Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Операционные системы»

Управление потоками в ОС. Обеспечение синхронизации между потоками

Студент: А. Р. Боташев
Преподаватель: Е. С. Миронов
Группа: М8О-201Б-21
Вариант: 16
Дата:
Оценка:
Подпись:

Москва, 2023

1 Постановка задачи

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

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

Задается радиус окружности. Необходимо с помощью метода Монте-Карло расчитать ее площадь

2 Сведения о программе

Программа написанна на Си в Unix подобной операционной системе на базе ядра Linux. Для компиляции требуется ключ -lpthread, для запуска программы нелбходимо указать в качестве аргумента количество потоков, которые максимально могут быть использованы.

Программа считывает со стандартного потока ввода радиус окружности и количество потоков, которые нужно создать.

Программа выводит в стандартный поток вычисленную площадь окружности методом Монте-Карло

3 Общий метод и алгоритм решения

Введем систему координат, центром которой будет центр окружности. Мысленно очерчим вокруг окружности квадрат со стороной 2r, так, чтобы окружность целиком находилась внутри квадрата.

Пусть изначально n = 0. Программа 10^7 раз выбирает случайную точку внутри квадрата. Если эта точка находится внутри окружности, то значение n увеличивается на 1. Искомая площадь будет равна $\frac{n}{10^7}$

Если m - количество используемых потоков, в таком случае количество попыток выбора случайной точки уменьшается до $\frac{10^7}{m}$

4 Листинг программы

main.cpp

```cpp
#include <iostream>
#include <pthread.h>
#include <vector>
#include "utils.h"
#include "lab3.h"
#include <chrono>


int main() {
    int threadCount;
    double r;
    std::cin >> r >> threadCount;

    int total = 0, success = 0, limit = 1e7;
    srandom(time(nullptr));

    std::vector<pthread_t> p(threadCount);
    std::vector<Args> a;

    for (int i = 0; i < threadCount; i++) {
        a.push_back({r, total, success, (limit + threadCount - 1) / threadCount});
    }

    auto start = std::chrono::high_resolution_clock::now();

    for (int i = 0; i < threadCount; i++) {
        pthread_create(&p[i], nullptr, &CalculateArea, &a[i]);
    }

    for (int i = 0; i < threadCount; i++) {
        pthread_join(p[i], nullptr);
    }

    auto end = std::chrono::high_resolution_clock::now();
    auto searchTime = std::chrono::duration_cast<std::chrono::milliseconds>(end - start
        ).count();


    int resSuccess = 0;
    int resTotal = 0;
    for (int i = 0; i < threadCount; i++) {
        resSuccess += a[i].success;
        resTotal += a[i].total;
    }

    std::cout << resSuccess * 4 * r * r / (double) resTotal << " " << searchTime;
```

```
46      return 0;
47  }
```

lab3.cpp

```
1   #include "lab3.h"
2   #include "utils.h"
3
4
5   bool InCircle(double x, double y, double r) {
6       if (x * x + y * y <= r * r + EPS) {
7           return true;
8       }
9       return false;
10  }
11
12  void *CalculateArea(void *args) {
13      auto *a = (struct Args *) args;
14      for (int i = 0; i < a->limit; i++) {
15          a->total++;
16          a->success += InCircle(GetRandomNumber(a->r),
17                                 GetRandomNumber(a->r),
18                                 a->r);
19      }
20      return a;
21  }
```

utils.cpp

```
1   #include "utils.h"
2
3
4   double GetRandomNumber(double max) {
5       return -max + (double) (random() % (long) 1e5) / 1e5 * max * 2;
6   }
```

5 Демонстрация работы программы

```
botashev@botashev-laptop:~/ClionProjects/os_labs/tests$ cat lab3_test.cpp
#include <gtest/gtest.h>

#include <lab3.h>
#include <utils.h>
#include <cmath>


TEST(ThirdLabTests,GetRandomNumberCorrectResults) {
```

```
EXPECT_LE(abs(GetRandomNumber(0)),0);

EXPECT_LE(abs(GetRandomNumber(1)),1);

EXPECT_LE(abs(GetRandomNumber(1000000)),1000000);

EXPECT_LE(abs(GetRandomNumber(99999.999)),99999.999);

EXPECT_LE(abs(GetRandomNumber(0.0001)),0.0001);

EXPECT_LE(abs(GetRandomNumber(7)),7);

EXPECT_LE(abs(GetRandomNumber(123.4567)),123.4567);
}


TEST(ThirdLabTests,InCircleCorrectResults) {
EXPECT_EQ(InCircle(1,0,1),true);
EXPECT_EQ(InCircle(0,1,1),true);
EXPECT_EQ(InCircle(-1,0,1),true);
EXPECT_EQ(InCircle(0,-1,1),true);
EXPECT_EQ(InCircle(1,1,1),false);
EXPECT_EQ(InCircle(-1,-1,1),false);
EXPECT_EQ(InCircle(99999,-99999,1),false);
EXPECT_EQ(InCircle(-99999,99999,1),false);

EXPECT_EQ(InCircle(0,0,0.1),true);
EXPECT_EQ(InCircle(-0.000001,-0.000001,0.1),true);
EXPECT_EQ(InCircle(-0.1,-0.1,0.000001),false);

EXPECT_EQ(InCircle(3,4,5),true);
EXPECT_EQ(InCircle(3.00001,4.00000001,5),false);

EXPECT_EQ(InCircle(1234.5678,9876.54321,99999999),true);

}

botashev@botashev-laptop:~/ClionProjects/os_labs/tests$ ./../cmake-build-debug/tests/l
Running main() from /home/botashev/ClionProjects/os_labs/cmake-build-debug/_deps/googl
[==========] Running 2 tests from 1 test suite.
[----------] Global test environment set-up.
```

```
[----------] 2 tests from ThirdLabTests
[ RUN      ] ThirdLabTests.GetRandomNumberCorrectResults
[       OK ] ThirdLabTests.GetRandomNumberCorrectResults (0 ms)
[ RUN      ] ThirdLabTests.InCircleCorrectResults
[       OK ] ThirdLabTests.InCircleCorrectResults (0 ms)
[----------] 2 tests from ThirdLabTests (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 2 tests.
```

6 Вывод

Многие языки программирования позволяют пользователю работать с потоками. Создание потоков происходит быстрее, чем создание процессов, за счет того, что при создании потока не копируется область памяти, а они все работают с одной областью памяти. Поэтому многопоточность используют для ускарения не зависящих друг от друга, однотипнях задач, которые будут работать параллельно.

Язык Си предоставляет данный функционал пользователям Unix-подобных операционных систем с помощью библиотеки pthread.h. Средствами языка Си можно совершать системные запросы на создание и ожидания завершения потока, а также использовать различные примитивы синхронизации.