An Empirical Investigation of OpenMP Based Implementation of Simplex Algorithm

Arkaprabha Banerjee, Pratvi Shah, Shivani Nandani, Shantanu Tyagi, Sidharth Kumar, and Bhaskar Chaudhury

Group in Computational Science and High Performance Computing, DA-IICT, Gandhinagar, India
Department of Computer Science,The University of Alabama at Birmingham, Birmingham, USA

Agenda

- Introduction
- Objectives
- Parallel Algorithm
- Experimental Results and Observations
- Conclusion

What is the Simplex Algorithm?

- The Simplex Algorithm is a widely used algorithm to solve optimization problems subject to a set of linear relationships.
- Linear problems usually involve scheduling, assignment, location, network flow, and associated domains which are often required in industries that deal with manufacturing, distribution, transportation and finance.
- The Standard Simplex Method refers to the original algorithm proposed by George Dantzig.

What is the Simplex Algorithm?

- The linear relations are represented in the form of a table and we iterate on the table finding entering and leaving variables with each iteration. It requires the previous **iteration** to be completed before the new solution can be computed, thus restricting the scope of parallelization.
- LP problems are usually modelled as :

$$\textit{maximize / minimize } \mathbf{Z} = \mathbf{CX}$$

$$\textit{subject to } \mathbf{AX} = \mathbf{B} \ (\mathbf{X >= 0})$$

where **A** : Constraint Matrix **B** : Constant Vector

C : Objective Function Coefficient Vector **X** : Solution Vector

What is the Simplex Algorithm?

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & \cdots & A_{1n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ A_{m1} & A_{m2} & \cdots & \cdots & A_{mn} \end{bmatrix}_{mxn} \quad \begin{bmatrix} B_1 \\ \cdots \\ \cdots \\ B_m \end{bmatrix}_{mx1} \quad \begin{bmatrix} C_1 & C_2 & \cdots & \cdots & C_n \end{bmatrix}_{1xn} \quad \begin{bmatrix} X_1 \\ \cdots \\ \cdots \\ X_n \end{bmatrix}_{nx1}$$

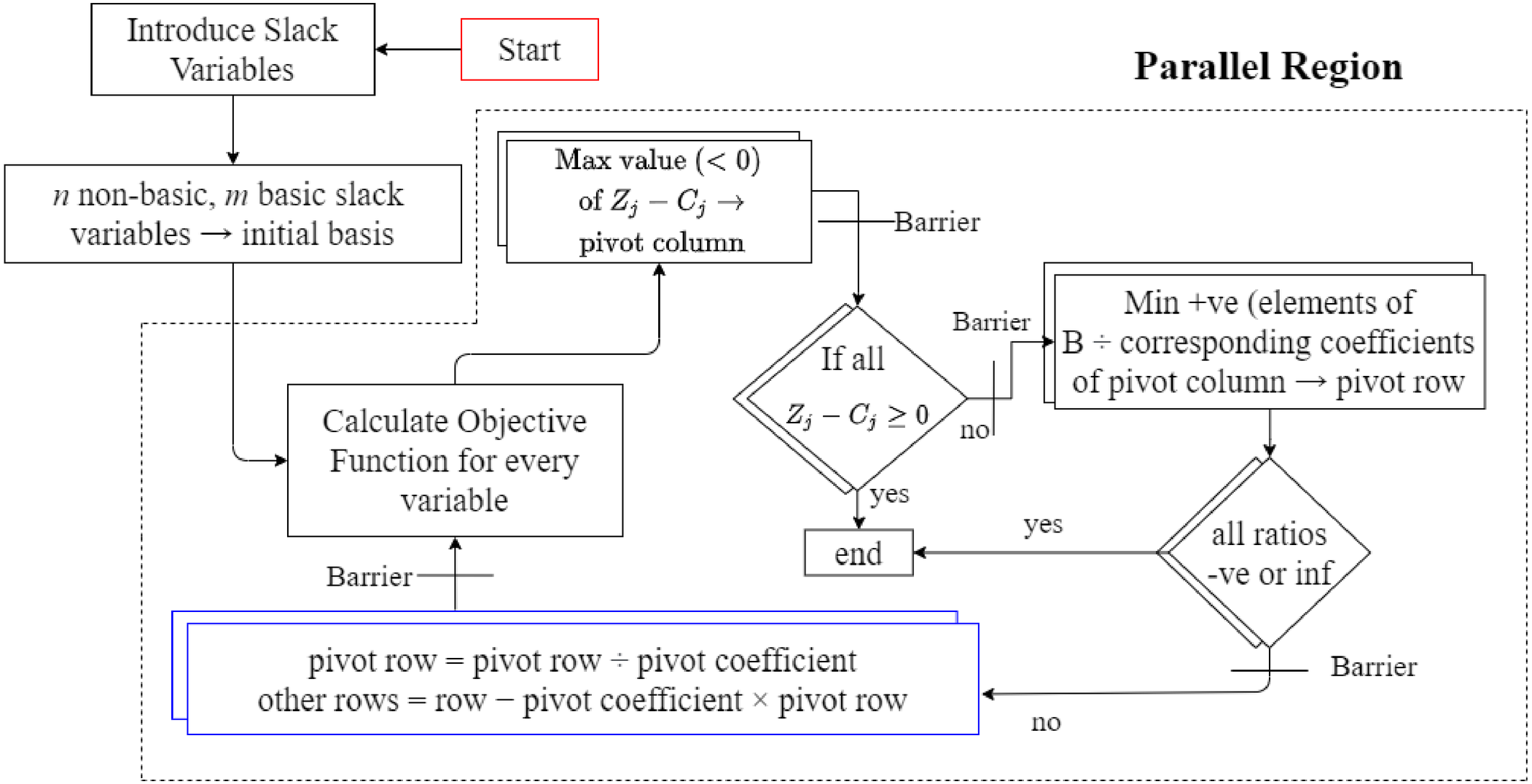
- Every row in the constraint matrix represent the coefficient of every variable for a given linear constraint.
- Every element in the constant vector corresponds to one linear constraint.
- Element in the Objective function coefficient vector corresponds to the coefficient of the variables in the objective function.
- Linear inequalities can be converted to equalities using slack and surplus variables.

