University at Buffalo The State University of New York

CSE 708 – Programming Massively Parallel Systems

Parallel BFS For Distributed Memory with 2D Partitioning

Under guidance of Dr. Russ Miller

Presentation by – Arshabh Semwal

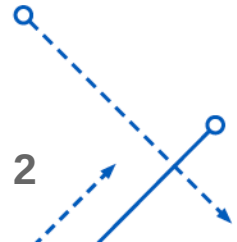
UB Person Number - 50419031

Motivations

Graph processing operates on a large volume of highly connected data.

Real-world applications of graph processing includes:

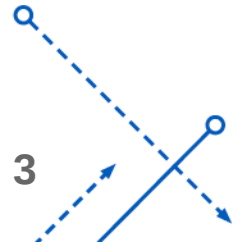
- Social network
- Digital maps
- Webpage hyperlinks
- Very Large-Scale Integration (VLSI) layout of integrated circuit (IC) and more



Applications of BFS

Finding Shortest Path: In an unweighted graph, the shortest path is the path with least number of edges. With Breadth First, we always reach a vertex from given source using the minimum number of edges.

Finding Minimum Spanning Tree for unweighted graph In an unweighted graph, in case of unweighted graphs, any spanning tree is Minimum Spanning Tree, and we can use either Depth or Breadth first traversal for finding a spanning tree.



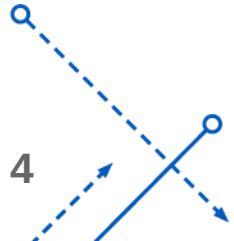
Applications of BFS

Peer to Peer Networks: In Peer-to-Peer Networks like BitTorrent, Breadth First Search is used to find all neighbor nodes.

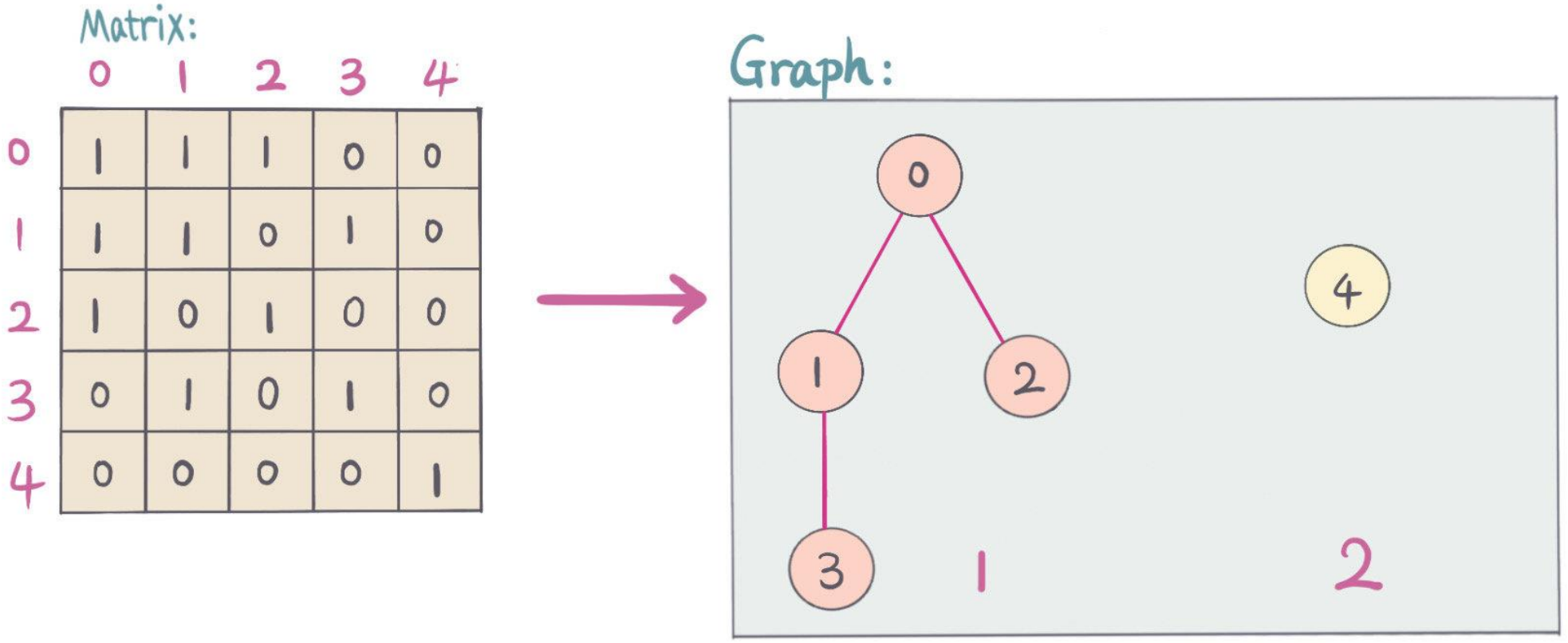
Social Networking Websites: In social networks, we can find people within a given distance ‘k’ from a person using Breadth First Search till ‘k’ levels.

