Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Группа: М8О-215Б-23

Студент: Шаталов М.А.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 17.10.24

Москва, 2024

**Постановка задачи**

**Вариант 13.**

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

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* open(const char \*pathname, mode\_t mode) - открытие\создание файла
* pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*func, void \*arg)
* exit(int status) - завершения выполнения процесса
* close(int fd) - закрыть файл
* pthread\_join(pthread\_t \_\_th, void \*\*return)
* int execl(const char \*path, const char \*arg, ...); - загружает и исполняет новый образ программы.

Для начала разберемся, что из себя представляет операция наложения матрицы свертки на матрицу. Для этой операции требуется две матрицы. Первая - матрица на которую накладывается матрица свёртки и вторая - сама матрица свёртки или же ядро. Ядро также называется окном.

Для удобного тестирования добавим в программу ключ запуска который может иметь значение либо “f”, ”i”. Где ‘f’ - означает, что программа использует данные из файла test.txt, флаг ‘i’ означает, что мы будем использовать ввод через консоль.

Первая часть программы выполняется в вызывающем потоке и отвечает за ввод матрицы и создания переменных.

Для операции свёртки используем формулу:

g(x, y) = w \* f(x, y) =

Где g(x,y) - отфильтрованное изображение, f(x,y) - исходное изображение, w - ядро фильтра. Каждый элемент ядра фильтра рассматривается при -a≤i≤a и -b≤j≤b.

Из формулы видно, что для применения фильтра нужно посчитать формулу для каждого элемента матрицы. Используем многопоточность и разделим линеаризованную версию матрицы на равные части, количество которых будет равно количеству доступных потоков.

Не забудем, что максимальное число потоков для матрицы равно M\*N где M, N размеры матрицы.

Для синхронизации потоков будем использовать барьерный паттерн. Его суть заключается в том, что мы запускаем несколько потоков и на определенном состоянии программы(барьере) ждем пока все потоки выполнятся. В нашем случае барьером будет K-ая итерация применения фильтра к матрице. Для каждой итерации K запускаем максимальное число размешенных потоков и ждём пока все потоки завершат свою работу.

Для опитимизации памяти мы не будем создавать много матриц для результатов каждого шага. Вместо этого будем использовать 2 матрицы, одна для чтения другая для записи. После каждого K будем менять их местами. Ответом будет являться либо первая либо вторая матрица в зависимости от чётности K.

Каждая функция потока будет принимать начало и конец интервала, который она должна обработать. Для удобства функцию вычисляющую формулу для элемента на позиции (x, y) будем использовать еще одну функцию принимающую координаты элемента, матрицу из которой будем читать элементы, матрицу в которую будем записывать результат и матрицу ядра.

**Код программы**

#include <iostream>

#include <fstream>

#include <unistd.h>

#include <pthread.h>

#include <vector>

using namespace std;

int K, M, N, H, W, tk;

vector<double> A;

vector<double> B;

vector<double> S;

void print\_matrix(vector<double> &mt, ostream \*out, int &M, int &N)

{

    (\*out) << M << " " << N << "\n";

    for (int i = 0; i < N \* M; i++)

    {

        (\*out) << mt[i];

        if ((i + 1) % N != 0)

            (\*out) << " ";

        if (i % N == N - 1 && i != N \* M - 1)

            (\*out) << "\n";

    }

    (\*out) << '\n';

}

void read\_matrix(vector<double> &mt, istream \*in, int &M, int &N)

{

    double el;

    for (int i = 0; i < N \* M; i++)

    {

        (\*in) >> el;

        mt.push\_back(el);

    }

}

double get\_x\_y(int x, int y, vector<double> &mt, int &M, int &N)

{

    if (x < 0 or x >= N)

        return 0;

    if (y < 0 or y >= M)

        return 0;

    return mt[y \* N + x];

}

void set\_x\_y(double val, int x, int y, vector<double> &mt, int &M, int &N)

{

    if (x < 0 or x >= N)

        return;

    if (y < 0 or y >= M)

        return;

    mt[y \* N + x] = val;

}

void calculate\_cell(int &x, int &y, vector<double> &A, vector<double> &B, vector<double> &S)

{

    double sum = 0;

    for (int i = 0; i < H; ++i)

    {

        for (int j = 0; j < W; j++)

