MST using Prim's Algorithm

Final presentation

SAI KIRAN MUNDRA

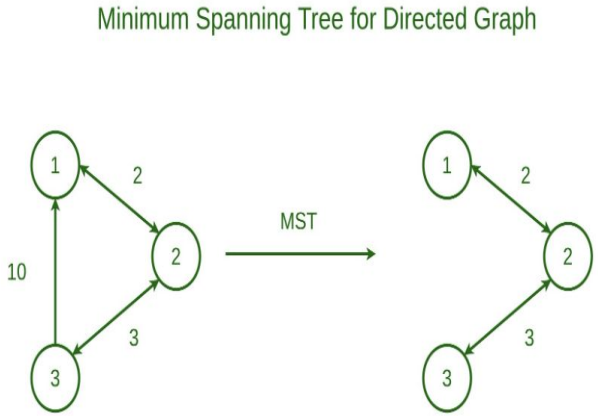
CSE 633 - Parallel Algorithms

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

Prim's algorithm - Sequential

- The algorithm starts with an empty spanning tree. The idea is to maintain two sets of vertices.
- The first set contains the vertices already included in the MST, and the other set contains the vertices not yet included.
- At every step, it considers all the edges that connect the two sets and picks the minimum weight edge from these edges.
- After picking the edge, it moves the other endpoint of the edge to the set containing MST.

Time Complexity:

O(V^2) - V is number of vertices.

Implementation

- Implementations of Prim's algorithm commonly use auxiliary array d of length n to store distances (weight) from each vertex to MST.
- In every iteration a lightest weight edge in d is added to MST and d is updated to reflect changes.

Approach for Parallelization

- Two steps can be parallelized:
  - selection of the minimum-weight edge connecting a vertex not in MST to a vertex in MST,
  - and updating array d after a vertex is added to MST

Pseudo code for parallelization

- Initialization:
  - Let the set of vertices V be divided into subsets V_1, V_2, V_3, ….. V_p
  - Assign each subset to a different process.
  - While the MSTvertices not equal to V:
    - For each process P_i:
      - Find the minimum-weight edge e_i connecting MST to vertices in Vi
      - Send e_i to the root process using MPI_Reduce to find the global minimum weight edge Emin.

Pseudo code for parallelization

- If current process is root:
  - Select the global minimum-weight edge Emin among the received edges
  - Add Emin to MST
  - Broadcast E min to all processes.
- Continue till all vertices are added in the MST.

- It can proved that Time Complexity is O((n^2/p) + (nlogp))

Implementation in MPI

- I have used MPI to implement the parallel Prim's Algorithm

Initial Results

10K vertices, 1% density ~ 500K edges

Nodes/ Processors	Total time
1	0.1687
2	0.1248
4	0.0692

Slurm script

#!/bin/bash

#SBATCH --nodes=4

#SBATCH --ntasks-per-node=1

#SBATCH --constraint=IB|OPA

#SBATCH --time=00:10:00

#SBATCH --partition=general-compute

#SBATCH --qos=general-compute

#SBATCH --job-name="parallel-prim-4node-1core"

#SBATCH --output=out-4-1.txt

#SBATCH --exclusive

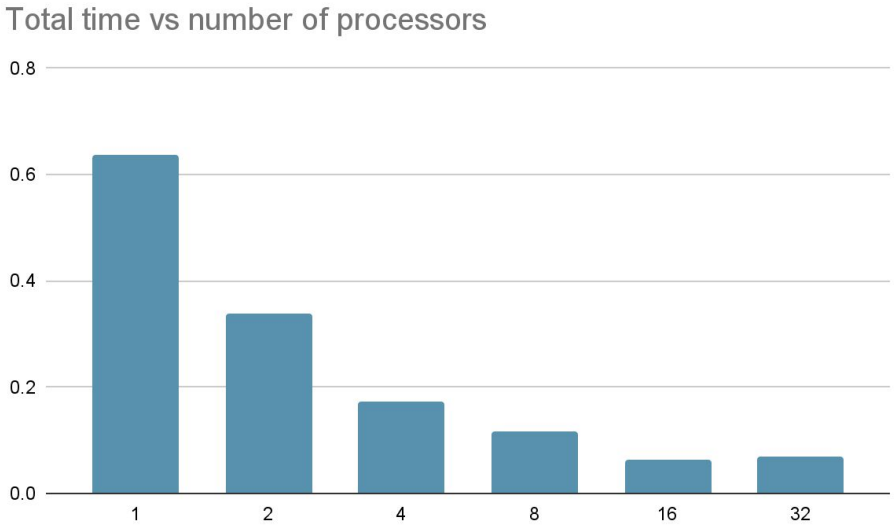
Slurm Script for MPI:

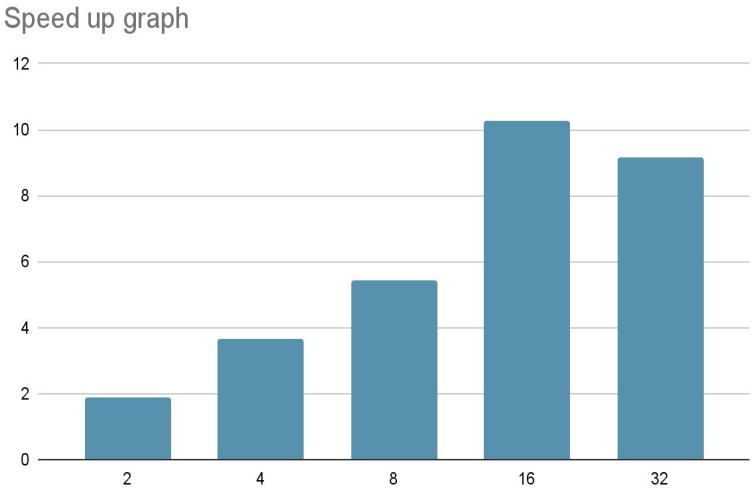
```
#!/bin/bash

#SBATCH --nodes=5
#SBATCH --ntasks-per-node=3
#SBATCH --constraint=IB|OPA
#SBATCH --time=00:10:00
#SBATCH --partition=general-compute
#SBATCH --qos=general-compute
#SBATCH --job-name="hello_world-5node-3core"
#SBATCH --output=output-15pe.txt
#SBATCH --exclusive
```