GPS Navigation systems: Breadth First Search is used to find all neighboring locations.

Broadcasting in Network: In networks, a broadcasted packet follows Breadth First Search to reach all nodes.



Adjacency Matrix Representation of a Graph



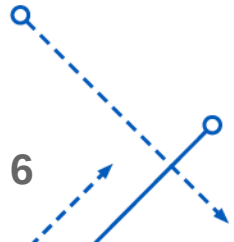
Note: Image Source- https://tva1.sinaimg.cn/large/007S8ZIlly1ghlud2fq7sj31bh0n4jub.jpg

Sequential BFS

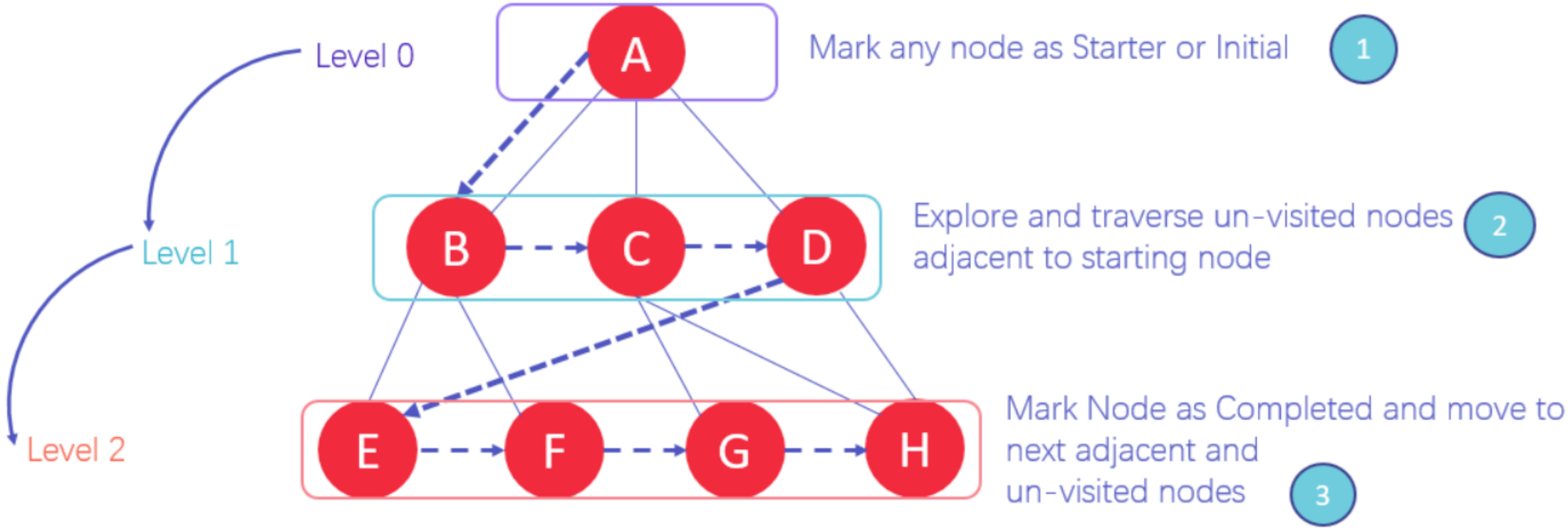
BredthFirstSerach(**G, A**): //G is graph and A is source node

1. Let **Q** be the queue
2. **Q**.enqueue(**A**)
3. Mark **A** node as visited.
4. While (**Q** is not empty)
5. **B** = **Q**.dequeue( )
6. Processing all the neighbors of **B**
7. For all neighbors **C** of **B**
8. If **C** is not visited, **Q**. enqueue(**C**)
9. Mark **C** a visited

For Graph representation in Adjacency Matrix of size *N x N*, the time complexity for BFS is $O(N^2)$