        {

            sum += get\_x\_y(j, i, B, H, W) \* get\_x\_y(x + j - W / 2, y + i - H / 2, A, M, N);

        }

    }

    set\_x\_y(sum, x, y, S, M, N);

}

void \*Thread\_func(void \*args)

{

    pair<int, int> \*st\_end = (pair<int, int> \*)args;

    int start = st\_end->first;

    int end = st\_end->second;

    for (int i = start; i < end; ++i)

    {

        int x, y;

        x = i % N;

        y = i / N;

        if (tk % 2 == 0)

            calculate\_cell(x, y, A, B, S);

        else

            calculate\_cell(x, y, S, B, A);

    }

    return NULL;

}

int main(int argc, char \*argv[])

{

    char Mode = 'f';

    argv[2][0];

    ostream \*out;

    istream \*in;

    ofstream outf;

    ifstream inf;

    switch (Mode)

    {

    case 'f': // file

        inf.open("test1.txt");

        in = &inf;

        outf.open("out.txt");

        out = &outf;

        break;

    case 'i': // console

        out = &cout;

        in = &cin;

        break;

    default:

        printf("incorrect mode: %c\n", Mode);

        break;

    }

    (\*in) >> K;

    (\*in) >> M >> N;

    read\_matrix(A, in, M, N);

    (\*in) >> H >> W;

    read\_matrix(B, in, H, W);

    S.resize(N \* M, 0);

    int THREAD\_COUNT = min(atoi(argv[1]), M \* N);

    pthread\_t \*thread\_ids = new pthread\_t[K]; // идентификаторы потоков

    int err = 0;

    vector<pair<int, int>> coords(THREAD\_COUNT);

    for (tk = 0; tk < K; ++tk)

    {

        for (int thread = 0; thread < THREAD\_COUNT; ++thread)

        {

            int p = N \* M;

            coords[thread] = {thread \* p / THREAD\_COUNT, (thread + 1) \* p / THREAD\_COUNT};

            err = pthread\_create(&thread\_ids[thread], NULL, Thread\_func, &coords[thread]);//0 if ok, other err code

            //(id, attr(приоритета потока, стека и других параметров), yказатель на функцию, args)

            if (err)

            {

                (\*out) << "Error in thread create";

                exit(1);

            }

        }

        for (int thread = 0; thread < THREAD\_COUNT; ++thread)

        {

            if (pthread\_join(thread\_ids[thread], NULL) != 0)

            { // (id, результат ф-ии потока)

                (\*out) << "Error with join threads";

                exit(1);

            }

        }

    }

    if (K % 2 == 0)

        print\_matrix(S, out, M, N);

    else

        print\_matrix(A, out, M, N);

    delete [] thread\_ids;

    if (Mode == 'f')

    {

        outf.close();

        inf.close();

    }

}

**Протокол работы программы**

**Тестирование:**

root@f5b21642cf59:/workspaces/MAI\_OS\_Labs/build# ./lab2\_exe 2 i

2

3 3

1 2 3

4 5 6

7 8 9

3 3

0 0 0

0 1 0

0 0 0

Out:

3 3

1 2 3

4 5 6

7 8 9

Демонстрация количества потоков:

root@f5b21642cf59:/workspaces/MAI\_OS\_Labs# ps hH p76424 | wc -l

5

**Strace:**

execve("./lab2\_exe", ["./lab2\_exe", "2", "f"], 0x7fff5ab2d628 /\* 30 vars \*/) = 0

brk(NULL) = 0x56413a55e000

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f320048c000

access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=26022, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 26022, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7f3200485000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f3200217000

mmap(0x7f32002bc000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7f32002bc000

mmap(0x7f32003e5000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7f32003e5000

mmap(0x7f3200473000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000) = 0x7f3200473000

mmap(0x7f3200481000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f3200481000

close(3) = 0

openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f32001ea000

mmap(0x7f32001ee000, 143360, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7f32001ee000

mmap(0x7f3200211000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7f3200211000

mmap(0x7f3200215000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7f3200215000

close(3) = 0

**openat**(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\20t\2\0\0\0\0\0"..., 832) = 832

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784

mmap(NULL, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f3200009000

mmap(0x7f320002f000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000) = 0x7f320002f000

mmap(0x7f3200184000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000) = 0x7f3200184000

mmap(0x7f32001d7000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7f32001d7000

mmap(0x7f32001dd000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7f32001dd000