Objectives

In this paper, we study:

- a shared-memory based parallel implementation of the standard simplex algorithm.
- the impact of the density of the underlying matrix on the overall performance.
- the **scalability** of the parallel algorithm over a range of matrix densities.
- the effect of increasing the number of constraints.
- the effect of varying the number of variables.

Simplex Algorithm



Parallel Algorithm

1. **Slack variables** are introduced to convert inequality constraint to equality constraint with **0 as their objective function coefficient**.
2. A tableau is made with **slack variables as the initial basis**. These 2 steps are done serially.
3. Threads are assigned to **individual columns** to calculate the value of the objective function and reduced costs**.**
4. **Maximum negative value** of reduced cost is found out using a custom class and reduction clause to get the **pivot column/ entering variable**. We terminate if not found.

Parallel Algorithm

5. Similarly, the **minimum row-wise ratio** is found out to get the pivot row/ leaving variable using reduction clause and custom-defined class. Unbounded and no solution condition is checked using no wait clause.
6. The **pivot coefficient** thus found is used to first update all the values in the pivot row. The remaining rows are updated using pragma for construct and within each row, columns use **SIMD vectorization.**
7. Local variables are updated by a single thread. **Repeat again from step 3** till the termination condition is satisfied.

Implementation Details

1. **Block Structured Notation**
   For efficient storage and modification fo the matrix elements we have made use of the block structure notation proposed by Dantzig:

$$\begin{bmatrix} A & B \\ -C & 0 \end{bmatrix}$$

2. **Class-based key-value pair** to store maximum/minimum value along with its index
3. **Optimal Scheduling clause and Load Balancing**
   Guided scheduling was used to tackle the load balancing problem
4. **Optimal SIMD Units**
   4 was found to be the optimal *simdlen()*

Pseudo Code


```
# pragma omp parallel <shared variables>
{
//Step 3 & 4
# pragma omp for <schedule> <reduction>
for(int j=0;j<C;j++){
   find max negative value (max_value)
   from reduced cost row
}

Corresponding index=max_index
& column=Pivot Col

#pragma omp single nowait
max_value=0
```