CONCEPT DIAGRAM



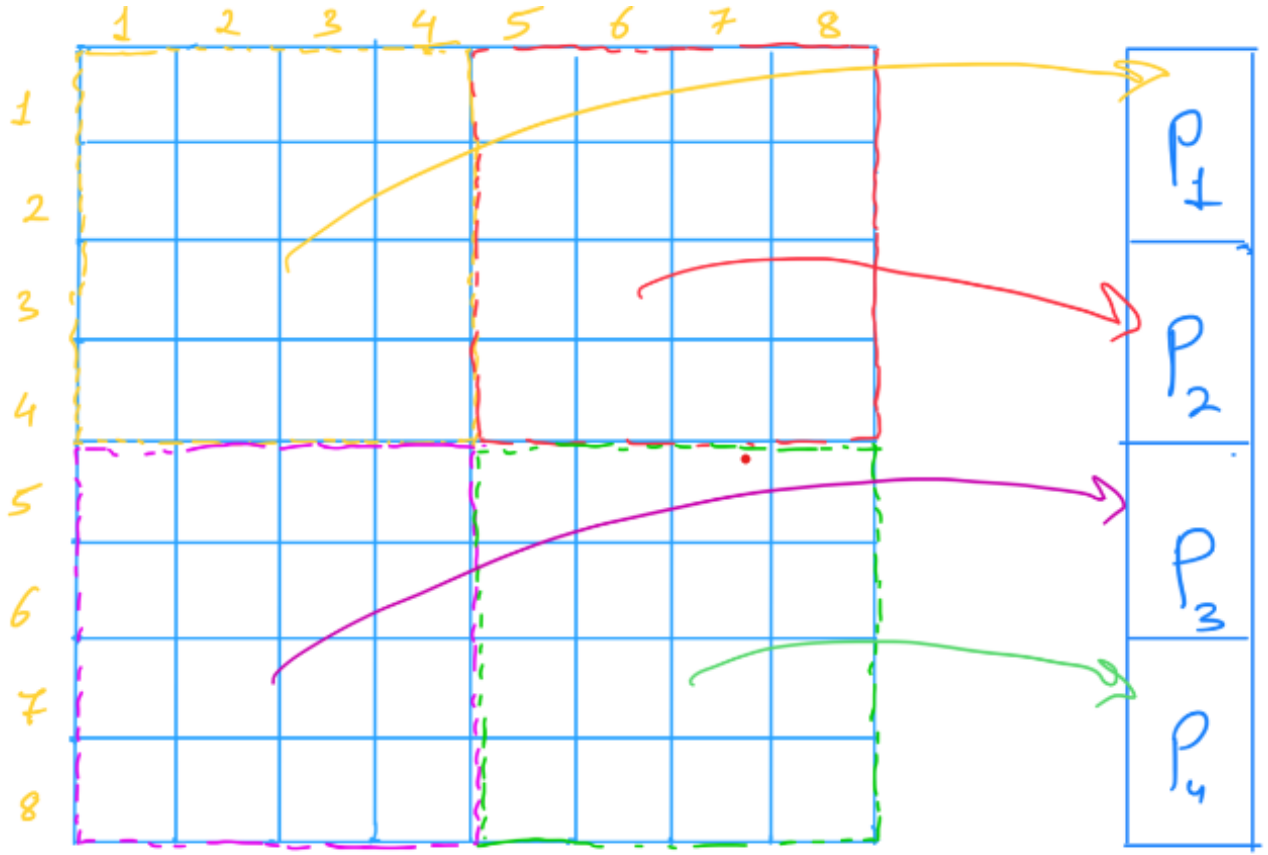
Note: Image Source from guru99.com (https://www.guru99.com/breadth-first-search-bfs-graph-example.html)

2-D Partition Parallel BFS Algorithm

```
1    define 2_D_distributed_memory_BFS( graph(V,E), source s):
2        // normal initialization
3        for all v in V do
4            d[v] = -1;
5        d[s] = 0;
6        // begin BFS traversal
7        for l = 0 to infinite do:
8            F = {the set of local vertexes with level l}
9            // all vertexes traversed
10           if F = {} for all processors then:
11               terminate the while loop
12           // traverse vertexes by sending message to selected processor
13           for all processor q in this processor-column do:
14               Send F to processor q
15               Receive F_q^r from q
16           // deal with the receiving information after the frontier traversal
17           F^r = Union{F_q^r} for all q
18           N = {neighbors of vertexes in F^r using edge lists on this processor}
19           // broadcast the neighbor vertexes by sending message to their owner processor
20           for all processor q in this processor-row do:
21               N_q = {vertexes in N owned by processor q}
22               Send N_q to processor q
23               Receive N_q^r from q
24           // form the next frontier used for next layer traversal
25           N^r = Union{N_q^r} for all q
26           // layer distance update
27           for v in N^r and d(v) = -1 do:
28               level = l + 1
```

Note : Image Source Wikipedia: https://en.wikipedia.org/wiki/Parallel_breadth-first_search

2-D Partition of Data



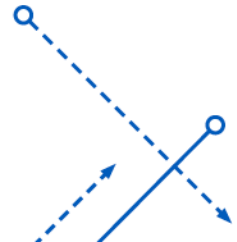
NOTE: The key idea in parallelizing the BFS algorithm is to synchronize levels in which a node is visited from the start node

2-D Partition of Data

- Let total number of processor are P
  Then P = M*M i.e. P is a perfect square.

- Then the adjacent matrix of size N*N is divided into (N/M)*(N/M) size.

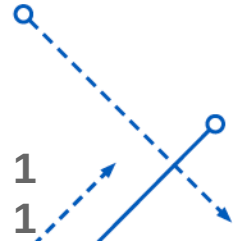
- Let the total vertices are N (since size of adjacency matrix is N*N) then each processor will have N/P vertices.



Parallel BFS 2-D Partition

The main steps of BFS traversal in the following algorithm are:

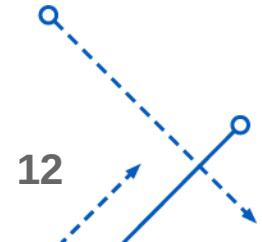
- Construct the frontier with vertexes from local storage.
- Terminate the traversal if frontier from all processors are empty.
- Expand phase - based on local vertexes, only send messages to processors in processor-column to tell them these vertexes are in the frontier, receive messages from these processors.
- Merge all receiving messages and form the net frontier N.
- Fold phase - based on the local vertexes in next frontier, send messages to owner processors of these vertexes in processor-row.
- Merge all receiving messages and update the distance value of vertexes in the next frontier.