close(3) = 0

**openat**(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832

newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7f31fff2a000

mmap(0x7f31fff3a000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000) = 0x7f31fff3a000

mmap(0x7f31fffad000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7f31fffad000

mmap(0x7f3200007000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000) = 0x7f3200007000

close(3) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f31fff28000

arch\_prctl(ARCH\_SET\_FS, 0x7f31fff29480) = 0

set\_tid\_address(0x7f31fff29750) = 5012

set\_robust\_list(0x7f31fff29760, 24) = 0

rseq(0x7f31fff29da0, 0x20, 0, 0x53053053) = 0

mprotect(0x7f32001d7000, 16384, PROT\_READ) = 0

mprotect(0x7f3200007000, 4096, PROT\_READ) = 0

mprotect(0x7f3200215000, 4096, PROT\_READ) = 0

mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7f31fff26000

mprotect(0x7f3200473000, 45056, PROT\_READ) = 0

mprotect(0x564138a7c000, 4096, PROT\_READ) = 0

mprotect(0x7f32004be000, 8192, PROT\_READ) = 0

prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

munmap(0x7f3200485000, 26022) = 0

futex(0x7f320048173c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

getrandom("\xe0\x21\xad\x8c\x53\xe4\x5f\xf1", 8, GRND\_NONBLOCK) = 8

brk(NULL) = 0x56413a55e000

brk(0x56413a57f000) = 0x56413a57f000

openat(AT\_FDCWD, "test.txt", O\_RDONLY) = 3

openat(AT\_FDCWD, "out.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666) = 4

read(3, "2\n3 3\n1.2 2.7 3.3\n4.1 5.4 6.8\n7."..., 8191) = 64

read(3, "", 8191) = 0

rt\_sigaction(SIGRT\_1, {sa\_handler=0x7f320008f6a0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7f3200045050}, NULL, 8) = 0

rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7f31ff725000

mprotect(0x7f31ff726000, 8388608, PROT\_READ|PROT\_WRITE) = 0

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

**clone3**({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f31fff25990, parent\_tid=0x7f31fff25990, exit\_signal=0, stack=0x7f31ff725000, stack\_size=0x7fff80, tls=0x7f31fff256c0}strace: Process 5013 attached

=> {parent\_tid=[5013]}, 88) = 5013

[pid 5013] rseq(0x7f31fff25fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 5012] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5013] <... rseq resumed>) = 0

[pid 5012] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5013] set\_robust\_list(0x7f31fff259a0, 24 <unfinished ...>

[pid 5012] mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

[pid 5013] <... set\_robust\_list resumed>) = 0

[pid 5012] <... mmap resumed>) = 0x7f31fef24000

[pid 5013] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5012] mprotect(0x7f31fef25000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

[pid 5013] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5012] <... mprotect resumed>) = 0

[pid 5013] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 5012] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 5013] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5012] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 5013] madvise(0x7f31ff725000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 5012] **clone3**({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f31ff724990, parent\_tid=0x7f31ff724990, exit\_signal=0, stack=0x7f31fef24000, stack\_size=0x7fff80, tls=0x7f31ff7246c0} <unfinished ...>

[pid 5013] <... madvise resumed>) = 0

[pid 5013] exit(0strace: Process 5014 attached

<unfinished ...>

[pid 5012] <... clone3 resumed> => {parent\_tid=[5014]}, 88) = 5014

[pid 5013] <... exit resumed>) = ?

[pid 5012] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5014] rseq(0x7f31ff724fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 5012] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5013] +++ exited with 0 +++

[pid 5014] <... rseq resumed>) = 0

[pid 5012] **futex**(0x7f31ff724990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 5014, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 5014] set\_robust\_list(0x7f31ff7249a0, 24) = 0

[pid 5014] rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

[pid 5014] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

[pid 5014] madvise(0x7f31fef24000, 8368128, MADV\_DONTNEED) = 0

[pid 5014] exit(0) = ?

