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: М80-206Б-22

Студент: Седов М.Д

Преподаватель: Миронов Е.С.

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

Дата: 17.11.2023

Москва, 2023

**Постановка задачи**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, 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 входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

**Вариант 19.**Дан массив координат (x, y). Пользователь вводит число кластеров. Проводится кластеризация методом k-средних.

**Общий метод и алгоритм решения**

Использованные системные вызовы:

* int **pthread\_create** (pthread\_t \* newthread, NULL, void \* func, void \* arg) - создает новый поток с номером newthread, который запускает функцию func и передает в нее аргументы arg.
* int **pthread\_join**(pthread\_t thread, NULL) - ждет завершение потока c номером thread .

Сначала считываем параметры, отвечающие за количество потоков и количество кластеров(я решил сделать реализацию кластеризации с установленным заранее количеством кластеров). Затем мы генерируем вектор points, содержащий точки, над которыми и будет производится кластеризация. Дальше мы вызываем функцию kMeansClustering(), которая возвращает вектор, содержащий информацию о том к какому кластеру каждая точка принадлежит.  
Перейдем к самой кластеризации, метод k-средних основан на том, что мы выбираем центры кластеров(берем их из данных точек), а дальше начинаем группировать около них другие точки. После этого разделения мы пересчитываем центры и начинаем все сначала до тех пор, пока центры не перестанут обновляться, тем самым находя нужные кластеры точек.  
Также здесь использовалась многопоточность, чтобы потоки распределялись по кластерам и запускали функцию updateCentroids(), то есть потоки обновляли только центроиды своих кластеров. Так как каждый поток работает над своими центроидами, проблем с синхронизацией данных не происходит, тем самым использование мьютекса не требуется.

**Код программы**

**main.cpp**

#include <iostream>

#include <vector>

#include <cmath>

#include <limits>

#include <pthread.h>

#include <cstdlib>

#include <random>

#include <algorithm>

struct Point {

    double x;

    double y;

};

struct Args {

    std::vector<Point>\* centroids;

    std::vector<int> clusters;

    std::vector<Point> points;

    int start;

    int end;

};

double calculateDistance(Point point1, Point point2) {

    double x = point2.x - point1.x;

    double y = point2.y - point1.y;

    return std::sqrt(x \* x + y \* y);

}

int findNearestCluster(std::vector<Point>& centroids, Point point) {

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

    int near\_cluster = -1;

    for (size\_t i = 0; i < centroids.size(); ++i) {

        double distance = calculateDistance(centroids[i], point);

        if (distance < min\_distance) {

            min\_distance = distance;

            near\_cluster = i;

        }

    }

    return near\_cluster;

}

void\* updateCentroids(void\* args\_void) {

    auto args = static\_cast<Args\*>(args\_void);

    std::vector<Point>& centroids = \*(args->centroids);

    std::vector<Point>& points = args->points;

    std::vector<int>& clusters = args->clusters;

    std::vector<int> count(centroids.size(), 0);

    std::vector<double> sumX(centroids.size(), 0.0);

    std::vector<double> sumY(centroids.size(), 0.0);

    for (size\_t i = 0; i < points.size(); ++i) {

        int cluster = clusters[i];

        count[cluster]++;

        sumX[cluster] += points[i].x;

        sumY[cluster] += points[i].y;

    }

    for (int i = args->start; i < args->end; ++i) {

        if (count[i] > 0) {

            centroids[i].x = sumX[i] / count[i];

            centroids[i].y = sumY[i] / count[i];

        }

    }

    return nullptr;

}

std::vector<int> kMeansClustering(std::vector<Point>& points, std::vector<Point>& init\_centroids, int thread\_num) {

    std::vector<Point> centroids = init\_centroids;

    std::vector<int> clusters(points.size(), -1);

    bool update\_centroid = true;

    while (update\_centroid) {

        update\_centroid = false;

        for (size\_t i = 0; i < points.size(); ++i) {

            int near\_cluster = findNearestCluster(centroids, points[i]);

            if (clusters[i] != near\_cluster) {

                clusters[i] = near\_cluster;

                update\_centroid = true;

            }

        }

        if (!update\_centroid) {

            break;

        }

        pthread\_t threads[thread\_num];

        std::vector<Args\*> args(thread\_num);

        int k = centroids.size() / thread\_num;

        int m = centroids.size() % thread\_num;

        int start = 0;

        int end = k;

        for (int i = 0; i < thread\_num; ++i) {

            if (i < m) {

                end++;