Parallel region starts

Scheduling and reduction clauses

```
do{
    // Step 5
    #pragma omp for <schedule><reduction>
    for(int j=0;j<R;j++) {
        find the min value
        (min_value) from Min Ratio column
        do count++ for negative values
    }

    #pragma omp single nowait
    {
        if count == R
            there is unbounded/no solution
            flag=False break
        else
           Row corresponding to
           min_value is the
           Pivot row with index= min_index
```

Single thread with nowait clause to check for terminating condition

Pseudo Code


```
// Step 6
pivot = table[min_index][max_index]
#pragma omp barrier

#pragma omp parallel for
for(int i=0;i<C;i++)
    update Pivot Row
#pragma omp parallel for
for(int i=0;i<R;i++)
    #pragma omp simd
    for(int i=0;i<C;i++)
         update all elements except
         that of Pivot row
}
```

barrier Construct

simd Construct

```
//Step 3 & 4 repeated
    #pragma omp for <schedule> <reduction>
        for(int i=0;i<C;i++)
          find the new reduced cost values
          for updated table
          countNegative++ for negative values

    #pragma omp single
         Update the initial conditions
    }while(countNegative and flag)
}
```

Single thread to update variables

Machine Architecture

Specifications	Bench 1	Bench 2
Model Name	Silver 4214R CPU @ 2.40GHz	CPU E5-2640 v3 @ 2.60GHz
Core(s) per socket	12	8
Socket(s)	2	2
L1d Cache	768KB	32 KB
L2 Cache	24 MB	0.265 MB
L3 Cache	33 MB	20.48 MB
GNU GCC Version	9.3.0	10.2.0

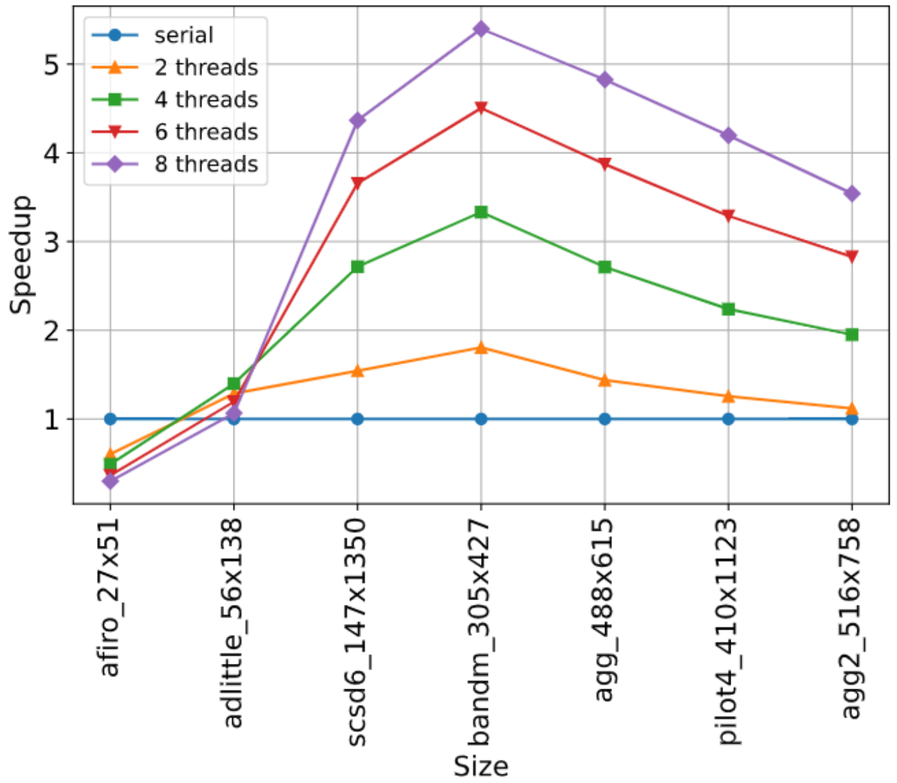
Datasets

- Netlib datasets [1]
- Computationally generated dataset with curated size and density
  - 256 constraints with varying number of variables with 0.5 density
  - 256 variables with varying number of constraints with 0.5 density
  - 512 variables and 512 constraints with varying density

