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-20

Студент: Голубев Т.Д.

Преподаватель: Миронов Е.С.

Оценка: _____

Дата: 16.11.2023

Москва, 2023

Постановка задачи

Вариант 2.

Отсортировать массив целых чисел при помощи параллельного алгоритма быстрой сортировки

Общий метод и алгоритм решения

Использованные системные вызовы:

- int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void *(*start_routine) (void *), void *arg); – создаёт новый поток;
- int pthread_join(pthread_t thread, void **retval); – ожидает завершения потока.

Программа разбивает заданный массив на N частей (N = количество потоков). Далее создаётся N потоков и для каждого куска массива вызывается быстрая сортировка. По окончании куски массива сливаются в один.

Код программы

sort.h

```
#pragma once

struct Piece{
    int* mas;
    int start;
    int end;
};

void sort(int* array, int n, int threads);
```

sort.cpp

```
#include "sort.h"
#include <pthread.h>
#include <iostream>
#include <algorithm>

void create_thread(pthread_t* thread, const pthread_attr_t* attr, void
*(*start)(void *), void* arg) {
    if (pthread_create(thread, attr, start, arg) != 0) {
        perror("create_thread error!");
        exit(-1);
    }
}

void* thread_sort(void* arg) {
    Piece* p = (Piece*) arg;
    int i = p->start;
    int j = p->end;
    int mid = p->mas[(i + j) / 2];
    int swaps = 0;
    do {
```

```
        while (p->mas[i] < mid) {
            ++i;
        }
        while (p->mas[j] > mid) {
            --j;
        }
        if (i <= j) {
            std::swap(p->mas[i], p->mas[j]);
            ++swaps;
            ++i;
            --j;
        }
    } while (i <= j);
    if (p->start < j) {
        Piece less = {p->mas, p->start, j};
        thread_sort(&less);
    }
    if (i < p->end) {
        Piece more = {p->mas, i, p->end};
        thread_sort(&more);
    }

    return 0;
}

int* merge(int* a, size_t size_a, int* b, size_t size_b) {
    size_t size_res = size_a + size_b;
    int* res = new int[size_res];
    int i = 0, j = 0, k = 0;
    while (i < size_a || j < size_b) {
        if (i >= size_a) {
            res[k] = b[j];
            ++j;
        } else if (j >= size_b) {
            res[k] = a[i];
            ++i;
        } else {
            if (a[i] < b[j]) {
                res[k] = a[i];
                ++i;
            } else {
                res[k] = b[j];
                ++j;
            }
        }
        ++k;
    }
    return res;
}

void sort(int* array, int n, int threads) {
    Piece p[threads];
    pthread_t tid[threads];
    for (int i = 0; i < threads; ++i) {
        int* array_piece = new int[n / threads];
        int counter = 0;
        for (int j = i * (n / threads); j < (i + 1) * (n / threads); ++j) {
```

```
            array_piece[counter] = array[j];
            ++counter;
        }
        p[i] = Piece{array_piece, 0, n / threads - 1};
        create_thread(&tid[i], NULL, thread_sort, &p[i]);
    }
    for (int i = 0; i < threads; ++i) {
        pthread_join(tid[i], NULL);
    }
    for (int i = 0; i < threads; ++i) {
        int counter = 0;
        for (int j = i * (n / threads); j < (i + 1) * (n / threads); ++j) {
            array[j] = p[i].mas[counter];
            ++counter;
        }
    }
    int* res = new int[0];
    size_t res_size = 0;
    for (int i = 0; i < threads; ++i) {
        res = merge(res, res_size, p[i].mas, n / threads);
        res_size += n / threads;
    }
    for (int i = 0; i < n; ++i) {
        array[i] = res[i];
    }
}
```

