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»

Выполнил: Т. В. Балдынов
Группа: М8О-208БВ-24
Преподаватель: Е. С. Миронов

Москва, 2025

Содержание

1. Условие

Приобрести практические навыки в параллельном программировании путем реализации четно-нечетной сортировки Бетчера с использованием нативных средств ОС.

Цель работы

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

Задание

Произвести сортировку массива из целых числел.

Вариант

5

2. Метод решения

Данная программа реализует многопоточную четно-нечетную сортировку Бетчера с использованием библиотеки pthread для межпоточной синхронизации. Основной алгоритм: программа случайный целочисленный массив размером 2^{17} и сортирует его с различным количеством потоков (1, 2, 4, 8, 16, 32, 64). Каждый поток обрабатывает определенный диапазон подотрезков данного массива.

Ключевые компоненты:
ThreadData - структура для передачи данных в потоки
sort_thread - функция потока для сортировки конкретного подотрезка массива
parallel_odd_even_sort - основная функция управления потоками

Системные вызовы:
pthread_create - создание потоков
pthread_join - ожидание завершения потоков
mmap/brk - управление памятью для потоков

Программа использует разделение массива на отдельные подотрезки, которые сортируются независимо, что обеспечивает эффективное распараллеливание и минимальные накладные расходы на синхронизацию.

2.1. График зависимости ускорения от количества потоков

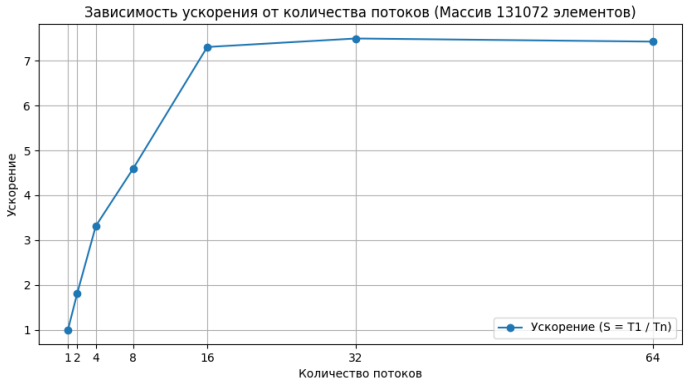

Рис. 1: Зависимость ускорения от количества потоков (Массив 2^{17} элементов)
На рисунке представлена зависимость ускорения $S = T_1/T_n$ от количества потоков.

2.2. График зависимости эффективности от количества потоков

Рис. 2: Зависимость эффективности от количества потоков (Массив 2^{17} элементов)

3. Исходный код

3.1. Основной модуль

kek.cpp - точка входа программы:

```
#include <bits/stdc++.h>

struct ThreadData {
    std::vector<int>& arr;
    int start;
    int end;
    int thread_id;
    ThreadData(std::vector<int>& a, int s, int e, int id)
        : arr(a), start(s), end(e), thread_id(id) {}
};

std::mt19937 rnd(123);

void unshuffle(std::vector<int>& a, int l, int r) {
    int half = (l + r) / 2;
    std::vector<int> tmp(a.size());

    int cnt = l;
    for (int i = l; i < r; i += 2) {
        tmp[cnt] = a[i];
        cnt++;
```

```
    }
    for (int i = l + 1; i < r; i += 2) {
        tmp[cnt] = a[i];
        cnt++;
    }

    for (auto i = 0; i < tmp.size(); i++){
        if (l <= i && i < r) {
            a[i] = tmp[i];
        }
    }
}

void Merge(std::vector<int>& v, int l, int r, int mid) {
    std::vector<int> tmp(v.size());

    int i = l, j = mid;
    int cnt = l;
    for (int it = l; it < r; ++it) {
        if (i < mid && (j >= r || v[i] < v[j])) {
            tmp[it] = v[i];
            i++;
        } else {
            tmp[it] = v[j];
            j++;
        }
    }

    for (int i = l; i < r; ++i) {
        v[i] = tmp[i];
    }
}

void OddEvenMergeSort(std::vector<int>& a, int l, int r) {
    if (r == l + 2) {
        if (a[l] > a[r - 1]) std::swap(a[l], a[r - 1]);
        return;
    }
    if (r <= l + 1) {
        return;
    }

    unshuffle(a, l, r);

    int half = (l + r) / 2;
    OddEvenMergeSort(a, l, half);
    OddEvenMergeSort(a, half, r);

    Merge(a, l, r, half);
```

```
}

void* sort_thread(void* arg) {
    ThreadData* data = static_cast<ThreadData*>(arg);
    OddEvenMergeSort(data->arr, data->start, data->end);
    pthread_exit(nullptr);
}

void parallel_odd_even_sort(std::vector<int>& arr, int
    num_threads = 1) {
    size_t n = arr.size();
    if (n <= 4 || num_threads == 1) {
        OddEvenMergeSort(arr, 0, n);
        return;
    }

    std::vector<pthread_t> threads(num_threads);
    std::vector<ThreadData*> thread_data(num_threads);

    int elements_per_thread = n / num_threads;
    int remaining_elements = n % num_threads;
    int current_start = 0;

    for (int i = 0; i < num_threads; i++) {
        int extra = (i < remaining_elements) ? 1 : 0;
        int end_pos = current_start + elements_per_thread +
    extra;
        thread_data[i] = new ThreadData(arr, current_start,
    end_pos, i);
        current_start = end_pos;
    }

    for (int i = 0; i < num_threads; i++) {
        int res = pthread_create(&threads[i], nullptr,
    sort_thread, thread_data[i]);
        if (res) {
            std::cerr << "Ошибка создания потока " << i << std::endl
    ;
            for (int j = 0; j <= i; j++) { delete thread_data[j
    ]; }
            OddEvenMergeSort(arr, 0, n);
            return;
        }
    }

    for (int i = 0; i < num_threads; i++) {
        pthread_join(threads[i], nullptr);
    }