1: http://www.netlib.org/

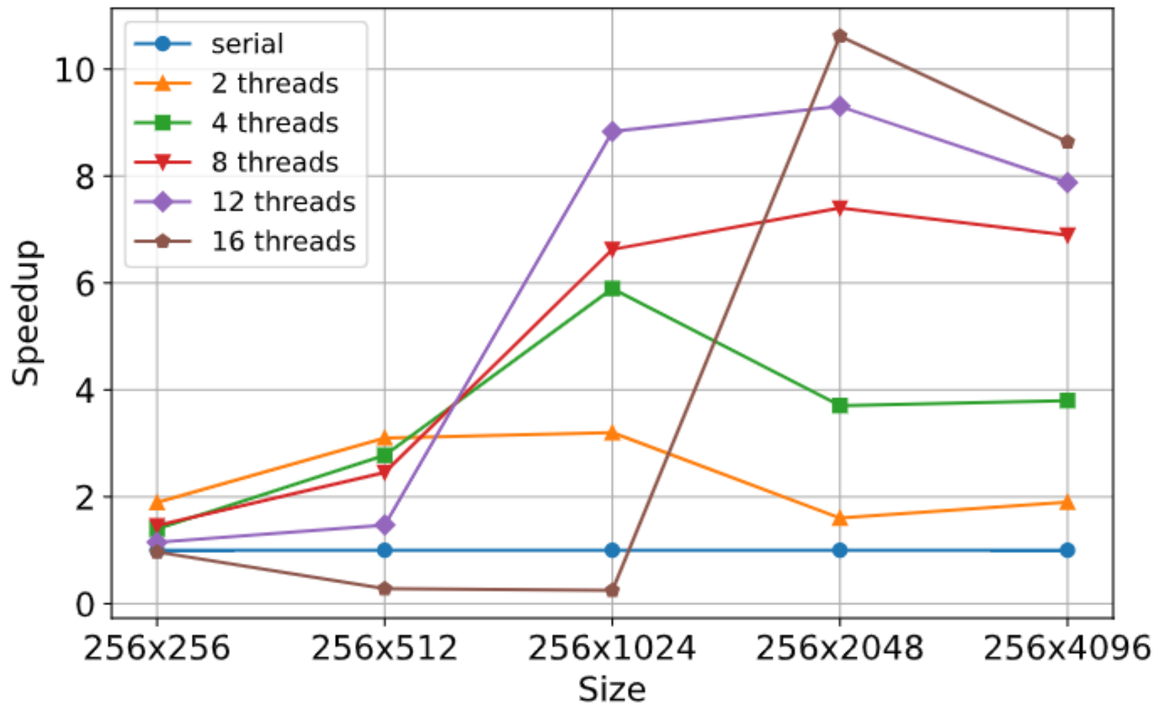
NETLIB Dataset Analysis

- Netlib dataset is the standard dataset for LP problems.
- For smaller problem sizes parallel algorithm does not lead to an improvement in performance due to thread overheads.
- The speedup for all the datasets remained within the linear upper bound.



