НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
з лабораторної роботи №4
з дисципліни Розподілені і хмарні обчислення на тему:
«OpenMP. Розв’язування СЛАР»

Виконав:
студент групи КМ-81
Піткевич І.В.

Керівник:
Ст.викладач
Ліскін В. О.

Київ — 2021

ЗМІСТ

ВСТУП

OpenMP (Open Multi-Processing) — це набір директив компілятора, бібліотечних процедур та змінних середовища, які призначені для програмування багатопоточних застосунків на багатопроцесорних системах із спільною пам'яттю на мовах C, C++ та Fortran.

Переваги OpenMP:

• За рахунок ідеї «інкрементального розпаралелювання» OpenMP ідеально підходить для розробників, що прагнуть швидко розпаралелювати свої обчислювальні програми з великими паралельними циклами. Розробник не створює нову паралельну програму, а просто послідовно додає в текст програми OpenMP-директиви.

• При цьому, OpenMP — досить гнучкий механізм, що надає розробникові великі можливості контролю над поведінкою паралельної програми.

• Передбачається, що OpenMP-програма на однопроцесорній платформі може бути використана як послідовна програма, тобто немає необхідності підтримувати послідовну та паралельну версії. Директиви OpenMP просто ігноруються послідовним компілятором, а для виклику процедур OpenMP можуть бути підставлені заглушки (stubs), текст яких приведений в специфікаціях.

• Однією з переваг OpenMP розробники вважають підтримку так званих «orphan» (відірваних) директив, тобто директиви синхронізації і розподілу роботи можуть не входити безпосередньо в лексичний контекст паралельної області.

І ПОСТАНОВКА ЗАДАЧІ

У даній лабораторній роботі потрібно ознайомитися з основами OpenMP та навчитися розв'язувати СЛАР паралельними алгоритмами.

У рамках виконання лабораторної роботи потрібно:

1. Розпаралелити метод Гаусса використовуючи omp_set_num_threads() та parallel.
2. Порівняти швидкодію програми для систем різної розмірності та при різній кількості робочих процесів.

II РЕЗУЛЬТАТ РОБОТИ ПРОГРАМИ

```
wiperse@wiperse-PC:~/Education/Networks/fourth_lab$ ./main

Threads: n=4
Dimensions: n=2
Printing the generated 2-dimensional matrix.
72.0x1  +       72.0x2  =24.0
49.0x1  +       44.0x2  =31.0
Algorithm wihtout omp: 1.3883e-05 sec.
Algorithm with omp 4 threads: 0.000309144 sec.
x_1: 3.26667
x_2: -2.93333

wiperse@wiperse-PC:~/Education/Networks/fourth_lab$ ./main

Threads: n=20
Dimensions: n=2000
Algorithm wihtout omp: 11.1139 sec.
Algorithm with omp 20 threads: 2.9055 sec.

wiperse@wiperse-PC:~/Education/Networks/fourth_lab$ ./main

Threads: n=2
Dimensions: n=2000
Algorithm wihtout omp: 11.0847 sec.
Algorithm with omp 2 threads: 5.77386 sec.

wiperse@wiperse-PC:~/Education/Networks/fourth_lab$ ./main

Threads: n=5
Dimensions: n=5
Printing the generated 5-dimensional matrix.
49.0x1  +       58.0x2  +       9.0x3   +       91.0x4  +       8.0x5   =38.0
75.0x1  +       16.0x2  +       90.0x3  +       55.0x4  +       20.0x5  =28.0
77.0x1  +       37.0x2  +       81.0x3  +       23.0x4  +       31.0x5  =43.0
48.0x1  +       53.0x2  +       51.0x3  +       96.0x4  +       13.0x5  =40.0
5.0x1   +       38.0x2  +       66.0x3  +       37.0x4  +       35.0x5  =62.0
Algorithm wihtout omp: 3.3962e-05 sec.
Algorithm with omp 5 threads: 0.000378976 sec.
x_1: -0.174699
x_2: 0.41735
x_3: -0.170274
x_4: 0.245648
x_5: 1.70064
```

ВИСНОВОК

Подивившись на результати, можна побачити, що швидкість виконання багатопоточного алгоритму в рази швидше для довгих задач, ніж виконання однопоточного.