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-210Б-23

Студент: Гула Дмитрий Александрович

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 01.11.24

Москва, 2024

Постановка задачи

Вариант 7.

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.

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

Общий метод и алгоритм решения

Использованные системные вызовы:

- **`pthread_create()`** — создаёт новый поток, выполняющий указанную функцию. Используется для запуска параллельных операций по обработке данных.
- **`pthread_join()`** — ожидает завершения указанного потока, чтобы продолжить выполнение основного кода.
- **`sem_init()`** — инициализирует семафор, ограничивающий одновременное количество потоков.
- **`sem_wait()`** — уменьшает значение семафора, блокируя выполнение, если значение семафора равно 0.
- **`sem_post()`** — увеличивает значение семафора, разблокируя один из ожидающих потоков.
- **`exit()`** — завершает выполнение программы.
- **`read()`** **и** **`write()`** — используются для чтения данных из стандартного ввода и записи в стандартный вывод.
- **`rand()`** — для выбора случайных начальных центроидов.

Разработка многопоточной программы для кластеризации точек на плоскости с использованием алгоритма k-средних. Основное внимание уделено реализации межпоточного взаимодействия, параллельной обработки данных и синхронизации потоков.

Метод и алгоритм решения

Программа реализует алгоритм кластеризации `k-средних` с использованием потоков для параллельной обработки данных. Алгоритм разбит на два основных этапа, каждый из которых выполняется потоками:

1. **Привязка точек к кластерам**
   Каждый поток обрабатывает свою подгруппу точек, определяя ближайший кластер для каждой из них.
2. **Пересчёт координат центроидов**
   Потоки вычисляют новые координаты центроидов как среднее точек, относящихся к соответствующему кластеру.

Работа программы

1. **Инициализация данных**
   Программа считывает входные данные — координаты точек и количество кластеров (`k`). На основе этих данных создаются массивы точек, начальных центроидов и привязок точек к кластерам.
2. **Создание потоков**
   Для выполнения каждого этапа алгоритма создаются потоки с помощью `pthread_create()`. Максимальное количество одновременно работающих потоков ограничивается с помощью семафора.
3. **Привязка точек к кластерам**
   Каждый поток обрабатывает определённую группу точек. На основе расстояния до центроидов определяется ближайший кластер для каждой точки.
4. **Пересчёт центроидов**
   Потоки пересчитывают координаты центроидов, используя данные о точках, принадлежащих каждому кластеру. Пересчёт выполняется параллельно для всех кластеров.
5. **Синхронизация потоков**
   Потоки синхронизируются с помощью семафоров, чтобы избежать конфликтов при одновременном доступе к общим данным.
6. **Итерации алгоритма**
   Привязка точек и пересчёт центроидов выполняются в цикле, пока алгоритм не достигнет заданного количества итераций (или пока изменения координат центроидов не станут незначительными).

