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-213Б-23

Студент: Германенко М. И.

Преподаватель: Бахарев В.Д.

Оценка: _____

Дата: 11.11.24

Москва, 2024

Постановка задачи

Вариант 14.

Есть набор 128 битных чисел, записанных в шестнадцатеричном представлении, хранящихся в файле. Необходимо посчитать их среднее арифметическое. Округлить результат до целых. Количество используемой оперативной памяти должно задаваться "ключом"

Общий метод и алгоритм решения

Программа предназначена для вычисления среднего арифметического набора 128-битных чисел, представленных в шестнадцатеричной форме, которые считываются из файла. Для этого используются многозадачность и обработка данных в потоках с ограничением по памяти.

Основные этапы работы программы:

1. Чтение данных: Программа открывает файл, содержащий 128-битные числа, каждое из которых представлено как строка из 32 символов в шестнадцатеричном формате. Эти строки считываются и преобразуются в 64-битные целые числа.

2. Параллельная обработка данных: Данные из файла считываются блоками. Размер блока определяется на основе ограничений по памяти. Для каждого блока создается поток, который вычисляет сумму чисел в этом блоке. Все потоки работают параллельно, что ускоряет процесс вычислений.

3. Использование многозадачности: Для управления количеством одновременно работающих потоков используется мьютекс и условная переменная. Потоки синхронизируются с главной программой, чтобы не возникало гонок за ресурсами, и все вычисления завершались корректно.

4. Вычисление среднего арифметического: После завершения работы всех потоков главная программа собирает результаты (сумму всех чисел) и делит на количество чисел, чтобы вычислить среднее арифметическое. Результат выводится на экран.

Входные параметры программы: Имя файла, содержащего 128-битные шестнадцатеричные числа. Максимальное количество потоков для параллельной обработки. Ограничение по памяти — максимальный объем памяти, который можно использовать для хранения данных за один раз.

Входной файл: Файл должен содержать 128-битные числа, записанные каждое с новой строки в шестнадцатеричной форме. Например:

00000000000000000000000000000010

00000000000000000000000000000020

Количество потоков K	**Время выполнения (сек)**	**Коэффициент прироста производительности** (T1/Tk)
1	0.8	1.0
2	0.45	1.78
4	0.27	2.96
8	0.23	3.48

Код программы

main.c

```c
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <pthread.h>

#define HEX_STRING_LENGTH 32
#define BUFFER_SIZE 64

uint64_t sum = 0;
size_t totalCount = 0;
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
int activeThreads = 0;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
size_t maxThreads;
size_t memoryLimit;

typedef struct {
    uint64_t *numbers;
    size_t count;
} ThreadData;

void *processChunk(void *arg) {
    ThreadData *data = (ThreadData *)arg;
    uint64_t localSum = 0;

    for (size_t i = 0; i < data->count; i++) {
        localSum += data->numbers[i];
    }

    pthread_mutex_lock(&mutex);
    sum += localSum;
    totalCount += data->count;
    activeThreads--;
    pthread_cond_signal(&cond);
    pthread_mutex_unlock(&mutex);

    free(data->numbers);
    free(data);
    return NULL;
}

void createThreads(uint64_t *numbers, size_t count) {
    ThreadData *data = malloc(sizeof(ThreadData));
    data->numbers = numbers;
    data->count = count;

    pthread_t thread;
    pthread_mutex_lock(&mutex);
```

```
    while (activeThreads >= maxThreads) {
        pthread_cond_wait(&cond, &mutex);
    }
    activeThreads++;
    pthread_mutex_unlock(&mutex);

    if (pthread_create(&thread, NULL, processChunk, data) != 0) {
        _exit(1);
    }
    pthread_detach(thread);
}

void uint64ToStr(uint64_t value, char *buffer) {
    char temp[20];
    int i = 0;

    if (value == 0) {
        buffer[0] = '0';
        buffer[1] = '\n';
        buffer[2] = '\0';
        return;
    }

    while (value > 0) {
        temp[i++] = '0' + (value % 10);
        value /= 10;
    }

    int j = 0;
    while (i > 0) {
        buffer[j++] = temp[--i];
    }
    buffer[j++] = '\n';
    buffer[j] = '\0';
}

int main(int argc, char *argv[]) {
    if (argc != 4) {
        const char *msg = "Использование: <имя файла> <макс. потоки> <память>\n";
        write(STDERR_FILENO, msg, strlen(msg));
        _exit(1);
    }

    const char *filename = argv[1];
    maxThreads = strtoul(argv[2], NULL, 10);
    memoryLimit = strtoul(argv[3], NULL, 10);

    int file = open(filename, O_RDONLY);
    if (file == -1) {
        const char *msg = "Ошибка открытия файла\n";
        write(STDERR_FILENO, msg, strlen(msg));
        _exit(1);
    }
```

```
    size_t numbersPerChunk = memoryLimit / sizeof(uint64_t);
    char buffer[BUFFER_SIZE];
    size_t bufferIndex = 0;
    uint64_t *numbers = malloc(numbersPerChunk * sizeof(uint64_t));
    size_t count = 0;

    while (1) {
        ssize_t bytesRead = read(file, buffer + bufferIndex, BUFFER_SIZE - bufferIndex);
        if (bytesRead <= 0) break;

        bytesRead += bufferIndex;
        size_t start = 0;

        for (size_t i = 0; i < bytesRead; i++) {
            if (buffer[i] == '\n') {
                buffer[i] = '\0';
                numbers[count++] = strtoull(buffer + start, NULL, 16);
                start = i + 1;

                if (count == numbersPerChunk) {
                    createThreads(numbers, count);
                    numbers = malloc(numbersPerChunk * sizeof(uint64_t));
                    count = 0;
                }
            }
        }

        bufferIndex = bytesRead - start;
        memmove(buffer, buffer + start, bufferIndex);
    }

    if (count > 0) {
        createThreads(numbers, count);
    } else {
        free(numbers);
    }

    pthread_mutex_lock(&mutex);
    while (activeThreads > 0) {
        pthread_cond_wait(&cond, &mutex);
    }
    pthread_mutex_unlock(&mutex);

    close(file);

    uint64_t average = (totalCount > 0) ? (sum / totalCount) : 0;

    char result[64];
    uint64ToStr(average, result);
    write(STDOUT_FILENO, "Среднее арифметическое: ", 45);
    write(STDOUT_FILENO, result, strlen(result));

    return 0;
}
```

