# MST using KRUSKAL's Algorithm

Final presentation

SAI KIRAN MUNDRA

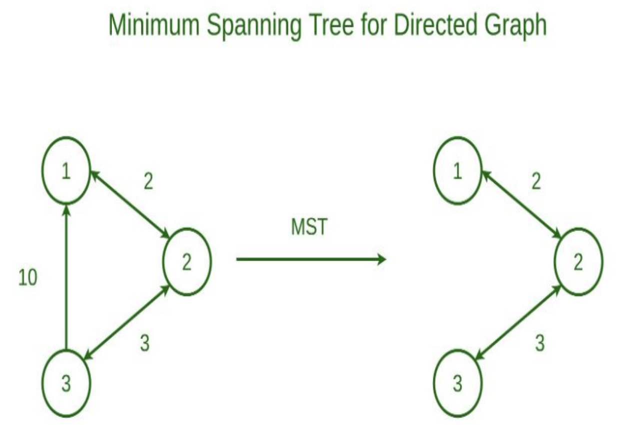
CSE 708 - Programming Massively Parallel Systems

Instructor - Dr. Russ Miller

University at Buffalo The State University of New York

# Minimum Spanning Tree

- A spanning tree is a subset of the edges of the graph that forms a acyclic tree  where every node of the graph is a part of the tree.
- MST is
    - a spanning tree
    - Total weight of edges is minimum



# Minimum Spanning Tree

- Number of vertices in graph and MST are same.
- Number of edges = V-1 where V is number of vertices
- Need not be unique, multiple MST are possible depending on input.
- Neither disconnected nor cyclic.

# Algorithms for MST

- Kruskal’s algorithm
- Prim’s algorithm
- Boruvka’s algorithm

# Kruskal’s algorithm

1. Sort all the edges in the non-decreasing order of their weights.
2. Select the smallest edge.
3. Check if the selected edge forms a cycle with the MST formed so far
4. Include the edge if no cycle is formed, else discard it.
5. Repeat steps from 2 to 5 till V-1 edges are included in the MST.

# Approach for Parallelization

- The data's spread across multiple processors.
- Every processor Pi sort the edges that are contained in it's partition Vi - parallely
- Every processor Pi finds the local MST using the edges in it's partition
  - Some edges are eliminated in this step across all processors

# Approach for Parallelization

- Processes merge their local MST's (or MSF's). Merging is performed in the following manner. Let a and b denote two processes which are to merge their local trees (or forests), and let Fa and Fb denote their respective set of local MST edges. Process a sends set Fa to b, which forms a new local MST (or MSF) from Fa U Fb.
- Merging continues until only one process remains. Its MST is the end result.

# Approach for Parallelization

- To create the new local MSF during merge step, we perform Kruskal's algorithm again on Fa U Fb.
- It can be shown that our approach is efficient for p = O(n/log n) number of processors.

# Communication b/w processors

- Communication between processors happen during the merging the local MSTs into new local MSTs.
- Processor A sends its local MST to Processor B and Processor B calculates the new local MST using A’s Local MST and B’s local MST.

