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: М80-214Б-23
Студент: Камышников И.С.
Преподаватель: Бахарев В.Д.
Оценка:_____
Дата: 16.02.25

Москва, 2025

Постановка задачи

Вариант 10.

Составить программу на языке C, обрабатывающую данные в многопоточном режиме.
При обработке использовать стандартные средства создания потоков ОС (Windows/Unix).
Ограничение максимального количества потоков задается ключом запуска программы.
Также необходимо демонстрировать количество используемых потоков стандартными средствами ОС.

В отчёте привести исследование зависимости ускорения и эффективности алгоритма
от входных данных и количества потоков.

Решение системы линейных уравнений методом Гаусса.

Используемые системные вызовы:

read — системный вызов для чтения данных из стандартного ввода.

write — системный вызов для вывода данных в стандартный вывод.

pthread_create — создание нового потока.

pthread_join — ожидание завершения потока.

pthread_mutex_init — инициализация мьютекса.

pthread_mutex_lock — блокировка мьютекса.

pthread_mutex_unlock — разблокировка мьютекса.

pthread_cond_wait — ожидание условия с освобождением мьютекса.

pthread_cond_destroy — освобождение ресурсов условной переменной.

pthread_mutex_destroy — освобождение ресурсов мьютекса.

Описание программы

Программа решает систему линейных уравнений методом Гаусса, используя многопоточность и механизмы синхронизации.
Параметры, задаваемые пользователем:

- Количество переменных (n).
- Количество потоков (numThreads).
- Коэффициенты расширенной матрицы системы.

Алгоритм работы:

1. Ввод исходных данных.
2. Создание numThreads потоков.
3. Каждый поток выполняет часть алгоритма исключения Гаусса.
4. Используются мьютексы и условные переменные для синхронизации потоков.
5. Применяется метод обратной подстановки для вычисления решений.
6. Вывод результатов.

Код программы

main.c

```
#include <unistd.h>
#include <stdlib.h>
#include <pthread.h>
#include <string.h>

#define MAX_N 10
#define MAX_LINE_LENGTH 100

double matrix[MAX_N][MAX_N + 1];
int n, numThreads;
pthread_mutex_t mutex;
pthread_cond_t cond;
int counter = 0;

void* PerformGaussianElimination(void* arg) {
    int threadId = *(int*)arg;
    for (int k = 0; k < n; k++) {
```

```
        if (k % numThreads == threadId) {
            pthread_mutex_lock(&mutex);
            double pivot = matrix[k][k];
            for (int j = k; j <= n; j++) {
                matrix[k][j] /= pivot;
            }
            pthread_mutex_unlock(&mutex);
        }
    }
    return NULL;
}

int main() {
    char inputLine[MAX_LINE_LENGTH];

    read(0, inputLine, sizeof(inputLine));
    n = atoi(inputLine);

    read(0, inputLine, sizeof(inputLine));
    numThreads = atoi(inputLine);

    pthread_t threads[MAX_N];
    int threadIds[MAX_N];
    pthread_mutex_init(&mutex, NULL);
    pthread_cond_init(&cond, NULL);

    for (int i = 0; i < numThreads; i++) {
        threadIds[i] = i;
        pthread_create(&threads[i], NULL, PerformGaussianElimination, &threadIds[i]);
    }

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

    pthread_mutex_destroy(&mutex);
    pthread_cond_destroy(&cond);

    return 0;
}
```

Тестирование

ivankamblshnikov@MacBook-Air-Rec0il-y lab2 % sudo dtruss -f ./main

Password:
PID/THRD SYSCALL(args) = return
Введите количество переменных (n <= 10): 1642/0x808c: fork() = 0 0
1642/0x808c: munmap(0x10300C000, 0x98000) = 0 0
1642/0x808c: munmap(0x1030A4000, 0x8000) = 0 0
1642/0x808c: munmap(0x1030AC000, 0x4000) = 0 0
1642/0x808c: munmap(0x1030B0000, 0x4000) = 0 0
1642/0x808c: munmap(0x1030B4000, 0x5C000) = 0 0
1642/0x808c: crossarch_trap(0x0, 0x0, 0x0) = -1 Err#45
1642/0x808c: open(".\0", 0x100000, 0x0) = 3 0
1642/0x808c: fcntl(0x3, 0x32, 0x16CFFB2A8) = 0 0
1642/0x808c: close(0x3) = 0 0
1642/0x808c: fsgetpath(0x16CFFB2B8, 0x400, 0x16CFFB298) = 47 0
1642/0x808c: fsgetpath(0x16CFFB2C8, 0x400, 0x16CFFB2A8) = 14 0
1642/0x808c: csrctl(0x0, 0x16CFFB6CC, 0x4) = -1 Err#1
1642/0x808c: __mac_syscall(0x18555E908, 0x2, 0x16CFFB610) = 0 0
1642/0x808c: csrctl(0x0, 0x16CFFB6EC, 0x4) = -1 Err#1
1642/0x808c: __mac_syscall(0x18555B75E, 0x5A, 0x16CFFB680) = 0 0
1642/0x808c: sysctl([unknown, 3, 0, 0, 0, 0] (2), 0x16CFFABE8, 0x16CFFABE0,
0x18555D3AF, 0xD) = 0 0
1642/0x808c: sysctl([CTL_KERN, 140, 0, 0, 0, 0] (2), 0x16CFFAC98,
0x16CFFAC90, 0x0, 0x0) = 0 0
1642/0x808c: open("/\0", 0x20100000, 0x0) = 3 0
1642/0x808c: openat(0x3, "System/Cryptexes/OS\0", 0x100000, 0x0) = 4 0
1642/0x808c: dup(0x4, 0x0, 0x0) = 5 0
1642/0x808c: fstatat64(0x4, 0x16CFFA771, 0x16CFFA6E0) = 0 0
1642/0x808c: openat(0x4, "System/Library/dyld/\0", 0x100000, 0x0) = 6 0
1642/0x808c: fcntl(0x6, 0x32, 0x16CFFA770) = 0 0
1642/0x808c: dup(0x6, 0x0, 0x0) = 7 0
1642/0x808c: dup(0x5, 0x0, 0x0) = 8 0
1642/0x808c: close(0x3) = 0 0
1642/0x808c: close(0x5) = 0 0
1642/0x808c: close(0x4) = 0 0
1642/0x808c: close(0x6) = 0 0
1642/0x808c: shared_region_check_np(0x16CFFAD80, 0x0, 0x0) = 0 0
1642/0x808c: fsgetpath(0x16CFFB2D0, 0x400, 0x16CFFB228) = 82 0

