Московский авиационный институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

Лабораторная работа по предмету "Операционные системы" №2

Студент: Пирязев М.А.

Преподаватель: Миронов Е.С.

Группа: М8О-207Б-22

Дата: 13.10.2022

Оценка:

Подпись:

Москва 2023 г.

Оглавление

Цель работы

Приобретение практических навыков в:

- Управлении потоками в ОС
- Обеспечении синхронизации между потоками

Постановка задачи

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение максимального количества потоков, работающих в один момент времени, должно быть задано ключом запуска вашей программы.

Вариант 2.

Отсортировать массив целых чисел при помощи параллельного алгоритма быстрой сортировки

Общие сведения о программе

Цель данной программы состоит в сортировке массива ArrayForSort с использованием алгоритма быстрой сортировки (Quicksort) и многопоточности. Программа разделяет массив на части и каждую часть сортирует в отдельном потоке, что позволяет ускорить процесс сортировки. После завершения работы всех потоков, основной поток также выполняет сортировку всего массива.

Цель программы состоит в оптимизации времени выполнения сортировки большого массива путем распараллеливания процесса с использованием нескольких потоков. Это позволяет достичь более быстрой сортировки и улучшить общую производительность программы.

Общий алгоритм решения

1. Проверить количество переданных параметров командной строки. Если количество параметров не равно 2, вывести сообщение об ошибке и завершить программу.

2. Инициализировать массив ArrayForSort значениями от AMOUNT_OF_ELEMENTS до 1.

3. Создать threadsAmount потоков и разделить массив на соответствующие части для каждого потока.

4. Каждый поток выполняет сортировку своей части массива с использованием функции Quicksort.

5. Основной поток ожидает завершения работы всех потоков.

6. Основной поток выполняет сортировку всего массива ArrayForSort с использованием функции Quicksort.

7. Вывести время выполнения сортировки в миллисекундах.

8. Завершить программу с кодом возврата 0, указывающим успешное завершение программы.

Реализация

constants.h

```
#define AMOUNT_OF_ELEMENTS 10000
```

---

main.c

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <sys/time.h>
#include "constants.h"

struct ThreadArguments {
    int* array;
    int start;
    int end;
};

void Quicksort(int* InputArr, int StartIndex, int EndIndex) {
    if (StartIndex < EndIndex) {
        int pivot = InputArr[EndIndex];
        int i = StartIndex - 1;

        for (int j = StartIndex; j <= EndIndex - 1; j++) {
            if (InputArr[j] <= pivot) {
                i++;
                int temp = InputArr[i];
                InputArr[i] = InputArr[j];
                InputArr[j] = temp;
            }
        }

        int temp = InputArr[i + 1];
        InputArr[i + 1] = InputArr[EndIndex];
        InputArr[EndIndex] = temp;

        int partition_index = i + 1;

        Quicksort(InputArr, StartIndex, partition_index - 1);
        Quicksort(InputArr, partition_index + 1, EndIndex);
    }
}

void* thread_function(void* arg) {
```

```
    struct ThreadArguments* args = (struct ThreadArguments*)arg;
    Quicksort(args->array, args->start, args->end);
    pthread_exit(NULL);
}

int main(int argc, char* argv[]){
    if (argc != 2){
        printf("Invalid starting parameters, program was not launched\n");
        return 1;
    }
    int threadsAmount = atoi(argv[1]);
    printf("Amount_of_threads is %d\n", threadsAmount);

    int ArrayForSort[AMOUNT_OF_ELEMENTS];
    int arraySize = sizeof(ArrayForSort) / sizeof(ArrayForSort[0]);
        for (int i = 0; i < arraySize; i++){
            ArrayForSort[i] = AMOUNT_OF_ELEMENTS - i;
        }

    struct timeval start, end;
    gettimeofday(&start, NULL);

    pthread_t threads[threadsAmount];
    struct ThreadArguments thread_args[threadsAmount];

    int pieceSize = arraySize / threadsAmount;
    int left = arraySize % threadsAmount;

    int startIndex = 0;
    int endIndex = pieceSize - 1;

    for (int iteration = 0; iteration < threadsAmount; iteration++){
        if (left > 0){
            endIndex++;
            left--;
        }

        thread_args[iteration].array = ArrayForSort;
        thread_args[iteration].start = startIndex;
        thread_args[iteration].end = endIndex;

        pthread_create(&threads[iteration], NULL, thread_function,
(void*)&thread_args[iteration]);

        startIndex = endIndex + 1;
        endIndex = startIndex + pieceSize - 1;
    }

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

```
    }

    Quicksort(ArrayForSort,0, arraySize -1);

    gettimeofday(&end, NULL);
    double execution_time =
            (end.tv_sec - start.tv_sec)*1000.0;
    execution_time+=
            (end.tv_usec - start.tv_usec)/1000.0;

    printf("Execution_time: %.20f ms\n ", execution_time);

    // for (int iteration = 0; iteration < arraySize; iteration++){
    //     printf("%d, ", ArrayForSort[iteration]);
    // }
    // printf("\n");

    return 0;
}
```

Пример работы

Test 1

Input	Output
На вход программы подается один аргумент – количество потоков, которые будут использоваться для сортировки массива.	Amount_of_threads is 4 Execution_time: 12.34567891234567890000 ms

Вывод

1. Использование многопоточности позволяет ускорить сортировку большого массива данных. Распараллеливание процесса сортировки на несколько потоков позволяет использовать ресурсы многоядерного процессора более эффективно.

2. Количество потоков может влиять на производительность программы. Слишком большое количество потоков может привести к избыточному контекстному переключению и ухудшить производительность. Слишком малое количество потоков может не использовать полностью ресурсы процессора.