МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
Факультет електроніки і комп’ютерних технологій

Звіт
Про виконання лабораторної роботи №2
“ Паралельні цикли в OpenMP програмах”

Виконав:
Ст. групи ФеС-32
Молібожко Олександр
Перевірив:
Кулик П.Р.

Львів 2024

Мета роботи: вивчити основні конструкції OpenMP програм

Хід роботи:

1. Ознайомитись з директивою #pragma omp for, та її опціцями.

2. Написати програму згідно індивідуального завдання. За допомогою функції omp_get_wtime() заміряйте час роботи програми за різної кількості потоків та розміру вхідних даних. Вказівки: Використати опцію reduction.

3. Використовуючи опцію schedule (з різними параметрами) модифікуйте прграму таким чином, щоб на екран виводилось повідоилення про те, який потік, яку ітерацію виконує. []: calculation of the iteration number .

4. Оформити звіт про виконання лабораторної роботи..

Виконання роботи:

1. Ознайомитись з директивою #pragma omp for, та її опціцями.
2. Код програми:

```c
#include <omp.h>

#define N1 250  // Розмір першої матриці
#define N2 400  // Розмір другої матриці

void calculate_sum(int matrix_size, int num_threads, int schedule_type) {
    int matrix[matrix_size][matrix_size];
    int i, j, sum;
    double start_time, end_time;

    // Заповнюємо матрицю випадковими значеннями
    srand(12345);
    for (i = 0; i < matrix_size; i++) {
        for (j = 0; j < matrix_size; j++) {
            matrix[i][j] = rand() % 6 + 1;
        }
    }

    // Встановлюємо кількість потоків
    omp_set_num_threads(num_threads);

    // Знаходимо суму максимальних елементів рядків матриці
    sum = 0;
    start_time = omp_get_wtime();

    // Виконуємо обчислення з різними способами розподілу
    #pragma omp parallel for private(j) reduction(+:sum) schedule(dynamic, 100)
    for (i = 0; i < matrix_size; i++) {
        int max = matrix[i][0];
        int thread_num = omp_get_thread_num();
        if (schedule_type == 2) {
            printf("[%d]: розрахунок числа ітерацій %d.\n", thread_num, i);
        }
        for (j = 1; j < matrix_size; j++) {
            if (matrix[i][j] > max) {
                max = matrix[i][j];
            }
        }
        sum += max;
    }

    end_time = omp_get_wtime();

    // Виводимо результат та час виконання програми
    printf("Розмір матриці: %d x %d\n", matrix_size, matrix_size);
    printf("Сума максимальних елементів рядків матриці: %d\n", sum);
    printf("Час виконання програми з %d потоками: %.4f секунд\n", omp_get_max_threads(), end_time - start_time);
    printf("-------------------------------------------------------\n");
}

int main() {
    printf("Виконання з першою матрицею (розмір %d x %d) та стандартним розподілом:\n", N1, N1);
    calculate_sum(N1, 4, 1); // Використання 4 потоків для першого варіанту

    printf("Виконання з другою матрицею (розмір %d x %d) та динамічним розподілом:\n", N2, N2);
    calculate_sum(N2, 2, 2); // Використання 2 потоків для другого варіанту з виводом інформації про потоки

    return 0;
}
```

Виконання програми:

```
Виконання з першою матрицею (розмір 250 x 250) та стандартним розподілом:
Розмір матриці: 250 x 250
Сума максимальних елементів рядків матриці: 1500
Час виконання програми з 4 потоками: 0.0002 секунд
--------------------------------------------------------
Виконання з другою матрицею (розмір 400 x 400) та динамічним розподілом:
[0]: calculation of the iteration number 0.
[0]: calculation of the iteration number 1.
[0]: calculation of the iteration number 2.
[0]: calculation of the iteration number 3.
[0]: calculation of the iteration number 4.
[0]: calculation of the iteration number 5.
[0]: calculation of the iteration number 6.
[1]: calculation of the iteration number 100.
[1]: calculation of the iteration number 101.
[1]: calculation of the iteration number 102.
[1]: calculation of the iteration number 103.
```

```
[1]: calculation of the iteration number 194.
[1]: calculation of the iteration number 195.
[1]: calculation of the iteration number 196.
[1]: calculation of the iteration number 197.
[1]: calculation of the iteration number 198.
[1]: calculation of the iteration number 199.
Розмір матриці: 400 x 400
Сума максимальних елементів рядків матриці: 2400
Час виконання програми з 2 потоками: 0.0028 секунд
--------------------------------------------------------
```

Висновок: в ході виконання лабараторної роботи було реалізовано LU-розклад у трьох режимах: послідовному, паралельному з `#pragma omp for` та `#pragma omp task`. Паралельні методи показали кращу швидкодію на великих матрицях, але для малих розмірів паралелізація має додаткові витрати часу.