1 Matrix multiplication

Implement a parallel version of a matrix multiplication function (assume matrices with fixed sizes) with parallel tasks. Experiment with placing tasks at the cell and line level.

Can you conclude on the most efficient (and scalable) approach? And comparing with the parallel loop implementation of the previous laboratory?

2 Quick Sort

Write a OpenMP program that sorts an array using quick sort with tasks. The size of the array can be fixed, and the array global. The decision on the pivot is irrelevant (e.g. take the first).

Determine the time it takes to perform the sorting, and compare it to the implementation of the previous laboratory.

3 Summing all values of an array

Implement a program that calculates the sum of all values of an array using:

a. Explicit handling of the array splitting and shared sum variable
b. Parallel for loop with reduction
c. Parallel taskgroup with reduction
d. Parallel taskloop with reduction

Compare the time to perform the calculation. Can you conclude on the most efficient (and scalable) approach?

4 Area of Mandelbrot Set

Parallelize the calculation of the estimate of the area of the Mandelbrot Set with worksharing constructs and with tasks. Compare the time to perform the calculation. Can you conclude on the most efficient (and scalable) approach?

5 Blocked matrix processing

a. Parallelize the following code, with tasks and dependencies:

```
for (i = 1; i < N - 1; i++)
        for (j = 1; j < N - 1; j++)
                M[i][j] = (M[i][j-1] + M[i-1][j] + M[i][j+1] + M[i+1][j])/4.0;
```

b. Change to divide the matrix in blocks of BS size. Inside each block the execution is sequential (assume N is multiple of BS).

6 Analyzing and Evaluating (Group Evaluation #1)

Analyze by profiling the execution of exercise 5 with tasks (with and without dependencies), in both GCC and LLVM implementations of OpenMP:

a. Compare the execution time of the different approaches to parallelize program
b. Compare the strategy used by both implementation for the execution of tasks by the threads

Extra Lab (Students may advance further outside of lab)

The Cholesky decomposition is a decomposition of a matrix into the product of a lower triangular matrix and its transpose. It is widely used in numerical methods, such as Monte Carlo Simulations.

a. Taking as a basis the provided parallel code (with worksharing constructs) of the Cholesky decomposition, implement parallel versions (with blocks) which use tasks, with and without dependencies.
b. Derive the task dependency graph of the computation.
c. Compare the time to perform the calculation. Can you conclude on the most efficient (and scalable) approach?

Credits:

- Version 1.0, Luis Miguel Pinho, with inputs from:
  - Christian Terboven, Programming OpenMP, Task Dependencies, https://blog.rwth-aachen.de/itc-events/files/2021/02/11-openmp-CT-tasking_dependencies.pdf
  - Xavier Teruel, OpenMP Tasking, Part 2:Performant Tasking, https://www.openmp.org/wp-content/uploads/OpenMP-UMT-Tasking-2.pdf