```

```
114     int step = 1;
115     while (step < num_threads) {
116         for (int i = 0; i < num_threads; i += 2 * step) {
117             if (i + step * 2 - 1 < num_threads) {
118                 int start = thread_data[i]->start;
119                 int end = thread_data[i + step * 2 - 1]->end;
120                 int mid = (start + end) / 2;
121                 Merge(arr, start, end, mid);
122             }
123         }
124         step *= 2;
125     }
126 
127     for (int i = 0; i < num_threads; i++) {
128         delete thread_data[i];
129     }
130 }
131 
132 std::vector<int> makerndV(int n) {
133     std::vector<int> v(n);
134     for (int i = 0; i < n; ++i) {
135         v[i] = rnd() % (int)1e8;
136     }
137     return v;
138 }
139 
140 int main() {
141     int n, threads;
142     std::cin >> n;
143     std::cin >> threads;
144 
145     auto v = makerndV(n);
146     auto v_ = v;
147     sort(v_.begin(), v_.end());
148 
149     auto start_parallel = std::chrono::high_resolution_clock::
    now();
150     parallel_odd_even_sort(v, threads);
151     auto end_parallel = std::chrono::high_resolution_clock::now
    ();
152 
153     auto duration_parallel = std::chrono::duration_cast<std::
    chrono::microseconds>(end_parallel - start_parallel);
154 
155 
156     if (v_ != v) return 1;
157 
158     std::cout << duration_parallel.count();
159     return 0;
```

```
160 }
```