Протокол работы программы

Тестирование:

migermanenko@MacBook-Pro-Matvej Lab2 % cat > test1.txt

000000000000000000000000000001000000000000000000000000000000020000000000000000000000000000A200000000000000000000000000000002%

migermanenko@MacBook-Pro-Matvej Lab2 % ./a.out test1.txt 5 1024

Среднее арифметическое: 53

migermanenko@MacBook-Pro-Matvej Lab2 % cat > test2.txt

00000000000000000000000000001000000000000000000000000020%

migermanenko@MacBook-Pro-Matvej Lab2 % ./a.out test2.txt 2 128

Среднее арифметическое: 24

migermanenko@MacBook-Pro-Matvej Lab2 %.

Dtruss:

migermanenko@MacBook-Pro-Matvej Lab2 % sudo dtruss -f ./a.out test.txt 5 1024

PID/THRD SYSCALL(args) = return

Среднее арифметическое: 53

1352/0x3b98: fork() = 0 0

1352/0x3b98: munmap(0x100A60000, 0x84000) = 0 0

1352/0x3b98: munmap(0x100AE4000, 0x8000) = 0 0

1352/0x3b98: munmap(0x100AEC000, 0x4000) = 0 0

1352/0x3b98: munmap(0x100AF0000, 0x4000) = 0 0

1352/0x3b98: munmap(0x100AF4000, 0x48000) = 0 0

1352/0x3b98: munmap(0x100B3C000, 0x4C000) = 0 0

1352/0x3b98: crossarch_trap(0x0, 0x0, 0x0) = -1 Err#45

1352/0x3b98: open(".\0", 0x100000, 0x0) = 3 0

1352/0x3b98: fcntl(0x3, 0x32, 0x16F52B098) = 0 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: fsgetpath(0x16F52B0A8, 0x400, 0x16F52B088) = 46 0

1352/0x3b98: fsgetpath(0x16F52B0B8, 0x400, 0x16F52B098) = 14 0

1352/0x3b98: csrctl(0x0, 0x16F52B4BC, 0x4) = -1 Err#1

1352/0x3b98: __mac_syscall(0x187203C12, 0x2, 0x16F52B400) = 0 0

1352/0x3b98: csrctl(0x0, 0x16F52B4AC, 0x4) = -1 Err#1

1352/0x3b98: __mac_syscall(0x187200A45, 0x5A, 0x16F52B440) = 0 0

1352/0x3b98: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16F52A9A8, 0x16F52A9A0, 0x187202738, 0xD) = 0 0

1352/0x3b98: sysctl([CTL_KERN, 157, 0, 0, 0, 0] (2), 0x16F52AA58, 0x16F52AA50, 0x0, 0x0) = 0 0

1352/0x3b98: open("/\0", 0x20100000, 0x0) = 3 0

1352/0x3b98: openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0

1352/0x3b98: dup(0x4, 0x0, 0x0) = 5 0

1352/0x3b98: fstatat64(0x4, 0x16F52A531, 0x16F52A4A0) = 0 0

1352/0x3b98: openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0

1352/0x3b98: fcntl(0x6, 0x32, 0x16F52A530) = 0 0

1352/0x3b98: dup(0x6, 0x0, 0x0) = 7 0

1352/0x3b98: dup(0x5, 0x0, 0x0) = 8 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: close(0x5) = 0 0

