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: Верменников М.В.

Преподаватель: Миронов Е.С.

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

Дата: 17.10.24

Москва, 2024

**Постановка задачи**

**Вариант 19.**

Дан массив координат (x, y). Пользователь вводит число кластеров. Проведите кластеризацию методом k-средних.

**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 \*(\*start\_routine)(void \*), void \*arg) — создание нового потока.
* exit(int status) — завершение выполнения процесса.
* close(int fd) — закрытие файла.
* pthread\_join(pthread\_t thread, void \*\*retval) — ожидание завершения потока.

Для начала разберемся, что представляет собой метод кластеризации k-средних. Данный алгоритм позволяет разделить множество точек на K кластеров, минимизируя суммарное расстояние между точками и соответствующими им центроидами кластеров.

Для удобного тестирования добавим в программу ключ запуска, который может иметь значение либо 'f', либо 'i'. Где 'f' означает, что программа использует данные из файла input.txt, а флаг 'i' означает, что мы будем использовать ввод через консоль.

Первая часть программы выполняется в главном потоке и отвечает за ввод массива точек и создание необходимых переменных.

Алгоритм k-средних включает следующие шаги:

1. Инициализация центроидов
2. Присвоение точек кластерам
3. Обновление центроидов
4. Проверка сходимости

Используем многопоточность для параллельной обработки точек при присвоении их кластерам. Разделим массив точек на равные части, количество которых будет равно количеству доступных потоков. Не забудем, что максимальное число потоков для обработки массива равно N — количеству точек.

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 завершение шага присвоения точек кластерам. После того как все потоки закончат присвоение, главный поток обновит центроиды.

Для оптимизации памяти мы не будем создавать множество дополнительных структур данных. Вместо этого каждый поток будет хранить локальные суммы координат и количество точек для каждого кластера. После завершения работы потоков эти данные будут объединены для обновления центроидов.

Каждая функция потока будет принимать начало и конец интервала точек, которые она должна обработать. Для удобства создадим функцию, вычисляющую ближайший центроид для заданной точки, которая принимает координаты точки, массив текущих центроидов и возвращает индекс ближайшего кластера.

**Код программы**

#include <iostream>

#include <fstream>

#include <vector>

#include <cstdlib> // для rand(), srand()

#include <ctime>   // для time()

#include <cmath>   // для sqrt(), pow()

#include <pthread.h>

#include <set>

#include <limits>

#include <unistd.h> //для sleep(10)

using namespace std;

// Структура для передачи данных в потоки

struct ThreadData

{

    int thread\_id;                          // Идентификатор потока

    int start\_idx;                          // Начальный индекс для обработки

    int end\_idx;                            // Конечный индекс для обработки

    const vector<pair<double, double>> \*points;     // Указатель на вектор точек

    const vector<pair<double, double>> \*centroids;  // Указатель на вектор центроидов

    vector<int> \*labels;                    // Указатель на вектор меток кластеров

    vector<vector<double>> local\_sums;      // Локальные суммы координат для каждого кластера

    vector<int> local\_counts;               // Локальное количество точек в каждом кластере

};

// Функция потока для шага присваивания

void \*assignment\_thread\_func(void \*arg)

{

    ThreadData \*data = (ThreadData \*)arg;

    int start = data->start\_idx;

    int end = data->end\_idx;

    const vector<pair<double, double>> &points = \*(data->points);

    const vector<pair<double, double>> &centroids = \*(data->centroids);

    vector<int> &labels = \*(data->labels);

    int K = centroids.size();

    // Инициализация локальных сумм и счетчиков

    data->local\_sums.assign(K, vector<double>(2, 0.0)); // sum\_x, sum\_y

    data->local\_counts.assign(K, 0);

    // Присваивание точек ближайшим центроидам

    for (int i = start; i < end; ++i)

    {

        double min\_dist = std::numeric\_limits<double>::max();

        int min\_idx = -1;

        for (int k = 0; k < K; ++k)

