Parallel Bar Chart Generation

Project Description:

This project involves designing a program that allows users to input an array of values (for simplicity, let's assume that the array values range from 1 up to a maximum number). The program will then generate and display a bar chart representing these values, with each bar in the chart drawn using asterisks ('*') as shown in Figure 1. The project then explores the use of parallel programming techniques, specifically OpenMP and MPI, to enhance the efficiency of the bar chart generation process.

```
Enter the dataset size: 7

Enter the dataset values (separated by spaces): 4 1 2 2 4 4 4

--- Bar chart ---

Data Point 1: *

Data Point 2: **

Data Point 3:

Data Point 4: ****
```

Figure 1 sample run

Project Tasks:

1. **Sequential Bar Chart Generation:** Begin by developing a sequential version of the program. First, allow the users to enter the length and values of an array (short array). Then, test the code on a large array with autogenerated values. This initial sequential version serves as the starting point for performance comparison.

2. **Parallel Bar Chart Generation with OpenMP:** Proceed to parallelize the bar chart generation process using OpenMP. Allow users to specify the number of threads to be used for parallelization. Also, just like the sequential code, test your program on both short and long array size. The goal here is to leverage OpenMP parallelism to enhance performance.

3. **Parallel Bar Chart Generation with MPI:** Following the OpenMP version, develop another parallel implementation, this time utilizing MPI for parallelization. Users can specify the number of processes to employ. Like the pervious programs, you must test your program on both short and long array size. This step explores parallelism using MPI.

4. **Performance Comparison:** Conduct a thorough analysis and performance comparison. Focus on understanding how the choice of parallelization technique affects the speed and accuracy of bar chart generation. Key considerations include evaluating task distribution between threads/processes, code complexity, scalability (performance with large data) and communication overhead between processes or threading overhead.

By completing these project tasks, you will create a program that allows users to visualize data through bar charts while exploring the benefits and trade-offs of parallel programming techniques for efficient data visualization.

Deliverables:

Sequential, OpenMP and MPI: DUE DATE: 5 May 2025

a. Source code for the sequential program, OpenMP program, and MPI program.
b. A report document for the sequential, OpenMP, and MPI codes, written according to the project report document *specifications*. (important)
c. Task distribution between team members.

Assessment:

You will be graded mainly on your code and report document, so please make sure that your report document is your own work, complete, and clear. Nonetheless, the source code will be checked to make sure that the results reported in your document are true and based on a valid code. In addition, it is your responsibility to make sure that you have a programming role in the project. You will not get a full mark if your role is minor, such as formatting the document. Table 1 shows a suggested distribution of tasks between team members; your task distribution should be similar, but not necessarily the same, to earn teamwork points. The grading rubric is provided in Table 2.

Table 1 Tasks Distribution

Task	Student
Introduction and problem definition	Student 1
Sequential part	Student 2
OpenMP part	Students 1
MPI	Students 3 and 2
Result and conclusion	All
Review	All

Table 2 Grading Rubric

Criteria	Points
Teamwork	2
Time Management	2
Format, quality, and organization	2
Problem definition and introduction	1
Sequential part	2.5
OpenMP part	4
MPI part	4
Results and conclusion	1.5
Appendices	1
Total	20