SimpleMultithreader Assignment

Group Members:
Asa(2022113),
Sargun(2022450)

GitHub Repository: https://github.com/sargun4/OS

Project Overview:

Divides the size into chunks and assigns each chunk to a separate thread.
Each thread iterates over its assigned first dimension chunk and the entire second dimension concurrently.
Handles any remainder elements not evenly distributed among threads.
Measures and prints the execution time.

1.Data Structures:

```
struct ThreadData
{
  int id;
  int low;
  int high;
  function<void(int)> lambda;
};
```

id: Thread identifier.
low: Lower bound of the range assigned to the thread.
high: Upper bound of the range assigned to the thread.
lambda: Lambda function representing the operation to be parallelized.

```
struct ThreadData2D
{
 int id;
 int low;
 int high;
 int size;
 function<void(int, int)> lambda;
 pthread_mutex_t *mutex;
};
```

2.Variables

- pthread_t threads_arr[] : Array of pthread identifiers for created threads.
- ThreadData thread_data_arr[]: Array of ThreadData structures, storing thread-specific data.

Function Details

1.void parallel_for(int low, int high, function<void(int)> &&lambda, int numThreads);

Parallelizes a one-dimensional operation on the range [low, high).
low: Lower bound of the range.
high: Upper bound of the range.
lambda: Lambda function representing the operation.
numThreads: Number of threads for parallel execution.

2.void parallel_for(int low1, int high1, int low2, int high2, function<void(int, int)> &&lambda, int numThreads);

Parallelizes a two-dimensional operation on the ranges [low1, high1) and [low2, high2).
low1: Lower bound of the first dimension range.
high1: Upper bound of the first dimension range.
low2: Lower bound of the second dimension range.
high2: Upper bound of the second dimension range.
lambda: Lambda function representing the operation.
numThreads: Number of threads for parallel execution.

1. parallel_for
    - The function creates an array of pthread identifiers (pthread_t threads_arr[]) and an array of ThreadData structures (ThreadData thread_data_arr[]).
    - It uses a helper lambda function (helper) to define the behavior of each thread.
    - The total range is divided among threads, and each thread is assigned a chunk of the range to process.
    - Threads are created using pthread_create, and each thread executes the specified lambda function on its assigned range.
    - After the threads complete their tasks, the main thread waits for their completion using pthread_join.
    - The execution time of the parallelized operation is measured and printed.
2. parallel_for_2d
    - Similar to parallel_for, this function creates an array of pthread identifiers and an array of ThreadData2D structures.
    - It uses a helper lambda function (helper2d) to define the behavior of each thread in two dimensions.
    - The total range in the first dimension is divided among threads, and each thread is assigned a chunk of this range.
    - Each thread iterates over the assigned range in the first dimension and the entire range in the second dimension.
    - Threads are created using pthread_create, and each thread executes the specified lambda function on its assigned ranges.
    - After the threads complete their tasks, the main thread waits for their completion using pthread_join.
    - The execution time of the parallelized operation is measured and printed.

Contributions:
Asa:

Sargun:

How to Run:

1. Make
2. Then : ./vector numthreads size