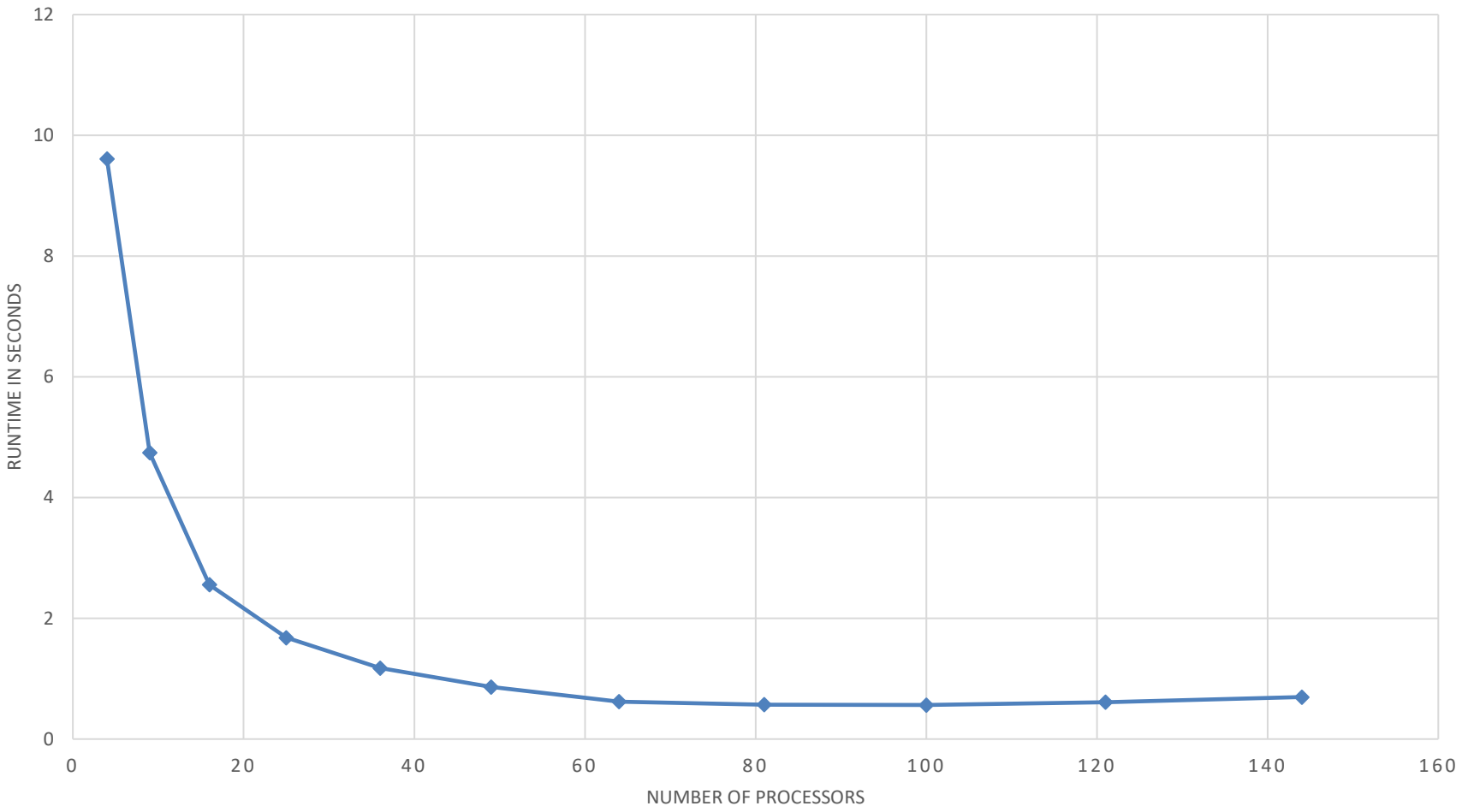
Data Used

Two data are used:

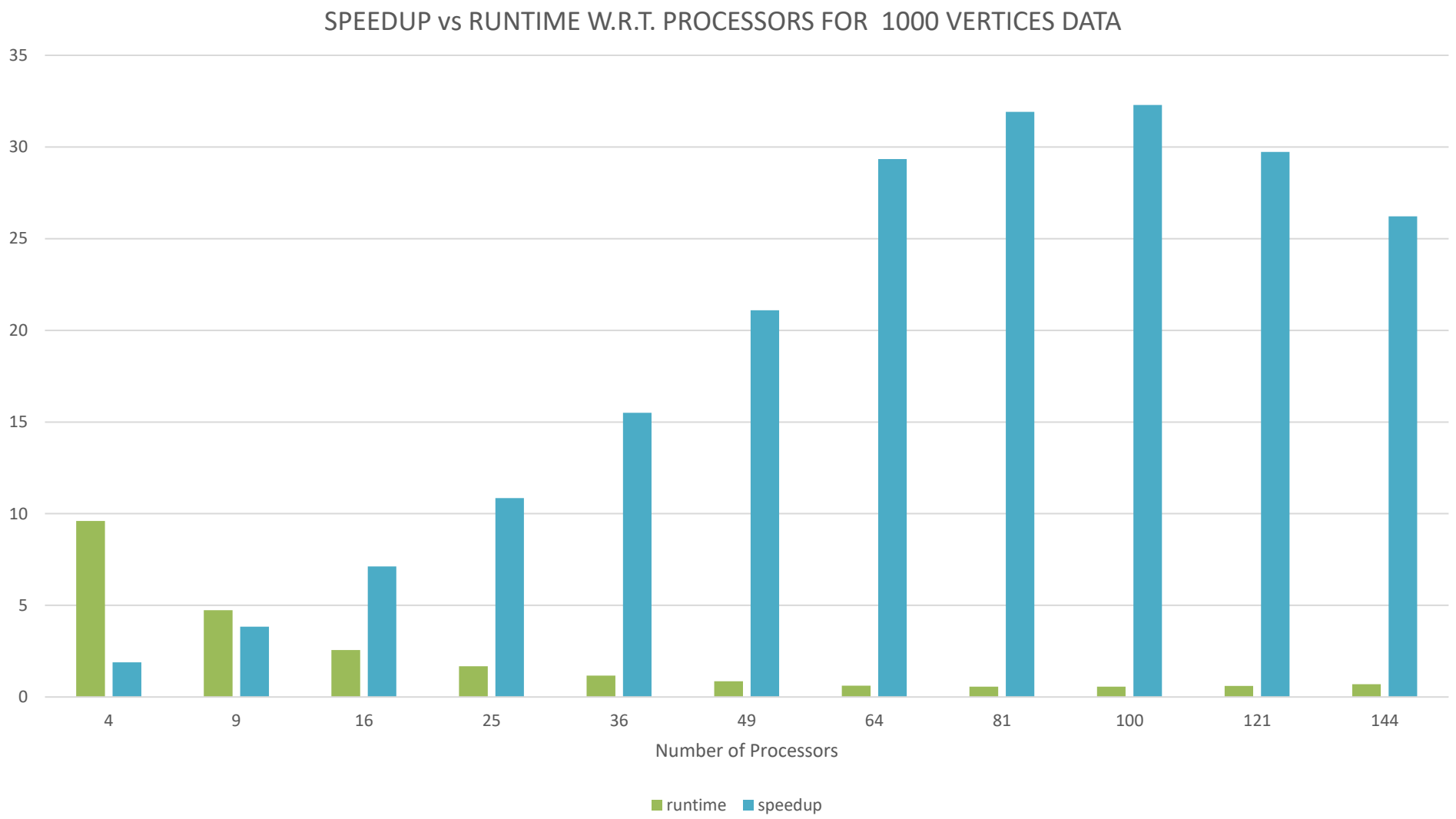
1. 1K Nodes data generated.
2. 400K Nodes data generated.