main.cpp

```
#include "sort.h"
#include "threadscount.h"
#include <iostream>
#include <chrono>

using namespace std::chrono;

int main(int argc, char* argv[]) {
    if (argc != 2) {
        perror("Using: ./lab02_exe num_of_threads");
        exit(-1);
    }
    int n;
    std::cout << "Enter the number of elements: ";
    std::cin >> n;
    int mas[n];
    std::cout << "Fill array: ";
    for (int i = 0; i < n; ++i) {
        std::cin >> mas[i];
    }
    int threads(atoi(argv[1]));
    auto start = std::chrono::high_resolution_clock::now();
    sort(mas, n, threads);
    auto end = std::chrono::high_resolution_clock::now();
    duration<double> sec = end - start;
    std::cout << "Result: ";
    std::cout << sec.count() << " s" << std::endl;

    return 0;
```

```
}
```

Протокол работы программы

Strace:

```
execve("./lab02_exe", ["./lab02_exe", "2"], 0x7ffe2ef2cd50 /* 60 vars */) = 0
brk(NULL)                               = 0x55d36770e000
arch_prctl(0x3001 /* ARCH_??? */, 0x7ffc04b6c520) = -1 EINVAL (Недопустимый аргумент)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdc0dc2d000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (Нет такого файла или каталога)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
newfstatat(3, "", {st_mode=S_IFREG|0644, st_size=75015, ...}, AT_EMPTY_PATH) = 0
mmap(NULL, 75015, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdc0dc1a000
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) = 0x7fdc0d9ee000
mprotect(0x7fdc0da88000, 1576960, PROT_NONE) = 0
mmap(0x7fdc0da88000, 1118208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x9a000)
= 0x7fdc0da88000
mmap(0x7fdc0db99000, 454656, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1ab000) =
0x7fdc0db99000
mmap(0x7fdc0dc09000, 57344, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21a000)
= 0x7fdc0dc09000
mmap(0x7fdc0dc17000, 10432, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fdc0dc17000
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) = 0x7fdc0d9ce000
mmap(0x7fdc0d9d1000, 94208, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x3000) =
0x7fdc0d9d1000
mmap(0x7fdc0d9e8000, 16384, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a000) =
0x7fdc0d9e8000
mmap(0x7fdc0d9ec000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1d000) =
0x7fdc0d9ec000
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\0"..., 48, 848) = 48
```

```
      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

      newfstatat(3, "", {st_mode=S_IFREG|0755, st_size=2216304, ...}, 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, 2260560, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdc0d7a6000

      mmap(0x7fdc0d7ce000, 1658880, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000)
= 0x7fdc0d7ce000

      mmap(0x7fdc0d963000, 360448, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1bd000) =
0x7fdc0d963000

      mmap(0x7fdc0d9bb000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x214000)
= 0x7fdc0d9bb000

      mmap(0x7fdc0d9c1000, 52816, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7fdc0d9c1000

      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) = 0x7fdc0d6bf000

      mmap(0x7fdc0d6cd000, 507904, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) =
0x7fdc0d6cd000

      mmap(0x7fdc0d749000, 372736, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x8a000) =
0x7fdc0d749000

      mmap(0x7fdc0d7a4000, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe4000) =
0x7fdc0d7a4000

      close(3)                                  = 0

      mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdc0d6bd000

      arch_prctl(ARCH_SET_FS, 0x7fdc0d6be3c0) = 0

      set_tid_address(0x7fdc0d6be690)         = 20093

      set_robust_list(0x7fdc0d6be6a0, 24)     = 0

      rseq(0x7fdc0d6bed60, 0x20, 0, 0x53053053) = 0

      mprotect(0x7fdc0d9bb000, 16384, PROT_READ) = 0

      mprotect(0x7fdc0d7a4000, 4096, PROT_READ) = 0

      mprotect(0x7fdc0d9ec000, 4096, PROT_READ) = 0

      mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdc0d6bb000

      mprotect(0x7fdc0dc09000, 45056, PROT_READ) = 0

      mprotect(0x55d366206000, 4096, PROT_READ) = 0

      mprotect(0x7fdc0dc67000, 8192, PROT_READ) = 0

      prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0

      munmap(0x7fdc0dc1a000, 75015)           = 0

      getrandom("\x55\x9a\xa1\xc2\xd2\x35\xe5\xe7", 8, GRND_NONBLOCK) = 8