        {

            double dx = points[i].first - centroids[k].first;

            double dy = points[i].second - centroids[k].second;

            double dist = dx \* dx + dy \* dy; // Квадрат расстояния

            if (dist < min\_dist)

            {

                min\_dist = dist;

                min\_idx = k;

            }

        }

        labels[i] = min\_idx;

        data->local\_sums[min\_idx][0] += points[i].first;

        data->local\_sums[min\_idx][1] += points[i].second;

        data->local\_counts[min\_idx]++;

    }

    sleep(10); //задерживаемся на 10 секунд, чтобы я поймал потоки

    pthread\_exit(NULL);

}

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

{

    // Разбор аргументов командной строки

    if (argc < 3)

    {

        cout << "Использование: " << argv[0] << " max\_threads mode(f/i)" << endl;

        return 1;

    }

    int max\_threads = atoi(argv[1]);

    char mode = argv[2][0];

    // Настройка потоков ввода и вывода

    ostream \*out;

    istream \*in;

    ofstream outf;

    ifstream inf;

    if (mode == 'f')

    {

        inf.open("input.txt");

        outf.open("output.txt");

        in = &inf;

        out = &outf;

    }

    else if (mode == 'i')

    {

        in = &cin;

        out = &cout;

    }

    else

    {

        cout << "Некорректный режим: " << mode << endl;

        return 1;

    }

    // Чтение количества кластеров K

    int K;

    (\*in) >> K;

    // Чтение количества точек N

    int N;

    (\*in) >> N;

    // Чтение N точек

    vector<pair<double, double>> points(N);

    for (int i = 0; i < N; ++i)

    {

        double x, y;

        (\*in) >> x >> y;

        points[i] = make\_pair(x, y);

    }

    // Инициализация центроидов случайным выбором K точек

    srand(time(NULL));

    set<int> selected\_indices;

    vector<pair<double, double>> centroids(K);

    for (int i = 0; i < K; ++i)

    {

        int idx;

        do

        {

            idx = rand() % N;

        } while (selected\_indices.find(idx) != selected\_indices.end());

        selected\_indices.insert(idx);

        centroids[i] = points[idx];

    }

    // Алгоритм k-средних

    bool converged = false;

    int max\_iterations = 100;

    int iteration = 0;

    vector<int> labels(N, -1); // Метки кластеров для каждой точки

    vector<vector<double>> new\_centroids\_sums(K, vector<double>(2, 0.0));

    vector<int> new\_centroids\_counts(K, 0);

    while (!converged && iteration < max\_iterations)

    {

        // Шаг присваивания

        int num\_threads = min(max\_threads, N);

        vector<ThreadData> thread\_data(num\_threads);

        vector<pthread\_t> threads(num\_threads);

        int points\_per\_thread = N / num\_threads;

        for (int t = 0; t < num\_threads; ++t)

        {

            thread\_data[t].thread\_id = t;

            thread\_data[t].start\_idx = t \* points\_per\_thread;

            if (t == num\_threads - 1)

                thread\_data[t].end\_idx = N;

            else

                thread\_data[t].end\_idx = (t + 1) \* points\_per\_thread;

            thread\_data[t].points = &points;

            thread\_data[t].centroids = &centroids;

            thread\_data[t].labels = &labels;

            int rc = pthread\_create(&threads[t], NULL, assignment\_thread\_func, (void \*)&thread\_data[t]);

            if (rc)

            {

                cout << "Ошибка: невозможно создать поток," << rc << endl;

                exit(-1);

            }

        }

        // Ожидание завершения всех потоков

        for (int t = 0; t < num\_threads; ++t)

        {

            pthread\_join(threads[t], NULL);

        }

        // Шаг обновления центроидов

        // Сбор локальных сумм и счетчиков из потоков

        for (int k = 0; k < K; ++k)

        {

            new\_centroids\_sums[k][0] = 0.0;

            new\_centroids\_sums[k][1] = 0.0;

            new\_centroids\_counts[k] = 0;

        }

        for (int t = 0; t < num\_threads; ++t)