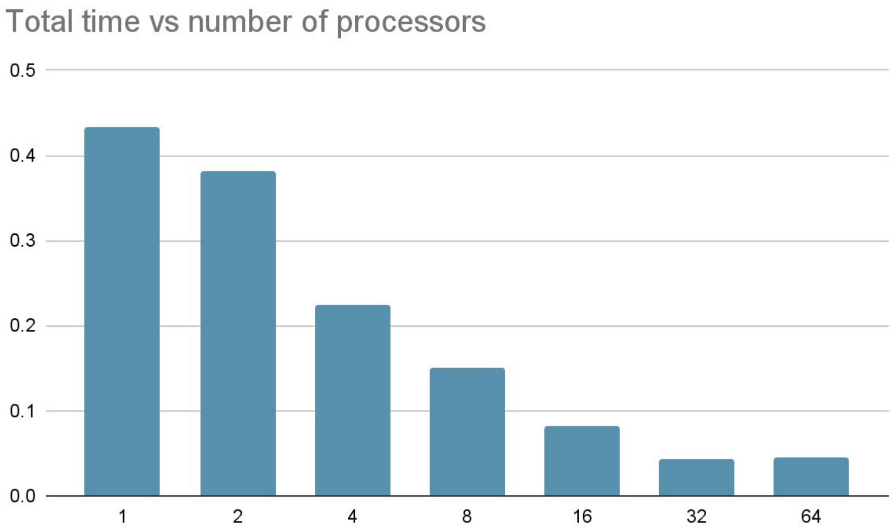
# Implementation in MPI

- I have used MPI to implement the parallel Kruskal Algorithm

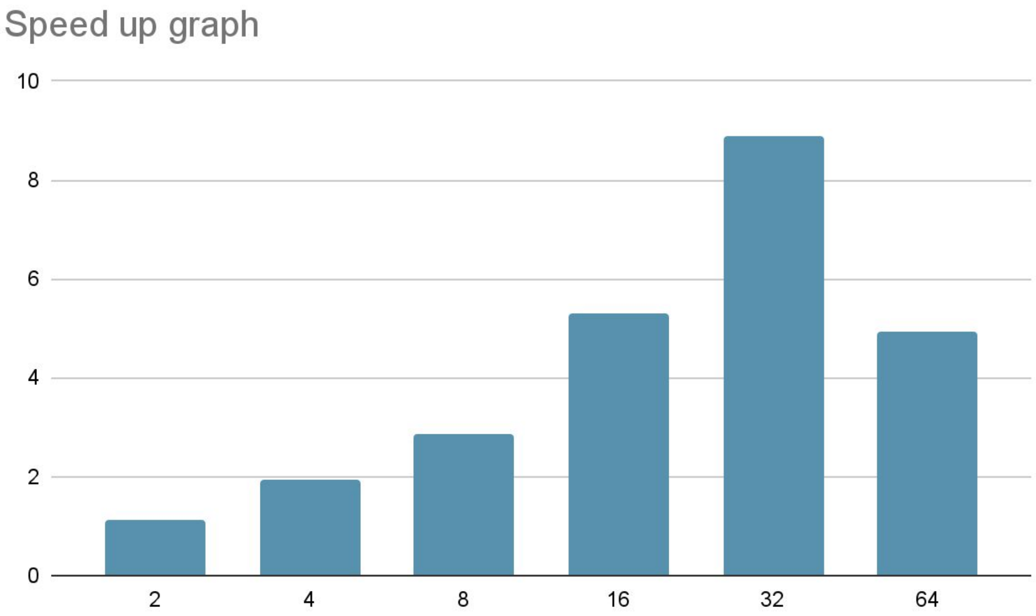
# Results

# 10k vertices - 5% density -2.5M edges

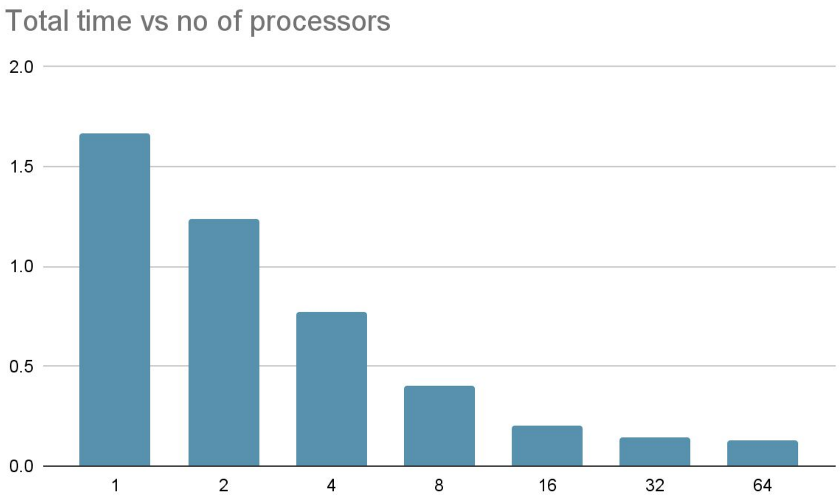
| Processors | Time | Speed up |
|---|---|---|
| 1 | 0.432887 | |
| 2 | 0.381065 | 1.135992547 |
| 4 | 0.224477 | 1.928424738 |
| 8 | 0.150779 | 2.871003256 |
| 16 | 0.081652 | 5.301609269 |
| 32 | 0.042923 | 8.877874333 |
| 64 | 0.045468 | 4.937032638 |

