A Scalable Task Parallelism Approach For LU Decomposition With Multicore CPUs

V. Rana[1,2] , M. Lin[2], B. Chapman[1,2]
[1] Stony Brook University [2] Brookhaven National Laboratory

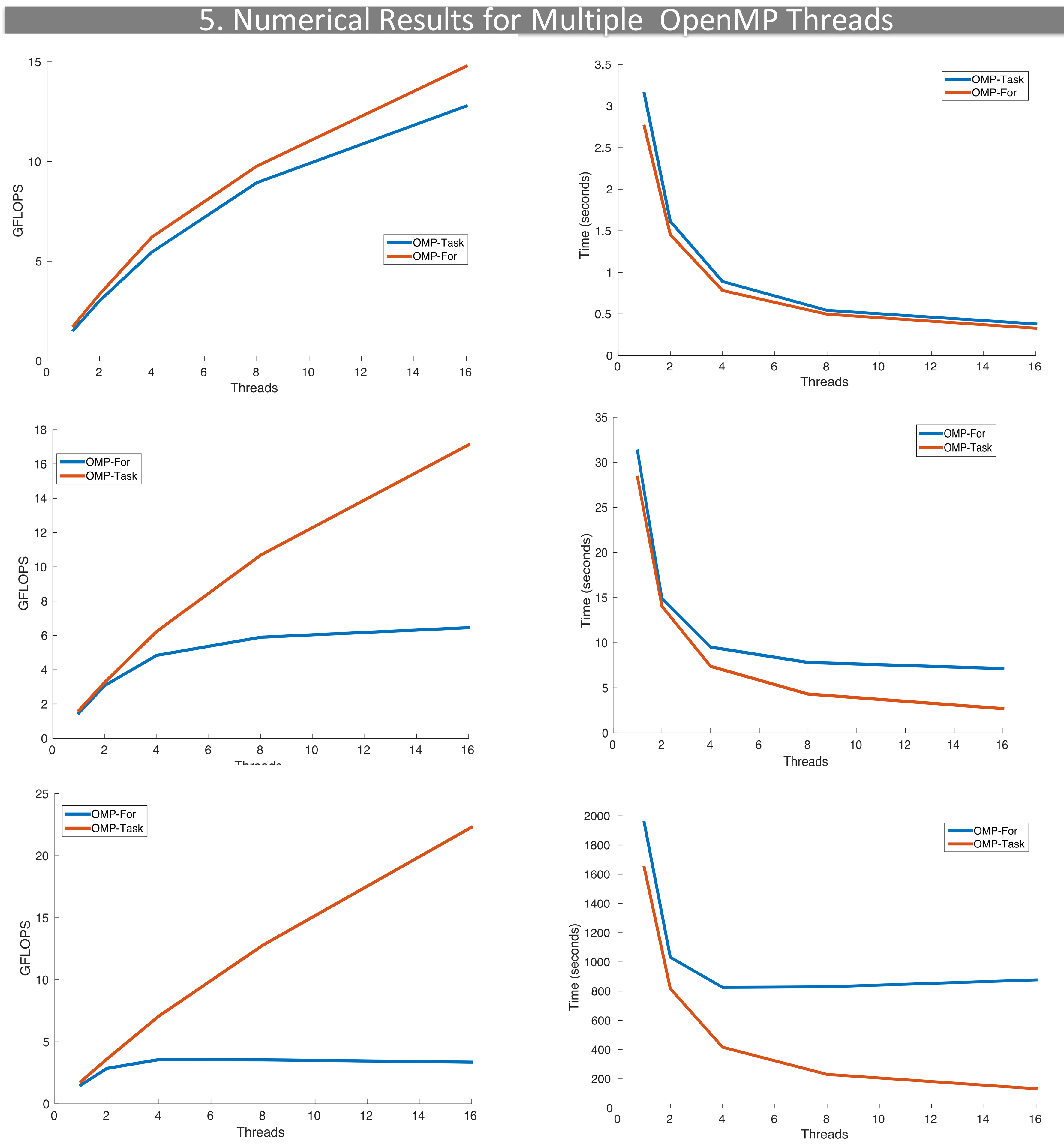
1. Introduction: Motivation and Goal

Task parallelism is a popular and relatively new approach to exploiting parallelism because it offers the potential for minimal synchronization. Tasks were introduced in OpenMP version 3.0 but there are not many instances of them being used in current programming applications. Many scientific applications have linear systems Ax = b which needs to be solved for different vectors b. Gaussian elimination, or otherwise known as LU decomposition, is an efficient technique to solve a linear system. The main idea of the LU decomposition is to factorize A into an upper (U) and a lower (L) matrix such that A = LU . This poster presents an OpenMP task parallel approach for the LU factorization. The tasking model is based on the individual computational tasks which occur during the block-wise LU factorization. The algorithm is especially suited for multi core processors and shows a much improved parallel scaling behavior compared to a naive parallel for based LU decomposition.

2. Problem Formulation

For our experimental evaluation we used an Intel Xeon CPU E5-2680 v2, a 2.80 GHz clock speed and 64 GB of memory. The system ran GNU/Linux, kernel version 2.6, for the x84 64 ISA. All programs were implemented in C++ using OpenMP and were compiled using GCC, version 6.1.0, with the −O3 optimization flag. A set of randomly generated input matrices with sizes ranging from 1024 x 1024 to 16384 x 16384 were used as the test matrices. To compare the parallel algorithms, the practical execution time was used as a measure. Execution measures the total time an algorithm completes the computation. Results were obtained using the cluster at Brookhaven National Laboratory.

5. Numerical Results for Multiple OpenMP Threads



7. Conclusions and Future Work

- For very large systems, using the block wise LU decomposition with OpenMP tasks on multiple threads is the optimal choice versus using a naive implementation of an OpenMP based parallel for loop.
