Deep Bodra
5801 1841

README

TEAM MEMBERS

Individual Project

RUNNING THE CODE

Run project2.fsx

```
dotnet fsi --langversion:preview .\project2.fsx numNodes topology algorithm
```

Run project2_bonus.fsx

```
dotnet fsi --langversion:preview .\project2_bonus.fsx numNodes topology algorithm numFailedNodes
```

WHAT IS WORKING

1. All algorithms converge for all the topologies- line, 2D, imp2D, full
2. However in case of PushSum, for line and 2D, the nodes don't calculate the exact sum when the number of nodes is very large. The reason for this is explained in the implementation section

IMPLEMENTATION

1. How did I make the line topology to converge?
    a. Earlier when I made the nodes to stop transmitting gossip once they converge, then the line took a long time to converge or did not converge at all
    b. It took a long time because some of the nodes sent the gossip back instead of sending it to the nodes that had not received any gossip. This resulted in the gossip being trapped in the network
    c. At times, it did not converge because some of the nodes converged and they did not participate thereafter and the gossip kind of died
    d. So, I changed the implementation a little bit. Before convergence, the nodes would send the gossip to a random neighbor every 300 ms (defined as Tick)
    e. After convergence, the nodes would send the gossip to a random neighbor only when they receive the gossip which made the line topology to converge
2. Convergence condition for PushSum
    a. To make sure the nodes calculate correct sum, I changed the convergence condition
    b. The nodes converge when the difference in the ratio (s/w) stays the same (precision: 10^{-10}) for 5 consecutive rounds/ticks instead of 3

c. This made sure that the sum is numerically correct for imp2D and full topologies
d. However for line and 2D, this value (5) is still low. It has to be much larger for them to converge. But it take a very long time. So I have kept it at 5 instead of going for 10 or 15