        {

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

            {

                new\_centroids\_sums[k][0] += thread\_data[t].local\_sums[k][0];

                new\_centroids\_sums[k][1] += thread\_data[t].local\_sums[k][1];

                new\_centroids\_counts[k] += thread\_data[t].local\_counts[k];

            }

        }

        converged = true;

        // Вычисление новых центроидов и проверка на сходимость

        for (int k = 0; k < K; ++k)

        {

            if (new\_centroids\_counts[k] > 0)

            {

                double new\_x = new\_centroids\_sums[k][0] / new\_centroids\_counts[k];

                double new\_y = new\_centroids\_sums[k][1] / new\_centroids\_counts[k];

                // Проверка, изменился ли центроид существенно

                double dx = centroids[k].first - new\_x;

                double dy = centroids[k].second - new\_y;

                if (sqrt(dx \* dx + dy \* dy) > 1e-4)

                {

                    converged = false;

                }

                centroids[k].first = new\_x;

                centroids[k].second = new\_y;

            }

            else

            {

                // В кластере нет точек, инициализируем центроид случайно

                int idx = rand() % N;

                centroids[k] = points[idx];

                converged = false;

            }

        }

        iteration++;

    }

    // Вывод финальных центроидов

    (\*out) << "Финальные центроиды:\n";

    for (int k = 0; k < K; ++k)

    {

        (\*out) << centroids[k].first << " " << centroids[k].second << "\n";

    }

    // Вывод меток кластеров

    (\*out) << "Метки кластеров:\n";

    for (int i = 0; i < N; ++i)

    {

        (\*out) << labels[i] << "\n";

    }

    if (mode == 'f')

    {

        inf.close();

        outf.close();

    }

    return 0;

}

}

**Протокол работы программы**

**Тестирование:**

root@f5b21642cf59:/workspaces/MAI\_OS\_Labs/Lab2/src# ./kmeans 3 i

3

9

1 1

2 1

1 2

8 8

9 8

8 9

5 5

6 5

5 6

Out:

Финальные центроиды:

1 1.5

2 1

6.83333 6.83333

Метки кластеров:

0

1

0

2

2

2

2

2

2

Демонстрация количества потоков:

root@7835a273c0d7:/workspaces/MAI\_OS\_Labs/Lab2/src# ps -eLf | grep kmeans

root 5383 3540 5383 0 4 23:54 pts/0 00:00:00 ./kmeans 3 f

root 5383 3540 5384 0 4 23:54 pts/0 00:00:00 ./kmeans 3 f

root 5383 3540 5385 0 4 23:54 pts/0 00:00:00 ./kmeans 3 f

root 5383 3540 5386 0 4 23:54 pts/0 00:00:00 ./kmeans 3 f

root 5462 3828 5462 0 1 23:54 pts/1 00:00:00 grep kmeans

**Strace:**

6088 execve("./kmeans", ["./kmeans", "3", "f"], 0x7ffd1af95c58 /\* 26 vars \*/) = 0

6088 brk(NULL) = 0x1a2f000

6088 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd42af9b000

6088 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

**6088 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3**

6088 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=25258, ...}, AT\_EMPTY\_PATH) = 0

6088 mmap(NULL, 25258, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7fd42af94000

**6088 close(3) = 0**

**6088 openat(AT\_FDCWD, "/usr/local/lib64/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3**

6088 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

6088 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2530008, ...}, AT\_EMPTY\_PATH) = 0

6088 mmap(NULL, 2543808, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fd42ad26000

6088 mmap(0x7fd42adcb000, 1216512, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xa5000) = 0x7fd42adcb000

6088 mmap(0x7fd42aef4000, 581632, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7fd42aef4000

6088 mmap(0x7fd42af82000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x25c000) = 0x7fd42af82000

6088 mmap(0x7fd42af90000, 12480, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fd42af90000

**6088 close(3) = 0**

**6088 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3**

6088 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

6088 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=907784, ...}, AT\_EMPTY\_PATH) = 0

6088 mmap(NULL, 909560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fd42ac47000