      brk(NULL)                                 = 0x55d36770e000

      brk(0x55d36772f000)                       = 0x55d36772f000

      futex(0x7fdc0dc1777c, FUTEX_WAKE_PRIVATE, 2147483647) = 0

      newfstatat(1, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0
```

```
write(1, "Enter the number of elements: ", 30Enter the number of elements: ) = 30

newfstatat(0, "", {st_mode=S_IFCHR|0620, st_rdev=makedev(0x88, 0), ...}, AT_EMPTY_PATH) = 0

read(0, 5

"5\n", 1024)                    = 2

write(1, "Fill array: ", 12Fill array: )            = 12

read(0, 5 1 4 3 2

"5 1 4 3 2\n", 1024)            = 10

rt_sigaction(SIGRT_1, {sa_handler=0x7fdc0d837870, sa_mask=[], sa_flags=SA_RESTORER|SA_ONSTACK|
SA_RESTART|SA_SIGINFO, sa_restorer=0x7fdc0d7e8520}, 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) = 0x7fdc0ceba000

mprotect(0x7fdc0cebb000, 8388608, PROT_READ|PROT_WRITE) = 0

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

clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|CLONE_SETTLS|
CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fdc0d6ba910, parent_tid=0x7fdc0d6ba910,
exit_signal=0, stack=0x7fdc0ceba000, stack_size=0x7fff00, tls=0x7fdc0d6ba640}strace: Process 20112
attached

 => {parent_tid=[20112]}, 88) = 20112

[pid 20093] rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0

[pid 20112] rseq(0x7fdc0d6bafe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 20093] mmap(NULL, 8392704, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_STACK, -1, 0
<unfinished ...>

[pid 20112] <... rseq resumed>)         = 0

[pid 20093] <... mmap resumed>)         = 0x7fdc0c6b9000

[pid 20093] mprotect(0x7fdc0c6ba000, 8388608, PROT_READ|PROT_WRITE <unfinished ...>

[pid 20112] set_robust_list(0x7fdc0d6ba920, 24 <unfinished ...>

[pid 20093] <... mprotect resumed>)     = 0

[pid 20093] rt_sigprocmask(SIG_BLOCK, ~[],  <unfinished ...>

[pid 20112] <... set_robust_list resumed>) = 0

[pid 20093] <... rt_sigprocmask resumed>[], 8) = 0

[pid 20112] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 20093] clone3({flags=CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD|CLONE_SYSVSEM|
CLONE_SETTLS|CLONE_PARENT_SETTID|CLONE_CHILD_CLEARTID, child_tid=0x7fdc0ceb9910,
parent_tid=0x7fdc0ceb9910, exit_signal=0, stack=0x7fdc0c6b9000, stack_size=0x7fff00, tls=0x7fdc0ceb9640}
<unfinished ...>

[pid 20112] <... rt_sigprocmask resumed>NULL, 8) = 0

strace: Process 20113 attached

[pid 20112] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>

[pid 20093] <... clone3 resumed> => {parent_tid=[20113]}, 88) = 20113

[pid 20113] rseq(0x7fdc0ceb9fe0, 0x20, 0, 0x53053053 <unfinished ...>

[pid 20093] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

[pid 20112] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 20093] <... rt_sigprocmask resumed>NULL, 8) = 0

[pid 20113] <... rseq resumed>)         = 0
```

```
      [pid 20093] futex(0x7fdc0d6ba910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 20112, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

      [pid 20112] madvise(0x7fdc0ceba000, 8368128, MADV_DONTNEED <unfinished ...>

      [pid 20113] set_robust_list(0x7fdc0ceb9920, 24 <unfinished ...>

      [pid 20112] <... madvise resumed>)      = 0

      [pid 20113] <... set_robust_list resumed>) = 0

      [pid 20112] exit(0 <unfinished ...>

      [pid 20113] rt_sigprocmask(SIG_SETMASK, [],  <unfinished ...>

      [pid 20112] <... exit resumed>)         = ?

      [pid 20113] <... rt_sigprocmask resumed>NULL, 8) = 0

      [pid 20093] <... futex resumed>)        = 0

      [pid 20113] rt_sigprocmask(SIG_BLOCK, ~[RT_1],  <unfinished ...>

      [pid 20112] +++ exited with 0 +++

      [pid 20093] futex(0x7fdc0ceb9910, FUTEX_WAIT_BITSET|FUTEX_CLOCK_REALTIME, 20113, NULL,
FUTEX_BITSET_MATCH_ANY <unfinished ...>

      [pid 20113] <... rt_sigprocmask resumed>NULL, 8) = 0

      [pid 20113] madvise(0x7fdc0c6b9000, 8368128, MADV_DONTNEED) = 0

      [pid 20113] exit(0)                     = ?

      [pid 20113] +++ exited with 0 +++

      <... futex resumed>)                    = 0

      write(1, "Result: 0.00226607 s\n", 21Result: 0.00226607 s

      )  = 21

      write(1, "1 2 3 4 5 \n", 111 2 3 4 5

      )            = 11

      lseek(0, -1, SEEK_CUR)                  = -1 ESPIPE (Недопустимая операция смещения)

      exit_group(0)                           = ?

      +++ exited with 0 +++
```