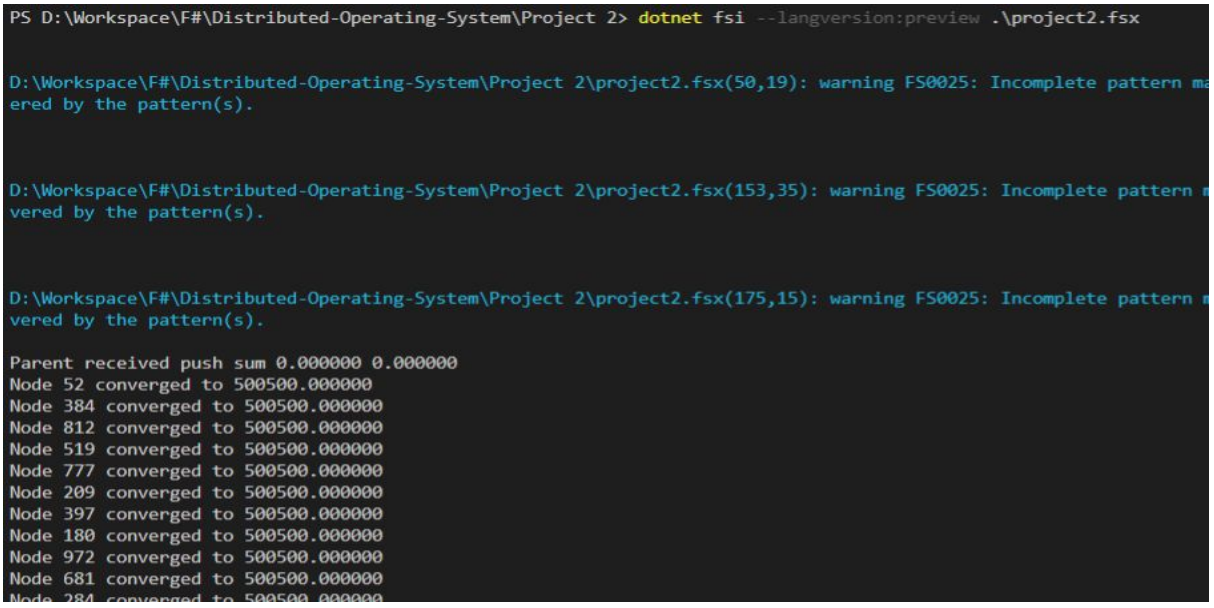
LARGEST PROBLEM

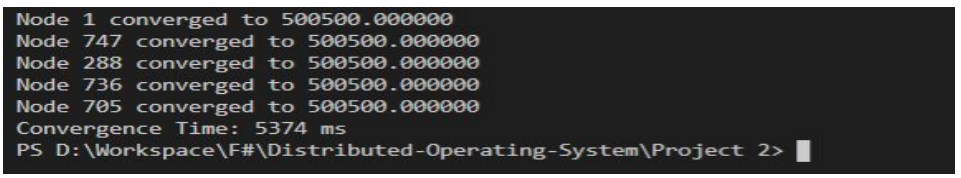
	Gossip	PushSum
Line	1000	1000
2D	10000	1000
imp2D	10000	5000
Full	10000	5000

1. Although I haven't tried larger number of nodes than this (because of execution time constraints) I am sure that all the nodes will converge because of the modification I made to the algorithm as explained in implementation section

RESULT

```
dotnet fsi --langversion:preview .\project2.fsx 1000 full pushsum
```



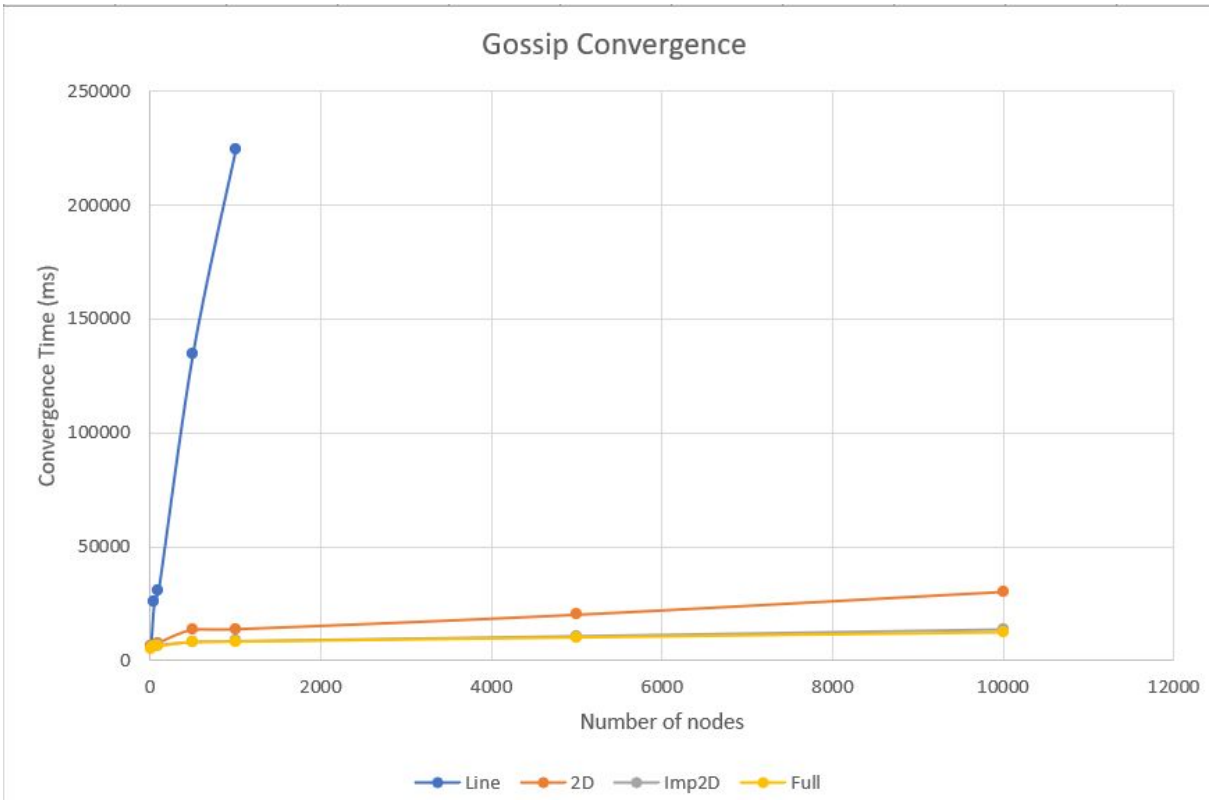


PLOTS

SEED NODE

For simplicity in explanation, let call a node that receives the gossip/ message for the first time until it converges as the seed node. If a node converges then it is no more a seed node

