NATIONAL UNIVERSITY
of Computer & Emerging Sciences

Parallel and Distributed Computing

Project Proposal

Warehouse Management System

1. Project Title

- **Project Title**: Warehouse Management System
- **Group Number**: 60
- **Group Members**:

Asghar Ali	22k-4415
Muhammad Bilal	22k-4242
Hafiz Abdullah	22k-4489

2. Code Repository and Selection

- **Repository URL**: Not chosen from provided repositories.

- **Code Description**: The Warehouse Management System (WMS) manages items in stock, tracks inventory levels, and processes bulk updates efficiently. The system includes core operations such as item addition, deletion, and retrieval for large datasets.
- **Complexity Analysis**:

  Single-thread execution for large datasets (~20 minutes) due to operations like sorting, searching, and inventory updates.

  Initial code can have more or less 50 #pragmas, so additional complexity will be added by parallelizing compute-intensive tasks such as bulk item processing and inventory analysis.

3. Parallelization Strategy

- **OpenMP and MPI Implementation**:
  - OpenMP: Introduce thread-level parallelism in compute-heavy tasks like sorting and inventory updates.
- **Section for Parallelism:**
  - OpenMP: Sorting inventory, calculating stock levels, updating multiple items concurrently.
- **Challenges and Limitations:**
  - Ensuring thread safety in shared data structures during parallel updates.
  - Overhead of communication in MPI for smaller datasets.
  - Debugging parallel race conditions and synchronization issues.

4. Execution Plan

- **Hardware Specifications**:
    - Processor: Intel Core i7, 8 cores, 16 threads.
    - Memory: 16GB DDR4.

- **Parallel Execution**:
    - **OpenMP Testing:** Vary thread counts (2, 4, 8, etc.) for tasks like sorting and updates.

5. Data and Performance Metrics

- **Data Size**:
    - Test data sets: 5 million, 10 million, and 20 million items.

- **Performance Metrics**:
    - **Execution Time:** Measure total runtime for both serial and parallel implementations.
    - **Speedup:** Calculate speedup
    - **Scalability:** Evaluate performance gains with increasing threads or processes.
    - **Efficiency:** Determine the efficiency of parallelism as E , where N is the number of threads/processes..

6. Numerical Results and Visualization

- **Expected Results**: Significant speedup with OpenMP for multi-threaded operations.
- **Graphical Representation**:
    - Execution time vs. number of threads/processes.
    - Speedup and efficiency graphs.
    - Tools: Python's Matplotlib and Excel.

7. Testing and Validation

- **Testing Methodology**:
    - Validate correctness by comparing outputs of serial and parallel implementations.
    - Stress tests: Large datasets and edge cases (e.g., zero inventory, large incoming stock).

- **Validation of Results**:
    - Cross-check outputs of parallel implementations with serial execution for accuracy.

        - Measure performance gains and ensure they align with theoretical expectations.

8. Plagiarism and Originality Declaration

- **Originality Assurance**:
    - No plagiarized content; code and analysis are original.
    - Enhancements include OpenMP parallelism for increased complexity.
- **Use of AI Tools**: ChatGPT and similar tools were used for debugging and learning new parallelization techniques without compromising originality.