- For a single thread, the block wise LU decomposition is much faster than a vectorized for loop (compiler optimized), for larger linear systems. The GNU scientific library is a benchmark and should not be used in any high performance application code.
- The results indicate that for smaller systems, the task based approach is not optimal. The reasons for that are embedded in task generation and the related task overhead costs. The future work is to strategize on how to eliminate such bottlenecks.

3. Block LU Decomposition

Fig 1: Example of a 3x3 block matrix.

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{bmatrix} \cdot \begin{bmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{bmatrix}$$

Fig 2: Equating the second column of L and U with that of A we obtain the following system of equations

$$A = L \cdot U.$$

$$A \cdot x = (L \cdot U) \cdot x = L \cdot (U \cdot x) = L \cdot y$$

4. Numerical Results for Single Thread

Table 1: Execution time comparison (seconds) of the three different LU algorithms

	OMP-Task	OMP-For	GSL
1024 x 1024	0.4268	0.4137	0.4831
1936 x 1936	3.15	2.759	3.585
4096 x 4096	28.32	31.24	36.25
16384 x 16384	164.69	195.44	287.8

Table 2: GFLOP comparison of the three different LU algorithms.

	OMP-Task	OMP-For	GSL
1024 x 1024	1.684	1.737	1.487
1936 x 1936	1.542	1.761	1.356
4096 x 4096	1.625	1.473	1.27
16384 x 16384	1.789	1.507	1.023

6. Algorithmic Description

Algorithm 1 Serial Crout LU

for $k \in$ n - 1 **do**
  for $k+1 \in$ n **do**
    $A_{i,k} = A_{i,k} \;/\; A_{k,k}$
  end for
  for $k+1 \in$ n **do**
    for $k+1 \in$ n **do**
      $A_{i,j} = A_{i,j} - A_{i,k} * A_{k,j}$
    end for
  end for
end for

Algorithm 2 Block LU

Factor $A_{11} = L11 \cdot U_{11}$
Solve $L_{11} \cdot U_{12} = A_{12}$ for U_{12}
Solve $L_{21} \cdot U_{11} = A_{21}$ for L_{21}
Form $S = A_{22} - L_{21} \cdot U_{12}$
Repeate 1-4 on S to obtain L_{22} and U_{22}

Algorithm 3 Task Based Block LU Decomposition

for step $\in$ numberOfBlocks **do**
  $LUPivot()$
  $LUPermutations()$
  #pragma omp taskgroup {
  #pragma omp task
  for j $\in$ steps **do**
    $LUSolve$
  end for
  for k $\in$ steps **do**
    for l $\in$ steps **do**
      #pragma omp task
      $LUmatrixMult$
    end for
  end for}
end for