            }

            args[i] = new Args{&centroids, clusters, points, start, end};

            pthread\_create(&threads[i], NULL, updateCentroids, static\_cast<void\*>(args[i]));

            start = end;

            end += k;

        }

        for (int i = 0; i < thread\_num; ++i) {

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

            delete args[i];

        }

    }

    return clusters;

}

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

    int thread\_num = 1;

    int k = 2;

    if (argc > 2) {

        thread\_num = std::atoi(argv[1]);

        k = std::atoi(argv[2]);

    }

    if (thread\_num < 1) {

        throw std::logic\_error("Num of threads must be positive");

    }

    if (k < 1) {

        throw std::logic\_error("Num of cluster must be positive");

    }

    std::vector<Point> points;

    std::vector<Point> init\_centroids(k);

    std::default\_random\_engine rng(9999);

    std::uniform\_real\_distribution<double> dist(-1111111.1, 1111111.1);

    for (int i = 0; i < 10000000; ++i) {

        double x = dist(rng);

        double y = dist(rng);

        points.emplace\_back(Point{x, y});

    }

    std::sample(points.begin(), points.end(), init\_centroids.begin(), k, rng);

    std::vector<int> clusters = kMeansClustering(points, init\_centroids, thread\_num);

    // for (int cluster = 0; cluster < k; ++cluster) {

    //     std::cout << "Cluster " << cluster + 1 << ":" << std::endl;

    //     for (size\_t i = 0; i < points.size(); ++i) {

    //         if (clusters[i] == cluster) {

    //             std::cout << "Point (" << points[i].x << ", " << points[i].y << ")" << std::endl;

    //         }

    //     }

    //     std::cout << std::endl;

    // }

    return 0;

}

**Протокол работы программы**

**Тестирование:**

maksim@caseofpeace:~/course2/sem3/os/lab2/src$ time ./main 1 6

real 1m52.407s

user 1m37.875s

sys 0m11.016s

maksim@caseofpeace:~/course2/sem3/os/lab2/src$ time ./main 2 6

real 2m3.624s

user 1m50.297s

sys 0m22.313s

maksim@caseofpeace:~/course2/sem3/os/lab2/src$ time ./main 4 6

real 2m27.958s

user 2m13.703s

sys 0m44.031s

maksim@caseofpeace:~/course2/sem3/os/lab2/src$ time ./main 6 6

real 2m54.866s

user 2m37.266s

sys 1m6.156s

**STRACE:**

10216 execve("./main", ["./main", "4", "6"], 0x7fffe72275b8 /\* 31 vars \*/) = 0

10216 brk(NULL) = 0x7fffcb479000

10216 arch\_prctl(0x3001 /\* ARCH\_??? \*/, 0x7fffd3f9a4f0) = -1 EINVAL (Invalid argument)

10216 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d6040000

10216 access("/etc/ld.so.preload", R\_OK) = -1 ENOENT (No such file or directory)

10216 openat(AT\_FDCWD, "/etc/ld.so.cache", O\_RDONLY|O\_CLOEXEC) = 3

10216 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=22607, ...}, AT\_EMPTY\_PATH) = 0

10216 mmap(NULL, 22607, PROT\_READ, MAP\_PRIVATE, 3, 0) = 0x7ff9d5ffa000

10216 close(3) = 0

10216 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libstdc++.so.6", O\_RDONLY|O\_CLOEXEC) = 3

10216 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

10216 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=2260296, ...}, AT\_EMPTY\_PATH) = 0

10216 mmap(NULL, 2275520, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff9d5dc0000

10216 mprotect(0x7ff9d5e5a000, 1576960, PROT\_NONE) = 0

10216 mmap(0x7ff9d5e5a000, 1118208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x9a000) = 0x7ff9d5e5a000

10216 mmap(0x7ff9d5f6b000, 454656, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1ab000) = 0x7ff9d5f6b000

10216 mmap(0x7ff9d5fdb000, 57344, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x21a000) = 0x7ff9d5fdb000

10216 mmap(0x7ff9d5fe9000, 10432, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5fe9000

10216 close(3) = 0

10216 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libm.so.6", O\_RDONLY|O\_CLOEXEC) = 3

10216 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

10216 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=940560, ...}, AT\_EMPTY\_PATH) = 0

10216 mmap(NULL, 942344, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff9d5cd0000

10216 mmap(0x7ff9d5cde000, 507904, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe000) = 0x7ff9d5cde000

10216 mmap(0x7ff9d5d5a000, 372736, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x8a000) = 0x7ff9d5d5a000

