Vincent Kipchoge

Parallel and Distributed Computing

OpenMP Matrix Multiplication Practice

Task 1: Write a parallel program using OpenMP based on this sequential solution of matrix multiplication.

A screenshot of a computer screen

Description automatically generated

*Figure 1*

***Code Logic***

***matrixInit():*** This code function initializes two matrices, firstMatrix and secondMatrix with random double-precision values mimicking a matrix of dimensions N x N.

***matrixMultiplication():***Matrix multiplication is performed within the *matrixMultiplication* function using triple nested loops. For each element in the resulting matrix matrixMultiResult, the function computes the products of corresponding elements from firstMatrix and secondMatrix.

**#pragma omp parallel:** The provided directive is included in the outermost loop which separates the calculations amongst multiple threads. The threads are thus responsible for individually calculating their portion of the result matrix. Note in addition there is an Inner loop parallelization using reduction.

***Analysis:***

In the section I parallelized the program with OpenMP based on the sequential solution and provided in Figure 1 are the average times of the implementation. As recognized in the figure, the implementation reduced the execution time needed compared to the original sequential solution for the matrices program. The reason being is that with parallelism we can manage large data values and process them amongst the processors thus speeding up the execution speed.

Task 2: Write a block-optimized matrix multiplication program and use OpenMP to parallel its

execution

A green and white grid with numbers

Description automatically generated

*Figure 2*

***Code Logic***

***matrixInit():*** This code function initializes two matrices, firstMatrix and secondMatrix with random double-precision values mimicking a matrix of dimensions N x N.

***#define BLOCKSIZE 64:*** Provided is a constant to define the size of the blocks for the block-wise multiplication. Adjusting the value based on hardware and performance requirements will change the overall performance of the program.

***#pragma omp parallel:*** Parallelized block-optimized matrix multiplication in the ***matrixMultiBlockOptimized()*** function in addition there is an Inner loop parallelization using reduction.

***Analysis:***

In the section, I implemented a block optimization for the matrix’s multiplication program using OpenMP. Thus, increasing the computation required in comparison to the parallel solution and sequential solution. For the section, the table provided in Figure 2 depicts the results of the different BLOCKSIZE which span from 32, 64, 128, and 256. Inside the nested loops, we compute block-wise multiplication, reducing cache misses and improving data locality. The division of the matrices into BLOCKSIZES and implementing the parallelization of them has increased the hit rate of the cache thus assisting in computation.

Main Function Code Logic:

The ***main()*** function contains a call to the ***matrixInit()*** which initializes the matrices following are the parts for each Task collecting start and end times before calculating the difference and then displaying it.