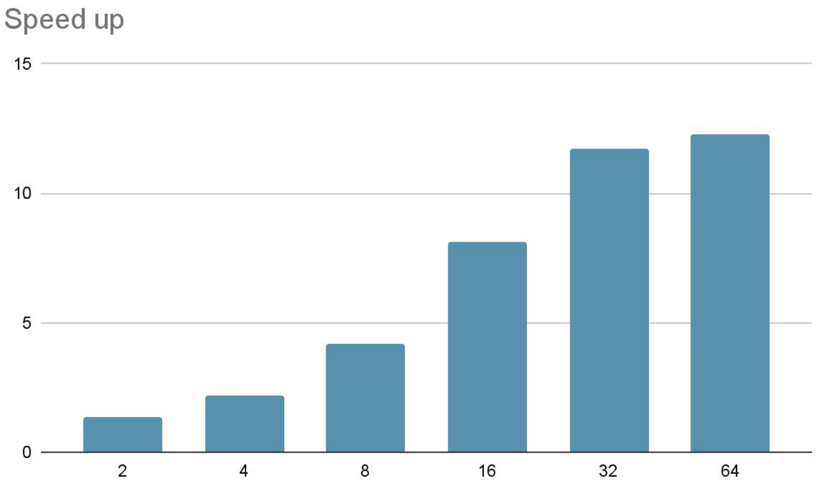


# 10k vertices - 5% density -2.5M edges

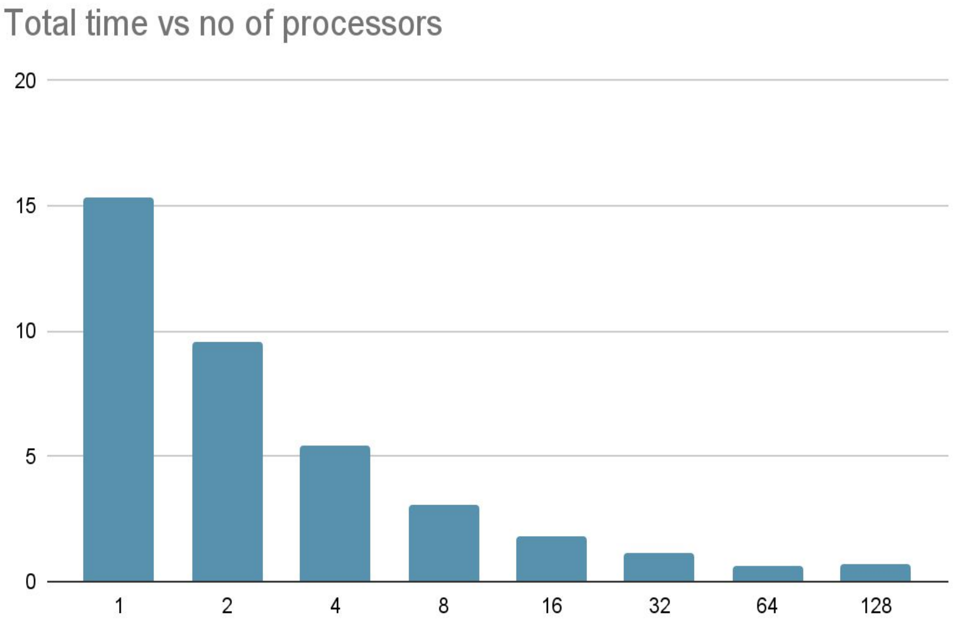


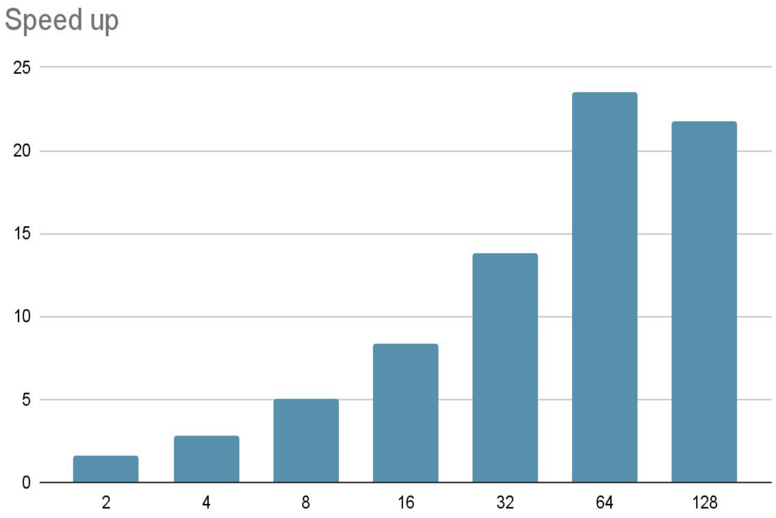
# 10k vertices - 10 M edges



# 10k vertices - 40M edges



# Observations

- Parallel Kruskal performs well on large data.
- The inflection point/ dip  is shifting rightwards as we increase the amount of data we are operating with.

# References


- Loncar-TET-Springer.pdf (scl.rs)
- Kruskal's Minimum Spanning Tree (MST) Algorithm - GeeksforGeeks

# Thank you