Listing 1: kek.cpp

4. Логи выполнения программы

```
execve("./kek", ["./kek"], 0x7ffedda17300 /* 94 vars */) = 0
brk(NULL)                               = 0x57f02cd10000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffda108d6f0) = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x78e8c7db8000
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=78560, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 78560, PROT_READ, MAP_PRIVATE, 3, 0) = 0x78e8c7da4000
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"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=2260296, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 2275520, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x78e8c7a00000
mprotect(0x78e8c7a9a000, 1576960, PROT_NONE) = 0
mmap(0x78e8c7a9a000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_
mmap(0x78e8c7bab000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
mmap(0x78e8c7c1b000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_
mmap(0x78e8c7c29000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_
1, 0) = 0x78e8c7c29000
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"..., 832) = 832
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=125488, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 127720, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x78e8c7d84000
mmap(0x78e8c7d87000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_D
mmap(0x78e8c7d9e000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3
mmap(0x78e8c7da2000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_D
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\0P\237\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
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"..., 48, 848) = 48
pread64(3, "\4\0\0\0\24\0\0\0\3\0\0\0GNU\0O{\f\225\\=\201\327\312\301P\32$\230\266\235"..., 68, 89
```

```
newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2220400, ...}, 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, 2264656, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x78e8c7600000
mprotect(0x78e8c7628000, 2023424, PROT_NONE) = 0
mmap(0x78e8c7628000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_
mmap(0x78e8c77bd000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
mmap(0x78e8c7816000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_
mmap(0x78e8c781c000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_A
1, 0) = 0x78e8c781c000
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=940560, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 942344, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x78e8c7c9d000
mmap(0x78e8c7cab000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_D
mmap(0x78e8c7d27000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
mmap(0x78e8c7d82000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_D
close(3)                                = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x78e8c7c9b000
arch_prctl(ARCH_SET_FS, 0x78e8c7c9c3c0) = 0
set_tid_address(0x78e8c7c9c690)         = 451072
set_robust_list(0x78e8c7c9c6a0, 24)     = 0
rseq(0x78e8c7c9cd60, 0x20, 0, 0x53053053) = 0
mprotect(0x78e8c7816000, 16384, PROT_READ) = 0
mprotect(0x78e8c7d82000, 4096, PROT_READ) = 0
mprotect(0x78e8c7da2000, 4096, PROT_READ) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -
1, 0) = 0x78e8c7c99000
mprotect(0x78e8c7c1b000, 45056, PROT_READ) = 0
mprotect(0x57eff52cd000, 4096, PROT_READ) = 0
mprotect(0x78e8c7df2000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x78e8c7da4000, 78560)           = 0
getrandom("\x5e\x11\x38\xbe\x21\xf2\xf1\xd7", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0x57f02cd10000
brk(0x57f02cd31000)                     = 0x57f02cd31000
futex(0x78e8c7c2977c, FUTEX_WAKE_PRIVATE, 2147483647) = 0
newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
read(0, "1024\n", 1024)                 = 5
read(0, "8\n", 1024)                    = 2
```

```
rt_sigaction(SIGRT_1, {sa_handler=0x78e8c7691870, sa_mask=[], sa_flags=SA_RESTORER|SA_ON
rt_sigprocmask(SIG_UNBLOCK, [RTMIN RT_1], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8c6dff000
mprotect(0x78e8c6e00000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8c65fe000
mprotect(0x78e8c65ff000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8c5dfd000
mprotect(0x78e8c5dfe000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8c55fc000
mprotect(0x78e8c55fd000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8c4dfb000
mprotect(0x78e8c4dfc000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8c45fa000
mprotect(0x78e8c45fb000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8b77ff000
mprotect(0x78e8b7800000, 8388608, PROT_READ|PROT_WRITE) = 0
```

```
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -
1, 0) = 0x78e8b6ffe000
mprotect(0x78e8b6fff000, 8388608, PROT_READ|PROT_WRITE) = 0
rt_sigprocmask(SIG_BLOCK, ~[], [], 8)   = 0
clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CL
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
munmap(0x78e8c6dff000, 8392704)         = 0
munmap(0x78e8c65fe000, 8392704)         = 0
futex(0x78e8b7fff910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 451561, NULL, FUTE
munmap(0x78e8c5dfd000, 8392704)         = 0
futex(0x78e8b77fe910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 451562, NULL, FUT
munmap(0x78e8c55fc000, 8392704)         = 0
newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
write(1, "4813", 4)                     = 4
lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Illegal seek)
exit_group(0)                           = ?
+++ exited with 0 +++
```

Программа демонстрирует корректную работу механизма многопоточности и эффективное распараллеливание вычислений для умножения комплексных матриц согласно варианту 5.

5. Результаты

Разработанная программа успешно реализует многопоточную архитектуру для параллельной четно-нечетной сортировки Бетчера.
В ходе решения были достигнуты следующие ключевые результаты:
Корректная работа системы многопоточной сортировки массива
Реализовано эффективное распределение работы между потоками по подотрезкам результирующего массива
Обеспечено четкое разделение данных между потоками для параллельных вычислений
Достигнута синхронизация потоков через механизм join и ожидание завершения
Кросс-платформенная функциональность
Реализована унифицированная абстракция для работы с потоками через pthread

6. Выводы

В ходе лабораторной работы успешно разработана многопоточная система четно-нечетной сортировки целочисленного массива с использованием библиотеки pthread. Программа демонстрирует корректную работу в многопоточном режиме и эффективное распараллеливание вычислений.

6.1. Анализ системных вызовов

В ходе выполнения программы были использованы следующие ключевые системные вызовы:

- pthread_create() - создание потоков для параллельной сортировки массивов
- pthread_join() - ожидание завершения работы потоков и синхронизация
- mmap() - выделение памяти для массивов и структур данных потоков
- futex() - механизм синхронизации между потоками
- mprotect() - защита областей памяти
- brk() - управление кучей для выделения памяти под массивы

Программа демонстрирует корректную работу механизма многопоточности и эффективное распределение вычислений между потоками для умножения комплексных матриц согласно варианту 5.