1352/0x3b98: close(0x4) = 0 0

1352/0x3b98: close(0x6) = 0 0

1352/0x3b98: __mac_syscall(0x187203C12, 0x2, 0x16F52AF20) = 0 0

1352/0x3b98: shared_region_check_np(0x16F52AB40, 0x0, 0x0) = 0 0

1352/0x3b98: fsgetpath(0x16F52B0C0, 0x400, 0x16F52AFE8) = 82 0

1352/0x3b98: fcntl(0x8, 0x32, 0x16F52B0C0) = 0 0

1352/0x3b98: close(0x8) = 0 0

1352/0x3b98: close(0x7) = 0 0

1352/0x3b98: getfsstat64(0x0, 0x0, 0x2) = 11 0

1352/0x3b98: getfsstat64(0x1008D4050, 0x5D28, 0x2) = 11 0

1352/0x3b98: getattrlist("/\0", 0x16F52B000, 0x16F52AF70) = 0 0

1352/0x3b98: stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm64e\0", 0x16F52B360, 0x0) = 0 0

dtrace: error on enabled probe ID 1696 (ID 845: syscall::stat64:return): invalid address (0x0) in action #12 at DIF offset 12

1352/0x3b98: stat64("/Users/migermanenko/Documents/C/OC/Lab2/a.out\0", 0x16F52A810, 0x0) = 0 0

1352/0x3b98: open("/Users/migermanenko/Documents/C/OC/Lab2/a.out\0", 0x0, 0x0) = 3 0

1352/0x3b98: mmap(0x0, 0xC868, 0x1, 0x40002, 0x3, 0x0) = 0x1008D4000 0

1352/0x3b98: fcntl(0x3, 0x32, 0x16F52A928) = 0 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: munmap(0x1008D4000, 0xC868) = 0 0

1352/0x3b98: stat64("/Users/migermanenko/Documents/C/OC/Lab2/a.out\0", 0x16F52AD80, 0x0) = 0 0

1352/0x3b98: stat64("/usr/lib/libSystem.B.dylib\0", 0x16F529CD0, 0x0) = -1 Err#2

1352/0x3b98: stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16F529C80, 0x0) = -1 Err#2

1352/0x3b98: open("/Users/migermanenko/Documents/C/OC/Lab2/a.out\0", 0x0, 0x0) = 3 0

1352/0x3b98: __mac_syscall(0x187203C12, 0x2, 0x16F5283D0) = 0 0

1352/0x3b98: map_with_linking_np(0x16F528260, 0x1, 0x16F528290) = 0 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: mprotect(0x1008C8000, 0x4000, 0x1) = 0 0

1352/0x3b98: open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0

1352/0x3b98: ioctl(0x3, 0x80086804, 0x16F527988) = 0 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

1352/0x3b98: access("/AppleInternal/XBS/.isChrooted\0", 0x0, 0x0) = -1 Err#2

1352/0x3b98: bsdthread_register(0x1875060F4, 0x1875060E8, 0x4000) = 1073746399 0

1352/0x3b98: getpid(0x0, 0x0, 0x0) = 1352 0

1352/0x3b98: shm_open(0x18739DF41, 0x0, 0xFFFFFFFF87544000) = 3 0

1352/0x3b98: fstat64(0x3, 0x16F528000, 0x0) = 0 0

1352/0x3b98: mmap(0x0, 0x8000, 0x1, 0x40001, 0x3, 0x0) = 0x1008DC000 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: csops(0x548, 0x0, 0x16F52813C) = 0 0