6088 mmap(0x7fd42ac57000, 471040, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x10000) = 0x7fd42ac570006088 mmap(0x7fd42acca000, 368640, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x83000) = 0x7fd42acca000

6088 mmap(0x7fd42ad24000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xdc000) = 0x7fd42ad24000

**6088 close(3) = 0**

**6088 openat(AT\_FDCWD, "/usr/local/lib64/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3**

6088 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

6088 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=906528, ...}, AT\_EMPTY\_PATH) = 0

6088 mmap(NULL, 181160, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fd42ac1a000

6088 mmap(0x7fd42ac1e000, 143360, PROT\_READ|PROT\_EopenatXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x4000) = 0x7fd42ac1e000

6088 mmap(0x7fd42ac41000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x27000) = 0x7fd42ac41000

6088 mmap(0x7fd42ac45000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x2b000) = 0x7fd42ac45000

**6088 close(3) = 0**

**6088 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3**

6088 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

6088 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

6088 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=1922136, ...}, AT\_EMPTY\_PATH) = 0

6088 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

6088 mmap(NULL, 1970000, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7fd42aa39000

6088 mmap(0x7fd42aa5f000, 1396736, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x26000) = 0x7fd42aa5f000

6088 mmap(0x7fd42abb4000, 339968, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x17b000) = 0x7fd42abb4000

6088 mmap(0x7fd42ac07000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ce000) = 0x7fd42ac07000

6088 mmap(0x7fd42ac0d000, 53072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7fd42ac0d000

**6088 close(3) = 0**

6088 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd42aa37000

6088 arch\_prctl(ARCH\_SET\_FS, 0x7fd42aa38480) = 0

6088 set\_tid\_address(0x7fd42aa38750) = 6088

6088 set\_robust\_list(0x7fd42aa38760, 24) = 0

6088 rseq(0x7fd42aa38da0, 0x20, 0, 0x53053053) = 0

6088 mprotect(0x7fd42ac07000, 16384, PROT\_READ) = 0

6088 mprotect(0x7fd42ac45000, 4096, PROT\_READ) = 0

6088 mprotect(0x7fd42ad24000, 4096, PROT\_READ) = 0

6088 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7fd42aa35000

6088 mprotect(0x7fd42af82000, 45056, PROT\_READ) = 0

6088 mprotect(0x40d000, 4096, PROT\_READ) = 0

6088 mprotect(0x7fd42afcd000, 8192, PROT\_READ) = 0

6088 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=RLIM64\_INFINITY}) = 0

6088 munmap(0x7fd42af94000, 25258) = 0

**6088 futex(0x7fd42af9073c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0**

6088 getrandom("\x0d\xc5\xb3\xd6\x84\x8c\xc6\xa2", 8, GRND\_NONBLOCK) = 8

6088 brk(NULL) = 0x1a2f000

6088 brk(0x1a50000) = 0x1a50000

**6088 openat(AT\_FDCWD, "input.txt", O\_RDONLY) = 3**

**6088 openat(AT\_FDCWD, "output.txt", O\_WRONLY|O\_CREAT|O\_TRUNC, 0666) = 4**

6088 read(3, "3\n9\n1 1\n2 1\n1 2\n8 8\n9 8\n8 9\n5 5\n"..., 8191) = 40

6088 rt\_sigaction(SIGRT\_1, {sa\_handler=0x7fd42aabf6a0, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7fd42aa75050}, NULL, 8) = 0

6088 rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

6088 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7fd42a234000

6088 mprotect(0x7fd42a235000, 8388608, PROT\_READ|PROT\_WRITE) = 0

6088 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

**6088 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fd42aa34990, parent\_tid=0x7fd42aa34990, exit\_signal=0, stack=0x7fd42a234000, stack\_size=0x7fff80, tls=0x7fd42aa346c0} => {parent\_tid=[6089]}, 88) = 6089**

6089 rseq(0x7fd42aa34fe0, 0x20, 0, 0x53053053 <unfinished ...>

6088 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6089 <... rseq resumed>) = 0