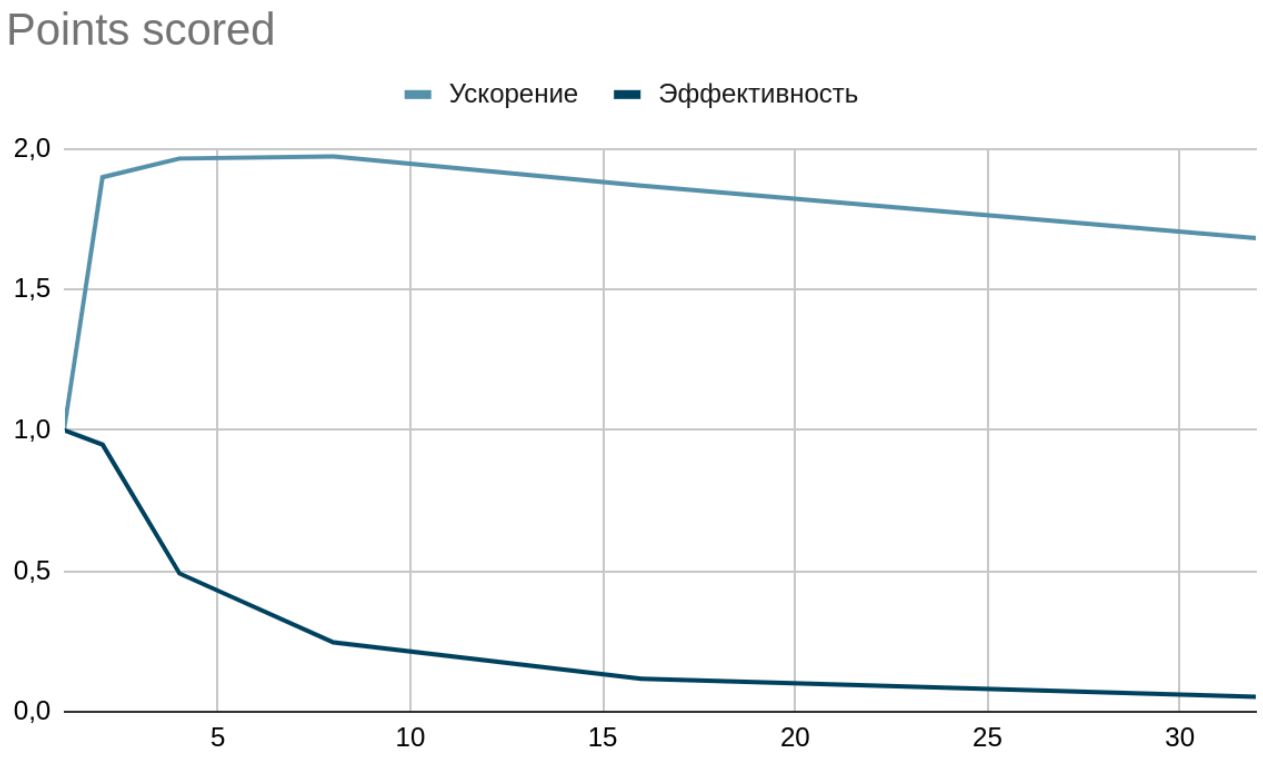
Используемые структуры данных

- **`Point`**
  Структура, представляющая точку с координатами `x` и `y`.
- **Массивы и векторы**:
  - `points`: массив всех точек.
  - `centroids`: массив координат центроидов.
  - `assignments`: массив, где для каждой точки указан её кластер.

Результаты выполнения программы

1. **Инициализация**: Пользователь задаёт количество кластеров и входные данные.
2. **Кластеризация**: Алгоритм разбивает точки на кластеры. Каждая итерация выполняется параллельно.
3. **Вывод результата**: Программа выводит итоговые координаты центроидов.

Число потоков	Время исполнения (с)	Ускорение	Эффективность
1	1.090827 seconds	1	1
2	0.574489 seconds	1.898777	0.94938
4	0.555066 seconds	1.965220	0.49130
8	0.552951 seconds	1.972737	0.24659
16	0.583781 seconds	1.868555	0.11678
32	0.648374 seconds	1.682403	0.05257