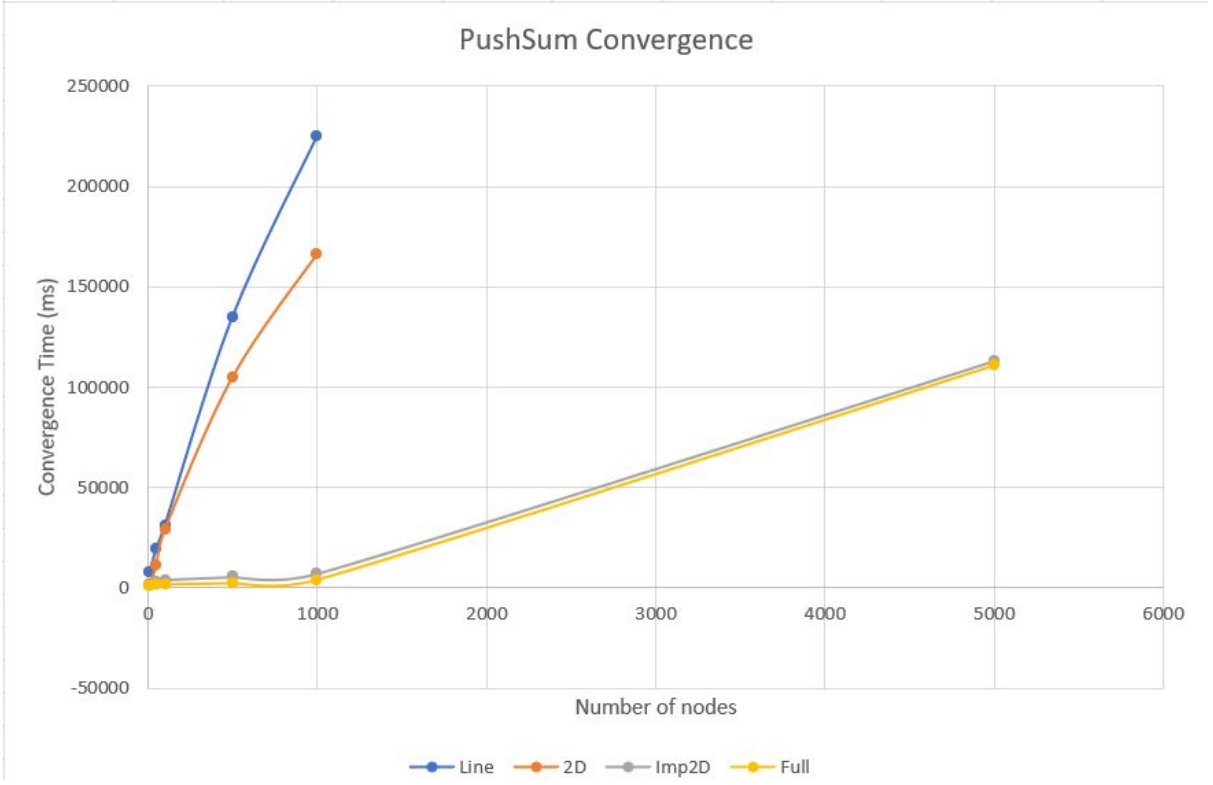
1. Gossip



   a. Line
      i. The convergence time for line grows exponentially because the gossip is trapped in portions of the network
      ii. Assuming that a gossip arrives from the left side at a node, there is a 50% chance that the gossip will be sent back to the same node. So the chances of the gossip going to the end of network is very low
      iii. For line topology a node can generate a maximum of 1 seed node
   b. 2D
      i. Each node(not on the boundary) can give rise to a maximum of 4 seed nodes
      ii. So the gossip propagates very quickly
      iii. The gossip seems to propagate in a radial fashion with the start node as the center

c. Imp2D
    i. Imp2D can generate a maximum of 5 seed node
    ii. Imp2D is significantly better than 2D and almost as good as Full
    iii. The gossip spreads radially like 2D but since each node has extra random neighbor, the radial centers are more than 2D
d. Full
    i. This is the fastest of all except that the difference between Imp2D and Full is not significant because the 5th random neighbor in Imp2D spreads the rumor everywhere