[pid 5012] <... futex resumed>) = 0

[pid 5014] +++ exited with 0 +++

rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

**clone3**({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f31ff724990, parent\_tid=0x7f31ff724990, exit\_signal=0, stack=0x7f31fef24000, stack\_size=0x7fff80, tls=0x7f31ff7246c0}strace: Process 5015 attached

=> {parent\_tid=[5015]}, 88) = 5015

[pid 5015] rseq(0x7f31ff724fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 5012] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5015] <... rseq resumed>) = 0

[pid 5012] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5015] set\_robust\_list(0x7f31ff7249a0, 24 <unfinished ...>

[pid 5012] rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

[pid 5015] <... set\_robust\_list resumed>) = 0

[pid 5012] <... rt\_sigprocmask resumed>[], 8) = 0

[pid 5015] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5012] **clone3**({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7f31fff25990, parent\_tid=0x7f31fff25990, exit\_signal=0, stack=0x7f31ff725000, stack\_size=0x7fff80, tls=0x7f31fff256c0} <unfinished ...>

[pid 5015] <... rt\_sigprocmask resumed>NULL, 8) = 0

strace: Process 5016 attached

[pid 5015] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 5012] <... clone3 resumed> => {parent\_tid=[5016]}, 88) = 5016

[pid 5016] rseq(0x7f31fff25fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 5012] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5015] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5012] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5016] <... rseq resumed>) = 0

[pid 5012] **futex**(0x7f31ff724990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 5015, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 5015] madvise(0x7f31fef24000, 8368128, MADV\_DONTNEED <unfinished ...>

[pid 5016] set\_robust\_list(0x7f31fff259a0, 24 <unfinished ...>

[pid 5015] <... madvise resumed>) = 0

[pid 5016] <... set\_robust\_list resumed>) = 0

[pid 5015] exit(0 <unfinished ...>

[pid 5016] rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

[pid 5015] <... exit resumed>) = ?

[pid 5016] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5015] +++ exited with 0 +++

[pid 5012] <... **futex** resumed>) = 0

[pid 5016] rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

[pid 5012] **futex**(0x7f31fff25990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 5016, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

[pid 5016] <... rt\_sigprocmask resumed>NULL, 8) = 0

[pid 5016] madvise(0x7f31ff725000, 8368128, MADV\_DONTNEED) = 0

[pid 5016] **exit**(0) = ?

[pid 5012] <... **futex resumed**>) = 0

[pid 5016] +++ exited with 0 +++

**write**(4, "3 3\n1.2 2.7 3.3\n4.1 5.4 6.8\n7.2 "..., 40) = 40

**close**(4) = 0

**close**(3) = 0

exit\_group(0) = ?

+++ exited with 0 +++

**Вывод**

Матрица 100 на 100.

|  |  |  |  |
| --- | --- | --- | --- |
| **Число потоков** | **Время исполнения (мс)** | **Ускорение** | **Эффективность** |
| **1** | 1892 | 1 | 1 |
| **2** | 1298 | 0.68 | 0.34 |
| **3** | 1205 | 0.63 | 0.21 |
| **4** | 1062 | 0.56 | 0.14 |
| **5** | 1092 | 0.57 | 0.114 |
| **6** | 1032 | 0.54 | 0.09 |

Матрица 1000 на 1000.

|  |  |  |  |
| --- | --- | --- | --- |
| **Число потоков** | **Время исполнения (мс)** | **Ускорение** | **Эффективность** |
| **1** | 22663 | 1 | 1 |
| **2** | 14223 | 0.62 | 0.313 |
| **3** | 11150 | 0.49 | 0.16 |
| **4** | 9954 | 0.43 | 0.1075 |
| **5** | 8860 | 0.39 | 0.078 |
| **6** | 8265 | 0.367 | 0.06111 |

**Ускорение** показывает во сколько раз применение параллельного алгоритма уменьшает время решения задачи по сравнению с последовательным алгоритмом. Ускорение определяется величиной SN=T1/TN, где Т1 - время выполнения на одном потоке, TN - время выполнения на N потоках.

**Эффективность** - величина EN = SN/N, где SN - ускорение, N - количество используемых потоков.

**Вывод**

Лабораторная работа была выполнена, поставленная задача решена. Мне удалось написать программу, выполняющую действие с использованием множества потоков. Все потоки удалось синхронизировать. Работу было очень увлекательно выполнять. Было трудно заставить компилятор запускать программу. Исходя из результатов тестирования программы с разным числом данных и с разным количеством потоков можно прийти к выводу, что при грамотном использование потоков существенно ускоряется работа программы. Для того, чтобы ускорение было максимальным требуется выбирать количество потоков исходя из вычислительной способности компьютера и объема задачи которая будет возложена на каждый из процессов.