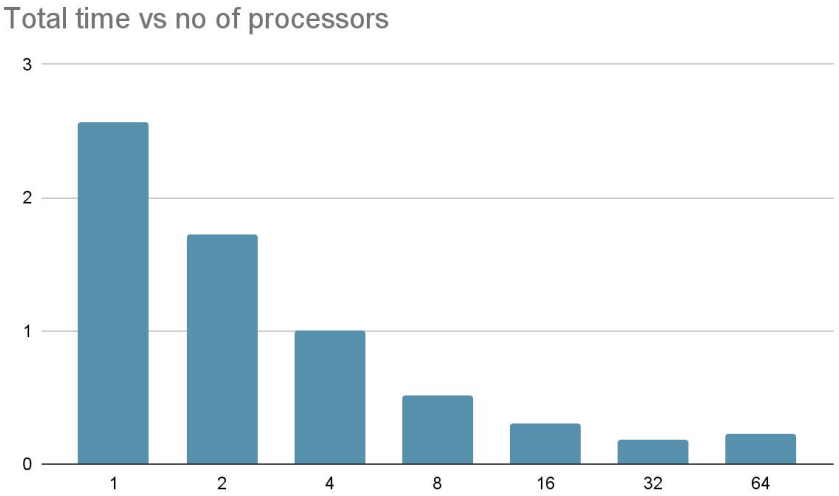
Final Results

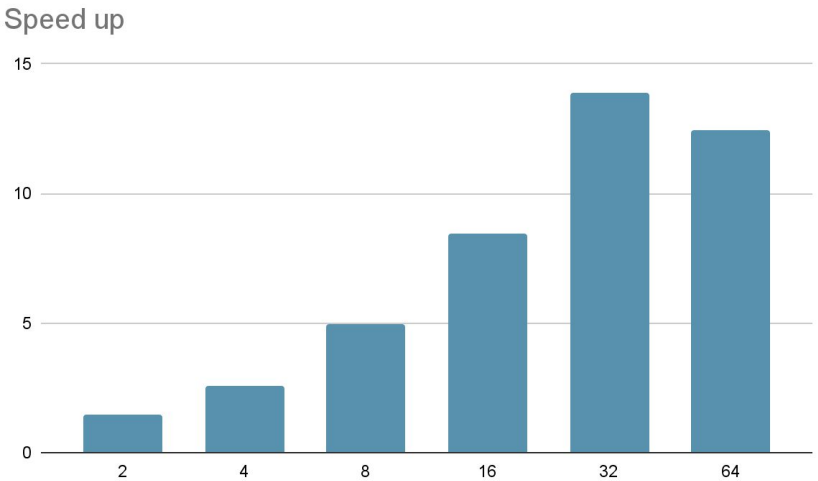
10k vertices - 5% density -2.5M edges



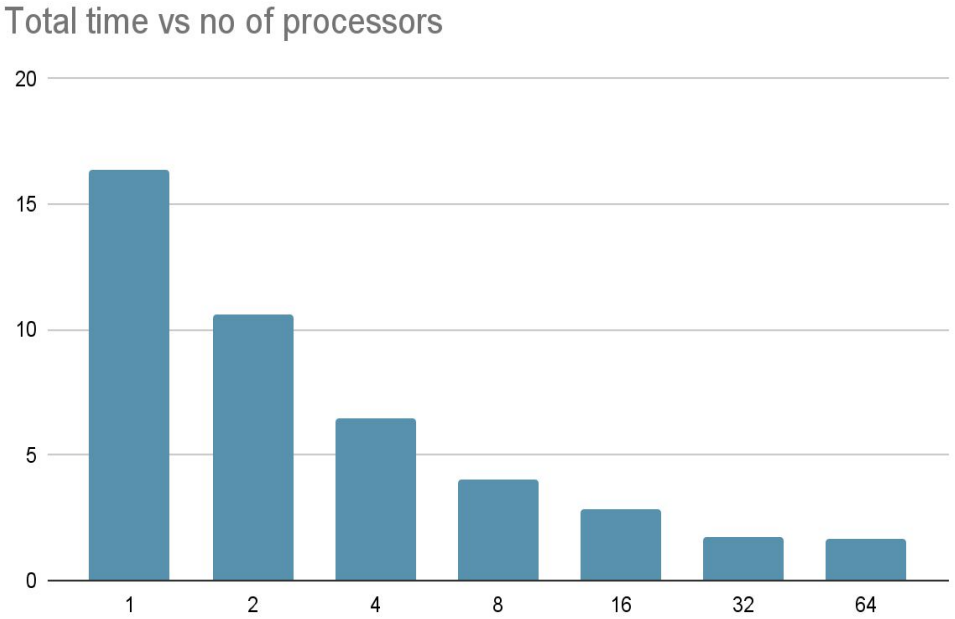


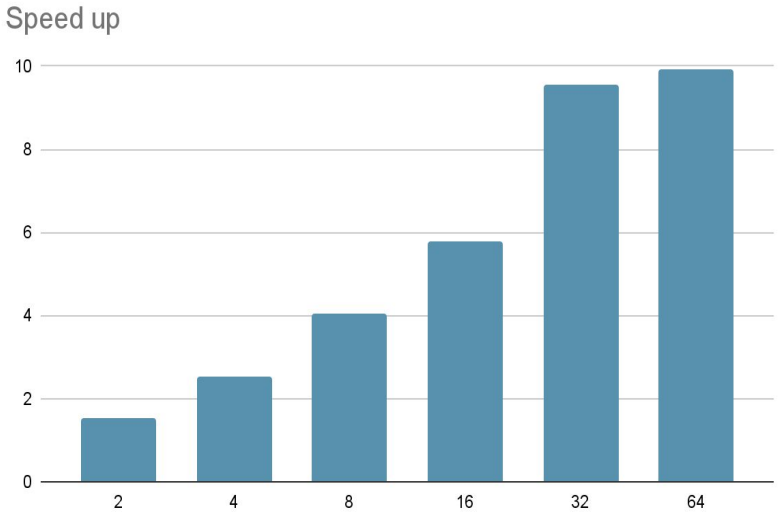
10k vertices - 10 M edges



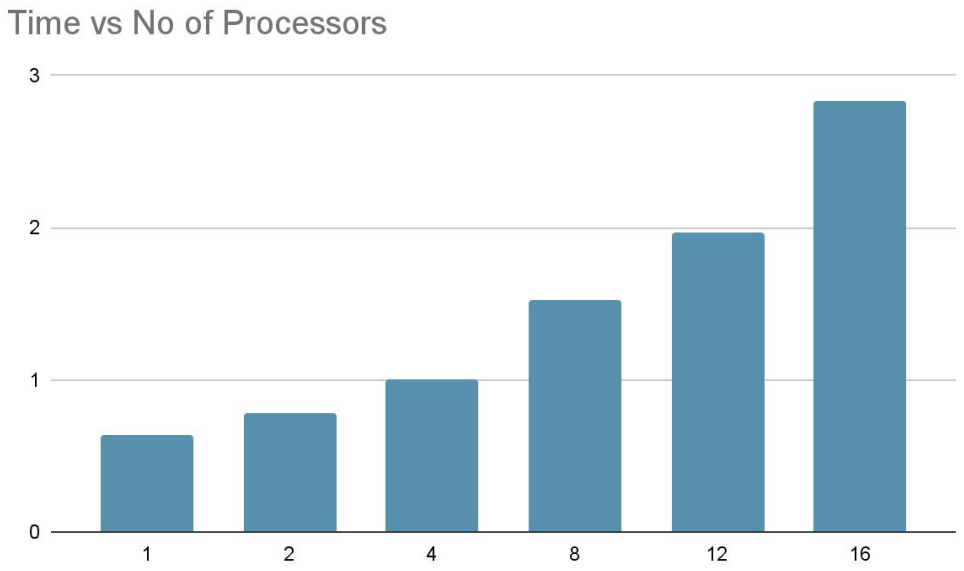


10k vertices - 40M edges





Weak Scaling - 2.5M edges per processor.



Observations

- Parallel Prim's Algorithm is performing better on larger data.
- As we increase the amount of data we are operating with the inflection point is shifting rightwards.
- In weak scaling, we can observe that despite we increase the data in proportion to the number of processors, we can observe the increase in completion time due to the sequential portion of the parallelised algorithm aligning with the Gustafson's law.

References

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

Thank you