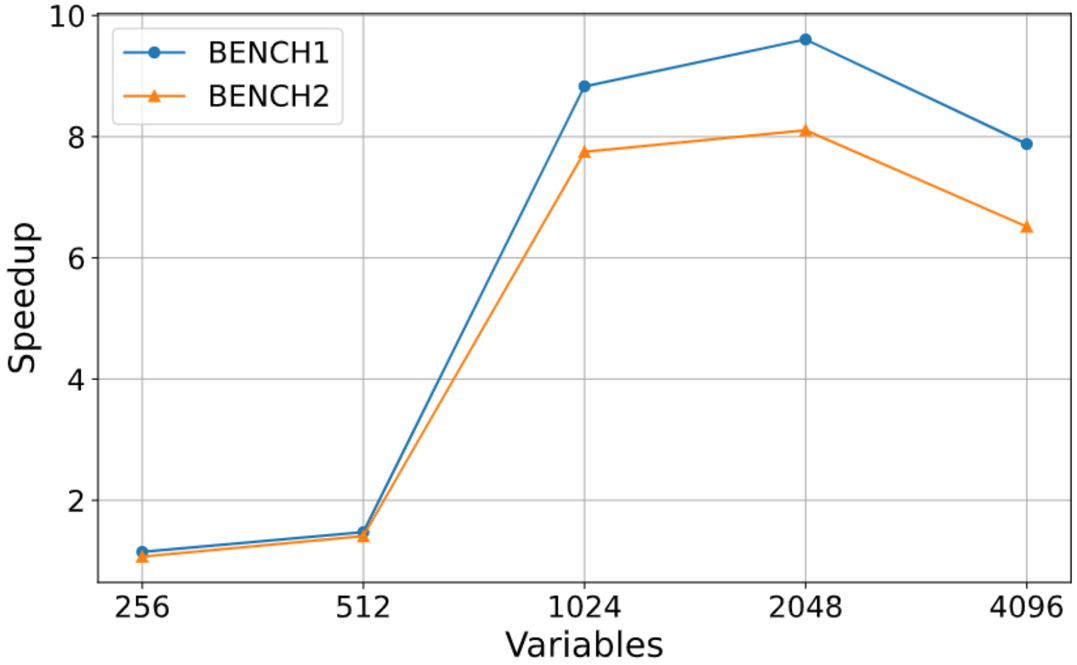
Variation of the Number of Variables

- it is observed that the speedup for each thread size increases (up to a point at mid-size) and then decreases. Peak values for large thread counts occur at larger problem sizes.
- At the largest problem size, the reduction in the peak performance is greatest for small thread counts (1, 2 and 4) and smallest for the larger thread counts (8,12 and 16)
- We could observe super-linear speedup in the case of 2 and 4 threads for certain problem sizes.