6088 <... rt\_sigprocmask resumed>NULL, 8) = 0

6089 set\_robust\_list(0x7fd42aa349a0, 24 <unfinished ...>

6088 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

6089 <... set\_robust\_list resumed>) = 0

6088 <... mmap resumed>) = 0x7fd429a33000

6089 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6088 mprotect(0x7fd429a34000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

6089 <... rt\_sigprocmask resumed>NULL, 8) = 0

6088 <... mprotect resumed>) = 0

6089 mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0 <unfinished ...>

6088 rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

6089 <... mmap resumed>) = 0x7fd421a33000

6088 <... rt\_sigprocmask resumed>[], 8) = 0

6089 munmap(0x7fd421a33000, 39636992 <unfinished ...>

**6088 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fd42a233990, parent\_tid=0x7fd42a233990, exit\_signal=0, stack=0x7fd429a33000, stack\_size=0x7fff80, tls=0x7fd42a2336c0} <unfinished ...>**

6089 <... munmap resumed>) = 0

**6088 <... clone3 resumed> => {parent\_tid=[6090]}, 88) = 6090**

6089 munmap(0x7fd428000000, 27471872 <unfinished ...>

6088 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6090 rseq(0x7fd42a233fe0, 0x20, 0, 0x53053053 <unfinished ...>

6088 <... rt\_sigprocmask resumed>NULL, 8) = 0

6089 <... munmap resumed>) = 0

6088 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0 <unfinished ...>

6090 <... rseq resumed>) = 0

6088 <... mmap resumed>) = 0x7fd429232000

6089 mprotect(0x7fd424000000, 135168, PROT\_READ|PROT\_WRITE <unfinished ...>

6088 mprotect(0x7fd429233000, 8388608, PROT\_READ|PROT\_WRITE <unfinished ...>

6090 set\_robust\_list(0x7fd42a2339a0, 24 <unfinished ...>

6088 <... mprotect resumed>) = 0

6089 <... mprotect resumed>) = 0

6088 rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

6090 <... set\_robust\_list resumed>) = 0

6088 <... rt\_sigprocmask resumed>[], 8) = 0

6089 clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=10, tv\_nsec=0}, <unfinished ...>

**6088 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fd429a32990, parent\_tid=0x7fd429a32990, exit\_signal=0, stack=0x7fd429232000, stack\_size=0x7fff80, tls=0x7fd429a326c0} <unfinished ...>**

6090 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

**6088 <... clone3 resumed> => {parent\_tid=[6091]}, 88) = 6091**

6090 <... rt\_sigprocmask resumed>NULL, 8) = 0

6088 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6091 rseq(0x7fd429a32fe0, 0x20, 0, 0x53053053 <unfinished ...>

6088 <... rt\_sigprocmask resumed>NULL, 8) = 0

6090 mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0 <unfinished ...>

**6088 futex(0x7fd42aa34990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 6089, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>**

6091 <... rseq resumed>) = 0

6090 <... mmap resumed>) = 0x7fd41c000000

6091 set\_robust\_list(0x7fd429a329a0, 24 <unfinished ...>

6090 munmap(0x7fd420000000, 67108864 <unfinished ...>

6091 <... set\_robust\_list resumed>) = 0

6090 <... munmap resumed>) = 0

6091 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6090 mprotect(0x7fd41c000000, 135168, PROT\_READ|PROT\_WRITE <unfinished ...>

6091 <... rt\_sigprocmask resumed>NULL, 8) = 0

6090 <... mprotect resumed>) = 0

6091 mmap(0x7fd420000000, 67108864, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0 <unfinished ...>

6090 clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=10, tv\_nsec=0}, <unfinished ...>

6091 <... mmap resumed>) = 0x7fd420000000

6091 mprotect(0x7fd420000000, 135168, PROT\_READ|PROT\_WRITE) = 0

6091 clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=10, tv\_nsec=0}, <unfinished ...>

