НИУ ВШЭ
Факультет компьютерных наук
Программная инженерия

Построение многопоточного приложения с использованием OpenMP на примере задачи определения множества взаимнопростых чисел

Куприхин Дмитрий Алексеевич, БПИ199

Table of Contents

1. Текст задания

Вариант 14. Определить множество индексов i, для которых A[i] и B[i] не имеют общих делителей (единицу в роли делителя не рассматривать). Входные данные: массивы целых положительных чисел A и B, произвольной длины ≥ 1000. Количество потоков является входным параметром.

2. Применяемый метод

Для решения этой задачи распределим цикл, вычисляющий НОД для каждой пары, по потокам с помощью прагмы. Для каждой пары чисел будем хранить результат вычислений и номер потока, который ее обработал. При таком подходе не будет возникать ситуаций одновременного доступа разных потоков в одни данные.

3. Тестирование

Для тестирования были сгенерированы 4 файла, содержащие одинаковые элементы массивов и различающиеся количеством используемых потоков. Количество элементов во всех файлах равно 1e6. Количество потоков равно 1, 2, 4 и 8. Для тестирования и запуска программ использовался скрипт в командной оболочке Bash.
Результаты тестирования:

```
dima@dimaPC:~/comp_arch/comp_arch_hw4$ ./test
1 thread:
Thread 0 processed 1000000 numbers
There are 608051 coprime numbers in the arrays

real    0m0,223s
user    0m0,219s
sys     0m0,004s

2 threads:
Thread 0 processed 500000 numbers
Thread 1 processed 500000 numbers
There are 608051 coprime numbers in the arrays

real    0m0,174s
user    0m0,230s
sys     0m0,000s

4 threads:
Thread 0 processed 250000 numbers
Thread 1 processed 250000 numbers
Thread 2 processed 250000 numbers
Thread 3 processed 250000 numbers
There are 608051 coprime numbers in the arrays

real    0m0,160s
user    0m0,278s
sys     0m0,000s

8 threads:
Thread 0 processed 125000 numbers
Thread 1 processed 125000 numbers
Thread 2 processed 125000 numbers
Thread 3 processed 125000 numbers
Thread 4 processed 125000 numbers
Thread 5 processed 125000 numbers
Thread 6 processed 125000 numbers
Thread 7 processed 125000 numbers
There are 608051 coprime numbers in the arrays

real    0m0,162s
user    0m0,229s
sys     0m0,004s
dima@dimaPC:~/comp_arch/comp_arch_hw4$
```

Приложение 1. Текст скрипта test.

```
#!/bin/bash
g++ -o hw4 hw4.cpp -fopenmp
printf "1 thread:\n"
time ./hw4 < inputs/input1.txt
printf "\n2 threads:\n"
time ./hw4 < inputs/input2.txt
printf "\n4 threads:\n"
time ./hw4 < inputs/input4.txt
printf "\n8 threads:\n"
time ./hw4 < inputs/input8.txt
```

Приложение 2. Текст файла hw4.cpp

```
#include <vector>
#include <iostream>
#include <omp.h>

// Куприхин Дима, БПИ199, Вариант 14
// Задание: определить такие индексы, для которых a[i] и b[i] не
имееют общих
// делителей больших единицы, т.е. НОД(a[i], b[i]) = 1.
// Входные параметры: размер массивов (>=1000), количество
потоков,
// элементы массивов.

// Для вычисления НОДа будем использовать алгоритм Евклида.
Результаты будем
// сохранять в вектор, в котором значение элемента 1 значит
// взаимнопростость чисел с этим индексом, а 0 - НОД > 1.

void inputIntVector(std::vector<int>& vector) {
    for(int i = 0; i < vector.size(); ++i){
        std::cin >> vector[i];
    }
}

// Алгоритм Евклида для поиска НОД'а двух чисел.
int gcd(int a, int b){
    if(a == 0)
        return b;
    return gcd(b % a, a);
}

int main() {

    std::ios_base::sync_with_stdio(0);
    std::cin.tie(0);
    std::cout.tie(0);

    // Вводим размеры массивов и количество потоков.
    int arraySize, threadsNumber;
    // Утанавливаем false, если не хотим выводить значения
индексов
    // взаимнопростых чисел.
    bool isIndexesOutputEnable = true;
    std::cin >> arraySize >> threadsNumber;
    std::cin >> isIndexesOutputEnable;
```

```
    // Инициализируем и заполняем массивы.
    // потоков и результата.
    std::vector<int> firstArray(arraySize);
    std::vector<int> secondArray(arraySize);
    inputIntVector(firstArray);
    inputIntVector(secondArray);
    std::vector<char> result(arraySize, 0);

    // Массив, хранящий номера потоков, обработовших итерации
цикла.
    std::vector<char> threadIndex(arraySize, -1);

    // Устанавливаем количество потоков из входных данных.
    omp_set_num_threads(threadsNumber);

    // Выполняем параллельлное вычисление НОД'ов чисел массивов.
    #pragma omp parallel for
    for(int i = 0; i < arraySize; ++i) {
        // Добавляем к текущему потоку одну итерацию.
        threadIndex[i] = omp_get_thread_num();
        if(gcd(firstArray[i], secondArray[i]) == 1) {
            result[i] = 1;
        }
    }


    // Массив, хранящий количество проихведенных итераций каждым
потоком.
    std::vector<int> threadIterations(threadsNumber, 0);
    // Считаем количество итераций для каждого потока.
    for(int i = 0; i < arraySize; ++i) {
        ++threadIterations[threadIndex[i]];
    }
    // Выводим количество итераций каждого потока.
    for(int i = 0; i < threadIterations.size(); ++i) {
        std::cout << "Thread " << i << " processed " <<
threadIterations[i] <<
                  " numbers" << std::endl;
    }

    // Считаем количество взаимнопростых чисел.
    int resultSize = 0;
    for(int i = 0; i < result.size(); ++i) {
        if(result[i]) {
            ++resultSize;
```

```
        }
    }
    // Выводим результат.
    std::cout << "There are " << resultSize <<
            " coprime numbers in the arrays" << std::endl;
    if(isIndexesOutputEnable) {
        for(int i = 0; i < result.size(); ++i) {
            if(result[i]) {
                std::cout << i << " ";
            }
        }
    }

    return 0;
}
```