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: Ядров Артем Леонидович
Группа: М8О-208Б-20
Вариант: 12
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/Yadroff/OS/tree/master/2_lab

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 16: Задаётся радиус окружности. Необходимо с помощью метода Монте-Карло рассчитать её площадь

Общие сведения о программе

Программа написана на языке Си в UNIX-подобной операционной системе (Fedora 34). Для компиляции программы требуется указать ключ `-pthread`. Для запуска программы в качестве 1 аргумента командной строки необходимо указать радиус окружности, в качестве 2 аргумента - количество проверяемых точек, в качестве 3 аргумента - количество потоков.

Общий метод и алгоритм решения

Пусть на вход от пользователя поступило n точек и m потоков. Тогда каждый поток будет обрабатывать n/m точек. Чтобы избежать работы с критической областью памяти будем хранить количество точек, попавших в круг и обрабатываемых i-м потоком, в динамическом массиве N. Тогда справедлива следующая формула:

$\frac{S_{кр}}{S_{кв}} = \frac{\sum_{i=0}^{m-1} N[i]}{n}$, которая равносильна формуле:

$\frac{S_{кр}}{S_{кв}} = \sum_{i=1}^{m-1} \frac{N[i]}{n}$, где $S_{кв}$ — площадь описанного вокруг окружности квадрата, то есть $S_{кв} = (2*R)^2 = 4*R^2$, где R — радиус окружности.

Итак, каждый поток обрабатывает n/m точек. Для этого генерируется случайная точка с координатами x, y, лежащая в пределах квадрата $(-r \leqslant x \leqslant r, -r \leqslant y \leqslant r)$. При этом если точка лежит в пределах круга (удовлетворяет неравенству $x^2 + y^2 \leqslant R^2$) , то инкрементируется значение N[i]. В качестве аргумента поток принимает количество обрабатываемых точек и номер i. После завершения работы всех потоков выводится площадь, вычисленная методом Монте-Карло и стандартным методом.

Исходный код

main.cpp

```
#include <stdio.h>
#include <math.h>
#include <pthread.h>
#include <stdlib.h>
#include <time.h>

int R;
int *N;
typedef struct arguments {
    int points;
    int i;
} Arg;

double get_rand() { // return random double from 0 to 1
```

```
    return ((double) rand()) / RAND_MAX;
}

double get_rand_range(double min, double max) { // return random double from min to max
    return get_rand() * (max - min) + min;
}

void *thread_function(void *args) { // create n random points of square size 2*R and check if point at circle
    Arg *arg = (Arg *) args;
    int n = arg->points;
    int i = arg->i;
    for (int j = 0; j < n; j++) {
        double x = get_rand_range(-R, R);
        double y = get_rand_range(-R, R);
        if (x * x + y * y <= R * R) {
            N[i]++;
        }
    }
    return NULL;
}

int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("Syntax: ./*executable_file_name* Radius Number_of_points Number_of_threads\n");
        exit(1);
    }
    R = atoi(argv[1]);
```

```
int points_num = atoi(argv[2]), threads_num = atoi(argv[3]);
N = (int *) calloc(threads_num, sizeof(int)); // array of number points at circle
srand(time(NULL));
pthread_t *threads = (pthread_t *) calloc(threads_num, sizeof(pthread_t));
if (threads == NULL) {
    printf("Can't allocate memory for threads\n");
    exit(1);
}
int points_for_thread = points_num / threads_num;
Arg a;
for (int i = 0; i < threads_num; i++) {
    a.points = points_for_thread + (i < (points_num % threads_num));
    a.i = i;
    if (pthread_create(&threads[i], NULL, thread_function, &a) != 0) {
        printf("Can not create thread\n");
        exit(1);
    }
}
for (int i = 0; i < threads_num; i++) {
    if (pthread_join(threads[i], NULL) != 0) {
        printf("Join error\n");
        exit(1);
    }
}
double n = 0;
for (int i = 0; i < threads_num; i++) { // calculate points at circle
    n += N[i] / 1.0 / points_num;
}
printf("Monte-Carlo Circle square is %.5f\n",
```

```
        (double) 4 * R * R * n); // (Circle Square)/(Square of square with size 2*R)
= N/M, where M = points_num
    printf("Real Circle square is %.5f\n", (double) M_PI * R * R);
    free(threads);
    return 0;
}
```

Демонстрация работы программы

```
[Temi4@localhost OS]$ cd 3_lab/src/
[Temi4@localhost src]$ gcc main.c -pthread
[Temi4@localhost src]$ ./a.out
Syntax: ./*executable_file_name* Radius Number_of_points
Number_of_threads
[Temi4@localhost src]$ ./a.out 1 100 5
Monte-Carlo Circle square is 3.12000
Real Circle square is 3.14159
[Temi4@localhost src]$ ./a.out 1 100000 8
Monte-Carlo Circle square is 3.14156
Real Circle square is 3.14159
[Temi4@localhost src]$ ./a.out 4 100000 8
Monte-Carlo Circle square is 50.08896
Real Circle square is 50.26548
```

Выводы

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого на Unix-подобных системах требуется подключить библиотеку pthread.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.