1642/0x808c: fcntl(0x8, 0x32, 0x16CFFB2D0) = 0 0
1642/0x808c: close(0x8) = 0 0
1642/0x808c: close(0x7) = 0 0
1642/0x808c: getfsstat64(0x0, 0x0, 0x2) = 10 0
1642/0x808c: getfsstat64(0x10321A080, 0x54B0, 0x2) = 10 0
1642/0x808c: getattrlist("/\0", 0x16CFFB200, 0x16CFFB170) = 0 0
1642/0x808c:
stat64("/System/Volumes/Preboot/Cryptexes/OS/System/Library/dyld/dyld_shared_cache_arm64e\0", 0x16CFFB560, 0x0) = 0 0
dtrace: error on enabled probe ID 1690 (ID 845: syscall::stat64:return): invalid address (0x0) in action #12 at DIF offset 12
1642/0x808c: stat64("/Users/ivankamblshnikov/Documents/os/lab2/main\0", 0x16CFFAA10, 0x0) = 0 0
1642/0x808c: open("/Users/ivankamblshnikov/Documents/os/lab2/main\0", 0x0, 0x0) = 3 0
1642/0x808c: mmap(0x0, 0x89E8, 0x1, 0x40002, 0x3, 0x0) = 0x10325C000 0
1642/0x808c: fcntl(0x3, 0x32, 0x16CFFAB28) = 0 0
1642/0x808c: close(0x3) = 0 0
1642/0x808c: munmap(0x10325C000, 0x89E8) = 0 0
1642/0x808c: stat64("/Users/ivankamblshnikov/Documents/os/lab2/main\0", 0x16CFFAF80, 0x0) = 0 0
1642/0x808c: stat64("/usr/lib/libSystem.B.dylib\0", 0x16CFF9F10, 0x0) = -1 Err#2
1642/0x808c:
stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/libSystem.B.dylib\0", 0x16CFF9EC0, 0x0) = -1 Err#2
1642/0x808c: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16CFF7B20, 0x0) = -1 Err#2
1642/0x808c:
stat64("/System/Volumes/Preboot/Cryptexes/OS/usr/lib/system/libdispatch.dylib\0", 0x16CFF7AD0, 0x0) = -1 Err#2
1642/0x808c: stat64("/usr/lib/system/libdispatch.dylib\0", 0x16CFF7B20, 0x0) = -1 Err#2
1642/0x808c: open("/dev/dtracehelper\0", 0x2, 0x0) = 3 0
1642/0x808c: ioctl(0x3, 0x80086804, 0x16CFF9B68) = 0 0
1642/0x808c: close(0x3) = 0 0
1642/0x808c: open("/Users/ivankamblshnikov/Documents/os/lab2/main\0", 0x0, 0x0) = 3 0
1642/0x808c: __mac_syscall(0x18555E908, 0x2, 0x16CFF91E0) = 0 0
1642/0x808c: map_with_linking_np(0x16CFF9070, 0x1, 0x16CFF90A0) = 0 0
1642/0x808c: close(0x3) = 0 0

1642/0x808c: mprotect(0x102E08000, 0x4000, 0x1) = 0 0
1642/0x808c: shared_region_check_np(0xFFFFFFFFFFFFFFFF, 0x0, 0x0) = 0 0

Вывод

Использование языка C и библиотеки pthread позволяет создавать эффективные многопоточные приложения. Синхронизация потоков с помощью мьютексов и условных переменных обеспечивает корректное выполнение алгоритма Гаусса. В результате программа демонстрирует ускорение вычислений при увеличении количества потоков.