

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу
«Операционные системы»**

Студент: Кривошапкин Егор Борисович
Группа: М8О-209Б-23
Вариант: 8
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Содержание

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

Репозиторий

https://github.com/EgorX2000/os_labs/tree/main/2

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Управление потоками в ОС
- Обеспечение синхронизации между потоками

Задание

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска программы.

Необходимо уметь продемонстрировать количество потоков, используемых программой, с помощью стандартных средств операционной системы. Привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Объяснить получившиеся результаты.

Вариант 8: Есть K массивов одинаковой длины. Необходимо сложить эти массивы. Необходимо предусмотреть стратегию, адаптирующуюся под количество массивов и их длину (по количеству операций)

Общие сведения о программе

Программа написана на языке Си в операционной системе Windows. Для запуска программы в качестве аргумента командной строки необходимо указать максимальное количество используемых потоков.

Общий метод и алгоритм решения

Пусть на вход от пользователя поступило n потоков. Тогда каждый поток будет обрабатывать k/n массивов, оставшиеся $k \% n$ также равномерно распределяются по потокам. Каждый поток поэлементно прибавляет доступные ему массивы к массиву-результату. Если заданное пользователем максимальное количество потоков превышает количество исходных массивов, то потоки создаются в количестве, равном количеству массивов, чтобы не создавались лишние потоки.

Исходный код

main.cpp

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

#define ARRAY_SIZE 20000

#define MAX_ARRAYS 100000

typedef struct {
    int* result;
    int** arrays;
    int start_index;
    int end_index;
    int array_length;
} ThreadData;

DWORD WINAPI SumArrays(LPVOID param) {
    ThreadData* data = (ThreadData*)param;

    for (int i = data->start_index; i < data->end_index; i++) {
        for (int j = 0; j < data->array_length; j++) {
            data->result[j] += data->arrays[i][j];
        }
    }

    return 0;
}

double getTime() {
    LARGE_INTEGER freq, val;

    QueryPerformanceFrequency(&freq);
```

```

    QueryPerformanceCounter(&val);
    return (double)val.QuadPart / (double)freq.QuadPart;
}

int main(int argc, char* argv[]) {
    double start_time = getTime();

    if (argc < 2) {
        printf("Usage: %s <max_threads>\n", argv[0]);
        return 1;
    }

    int max_threads = atoi(argv[1]);
    if (max_threads <= 0) {
        printf("Invalid max_threads value.\n");
        return 1;
    }

    int** arrays = (int**)malloc(MAX_ARRAYS * sizeof(int*));
    // FILE* data = fopen("data.txt", "w");
    // srand(time(NULL));
    for (int i = 0; i < MAX_ARRAYS; i++) {
        arrays[i] = (int*)malloc(ARRAY_SIZE * sizeof(int));
        for (int j = 0; j < ARRAY_SIZE; j++) {
            arrays[i][j] = rand() % 100;
            // fprintf(data, "%d ", arrays[i][j]);
        }
        // fputc('\n', data);
    }

    int* result = (int*)calloc(ARRAY_SIZE, sizeof(int));

    int threads_to_use = (MAX_ARRAYS < max_threads) ? MAX_ARRAYS : max_threads;

```

```

int arrays_per_thread = MAX_ARRAYS / threads_to_use;
int remaining_arrays = MAX_ARRAYS % threads_to_use;

HANDLE* threads = (HANDLE*)malloc(threads_to_use * sizeof(HANDLE));
ThreadData* thread_data =
    (ThreadData*)malloc(threads_to_use * sizeof(ThreadData));

int current_array = 0;

for (int i = 0; i < threads_to_use; i++) {
    thread_data[i].result = result;
    thread_data[i].arrays = arrays;
    thread_data[i].array_length = ARRAY_SIZE;

    thread_data[i].start_index = current_array;
    if (i < remaining_arrays) {
        current_array += arrays_per_thread + 1;
    } else {
        current_array += arrays_per_thread;
    }
    thread_data[i].end_index = current_array;

    threads[i] = CreateThread(NULL, 0, SumArrays, &thread_data[i], 0, NULL);
}

WaitForMultipleObjects(threads_to_use, threads, TRUE, INFINITE);

/*
for (int i = 0; i < ARRAY_SIZE; i++) {
    printf("%d ", result[i]);
}
putchar('\n');

```

```

for (int i = 0; i < MAX_ARRAYS; i++) {
    CloseHandle(threads[i]);
    free(arrays[i]);
}
*/

free(arrays);
free(result);
free(threads);
free(thread_data);

printf("%0.2lfms\n", (getTime() - start_time) * 1000);

return 0;
}

```

Демонстрация работы программы

```

PS C:\Users\asUS\Documents\C++\os_labs\2\src> cd .\build\
PS C:\Users\asUS\Documents\C++\os_labs\2\src\build> ./main 1
33458.84ms
PS C:\Users\asUS\Documents\C++\os_labs\2\src\build> ./main 2
31183.74ms
PS C:\Users\asUS\Documents\C++\os_labs\2\src\build> ./main 5
30220.99ms
PS C:\Users\asUS\Documents\C++\os_labs\2\src\build> ./main 10
29515.58ms

```

Выводы

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого при работе на системе Windows требуется подключить библиотеку `<windows.h>`.

Создание потоков происходит быстрее, чем создание процессов, а все потоки используют одну и ту же область данных. Поэтому многопоточность – один

из способов ускорить обработку каких-либо данных: выполнение однотипных, не зависящих друг от друга задач, можно поручить отдельным потокам, которые будут работать параллельно.

Средствами языка Си можно совершать системные запросы на создание потока, ожидания завершения потока, а также использовать различные примитивы синхронизации.