка* |
|  |  |  |  | Иванов И.И. | гр. А-??? |  |
|  |  |  |  | Петров П.П. | гр АУ-??? |  |

Таганрог 2020

# **1 Цель работы**

* 1. **Дидактическая цель работы**

В ходе выполнения данной работы изучить механизм псевдопараллелизма и его влияние на производительность.

**1.2 Практическая цель работы**

Научиться использовать на практике методы управления потоками в языке C++.

**2. Вариант задания:**

Изучить влияние увеличения количества потоков на скорость перемножения матриц.

**3. Ход работы**

**3.1 Теоретическое описание проблемы**

Когда мы используем компьютеры, создается ощущение, что они могут делать сразу несколько дел одновременно. На самом деле процессор переключается между программами, предоставляя каждой от десятков до сотен миллисекунд – кванты времени. Это явление называется псевдопараллелизм.

Для распараллеливания разных подзадач одной программы могут использоваться процессы или потоки. Каждый процесс имеет свое адресное пространство, в то время как потоки внутри одного процесса разделяют между собой адресное пространство и, например, один поток может поменять значение переменной, используемой другим потоком. Следовательно полученные данные могут быть не корректны или же мы можем получить ошибку. К примеру, выйти за границу массива.

Чтобы этого не произошло используются различные механизмы синхронизации. В данной работе использованы мьютексы (от англ. **MUT**ual **EX**clusion). Мьютекс — переменная, которая может находиться в одном из двух состояний: блокированном или неблокированном.

В данной работе мьютекс блокируется в том месте кода, где происходит изменение общих данных

**3.2 Описание алгоритма перемножения матриц**.

|  |  |
| --- | --- |
| 1 | N = размерность матрицы |
| 2 | i=0 |
| 3 | j=0 |
| 4 | k=0 |
| 5 | While(i>N) |
| 6 | mutex.lock() //блокируется доступ других потоков к общим переменным |
| 7 | If (i>=N) then end while |
| 8 | mat3[i][j]=mat1[i][k]\*mat2[k][j] |
| 9 | k=k+1 |
| 10 | If (k>=N) then |
| 11 | k=0 |
| 12 | j=j+1 |
| 13 | If (j>=N) then |
| 14 | j=0 |
| 15 | i=i+1 |
| 16 | mutex.unlock() |
| 17 | End While |

**3.3** Для написания программы была использована платформа Visual Studio и язык C++. Для разработки программы были использованы модули: <chrono>, <thread>, <mutex>.

**3.4 Полученные результаты**:

**4. Вывод**

В ходе выполнения лабораторной работыбыли процесс псевдопараллелизма и синхронизация потоков. Была написана программа, позволяющая оценить влияние псевдопараллелизма на производительность.

**5. Приложения**

Листинг:

#include <iostream>

#include <fstream>

#include <chrono>

#include <vector>

#include <thread>

#include <mutex>

#include <fstream>

#include <string>

using namespace std;

mutex mtx;

void new\_matrix\_null(int\*\* A, int n, int m)

{

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

{

A[i] = new int[n];

for (int j = 0; j < m; j++)

{

A[i][j] = 0;

}

}

}

void new\_matrix\_random(int\*\* A, int n, int m)

{

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

{

A[i] = new int[n];

for (int j = 0; j < m; j++)

{

A[i][j] = rand()%1000;

}

}

}

int\*\* mul\_matrix(int\*\* mat1, int\*\* mat2, int n)

{

int\*\* res = new int\*[n];

new\_matrix\_null(res, n, n);

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

for (int j = 0; j < n; ++j)

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

res[i][j] += mat1[i][k] \* mat2[k][j];

return res;

}

int i = 0;

int j = 0;

int k = 0;

void mul\_matrix2(int\*\* mat1, int\*\* mat2, int\*\* mat3, int n)

{

this\_thread::sleep\_for(chrono::milliseconds(1000));

while (i < n)

{

this\_thread::get\_id();

mtx.lock();

if (i >= n)

{

mtx.unlock(); break;

}

mat3[i][j] += mat1[i][k] \* mat2[k][j];

k++;

if (k >= n)

{

k = 0; j++;

}

if (j >= n)

{

j = 0; i++;

}

mtx.unlock();

}

}

void show\_matrix(int\*\* A, int n, int m)

{

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

{

//cout << A[i] << endl << endl;

for (int j = 0; j < m; ++j)

{

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

}

cout << endl;

}

}

void threadFunction(int &a)

{

a++;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int n = 10;

int\*\* A = new int\*[n];

new\_matrix\_random(A, n,n);

int\*\* B = new int\*[n];

new\_matrix\_random(B, n, n);

int\*\* C = new int\*[n];

new\_matrix\_null(C, n, n);

thread th1(mul\_matrix2, A, B, C, n);

thread th2(mul\_matrix2, A, B, C, n);

th1.join();

th2.join();

show\_matrix(C, n, n);

//th1.detach();

//th2.detach();

//int a = 1;

//std::thread thr(threadFunction, std::ref(a));

//thr.join();

//std::cout << a << std::endl;

return 0;

}