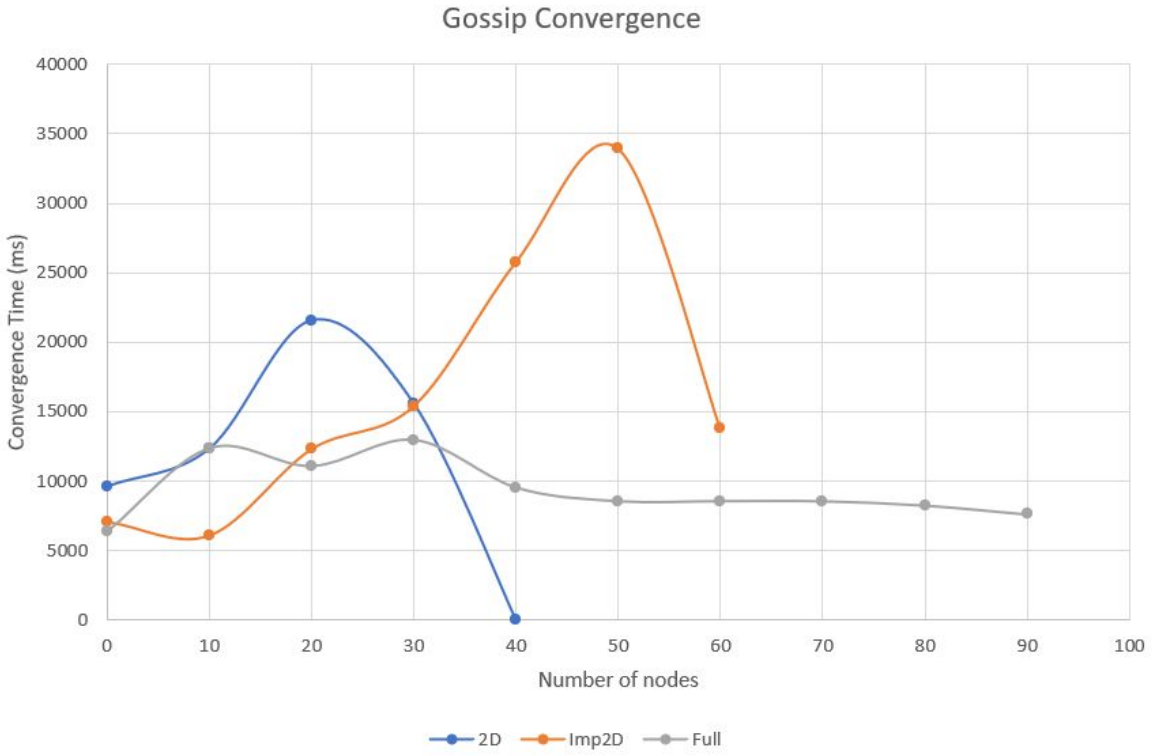
2. PushSum



a. Line
    i. Line is as bad as anyone would expect
b. 2D
    i. 2D is faster than line but the convergence time increases exponentially too
c. Imp2D
    i. The random 5th neighbor is really very powerful for this topology
    ii. It keeps the large portion of the network busy and doesn't allow the nodes to converge at the start
    iii. This makes the nodes converge to the exact value of the SUM

d. Full
    i. Although the difference between Full2D and Imp2D does not seem much in the graph, Full is actually 1.8 times faster than Imp2D