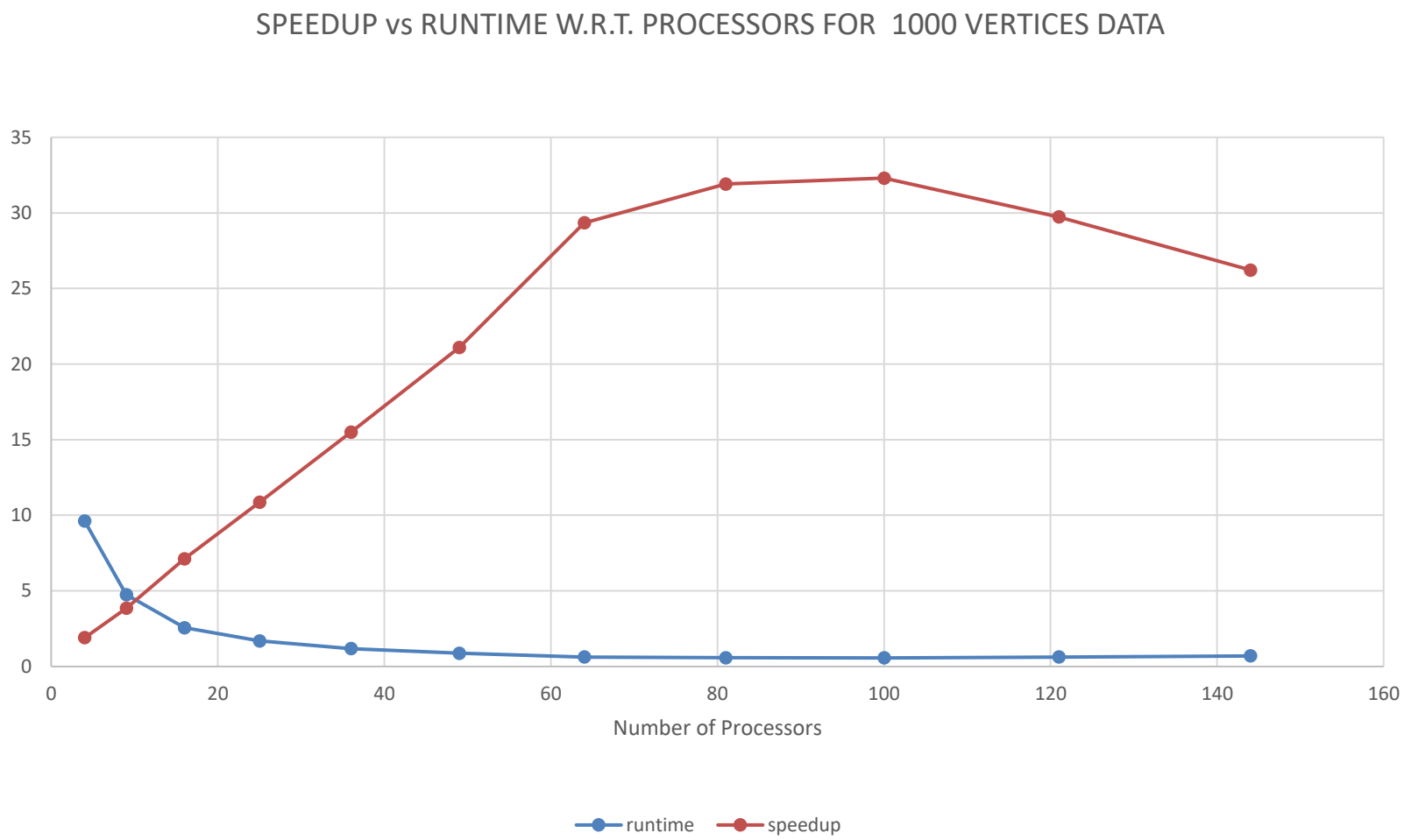
Runtime as a Function of Processors on 1000 vertices dataset



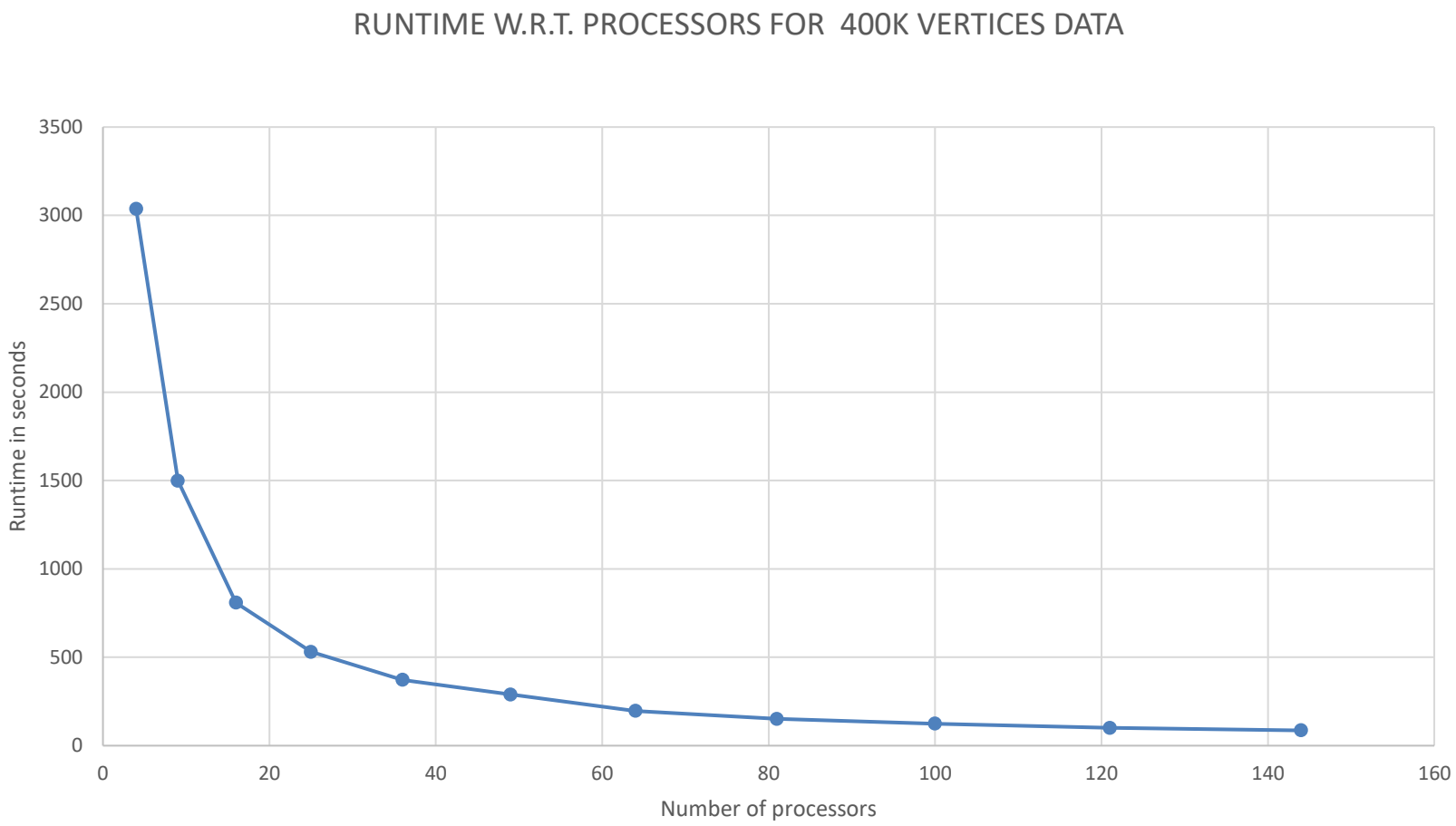
Runtime Vs Speedup on 1000 nodes dataset