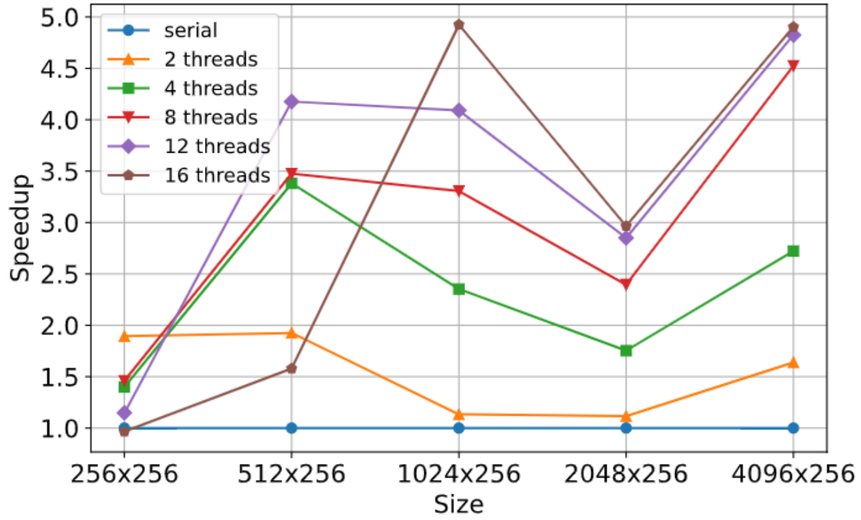
Variation of the Number of Variables

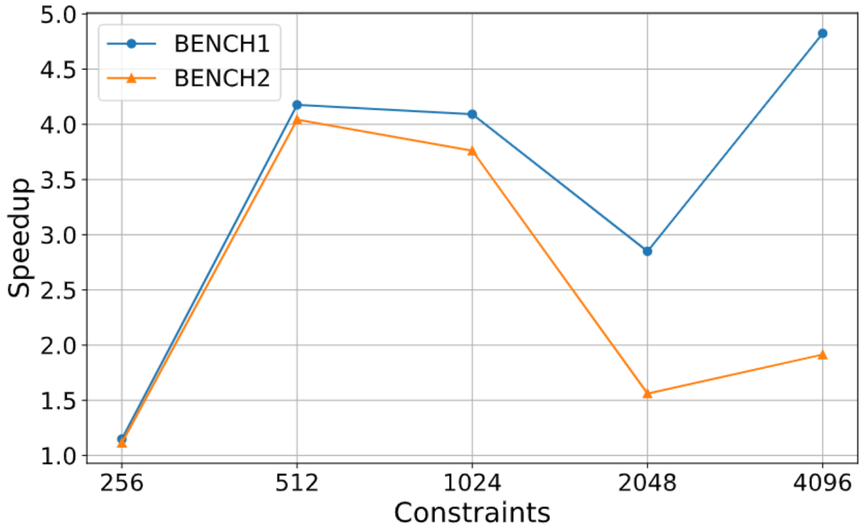
- The speedup achieved in the BENCH2 is similar to the one achieved in the BENCH1, till the problem size fits in the L1 and L2 cache of the respective systems.
- In the case of BENCH2, when the work allocated per thread exceeds the L2 cache size and results in data being continuously fetched from the L3 cache
- Our implementation observes a maximum speedup of 10.2 with 16 threads for 256 × 2048 which is comparable to the maximum speedup for a problem size of 256 constraints in the state-of-the-art implementation



Variation of the Number of Constraints

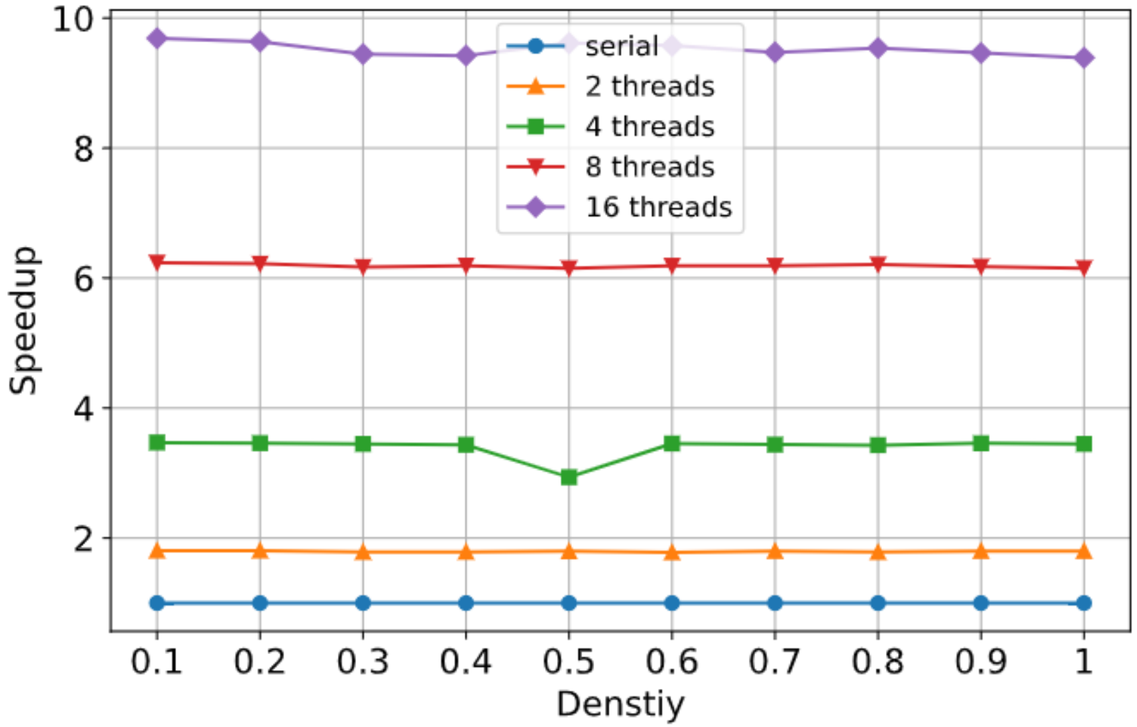
- On increasing the constraints for 256 variables, the speedup increases faster , as compared to when the number of variables increased.
- For 2048 constraints with 256 variables the problem size exceeds the L3 cache limit for the BENCH1. Thus we observe a drop.
- The speedup increases for larger problem size as threads increase to 16 and vice versa is seen for smaller problem sizes.