10216 mmap(0x7ff9d5db5000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0xe4000) = 0x7ff9d5db5000

10216 close(3) = 0

10216 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libgcc\_s.so.1", O\_RDONLY|O\_CLOEXEC) = 3

10216 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

10216 newfstatat(3, "", {st\_mode=S\_IFREG|0644, st\_size=125488, ...}, AT\_EMPTY\_PATH) = 0

10216 mmap(NULL, 127720, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff9d5cb0000

10216 mmap(0x7ff9d5cb3000, 94208, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x3000) = 0x7ff9d5cb3000

10216 mmap(0x7ff9d5cca000, 16384, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1a000) = 0x7ff9d5cca000

10216 mmap(0x7ff9d5cce000, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1d000) = 0x7ff9d5cce000

10216 close(3) = 0

10216 openat(AT\_FDCWD, "/lib/x86\_64-linux-gnu/libc.so.6", O\_RDONLY|O\_CLOEXEC) = 3

10216 read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\237\2\0\0\0\0\0"..., 832) = 832

10216 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

10216 pread64(3, "\4\0\0\0 \0\0\0\5\0\0\0GNU\0\2\0\0\300\4\0\0\0\3\0\0\0\0\0\0\0"..., 48, 848) = 48

10216 pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0\244;\374\204(\337f#\315I\214\234\f\256\271\32"..., 68, 896) = 68

10216 newfstatat(3, "", {st\_mode=S\_IFREG|0755, st\_size=2216304, ...}, AT\_EMPTY\_PATH) = 0

10216 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

10216 mmap(NULL, 2260560, PROT\_READ, MAP\_PRIVATE|MAP\_DENYWRITE, 3, 0) = 0x7ff9d5a80000

10216 mmap(0x7ff9d5aa8000, 1658880, PROT\_READ|PROT\_EXEC, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x28000) = 0x7ff9d5aa8000

10216 mmap(0x7ff9d5c3d000, 360448, PROT\_READ, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x1bd000) = 0x7ff9d5c3d000

10216 mmap(0x7ff9d5c95000, 24576, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_DENYWRITE, 3, 0x214000) = 0x7ff9d5c95000

10216 mmap(0x7ff9d5c9b000, 52816, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_FIXED|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5c9b000

10216 close(3) = 0

10216 mmap(NULL, 8192, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5ff0000

10216 mmap(NULL, 12288, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5a70000

10216 arch\_prctl(ARCH\_SET\_FS, 0x7ff9d5a70740) = 0

10216 set\_tid\_address(0x7ff9d5a70a10) = 10216

10216 set\_robust\_list(0x7ff9d5a70a20, 24) = 0

10216 rseq(0x7ff9d5a710e0, 0x20, 0, 0x53053053) = -1 ENOSYS (Function not implemented)

10216 mprotect(0x7ff9d5c95000, 16384, PROT\_READ) = 0

10216 mprotect(0x7ff9d5cce000, 4096, PROT\_READ) = 0

10216 mprotect(0x7ff9d5db5000, 4096, PROT\_READ) = 0

10216 mprotect(0x7ff9d5fdb000, 45056, PROT\_READ) = 0

10216 mprotect(0x7ff9d6050000, 4096, PROT\_READ) = 0

10216 mprotect(0x7ff9d6038000, 8192, PROT\_READ) = 0

10216 prlimit64(0, RLIMIT\_STACK, NULL, {rlim\_cur=8192\*1024, rlim\_max=8192\*1024}) = 0

10216 munmap(0x7ff9d5ffa000, 22607) = 0

10216 getrandom("\x68\x27\x9e\x7b\xcf\xbf\x90\xb2", 8, GRND\_NONBLOCK) = 8

10216 brk(NULL) = 0x7fffcb479000

10216 brk(0x7fffcb49a000) = 0x7fffcb49a000

10216 futex(0x7ff9d5fe977c, FUTEX\_WAKE\_PRIVATE, 2147483647) = 0

10216 brk(0x7fffcb4bb000) = 0x7fffcb4bb000

10216 mmap(NULL, 135168, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5a40000

10216 brk(0x7fffcb4ac000) = 0x7fffcb4ac000

10216 mmap(NULL, 266240, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d59f0000

10216 munmap(0x7ff9d5a40000, 135168) = 0

10216 mmap(NULL, 528384, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5960000

10216 munmap(0x7ff9d59f0000, 266240) = 0

10216 mmap(NULL, 1052672, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5850000

10216 munmap(0x7ff9d5960000, 528384) = 0

10216 mmap(NULL, 2101248, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5640000