Тестирование:

```
cat_mood@nuclear-box:~/programming/mai-os-labs/lab02/build$ ./lab02_exe 1

Enter the number of elements: 1

Fill array: 1

Result: 0.00027422 s

1

cat_mood@nuclear-box:~/programming/mai-os-labs/lab02/build$ ./lab02_exe 2

Enter the number of elements: 1

Fill array: 1

Result: 0.000105309 s

1

cat_mood@nuclear-box:~/programming/mai-os-labs/lab02/build$ ./lab02_exe 1
Enter the number of elements: 5
Fill array: 1 2 3 4 5
```

```
Result: 0.000260325 s
1 2 3 4 5
cat_mood@nuclear-box:~/programming/mai-os-labs/lab02/build$ ./lab02_exe 2
Enter the number of elements: 5
Fill array: 1 2 3 4 5
Result: 0.000328844 s
1 2 3 4 5
cat_mood@nuclear-box:~/programming/mai-os-labs/lab02/build$ ./lab02_exe 1
Enter the number of elements: 5
Fill array: 5 3 4 2 1
Result: 0.000253752 s
1 2 3 4 5
cat_mood@nuclear-box:~/programming/mai-os-labs/lab02/build$ ./lab02_exe 2
Enter the number of elements: 5
Fill array: 5 3 4 2 1
Result: 0.000233403 s
1 2 3 4 5
```

Вывод

Число потоков	Время исполнения (с)	Ускорение	Эффективность
1	0.328418	1	1
2	0.228291	1.44	0.72
3	0.199181	1.65	0.55
4	0.169378	1.94	0.49
5	0.242937	1.35	0.27
6	0.172961	1.9	0.32
7	0.179753	1.83	0.26
8	0.186823	1.76	0.22
9	0.208921	1.57	0.17
10	0.214146	1.53	0.15
11	0.226098	1.45	0.13
12	0.244072	1.35	0.11

13	0.253523	1.3	0.1
14	0.252323	1.3	0.09
15	0.266768	1.23	0.082
16	0.276207	1.19	0.074

Из выше приведённой таблицы можно сделать вывод, что выбранный способ параллельной сортировки имеет небольшой выигрыш по сравнению с последовательной сортировкой. Наибольшее ускорение достигается на 4 потоках.

В ходе работы я столкнулся с многими проблемами, одной из которых является false sharing. False sharing (от лукавого) – это доступ к разным данным, но по каким-то причинам, оказавшимся в одной кэш-линии процессора. В моей программе разные потоки обращались к одному участку памяти, из-за чего происходила деградация алгоритма с увеличением потоков.