6089 <... clock\_nanosleep resumed>0x7fd42aa33dd0) = 0

**6089 futex(0x7fd42ac46230, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0**

6089 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

6090 <... clock\_nanosleep resumed>0x7fd42a232dd0) = 0

6091 <... clock\_nanosleep resumed>0x7fd429a31dd0) = 0

6089 madvise(0x7fd42a234000, 8368128, MADV\_DONTNEED <unfinished ...>

6091 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

6090 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

6091 <... rt\_sigprocmask resumed>NULL, 8) = 0

6089 <... madvise resumed>) = 0

6091 madvise(0x7fd429232000, 8368128, MADV\_DONTNEED <unfinished ...>

6090 <... rt\_sigprocmask resumed>NULL, 8) = 0

6091 <... madvise resumed>) = 0

6089 exit(0 <unfinished ...>

6091 exit(0 <unfinished ...>

6090 madvise(0x7fd429a33000, 8368128, MADV\_DONTNEED <unfinished ...>

6091 <... exit resumed>) = ?

6089 <... exit resumed>) = ?

6091 +++ exited with 0 +++

6090 <... madvise resumed>) = 0

**6088 <... futex resumed>) = 0**

6089 +++ exited with 0 +++

**6088 futex(0x7fd42a233990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 6090, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>**

**6090 exit(0) = ?**

**6088 <... futex resumed>) = 0**

6090 +++ exited with 0 +++

6088 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

**6088 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fd429a32990, parent\_tid=0x7fd429a32990, exit\_signal=0, stack=0x7fd429232000, stack\_size=0x7fff80, tls=0x7fd429a326c0} => {parent\_tid=[6153]}, 88) = 6153**

6153 rseq(0x7fd429a32fe0, 0x20, 0, 0x53053053 <unfinished ...>

6088 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6153 <... rseq resumed>) = 0

6088 <... rt\_sigprocmask resumed>NULL, 8) = 0

6153 set\_robust\_list(0x7fd429a329a0, 24 <unfinished ...>

6088 rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

6153 <... set\_robust\_list resumed>) = 0

6088 <... rt\_sigprocmask resumed>[], 8) = 0

6153 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

**6088 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fd42a233990, parent\_tid=0x7fd42a233990, exit\_signal=0, stack=0x7fd429a33000, stack\_size=0x7fff80, tls=0x7fd42a2336c0} <unfinished ...>**

6153 <... rt\_sigprocmask resumed>NULL, 8) = 0

**6088 <... clone3 resumed> => {parent\_tid=[6154]}, 88) = 6154**

6154 rseq(0x7fd42a233fe0, 0x20, 0, 0x53053053 <unfinished ...>

6153 clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=10, tv\_nsec=0}, <unfinished ...>

6088 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6154 <... rseq resumed>) = 0

6088 <... rt\_sigprocmask resumed>NULL, 8) = 0

6154 set\_robust\_list(0x7fd42a2339a0, 24 <unfinished ...>

6088 rt\_sigprocmask(SIG\_BLOCK, ~[], <unfinished ...>

6154 <... set\_robust\_list resumed>) = 0

6088 <... rt\_sigprocmask resumed>[], 8) = 0

6154 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

**6088 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7fd42aa34990, parent\_tid=0x7fd42aa34990, exit\_signal=0, stack=0x7fd42a234000, stack\_size=0x7fff80, tls=0x7fd42aa346c0} <unfinished ...>**

6154 <... rt\_sigprocmask resumed>NULL, 8) = 0

**6088 <... clone3 resumed> => {parent\_tid=[6155]}, 88) = 6155**

6155 rseq(0x7fd42aa34fe0, 0x20, 0, 0x53053053 <unfinished ...>

6154 clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=10, tv\_nsec=0}, <unfinished ...>

6088 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

6155 <... rseq resumed>) = 0

6088 <... rt\_sigprocmask resumed>NULL, 8) = 0

6155 set\_robust\_list(0x7fd42aa349a0, 24 <unfinished ...>

**6088 futex(0x7fd429a32990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 6153, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>**

