МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет «ИТМО» (НИУ ИТМО)

Лабораторная работа №4 «C++ PROCESSES/THREADS» по курсу «C++ и UNIX системы»

Выполнил студент 3 курса группы К3333: Буданцев А.А.

Проверил: Маслов И.Д.

Санкт-Петербург - 2023

Цель работы

Познакомить студента с принципами параллельных вычислений. Составить несколько программ в простейшими вычислительными действиями, чтобы освоить принципы параллельных вычислений (когда одни алгоритмы зависят / не зависят от других).

Задача

1. [C++ SEQUENCE] Последовательные вычисления

Требуется последовательно выполнить вычисления по формуле 1, вычисления по формуле 2, после чего выполнить вычисления по формуле 3, которые выглядят следующим образом: результат вычислений 1 + результат вычислений 2 – результат вычислений 1.

Выполнить последовательно на 10 000 итераций и 100 000 итераций

Формула 1: f(x) = x^2-x^2+x*4-x*5+x+x

Формула 2: f(x) = x+x

Вывести длительность выполнения всех 10 000 итераций и 100 000 итераций в сек.

2. [C++ THREADS] Параллельные вычисления через потоки

Требуется параллельно (насколько возможно с помощью потоков) выполнить вычисления по формуле 1, вычисления по формуле 2, после чего выполнить вычисления по формуле 3, которые выглядят следующим образом: результат вычислений 1 + результат вычислений 2 – результат вычислений 1

Выполнить последовательно на 10 000 итераций и 100 000 итераций

Формула 1: f(x) = x^2-x^2+x*4-x*5+x+x

Формула 2: f(x) = x+x

Вывести длительность выполнения всех 10 000 итераций и 100 000 итераций в сек. в разбивке по шагам вычислений 1, 2 и 3

3. [C++ PROCESS] Параллельные вычисления через процессы

Требуется параллельно (насколько возможно с помощью процессов) выполнить вычисления по формуле 1, вычисления по формуле 2, после чего выполнить вычисления по формуле 3, которые выглядят следующим образом: результат вычислений 1 + результат вычислений 2 – результат вычислений 1

Выполнить последовательно на 10 000 итераций и 100 000 итераций

Формула 1: f(x) = x^2-x^2+x*4-x*5+x+x

Формула 2: f(x) = x+x

Вывести длительность выполнения всех 10 000 итераций и 100 000 итераций в сек. в разбивке по шагам вычислений 1, 2 и 3

4. **[SAVE] Результат всех вышеперечисленных шагов сохранить в репозиторий (+ отчет по данной ЛР в папку doc)**

Фиксацию ревизий производить строго через ветку dev. С помощью скриптов накатить ревизии на stg и на prd. Скрипты разместить в корне репозитория. Также создать скрипты по возврату к виду текущей ревизии (даже если в папке имеются несохраненные изменения + новые файлы).

Решение

1. [C++ SEQUENCE] Последовательные вычисления

```cpp
void calculations(const int N) {
    auto start = chrono::high_resolution_clock::now();
    int first_result[N], second_result[N], third_result[N];

    for(int i=0;i<N;i++)
        first_result[i] = i * i - i * i  + 4 * i  - 5 * i + i + i;

    for(int i=0;i<N;i++)
        second_result[i] = i + i;

    for(int i=0;i<N;i++)
        third_result[i] = first_result[i] + second_result[i] - first_result[i];

    auto end = chrono::high_resolution_clock::now();
    auto duration = chrono::duration<double>(end - start).count();
    cout << "Duration " << fixed << setprecision(4) << duration << " seconds" << "\n";
}
```

2. [C++ THREADS] Параллельные вычисления через потоки

```cpp
void calculations(const int N){
    auto start = chrono::high_resolution_clock::now();
    int first_result[N], second_result[N], third_result[N];

    thread th([&N, &first_result]() {
        for(int i=0;i<N;i++)
            first_result[i] = i * i - i * i  + 4 * i  - 5 * i + i + i;
    });

    thread th1([&N, &second_result]() {
        for(int i=0;i<N;i++)
            second_result[i] = i + i;
    });

    thread th2([&N, &first_result, &second_result, &third_result]() {
        for(int i=0;i<N;i++)
            third_result[i] = first_result[i] + second_result[i] - first_result[i];
    });

    th.join();
    th1.join();
    th2.join();

    auto end = chrono::high_resolution_clock::now();
    auto duration = chrono::duration<double>(end - start).count();
    cout << "Duration " << fixed << setprecision(4) << duration << " seconds" << "\n";
}
```

3. [C++ PROCESS] Параллельные вычисления через процессы

```
if (pid1 == 0) {
    for (int i=0;i<N;i++) {
        close(fd[0]);
        close(fd1[0]);
        close(fd1[1]);
        first_result[i] = i * i - i * i  + 4 * i  - 5 * i + i + i;
        write(fd[1], &first_result, sizeof(first_result));
        close(fd[1]);
    }

    return 0;
}
```

```
if (pid2 == 0) {
    for (int i=0;i<N;i++)  {
        close(fd[0]);
        close(fd1[0]);
        close(fd[1]);
        second_result[i] = i + i;
        write(fd1[1], &second_result, sizeof(second_result));
        close(fd1[1]);
    }

    return 0;
}
```

```
close(fd[1]);
close(fd1[1]);

read(fd[0], &first_result, sizeof(first_result));
read(fd1[0], &second_result, sizeof(second_result));


for(int i=0;i<N;i++)
    third_result[i] = first_result[i] + second_result[i] - first_result[i];

close(fd[0]);
close(fd1[0]);


int status;
waitpid(pid1, &status, 0);
waitpid(pid2, &status, 0);
```

Вывод

При вычислении функции, используя различные способы вычисления, получены следующие результаты:

Без использования функции задержки sleep			
N	Последовательное, сек	Многопоточное, сек	Процессное, сек
10000	0.0003	0.0004	0.018
100000	0.0026	0.0011	0.002

Таким образом, при выполнении небольшого количества итераций достаточно использовать последовательный метод. Однако при больших количествах итераций наиболее эффективным оказался многопоточный метод.