COL730 Assignment 2: LU Decomposition with OpenMP

Problem statement:

Read the publication referenced below and implement the OpenMP task-based block LU decomposition algorithm (Algorithm 4) described. OpenMP tasks must be used to implement the algorithm. The implementation is expected to scale with increasing number of CPUs (at least up to 8) for matrices of size at least 4096.

V. S. Rana, M. Lin and B. Chapman, "A Scalable Task Parallelism Approach for LU Decomposition with Multicore CPUs," 2016 Second International Workshop on Extreme Scale Programming Models and Middleware (ESPM2), 2016, pp. 17-23, doi: 10.1109/ESPM2.2016.008.

Submission Instruction:

Submit a single zip file named [Your Entry Number].zip on moodle with the following:

1. An outline of the implementation decisions made and why.
2. Graphs of number of CPUs vs. execution time for (at least up to) 8 CPUs and (at least up to) 4096 x 4096 matrix.
3. Sources implementing the function signatures provided in the header “plu.h”.
4. A makefile that builds a library named “plu” (libplu.a or libplu.so) with the implementation of the functions provided in “plu.h”.

Check course webpage for last date of submission.

Note:

All floating-point computations should be done using double precision floating-point arithmetic.

The function *init : () -> void* is called once before any calls to function lu is made and *close : () -> void* is called once after all calls to lu are complete. Any global OpenMP configuration should be performed by init and reverted by close (if any).

The function *lu : (const double **A, double **L, double **U, int n) -> void* should not mutate A and write the results into the matrices L and U.

You are expected to write your own driver program for testing and documenting the execution times of the lu function for different input sizes as necessary, but it may not be included in your submission.