SimpleMultithreader Library Documentation of Group-44(Sec-A)

Overview

The `SimpleMultithreader` library is designed to facilitate parallel execution using Pthreads in C++ programs. It offers functionalities for creating and managing threads, distributing work among threads, executing lambda expressions in parallel, measuring execution time, and ensuring synchronization where necessary

File Structure

- **`vector.cpp`**: Sample program demonstrating parallel addition of vectors using the `SimpleMultithreader` library.
- **`matrix.cpp`**: Sample program demonstrating parallel matrix multiplication using the `SimpleMultithreader` library.
- **`Simple-Multithreader.h`**: Header-only implementation of the `SimpleMultithreader` library with functions for parallel execution.

Functions

`parallel_for(int low, int high, std::function<void(int)> &&lambda, int numThreads)`

- **Description**: Executes a for loop in parallel, distributing work among threads based on the provided lambda expression.
- **Parameters**:
  - `low`: Lower bound of the loop.
  - `high`: Upper bound of the loop.
  - `lambda`: Lambda function representing the loop body to be executed in parallel.
  - `numThreads`: Number of threads to be created for parallel execution.
- **Execution**:
  - Divides the loop range into sections for each thread.
  - Creates threads and assigns work sections to each thread.
  - Executes the provided lambda expression in parallel.
  - Handles synchronization and measures execution time.

`parallel_for(int low1, int high1, int low2, int high2, std::function<void(int, int)> &&lambda, int numThreads)`

- **Description**: Executes a 2D for loop in parallel.

- **Parameters**:
  - `low1`, `high1`: Bounds for the outer loop.
  - `low2`, `high2`: Bounds for the inner loop.
  - `lambda`: Lambda function representing the 2D loop body to be executed in parallel.
  - `numThreads`: Number of threads for parallel execution.
- **Execution**:
  - Divides the 2D loop range into sections for each thread.
  - Distributes work among threads for parallel execution.
  - Executes the provided lambda expression in parallel.
  - Manages synchronization and measures execution time.

Sample Program Output

- **`vector.cpp`**:
  - Displays a welcome message.
  - Performs parallel addition of vectors, verifies the result, and prints execution time.
  - Displays a closing message.
- **`matrix.cpp`**:
  - Displays a welcome message.
  - Initializes matrices, performs parallel matrix multiplication, verifies the result, and prints execution times for both loops.
  - Displays a closing message.

Error Handling

- Implemented robust error handling mechanisms to ensure smooth execution in scenarios like thread creation and joining failures

Additional Notes

- Execution times may vary based on the system configuration and workload.

Contributors

- **Group 44 Section-A**
  - **Athiyo Chakma**:
    - Implemented the `parallel_for` function for 1D loops:
      - Managed the creation of threads for parallel execution.
      - Designed the logic for dividing work among threads for 1D loops.
      - Developed the mechanism for executing the provided lambda expression in parallel.
    - Contributed to error handling mechanisms:

        - Ensured robustness by implementing error checks during thread creation and joining.
        - Assisted in thorough testing to validate the library's functionality.
- **Yash Goyal**:
    - Implemented the `parallel_for` function for 2D loops:
        - Engineered the logic for distributing work among threads in 2D loops.
        - Orchestrated the synchronization mechanisms for parallel execution of 2D loops.
        - Optimized the library's efficiency for handling 2D loop computations.
    - Contributed to execution time measurement:
        - Integrated mechanisms to accurately measure the execution time of parallel sections.
        - Assisted in benchmarking and profiling to evaluate the library's performance.

GitHub Link:-

https://github.com/Alpha-rgb-cell/OS_Assignments.git