Variation in Matrix Density

- We have considered 512x512 matrices with densities varying from 0.1 to 1 in steps of 0.1.
- The speedup in all the cases remains almost constant or increases a little when the density of the matrices increases.
- Hence, the parallel algorithm is scalable in that nature.

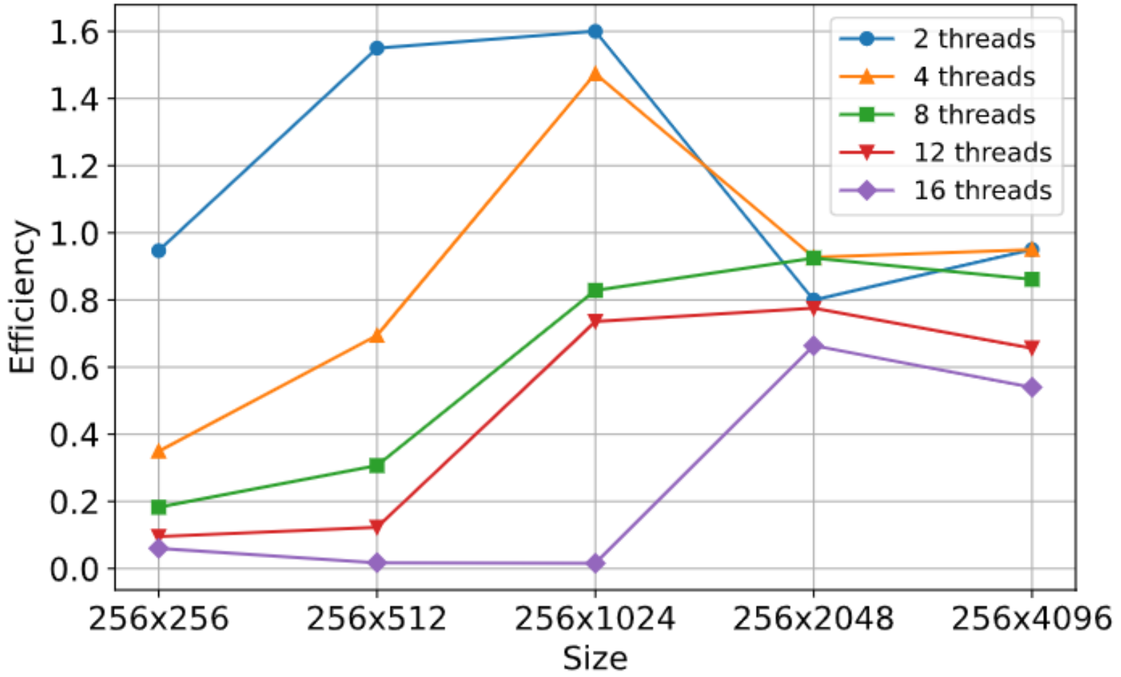


Key Observations

- Using more threads increases the synchronization overhead, while using a lower number of threads reduces the parallelization. The optimal limit on the number of threads lies in between the two extremes.
- There exists a critical problem size for each thread where the nature of the speedup changes from increasing to decreasing on either side of that critical number. This critical value is achieved at a larger problem size when using a larger thread count.
- Smaller problems performed better with a lower number of threads while larger problems perform better with a higher number of threads.

Key Observations

- Efficiency decreases with an increase in the number of threads mainly due to **synchronization overhead**.
- With the increase in problem size, **cache fulfillment** will also contribute significantly to the time needed to fetch the data leading to a drop in performance metrics



Conclusions

Vectorization	Number of constraints dominate more than number of variables	Scalable Algorithm
• Contributes significantly towards improving the performance • Constrained by the hardware properties of the system as well as the problem structure.	• The number of computations increases more with the increase in the number of constraints in comparison to the number of variables	• Our parallel algorithm proved to be fairly scalable, in terms of relative speedup, for the matrices of varying densities (in range of 0.1 to 1)

Questions ?


THE UNIVERSITY OF
ALABAMA AT BIRMINGHAM


Dhirubhai Ambani Institute of Information and Communication Technology
DA-IICT


IWOMP

Thank You


UAB
THE UNIVERSITY OF
ALABAMA AT BIRMINGHAM


Dhirubhai Ambani Institute of Information and Communication Technology
DA-IICT


IWOMP