10216 munmap(0x7ff9d5850000, 1052672) = 0

10216 mmap(NULL, 4198400, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d5230000

10216 munmap(0x7ff9d5640000, 2101248) = 0

10216 mmap(NULL, 8392704, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d4a20000

10216 munmap(0x7ff9d5230000, 4198400) = 0

10216 mmap(NULL, 16781312, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d3a10000

10216 munmap(0x7ff9d4a20000, 8392704) = 0

10216 brk(0x7fffcb87d000) = 0x7fffcb87d000

10216 brk(0x7fffcbc4d000) = 0x7fffcbc4d000

10216 mmap(NULL, 16003072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d4b20000

10216 rt\_sigaction(SIGRT\_1, {sa\_handler=0x7ff9d5b11870, sa\_mask=[], sa\_flags=SA\_RESTORER|SA\_ONSTACK|SA\_RESTART|SA\_SIGINFO, sa\_restorer=0x7ff9d5ac2520}, NULL, 8) = 0

10216 rt\_sigprocmask(SIG\_UNBLOCK, [RTMIN RT\_1], NULL, 8) = 0

10216 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7ff9d3200000

10216 mprotect(0x7ff9d3201000, 8388608, PROT\_READ|PROT\_WRITE) = 0

10216 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

10216 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7ff9d3a00910, parent\_tid=0x7ff9d3a00910, exit\_signal=0, stack=0x7ff9d3200000, stack\_size=0x7fff00, tls=0x7ff9d3a00640}, 88) = -1 ENOSYS (Function not implemented)

**10216 clone(child\_stack=0x7ff9d39ffef0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[10217], tls=0x7ff9d3a00640, child\_tidptr=0x7ff9d3a00910) = 10217**

10217 set\_robust\_list(0x7ff9d3a00920, 24 <unfinished ...>

10216 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

10217 <... set\_robust\_list resumed>) = 0

10217 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

10216 <... rt\_sigprocmask resumed>NULL, 8) = 0

10217 <... rt\_sigprocmask resumed>NULL, 8) = 0

10217 mmap(NULL, 134217728, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_NORESERVE, -1, 0 <unfinished ...>

10216 brk(0x7fffcc01e000 <unfinished ...>

10217 <... mmap resumed>) = 0x7ff9cb200000

10216 <... brk resumed>) = 0x7fffcc01e000

10217 munmap(0x7ff9cb200000, 14680064) = 0

10217 munmap(0x7ff9d0000000, 52428800) = 0

10217 mprotect(0x7ff9cc000000, 135168, PROT\_READ|PROT\_WRITE) = 0

10216 mmap(NULL, 16003072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d22b0000

10217 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

10217 madvise(0x7ff9d3200000, 8368128, MADV\_DONTNEED) = 0

10217 exit(0) = ?

10217 +++ exited with 0 +++

10216 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7ff9d1aa0000

10216 mprotect(0x7ff9d1aa1000, 8388608, PROT\_READ|PROT\_WRITE) = 0

10216 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

10216 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7ff9d22a0910, parent\_tid=0x7ff9d22a0910, exit\_signal=0, stack=0x7ff9d1aa0000, stack\_size=0x7fff00, tls=0x7ff9d22a0640}, 88) = -1 ENOSYS (Function not implemented)

**10216 clone(child\_stack=0x7ff9d229fef0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[10218], tls=0x7ff9d22a0640, child\_tidptr=0x7ff9d22a0910) = 10218**

10218 set\_robust\_list(0x7ff9d22a0920, 24 <unfinished ...>

10216 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

10218 <... set\_robust\_list resumed>) = 0

10218 rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

10216 <... rt\_sigprocmask resumed>NULL, 8) = 0

10216 brk(0x7fffcc3ee000) = 0x7fffcc3ee000

10216 mmap(NULL, 16003072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9d0b50000

10218 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

10218 madvise(0x7ff9d1aa0000, 8368128, MADV\_DONTNEED) = 0

10218 exit(0) = ?