3. Gossip convergence with failure

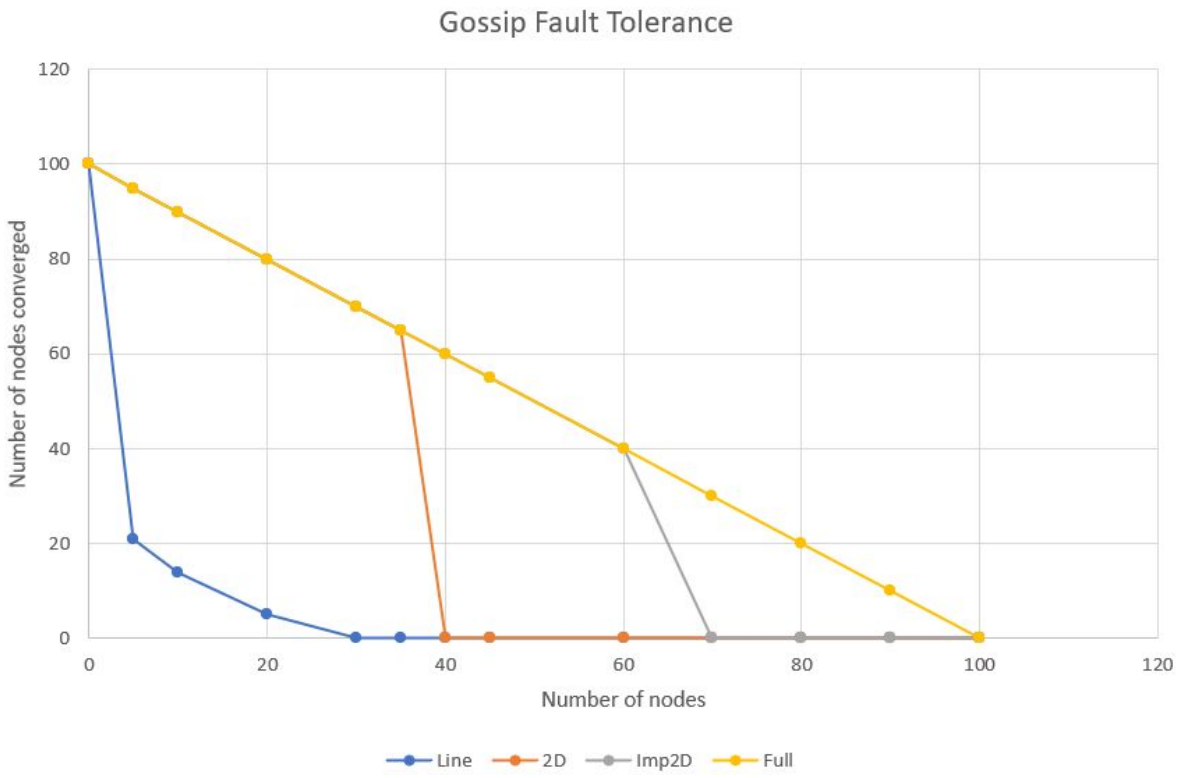


a. 2D
    i. The convergence time increases initially because the effective number of neighbors of a node has decreased
    ii. It reaches its maximum and then stops decreasing because the number of active nodes is very less and they can converge in short time

b. imp2D
    i. The abnormal peak at 50% failure is like a line topology behavior
    ii. Half of the nodes have failed and so a node only has 3 neighbors (2 adjacent) and 1 random link which may lead to a failed node with a probability of 0.5
    iii. The network has turned into a line kind of topology with some random links that offer convergence but with a delay

c. Full
    i. Logically the number of failed nodes shouldn't impact the convergence time a lot because the active nodes are always connected directly
    ii. The variation we see in the graph is due to the fact that the neighbors are picked at random while the gossip is to be sent

4. Gossip Fault Tolerance with failure



a. Line
    i. As expected the line topology is absolutely useless when the nodes may fail
    ii. It splits the network into n+1 parts where n is the number failure nodes
    iii. The maximum number of nodes that can converge is equal to the maximum size of the subnetwork
b. 2D
    i. It can converge even if 30% of the nodes fail
    ii. Thus the drop is linear with respect to the number of failure nodes
    iii. The number of nodes that can converge falls to zero after some threshold value (30<threshold<40)
c. Imp2D
    i. The same goes for Imp2D except that the threshold value is somewhere between 60 and 70
    ii. The remaining nodes can converge even if 60% of the nodes fail

d. Full
    i. This will converge no matter what because there is always a connection between active nodes

CONCLUSION

We used Actor model to implement gossip and possum algorithm without having to worry about threads and mutexes