Код программы

```
#include <vector>

#include <cstdlib>

#include <cmath>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#include <cstring>

#include <cstdio>


struct Point {

    double x;

    double y;

};


std::vector<Point> points;

std::vector<Point> centroids;

std::vector<int> assignments;

int maxThreads;

sem_t semaphore;


double distance(const Point& a, const Point& b) {

    return std::sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));

}


void* assignClusters(void* arg) {

    long startIdx = (long)arg;

    for (size_t i = startIdx; i < points.size(); i += maxThreads) {
```

```
        double minDist = std::numeric_limits<double>::max();
        int closestCluster = -1;

        for (size_t j = 0; j < centroids.size(); ++j) {
            double dist = distance(points[i], centroids[j]);
            if (dist < minDist) {
                minDist = dist;
                closestCluster = j;
            }
        }

        assignments[i] = closestCluster;
    }
    sem_post(&semaphore);
    pthread_exit(nullptr);
}

void* updateCentroids(void* arg) {
    long clusterIdx = (long)arg;
    double sumX = 0, sumY = 0;
    int count = 0;

    for (size_t i = 0; i < points.size(); ++i) {
        if (assignments[i] == clusterIdx) {
            sumX += points[i].x;
            sumY += points[i].y;
            count++;
        }
    }
```

```
    if (count > 0) {
        centroids[clusterIdx].x = sumX / count;
        centroids[clusterIdx].y = sumY / count;
    }
    sem_post(&semaphore);
    pthread_exit(nullptr);
}

void kMeansClustering(int k) {
    centroids.resize(k);
    for (int i = 0; i < k; ++i) {
        centroids[i] = points[rand() % points.size()];
    }
    assignments.resize(points.size(), -1);

    for (int iteration = 0; iteration < 100; ++iteration) {
        std::vector<pthread_t> threads(maxThreads);

        for (long i = 0; i < maxThreads; ++i) {
            sem_wait(&semaphore);
            pthread_create(&threads[i], nullptr, assignClusters, (void*)i);
        }
        for (int i = 0; i < maxThreads; ++i) {
            pthread_join(threads[i], nullptr);
        }

        for (long i = 0; i < k; ++i) {
            sem_wait(&semaphore);
```

```
            pthread_create(&threads[i], nullptr, updateCentroids, (void*)i);
        }
        for (int i = 0; i < k; ++i) {
            pthread_join(threads[i], nullptr);
        }
    }
}

int readInt() {
    char buffer[16];
    int bytesRead = read(STDIN_FILENO, buffer, sizeof(buffer) - 1);
    if (bytesRead <= 0) {
        write(STDERR_FILENO, "Error reading input\n", 20);
        exit(1);
    }
    buffer[bytesRead] = '\0';
    return std::atoi(buffer);
}

void writeString(const char* str) {
    write(STDOUT_FILENO, str, std::strlen(str));
}

int main(int argc, char* argv[]) {
    if (argc < 3) {
        writeString("Usage: ./program <maxThreads> <numClusters>\n");
        return 1;
    }
```

```
    maxThreads = std::atoi(argv[1]);
    int numClusters = std::atoi(argv[2]);

    writeString("Enter the number of points: ");
    int numPoints = readInt();

    points.resize(numPoints);
    writeString("Enter the coordinates (x y) for each point:\n");
    for (int i = 0; i < numPoints; ++i) {
        writeString("Point ");
        char indexBuffer[16];
        snprintf(indexBuffer, sizeof(indexBuffer), "%d: ", i + 1);
        writeString(indexBuffer);

        char inputBuffer[64];
        int bytesRead = read(STDIN_FILENO, inputBuffer, sizeof(inputBuffer) - 1);
        if (bytesRead <= 0) {
            write(STDERR_FILENO, "Error reading point\n", 20);
            exit(1);
        }
        inputBuffer[bytesRead] = '\0';
        sscanf(inputBuffer, "%lf %lf", &points[i].x, &points[i].y);
    }

    sem_init(&semaphore, 0, maxThreads);
    kMeansClustering(numClusters);

    writeString("Clusters:\n");
    for (size_t i = 0; i < centroids.size(); ++i) {
```

```
        char outputBuffer[128];
        snprintf(outputBuffer, sizeof(outputBuffer), "Cluster %zu: (%.2f,
%.2f)\n",
                 i, centroids[i].x, centroids[i].y);
        writeString(outputBuffer);
    }

    sem_destroy(&semaphore);
    return 0;
}
```

Протокол работы программы

Тестирование:

```
1)dmitry@dmitry-HP-Laptop-15-da3xxx:~/C/Labs-OS/lab2/src/build$ ./lab2 5 2
Enter the number of points: 4
Enter the coordinates (x y) for each point:
Point 1: 1 1
Point 2: 2 2
Point 3: 100 100
Point 4: 121 121
Clusters:
Cluster 0: (110.50, 110.50)
Cluster 1: (1.50, 1.50)

2)dmitry@dmitry-HP-Laptop-15-da3xxx:~/C/Labs-OS/lab2/src/build$ ./lab2 5 3
Enter the number of points: 9
Enter the coordinates (x y) for each point:
Point 1: 1 1
Point 2: 1 0
Point 3: 0 1
Point 4: 5 5
Point 5: 5 6
Point 6: 6 3
Point 7: 12 12
Point 8: 21 21
```

```
Point 9: 10 10
Clusters:
Cluster 0: (0.67, 0.67)
Cluster 1: (16.50, 16.50)
Cluster 2: (6.50, 6.00)
```

