Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Операционные системы»

Студент: Рокотянский А.Е.
Группа: М80-201Б-21
Вариант: 12
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022

Содержание

Репозиторий

https://github.com/Sly-al/OS-labs

Постановка задачи

Цель работы

Целью является приобретение практических навыков в:
Управление потоками в ОС
Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.
Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.
В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 12

Наложить K раз фильтр, использующий матрицу свертки, на матрицу, состоящую из вещественных чисел. Размер окна 3x3

Общие сведения о программе

Программа компилируется из файла main.cpp. Также используется заголовочные файлы: iostream, unistd.h, stdio.h, stdlib.h, vector, pthread.h. В программе используются следующие системные вызовы:

1. pthread_create() – создание нового потока
2. pthread_join() – ожидание окончания потока

Общий метод и алгоритм решения

На вход подается размер матрицы (использую квадратную) и размер матрицы фильтра. В функции фильтра происходит преобразование. Так каждому потоку передано необходимые строки матрицы. Дальше происходит наложения фильтра на матрицу.

Минусы распараллеливания на данной задаче происходит при довольно маленьких размерах матрицы, что при малых потоках вычисления выполняются быстрее нежели на больших количествах потоков.

Исходный код

Main.cpp

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <vector>

#include "lab3.h"

using namespace std;

int main(){
    int n, m, maxThread, loop;

    cin >> n >> m >> maxThread >> loop;

    TMatrix mas1(n, vector<double>(m,0));
    TMatrix mas2(KERNEL_MATRIX_SIZE, vector<double>(KERNEL_MATRIX_SIZE,0));

    double x;
    for(int i = 0; i < n; ++i){
        for(int j = 0; j < m; ++j){
            cin >> x;
            mas1[i][j] = x;
        }
    }

    for(int i = 0; i < KERNEL_MATRIX_SIZE; ++i){
        for(int j = 0;j < KERNEL_MATRIX_SIZE; ++j){
            cin >> x;
            mas2[i][j] = x;
        }
    }

    TMatrix result = MatrixConvolution(mas1, mas2, maxThread, loop);

    cout << " result \n";
    for(int i = 0; i < n; ++i){
        for(int j = 0;j < m; ++j){
            cout << result[i][j] << " ";

        }
        cout << '\n';
    }

}
```

Lab3.cpp

```
 #include "lab3.h"
 #include "utils.h"

 #include <iostream>
 #include <stdio.h>
 #include <stdlib.h>
 #include <unistd.h>
 #include <pthread.h>
 #include <vector>

 using namespace std;

 struct TData {
```

```
    vector<vector<double> > &mas1;
    vector<vector<double> > &mas2;
    vector<vector<double> > &res1;

    int threadNum;
    int threadCount;
};

void *MatrixMult(void *args){

    vector<vector<double>>&mas1 = ((TData *)args)->mas1;
    vector<vector<double>>&mas2 = ((TData *)args)->mas2;
    vector<vector<double>>&res1 = ((TData *)args)->res1;

    int threadNum = ((TData *)args)->threadNum;  // номер потока
    int thCount = ((TData *)args)->threadCount;  // количество потоков
    int cols = isize(mas1[0]);
    int rows = isize(mas1);
    int dir[3] = {-1, 0, 1};

     for (int thRow = threadNum; thRow < rows; thRow += thCount) {

        for (int thCol = 0; thCol < cols; thCol++) {

            double div = 0.0;
            double newVal = 0.0;

            for (int i = 0; i < KERNEL_MATRIX_SIZE; ++i) {
                for (int j = 0; j < KERNEL_MATRIX_SIZE; ++j) {
                    div += mas2[i][j];
                    if ( (thRow + dir[i] >= 0) && (thRow + dir[i] < rows) &&
                    (thCol + dir[j] >= 0) && (thCol + dir[j] < cols) ) {
                        newVal += mas1[thRow + dir[i]][thCol + dir[j]] * mas2[j][i];
                    }
                }
            }
            res1[thRow][thCol] = newVal/div;
        }

    }

    return NULL;
}


TMatrix MatrixConvolution(TMatrix mas1, TMatrix mas2, int maxThread, int loop){

    int n = isize(mas1);
    int m = isize(mas1[0]);

    TMatrix res1(n, vector<double>(m,0));

    vector<TData> data = vector<TData>(maxThread,{mas1,mas2,res1,0,0});

    vector<pthread_t> threads(maxThread);

    for(int k = 0; k < loop; ++k){

        for(int p = 0; p < maxThread; p++){
            data[p].threadCount = maxThread;
            data[p].threadNum = p;
            if(p >= n*m){
                break;
```

```
            }

            if(int err = pthread_create(&threads[p], NULL, MatrixMult, (void
 *)&data[p])){
                cout << "Thread create error: " << err << '\n';
                exit(EXIT_FAILURE);
            }
        }

        //join
        for(int i = 0; i < maxThread; i++) {
            if (pthread_join(threads[i],NULL) != 0) {
                cout << "Can't wait for thread\n";
            }
        }

        swap(data[0].mas1,data[0].res1);

    }
    swap(data[0].mas1,data[0].res1);

    return res1;
 }
```

Lab3.h

```
#ifndef LAB3_H
#define LAB3_H
#include <vector>

using namespace std;

constexpr int KERNEL_MATRIX_SIZE = 3;

using TMatrix = vector<vector<double>>;

TMatrix MatrixConvolution(TMatrix mas1, TMatrix mas2, int maxThread, int loop);

#endif
```

Utils.h

```
#ifndef UTILS_H
#define UTILS_H

template <typename Container>
inline int isize(const Container& c) {
    return static_cast<int>(c.size());
}

#endif
```

Демонстрация работы программы

alex@alex-VirtualBox:~/Рабочий стол/OS-labs/build/lab3$ ./lab3

3 3 2 1

2 2 2

2 2 2

2 2 2

0.5 0.5 0.5

0.5 0.5 0.5

0.5 0.5 0.5

result

0.888889 1.33333 0.888889

1.33333 2 1.33333

0.888889 1.33333 0.888889

Выводы

Составлена и отлажена многопоточная программа на языке Си, выполняющая наложение матрицы свёртки на матрицу. Тем самым, приобретены навыки в распараллеливании вычислений, управлении потоками и обеспечении синхронизации между ними.