10218 +++ exited with 0 +++

10216 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7ff9d0340000

10216 mprotect(0x7ff9d0341000, 8388608, PROT\_READ|PROT\_WRITE) = 0

10216 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

10216 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7ff9d0b40910, parent\_tid=0x7ff9d0b40910, exit\_signal=0, stack=0x7ff9d0340000, stack\_size=0x7fff00, tls=0x7ff9d0b40640}, 88) = -1 ENOSYS (Function not implemented)

**10216 clone(child\_stack=0x7ff9d0b3fef0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[10219], tls=0x7ff9d0b40640, child\_tidptr=0x7ff9d0b40910) = 10219**

10219 set\_robust\_list(0x7ff9d0b40920, 24 <unfinished ...>

10216 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

10219 <... set\_robust\_list resumed>) = 0

10216 <... rt\_sigprocmask resumed>NULL, 8) = 0

10219 rt\_sigprocmask(SIG\_SETMASK, [], <unfinished ...>

10216 brk(0x7fffcc7bf000 <unfinished ...>

10219 <... rt\_sigprocmask resumed>NULL, 8) = 0

10216 <... brk resumed>) = 0x7fffcc7bf000

10216 mmap(NULL, 16003072, PROT\_READ|PROT\_WRITE, MAP\_PRIVATE|MAP\_ANONYMOUS, -1, 0) = 0x7ff9cb0b0000

10219 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

10219 madvise(0x7ff9d0340000, 8368128, MADV\_DONTNEED) = 0

10219 exit(0) = ?

10219 +++ exited with 0 +++

10216 mmap(NULL, 8392704, PROT\_NONE, MAP\_PRIVATE|MAP\_ANONYMOUS|MAP\_STACK, -1, 0) = 0x7ff9ca8a0000

10216 mprotect(0x7ff9ca8a1000, 8388608, PROT\_READ|PROT\_WRITE) = 0

10216 rt\_sigprocmask(SIG\_BLOCK, ~[], [], 8) = 0

10216 clone3({flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, child\_tid=0x7ff9cb0a0910, parent\_tid=0x7ff9cb0a0910, exit\_signal=0, stack=0x7ff9ca8a0000, stack\_size=0x7fff00, tls=0x7ff9cb0a0640}, 88) = -1 ENOSYS (Function not implemented)

**10216 clone(child\_stack=0x7ff9cb09fef0, flags=CLONE\_VM|CLONE\_FS|CLONE\_FILES|CLONE\_SIGHAND|CLONE\_THREAD|CLONE\_SYSVSEM|CLONE\_SETTLS|CLONE\_PARENT\_SETTID|CLONE\_CHILD\_CLEARTID, parent\_tid=[10220], tls=0x7ff9cb0a0640, child\_tidptr=0x7ff9cb0a0910) = 10220**

10220 set\_robust\_list(0x7ff9cb0a0920, 24 <unfinished ...>

10216 rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

10220 <... set\_robust\_list resumed>) = 0

10216 munmap(0x7ff9d4b20000, 16003072 <unfinished ...>

10220 rt\_sigprocmask(SIG\_SETMASK, [], NULL, 8) = 0

10216 <... munmap resumed>) = 0

10216 munmap(0x7ff9d22b0000, 16003072) = 0

10216 munmap(0x7ff9d0b50000, 16003072) = 0

10216 futex(0x7ff9cb0a0910, FUTEX\_WAIT\_BITSET|FUTEX\_CLOCK\_REALTIME, 10220, NULL, FUTEX\_BITSET\_MATCH\_ANY <unfinished ...>

10220 rt\_sigprocmask(SIG\_BLOCK, ~[RT\_1], NULL, 8) = 0

10220 madvise(0x7ff9ca8a0000, 8368128, MADV\_DONTNEED) = 0

10220 exit(0) = ?

10216 <... futex resumed>) = 0

10220 +++ exited with 0 +++  
 …………………………………………………………………………

**Объем данных:** 2 \* 8 байт \* 10^7 = 160 МБ

|  |  |  |  |
| --- | --- | --- | --- |
| Число потоков | Время исполнения(с) | Ускорение | Эффективность |
| 1 | 112,407 | 1 | 1 |
| 2 | 123,624 | 0,9092 | 0,4546 |
| 4 | 147,958 | 0,7597 | 0,189925 |
| 6 | 174,866 | 0,6428 | 0,1071 |

Так как при увеличение потоков время выполнения не уменьшается, а растет, это говорит нам о том, что выделение ресурсов на потоки и также их контроль нивелируют ускорение от параллельного выполнения.

**Вывод**

Во время выполнения работы разобрался в том, как можно использовать многопоточность и на собственном примере увидел, что она не всегда может быть эффективна, как кажется на первый взгляд. Было интересно поэкспериментировать с мьютекс(в итоге не используя его), а точнее посмотреть на то, как он влияет и на то, что происходит без его использования. Много времени было потрачено для того, чтобы осознать как и для чего использовать потоки, ушло и время на отладку и изменение ситуации с неэффективностью многопоточности для входных данных и представленного алгоритма кластеризации.