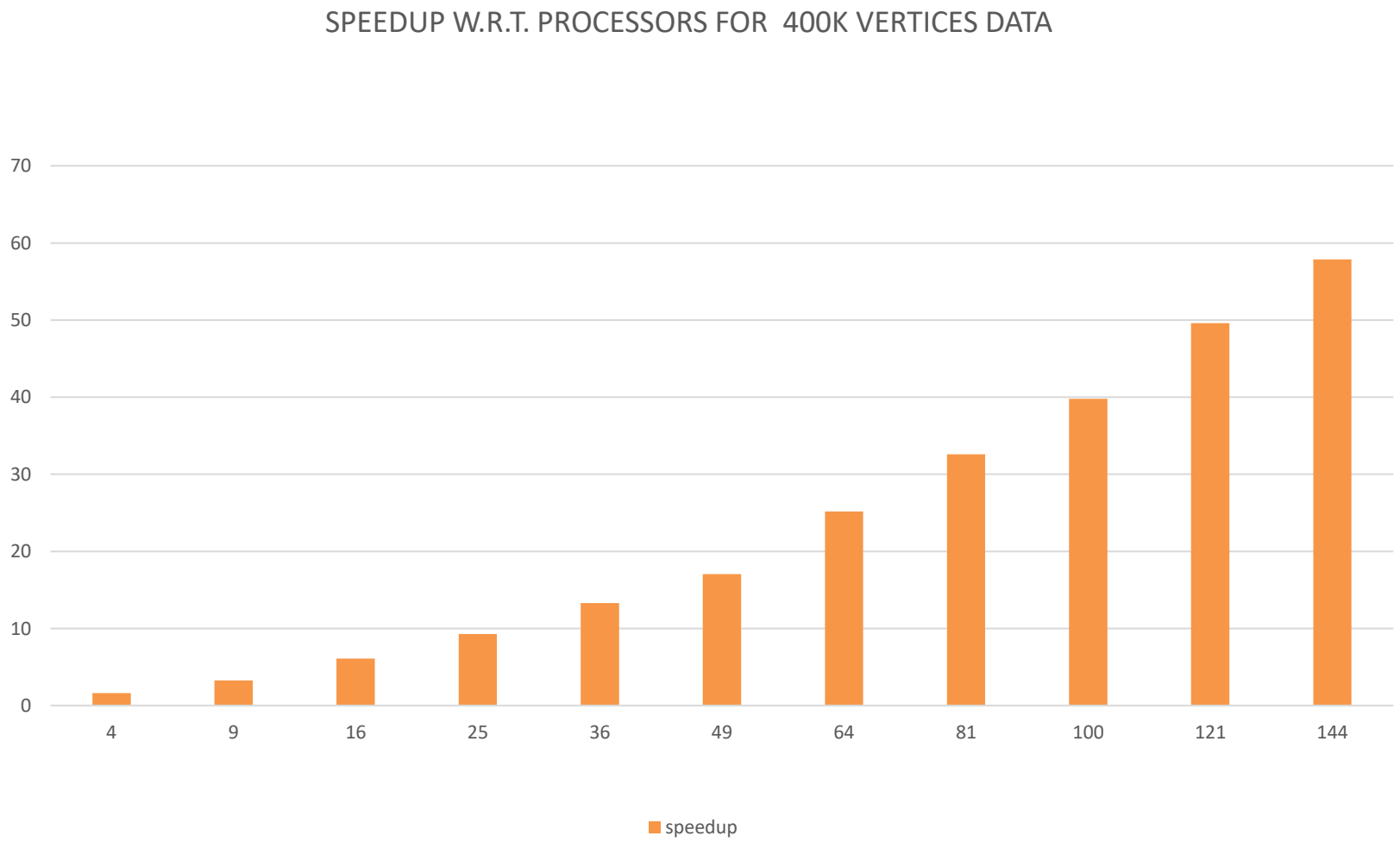
Runtime Vs Speedup on 1000 nodes dataset