6155 <... set\_robust\_list resumed>) = 0

6155 rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

6155 clock\_nanosleep(CLOCK\_REALTIME, 0, {tv\_sec=10, tv\_nsec=0}, <unfinished ...>

6153 <... clock\_nanosleep resumed>0x7fd429a31dd0) = 0

6153 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

6153 madvise(0x7fd429232000, 8368128, MADV\_DONTNEED) = 0

6154 <... clock\_nanosleep resumed>0x7fd42a232dd0) = 0

6153 exit(0 <unfinished ...>

6154 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

6153 <... exit resumed>) = ?

6155 <... clock\_nanosleep resumed>0x7fd42aa33dd0) = 0

6154 <... rt\_sigprocmask resumed>NULL, 8) = 0

**6088 <... futex resumed>) = 0**

6155 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], <unfinished ...>

6153 +++ exited with 0 +++

**6088 futex(0x7fd42a233990, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 6154, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>**

6155 <... rt\_sigprocmask resumed>NULL, 8) = 0

6154 madvise(0x7fd429a33000, 8368128, MADV\_DONTNEED <unfinished ...>

6155 madvise(0x7fd42a234000, 8368128, MADV\_DONTNEED <unfinished ...>

6154 <... madvise resumed>) = 0

6155 <... madvise resumed>) = 0

6154 exit(0 <unfinished ...>

6155 exit(0) = ?

6154 <... exit resumed>) = ?

6155 +++ exited with 0 +++

**6088 <... futex resumed>) = 0**

6154 +++ exited with 0 +++

**6088 close(3) = 0**

**6088 write(4, "\320\244\320\270\320\275\320\260\320\273\321\214\320\275\321\213\320\265 \321\206\320\265\320\275\321\202\321\200\320\276\320"..., 114) = 114**

**6088 close(4) = 0**

6088 exit\_group(0) = ?

6088 +++ exited with 0 +++

**Вывод**

Датасет из 10.000 точек.

|  |  |  |  |
| --- | --- | --- | --- |
| **Число потоков** | **Время исполнения (мс)** | **Ускорение** | **Эффективность** |
| **1** | 2343 | 1 | 1 |
| **2** | 1205 | 1,6667 | 0,8333 |
| **3** | 956 | 2,2222 | 0,7407 |
| **4** | 742 | 2,6667 | 0,6667 |
| **5** | 700 | 2,8571 | |  | | --- | |  |  |  | | --- | | 0,5714 | |
| **6** | 654 | 3,0769 | |  | | --- | |  |  |  | | --- | | 0,5128 | |

Датасет из 100 000 точек.

|  |  |  |  |
| --- | --- | --- | --- |
| **Число потоков** | **Время исполнения (мс)** | **Ускорение** | **Эффективность** |
| **1** | 22663 | 1 | 1 |
| **2** | 11150 | 1,8182 | 0,9091 |
| **3** | 8271 | 2,5 | |  | | --- | |  |  |  | | --- | | 0,8333 | |
| **4** | 6578 | |  | | --- | |  |  |  | | --- | | 3,3333 | | |  | | --- | |  |  |  | | --- | | 0,8333 | |
| **5** | 5512 | 3,6364 | 0,511580135 |
| **6** | 5063 | 4 | |  | | --- | |  |  |  | | --- | | 0,6667 | |

**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, которая реализует кластеризацию методом k-средних с использованием многопоточности, и синхронизировать потоки для корректной работы алгоритма. Выполнение работы было увлекательным и дало мне полезные навыки работы с многопоточными вычислениями и синхронизацией. В процессе тестирования программы с различным числом точек и количеством потоков я установил, что использование потоков позволяет значительно ускорить выполнение алгоритма. Однако, чтобы добиться наибольшего ускорения, необходимо учитывать как количество доступных аппаратных ядер, так и объем обрабатываемых данных. Это подтверждает, что для максимальной эффективности следует выбирать количество потоков в соответствии с возможностями компьютера и с учетом ресурсоемкости задачи.