1352/0x3b98: ioctl(0x2, 0x4004667A, 0x16F5280AC) = 0 0

1352/0x3b98: mprotect(0x1008EC000, 0x4000, 0x0) = 0 0

1352/0x3b98: mprotect(0x1008F8000, 0x4000, 0x0) = 0 0

1352/0x3b98: mprotect(0x1008FC000, 0x4000, 0x0) = 0 0

1352/0x3b98: mprotect(0x100908000, 0x4000, 0x0) = 0 0

1352/0x3b98: mprotect(0x10090C000, 0x4000, 0x0) = 0 0

1352/0x3b98: mprotect(0x100918000, 0x4000, 0x0) = 0 0

1352/0x3b98: mprotect(0x1008E4000, 0xC8, 0x1) = 0 0

1352/0x3b98: mprotect(0x1008E4000, 0xC8, 0x3) = 0 0

1352/0x3b98: mprotect(0x1008E4000, 0xC8, 0x1) = 0 0

1352/0x3b98: mprotect(0x10091C000, 0x4000, 0x1) = 0 0

1352/0x3b98: mprotect(0x100920000, 0xC8, 0x1) = 0 0

1352/0x3b98: mprotect(0x100920000, 0xC8, 0x3) = 0 0

1352/0x3b98: mprotect(0x100920000, 0xC8, 0x1) = 0 0

1352/0x3b98: mprotect(0x1008E4000, 0xC8, 0x3) = 0 0

1352/0x3b98: mprotect(0x1008E4000, 0xC8, 0x1) = 0 0

1352/0x3b98: mprotect(0x10091C000, 0x4000, 0x3) = 0 0

1352/0x3b98: mprotect(0x10091C000, 0x4000, 0x1) = 0 0

1352/0x3b98: issetugid(0x0, 0x0, 0x0) = 0 0

1352/0x3b98: getentropy(0x16F527708, 0x20, 0x0) = 0 0

1352/0x3b98: getattrlist("/Users/migermanenko/Documents/C/OC/Lab2/a.out\0", 0x16F527FA0, 0x16F527FBC) = 0 0

1352/0x3b98: access("/Users/migermanenko/Documents/C/OC/Lab2\0", 0x4, 0x0) = 0 0

1352/0x3b98: open("/Users/migermanenko/Documents/C/OC/Lab2\0", 0x0, 0x0) = 3 0

1352/0x3b98: fstat64(0x3, 0x14CE04500, 0x0) = 0 0

1352/0x3b98: csrctl(0x0, 0x16F52818C, 0x4) = -1 Err#1

1352/0x3b98: fgetattrlist(0x3, 0x16F528230, 0x16F5281B0) = 0 0

1352/0x3b98: __mac_syscall(0x19346B505, 0x2, 0x16F5281B0) = 0 0

1352/0x3b98: fcntl(0x3, 0x32, 0x16F527E88) = 0 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: open("/Users/migermanenko/Documents/C/OC/Lab2/Info.plist\0", 0x0, 0x0) = -1 Err#2

1352/0x3b98: proc_info(0x2, 0x548, 0xD) = 64 0

1352/0x3b98: csops_audittoken(0x548, 0x10, 0x16F528210) = 0 0

1352/0x3b98: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16F528568, 0x16F528560, 0x18AC30D3A, 0x15) = 0 0

1352/0x3b98: sysctl([CTL_KERN, 155, 0, 0, 0, 0] (2), 0x16F5285F8, 0x16F5285F0, 0x0, 0x0) = 0 0

1352/0x3b98: open("test.txt\0", 0x0, 0x0) = 3 0

1352/0x3b98: read(0x3, "00000000000000000000000000000010\0", 0x20) = 32 0

1352/0x3b98: read(0x3, "00000000000000000000000000000020\0", 0x20) = 32 0

1352/0x3b98: read(0x3, "000000000000000000000000000000A2\0", 0x20) = 32 0

1352/0x3b98: read(0x3, "00000000000000000000000000000002\0", 0x20) = 32 0

1352/0x3b98: read(0x3, "\0", 0x20) = 0 0

1352/0x3b98: bsdthread_create(0x1008C76D4, 0x600001F70030, 0x16F5C3000) = 1868312576 0

1352/0x3b98: read(0x3, "\0", 0x20) = 0 0

1352/0x3b9d: fork() = 0 0

1352/0x3b9d: thread_selfid(0x0, 0x0, 0x0) = 15261 0

1352/0x3b9d: psynch_cvsignal(0x1008CC040, 0x100, 0x0) = 257 0

1352/0x3b9d: __disable_threadsignal(0x1, 0x0, 0x0) = 0 0

1352/0x3b98: psynch_cvwait(0x1008CC040, 0x100000100, 0x0) = 0 0

1352/0x3b98: close(0x3) = 0 0

1352/0x3b98: write(0x1, "\320\241\321\200\320\265\320\264\320\275\320\265\320\265 \320\260\321\200\320\270\321\204\320\274\320\265\321\202\320\270\321\207\320\265\321\201 \320\272\320\276\320\265: \0", 0x2D) = 45 0

1352/0x3b98: write(0x1, "53\n\0", 0x3) = 3 0

Вывод

В ходе выполнения работы была разработана программа на языке C, которая реализует многопоточную обработку данных с использованием стандартных средств создания потоков операционной системы Unix (pthread). Программа обрабатывает набор 128-битных чисел, записанных в шестнадцатеричном представлении, находящихся в файле. Для обеспечения корректной и безопасной работы программы были применены механизмы синхронизации (мьютексы и условные переменные) для ограничения числа одновременно активных потоков, что задаётся ключом при запуске.

Программа успешно считывает данные, делит их на блоки на основе ограничения по оперативной памяти и обрабатывает каждый блок в отдельном потоке. После завершения всех потоков программа вычисляет и выводит округлённое до целого среднее арифметическое чисел.