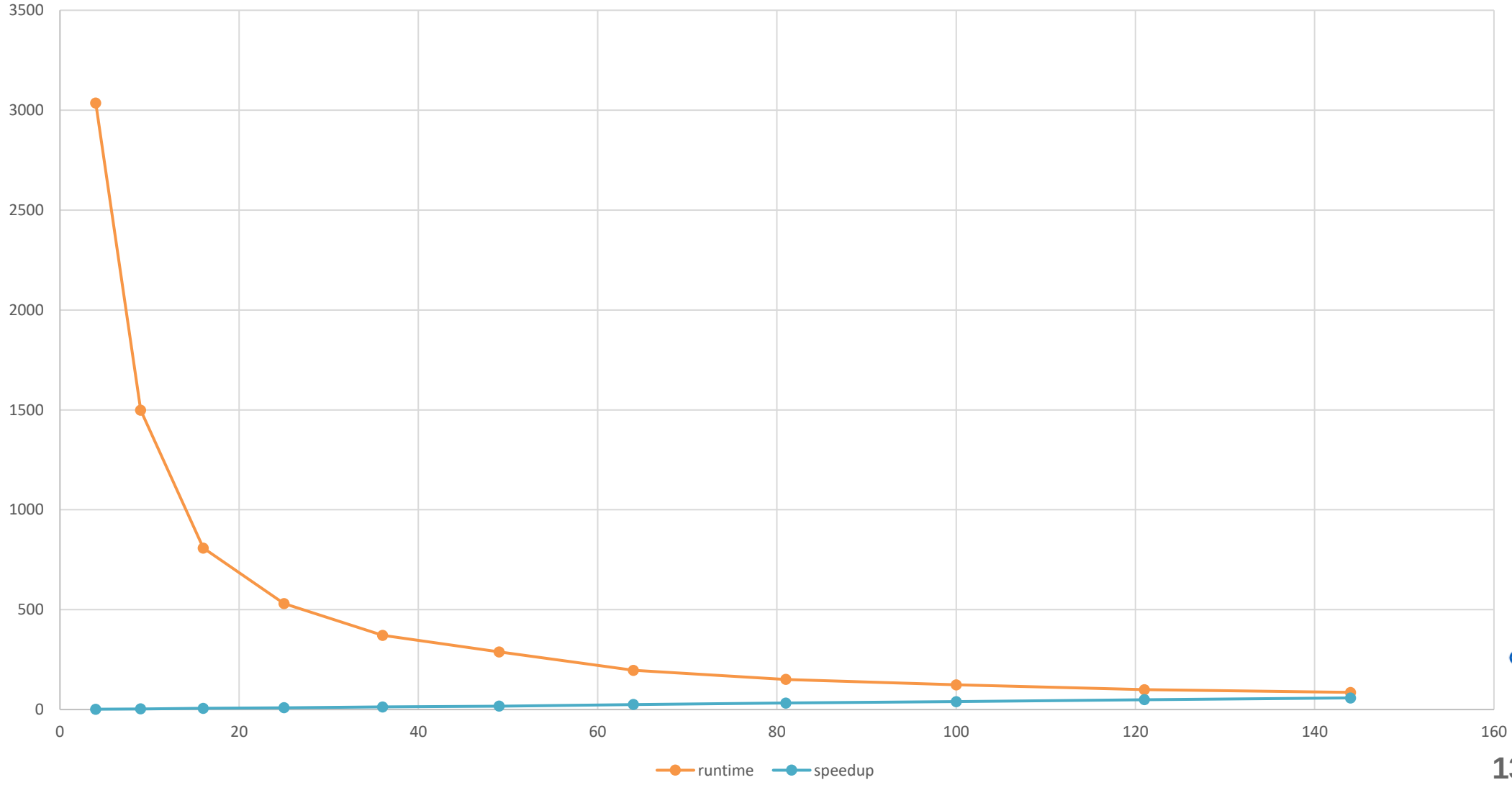
Runtime as a Function of Processors on 400K vertices dataset



Speedup on 400K nodes dataset



SPEEDUP vs RUNTIME W.R.T. PROCESSORS FOR 400K VERTICES DATA



Observations:

1. For smaller input of 1000 vertices the runtime slowed done after 100 processors. This shows that the network overhead become costlier in comparison to the speedup achieved by the division of computation across processors.

2. For 400K vertex dataset runtime kept on decreasing as the number of processors increased from 4 to 144. This shows that for larger dataset this algorithm performs very well and achieve consistent speedup.

3. This algorithm works best with larger dataset where computation per processors are higher which enables it to take advantage of parallel processing and gain considerable sustainable speedups. Also, higher density graph are much better suited than lower density graph as adjacency matrix is not a suitable data structure to store lower density graphs.

References

1. Parallel BFS 2-D Partition - https://en.wikipedia.org/wiki/Parallel_breadth-first_search

2. Parallel Breadth-First Search on Distributed Memory Systems - https://people.eecs.berkeley.edu/~aydin/sc11_bfs.pdf

3. Image reference on slide 7: https://www.guru99.com/breadth-first-search-bfs-graph-example.html

4. Image reference on slide 5: https://tva1.sinaimg.cn/large/007S8ZIlly1ghlud2fq7sj31bh0n4jub.jpg