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

Лабораторная работа №3 по курсу

«Операционные системы»

Тема работы

“Потоки”

Студент: Ханнанов Руслан Маратович
Группа: М8О-208Б-20
Вариант:8
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

https://github.com/Naksen/OS

Постановка задачи

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

Общие сведения о программе

Для реализации поставленной задачи нам нужны следующие библиотеки:

<iostream> - заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++.

<chrono> - для функций, работающих со временем.

<thread> - для работы с потоками.

<vector> - для работы с динамическими массивами - векторами.

<string> - для работы со строками.

<sstream> - заголовочный файл с классами, функциями и переменными для организации работы со строками через интерфейс потоков.

Для работы с потоками я использую библиотеку thread в C++. Я создаю вектор потоков и заполняю его ими по необходимости. В моей реализации потоки работают с регулярными фунцкиями, а именно с фунцкией sum, как раз и производящей нужные вычисления. Вызовом join я блокирую вызывающий поток до тех пор, пока он не выполнит работу. С помощью библиотеки chrono я замеряю время выполнения нужных вычислений для сравнения между собой запуска программы с разным количеством потоков.

Общий метод и алгоритм решения

Для реализации заданной задачи я использую массивы с уже заданными длинами, этим я избегаю “лишнего” ввода, который при необходимости можно добавить достаточно быстро. В программе есть такие переменные: k – количество массивов и n – их длина. Все массивы я заполняю единицами, для облегчения проверки. После программе подается на вход количество потоков, число которых не должно превышать 8, в таком случае оно уменьшается до заданного значения. Сделано это по причине отсутствия явного ускорения работы после излишнего увеличения числа потоков. Объем работы между потоками я разделяю поровну в зависимости от их количества. Таким образом каждый поток производит суммирования всех массивов только на диапазоне с определенными для него индексами, что позволяет избежать вмешательство одного потока к информации задействованной другим потоком. Каждый поток запускается с регулярной функцией sum, которая производит суммирования всех массивов на определенном диапазоне индексов.

Исходный код

```cpp
#include <iostream>
#include <thread>
#include <vector>
#include <string>
#include <sstream>
#include <chrono>

void Sum(int left, int right, const std::vector<std::vector<int>>& v, std::vector<int>& ans) {
    for (int i = left; i < right; ++i) {
        for (int j = 0; j < v.size(); ++j) {
            ans[i] += v[j][i];
        }
    }
}

int main(){
    int k = 10000; // Number of Arrays
    int n = 10000; // Length of Arrays

    std::vector<std::vector<int>> v(k,std::vector<int>(n));
    std::vector<int> ans(v[0].size());

    for (size_t i = 0; i < k; ++i) {
        for (size_t j = 0; j < n; ++j) {
            v[i][j] = 1;
        }
    }

    std::cout << "Enter number of threads:\n";
    int threads_num;
    std::cin >> threads_num;
    if (threads_num > n) {
        threads_num = n;
    }
    if (threads_num > 8) {
        threads_num = 8;
    }

    auto begin = std::chrono::steady_clock::now(); // Initial moment of time
    if (threads_num == 0) {
        for (size_t i = 0; i < n; ++i) {
            for (size_t j = 0; j < k; ++j) {
                ans[i] += v[j][i];
            }
        }
    } else {
        std::vector<std::thread> th;
        int left = 0;
        int delta = n / threads_num;
        int right = delta;
        th.reserve(threads_num);
        for (int i = 0; i < threads_num; ++i) {
            th.emplace_back(Sum, left, right,std::ref(v), std::ref(ans));
            left = right;
            right = right + delta;
            if (i == threads_num - 2) {
                right = n;
            }
        }
        for (int i = 0; i < threads_num; ++i) {
            th[i].join();
        }
    }
    auto end = std::chrono::steady_clock::now(); // End moment of time

    for (size_t i = 0; i < n; ++i) {

        std::cout << ans[i] << " ";

    }

    auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
    std::cout << "\nThe time: " << elapsed_ms.count() << " ms\n";
}
```

Демонстрация работы программы

Тест 1:

5

```
C:\Windows\system32\wsl.exe --distribution Ubuntu-20.04 --exec /bin/sh -c "cd /mnt/d/CProgramms/tests/test_lab3_os/cmake-build-debug &&
 /mnt/d/CProgramms/tests/test_lab3_os/cmake-build-debug/test_lab3_os; sleep 0.001"
Enter number of threads:
0

The time: 1715 ms

Process finished with exit code 0
```

Тест 2:

```
C:\Windows\system32\wsl.exe --distribution Ubuntu-20.04 --exec /bin/sh -c "cd /mnt/d/CProgramms/tests/test_lab3_os/cmake-build-debug &&
 /mnt/d/CProgramms/tests/test_lab3_os/cmake-build-debug/test_lab3_os; sleep 0.001"
Enter number of threads:
6

The time: 563 ms

Process finished with exit code 0
```

Выводы

Благодаря данной лабораторной я успешно ознакомился с работой потоков в Linux и тем, как они устроены. Во время выполнения своего задания я изучил многие системные вызовы и научился применять их в своей программе, а также я узнал многие тонкости работы с потоками.