Dtruss:

```
execve("./lab2", ["./lab2", "2", "2"], 0x7ffe56dc0b60 /* 73 vars */) = 0
brk(NULL)                               = 0x57c7a6796000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x75d719308000
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
fstat(3, {st_mode=S_IFREG|0644, st_size=56103, ...}) = 0
mmap(NULL, 56103, PROT_READ, MAP_PRIVATE, 3, 0) = 0x75d7192fa000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/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
fstat(3, {st_mode=S_IFREG|0644, st_size=2592224, ...}) = 0
mmap(NULL, 2609472, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75d719000000
mmap(0x75d71909d000, 1343488, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x9d000) = 0x75d71909d000
mmap(0x75d7191e5000, 552960, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1e5000) = 0x75d7191e5000
mmap(0x75d71926c000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x26b000) = 0x75d71926c000
mmap(0x75d71927a000, 12608, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x75d71927a000
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
fstat(3, {st_mode=S_IFREG|0644, st_size=952616, ...}) = 0
mmap(NULL, 950296, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75d718f17000
mmap(0x75d718f27000, 520192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x10000) = 0x75d718f27000
mmap(0x75d718fa6000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x8f000) = 0x75d718fa6000
mmap(0x75d718ffe000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0xe7000) = 0x75d718ffe000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libgcc_s.so.1", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\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
fstat(3, {st_mode=S_IFREG|0644, st_size=183024, ...}) = 0
mmap(NULL, 185256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75d7192cc000
mmap(0x75d7192d0000, 147456, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
```

```
3, 0x4000) = 0x75d7192d0000
mmap(0x75d7192f4000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000)
= 0x75d7192f4000
mmap(0x75d7192f8000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x2b000) = 0x75d7192f8000
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\220\243\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
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 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, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x75d718c00000
mmap(0x75d718c28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x28000) = 0x75d718c28000
mmap(0x75d718db0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1b0000) = 0x75d718db0000
mmap(0x75d718dff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1fe000) = 0x75d718dff000
mmap(0x75d718e05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0x75d718e05000
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x75d7192ca000
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x75d7192c7000
arch_prctl(ARCH_SET_FS, 0x75d7192c7740) = 0
set_tid_address(0x75d7192c7a10)         = 33121
set_robust_list(0x75d7192c7a20, 24)     = 0
rseq(0x75d7192c8060, 0x20, 0, 0x53053053) = 0
mprotect(0x75d718dff000, 16384, PROT_READ) = 0
mprotect(0x75d7192f8000, 4096, PROT_READ) = 0
mprotect(0x75d718ffe000, 4096, PROT_READ) = 0
mprotect(0x75d71926c000, 45056, PROT_READ) = 0
mprotect(0x57c7a491b000, 4096, PROT_READ) = 0
mprotect(0x75d719340000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x75d7192fa000, 56103)           = 0
futex(0x75d71927a7bc, FUTEX_WAKE_PRIVATE, 2147483647) = 0
getrandom("\x7e\x38\x2a\xc0\xdf\x95\xe9\x71", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x57c7a6796000
brk(0x57c7a67b7000)                     = 0x57c7a67b7000
write(1, "Enter the number of points: ", 28) = 28
read(0, "2\n", 15)                      = 2
write(1, "Enter the coordinates (x y) for "..., 44) = 44
write(1, "Point ", 6)                   = 6
write(1, "1: ", 3)                      = 3
```

```
read(0, "1 1\n", 63)                    = 4
write(1, "Point ", 6)                   = 6
write(1, "2: ", 3)                      = 3
read(0, "10 10\n", 63)                  = 6
rt_sigaction(SIGRT_1, {sa_handler=0x75d718c99520, sa_mask=[],
sa_flags=SA_RESTORER|SA_ONSTACK|SA_RESTART|SA_SIGINFO, sa_restorer=0x75d718c45320},
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) =
0x75d718200000
mprotect(0x75d718201000, 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|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[33173]}, 88) = 33173

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0) =
0x75d713600000
mprotect(0x75d713601000, 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|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33174

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33175

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33176

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
```

ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33177

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33178

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33179

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33180

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[33181]}, 88) = 33181

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33182

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990,

parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[33183]}, 88) = 33183

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33184

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[33185]}, 88) = 33185

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33186

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33187

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33188

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,

tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33189

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[33190]}, 88) = 33190**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33191**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[33192]}, 88) = 33192**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33193**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33194**

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

**clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33195**

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[33196]}, 88) = 33196

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33197

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[33198]}, 88) = 33198

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d713e00990,
parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80,
tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33199

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80,
tls=0x75d718a006c0} => {parent_tid=[33200]}, 88) = 33200

rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CL
ONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x75d718a00990,
```

parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33568

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[33569]}, 88) = 33569

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33570

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d713e00990, parent_tid=0x75d713e00990, exit_signal=0, stack=0x75d713600000, stack_size=0x7fff80, tls=0x75d713e006c0} => {parent_tid=[0]}, 88) = 33571

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 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=0x75d718a00990, parent_tid=0x75d718a00990, exit_signal=0, stack=0x75d718200000, stack_size=0x7fff80, tls=0x75d718a006c0} => {parent_tid=[0]}, 88) = 33572

```
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
write(1, "Clusters:\n", 10)             = 10
write(1, "Cluster 0: (10.00, 10.00)\n", 26) = 26
write(1, "Cluster 1: (1.00, 1.00)\n", 24) = 24
exit_group(0)                           = ?
+++ exited with 0 +++
```

Вывод

Выполненная работа продемонстрировала возможности многопоточного программирования для решения задач кластеризации. Использование потоков и механизмов синхронизации позволило эффективно распределить вычисления между процессорами, ускорив выполнение алгоритма. Полученные знания о работе с `pthread` и семафорами могут быть полезны для реализации других задач, требующих параллельной обработки данных.