Faculty of Computers and Artificial Intelligence
Cairo University
Spring-2025

HPC- 2025

Assignment 3 – OPENMP

Deadline & Submission:

1. **Teams:** Max three students in the team.
2. Upload it on Classroom with file named
   A3_student1ID_student2ID_GroupName.zip
   eg. A3_20130002_20130001_S1_S2.zip
3. Attach a screen shot from the console output for each problem.
4. **Cheating could lead to serious consequences.**
5. **The team members must be different for each assignment.**

Problem1 Statement:

Write a C program using OpenMP to perform matrix-vector multiplication. Given a matrix A of size n x n and a vector v of size n, compute the resulting vector r = A * v. Parallelize the multiplication operation using OpenMP."

Requirements:

1. **Input**:
   - A square matrix A of size n x n.
   - A vector v of size n.
2. **Output**: The resulting vector r of size n.
3. **Parallelization**: Use OpenMP to parallelize the matrix-vector multiplication.

Example Input:

```
Matrix A:
1 2 3
4 5 6
7 8 9

Vector v:
1 1 1

Resulting vector r:
6 15 24
```

Problem2:

Write a C program using OpenMP to compute the standard deviation of an array of n integers. The program should first compute the mean of the array, then compute the variance, and finally calculate the standard deviation. Parallelize the calculations using OpenMP."

Requirements:

1. **Input**: An array of n integers.
2. **Output**: The standard deviation of the array.
3. **Parallelization**: Use OpenMP to parallelize the calculations of the sum of squares and variance.

Example Input:

```
Array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Example Output:

```
Standard Deviation: 2.872281
```

Formulae:

- Mean:

$$\text{mean} = \frac{1}{n}\sum_{i=0}^{n-1} x_i$$

- Variance:

$$\text{variance} = \frac{1}{n}\sum_{i=0}^{n-1} (x_i - \text{mean})^2$$

- Standard Deviation:

$$\text{standard deviation} = \sqrt{\text{variance}}$$

Problem3:

Write a C program that uses both MPI and OpenMP to compute the sum of all elements in a large array.
Each MPI process should handle a chunk of the array, and within each process, multiple OpenMP threads should compute the local sum in parallel. Finally, the global sum should be computed across all MPI processes."

Breakdown of What the Program Should Do:

1. **Initialize MPI** and get process rank and size.
2. **Divide the array** equally among MPI processes.
3. Each process:
    - Uses **OpenMP** to compute the **local sum** of its chunk using threads.
4. **MPI_Reduce** to combine local sums into a global sum.
5. The **root process prints** the final result.