Distributed Operating System Principles Project-2

Kavya Gopal Rema Veeranna Gowda

1 Problem Statement

The goal of this project is to determine the convergence of Gossip and push sum algorithms through a simulator based on actors written in F. The efficiency of these algorithms depends on the underlying topology. Another goal of this project is to build and experiment with various topologies like line, full network, 3D, and Imperfect 3D grids. The topology determines who is considered a neighbor in the above algorithms.

2 Algorithm

Gossip Algorithm

Gossip Algorithm for information propagation.The Gossip algorithm involves the following:

- Starting: A participant(actor) it told/sent a rumor(fact) by the main process
- Step: Each actor selects a random neighbor and tells it the rumor
- Termination: Each actor keeps track of rumors and how many times it has heard the rumor. It stops transmitting once it has heard the rumor 10 times (10 is arbitrary, you can select other values).

Push-Sum algorithm for sum computation

- State: Each actor Ai maintains two quantities: s and w. Initially, s = xi = i (that is actor number i has value i, play with other distribution if you so desire) and w = 1
- Starting: Ask one of the actors to start from the main process.
- Receive: Messages sent and received are pairs of the form (s, w). Upon receiving, an actor should add the received pair to its own corresponding values. Upon receiving, each actor selects a random neighbor and sends it a message.

- Send: When sending a message to another actor, half of s and w is kept by the sending actor, and half is placed in the message. Sum estimate: At any given moment of time, the sum estimate is was where s and w are the current values of an actor.
- Termination: If an actor's ratio was did not change more than 1010 in 3 consecutive rounds the actor terminates. WARNING: the values s and w independently never converge, only the ratio does.

Topologies

The actual network topology plays a critical role in the dissemination speed of Gossip protocols. Another goal of this project is to experiment with various topologies. The topology determines who is considered a neighbor in the above algorithms.

⇒ Full Network: Every actor is a neighbor of all other actors. That is, every actor can talk directly to any other actor.

⇒ 3DGrid:Actorsforma2Dgrid.Theactorscanonlytalktothegrid neighbors.

⇒ Line: Actors are arranged in a line. Each actor has only 2 neighbors (one left and one right, unless its the first or last actor).

⇒ Imperfect3DGrid: Grid arrangement but one random other neighbor is selected from the list of all actors (6+1 neighbors).

3 Solution

Implementation Details

1. Firstly, we find a value 'n' such that the number of nodes is a perfect cube. We then use this value as the new number of nodes. This is a pre-requisite 3D and Imperfect 3D topologies to generate grids.
2. Build topology: We start by constructing a list of neighbors for each actor during its initiation. The neighbor's list consists of only the nodes that the specific node can send messages to and receive messages from.

Gossip Algorithm

Parent Actor

1. The parent actor acts as a supervisor for all nodes

2. It triggers the algorithm provided by the user as an argument for the provided topology by activating a child actor.

3. t periodically activates child actors to send messages to one of its neighbors.

4. The parent actor receives messages from each actor after it is converged. It maintains a dictionary for actors and sets the bool value to true.

5. When all actors send a message to the parent that it converged, the parent calculates the time of convergence and terminates the program.

Child Actor

1. The child actor initializes itself on receiving a message from the parent actor. child actor maintains references to its neighbors when the topology is built.

2. These actors have a dictionary that has a boolean value to determine whether the actor is converged or not. It is set to true when the actor receives a message at least 10 times.

3. When an actor receives a message, it updates the counter and picks one of its active neighbors from the list called "neighbours", and sends a message to it.

4. At a periodic interval, the actor wakes up and sends the message to itself from the time the first message to it is received until it terminates.

5. Once the child actor receives messages 10 times, it sends a message to the parent actor that it's complete.

Push Sum Algorithm

Parent Actor

1. The parent actor acts as a supervisor for all nodes

2. It triggers the algorithm activating a child actor and it sends a message which contain (S=i, W=1).

3. It periodically activates child actors to send messages to one of its neighbors.

4. The actor uses the list - "neighbors" to randomly select a neighbor to send a message to.

5. Once the counter reaches the total number of nodes, which means all nodes get a converged ratio R, the parent actor ends the algorithm and shows the time duration.

6. For push sum, we assume that convergence occurs when the nodes' ratio has not changed more than 10exp-10 in three consecutive rounds and the program terminates when all actors are terminated.

Child Actor

1. The child actor initializes itself on receiving a message from the parent actor.
2. Each child actor maintains references to its neighbors when the topology is built.
3. The actor nodes to send (s, w) pair messages in the push-sum algorithm
4. When an actor receives a message, the actor processes message. It adds sum and weight to itself and sends half of the sum and weight to a random neighbor and keeps the remaining half.
5. These actors have a dictionary that has a boolean value to determine whether the actor is converged or not. It is set to true when the actor receives a message at least 10 times.
6. At a periodic interval, the actor wakes up and sends the message to itself from the time the first message to it is received until it terminates.
7. Child actor also checks if the ratio R=S/W changes no more than in C consecutive rounds (default: C=3) before sending any Push message.

Running the code

To run the program, we need to provide 3 inputs:

dotnet fsi –langversion:preview project2.fix numOfNodes topology algorithm

- numOfNodes: number of nodes for the topology
- topology: full, 3D, line or Imperfect3D
- algorithm: gossip or push-sum

Output

Output*: Time duration: shows the millisecond time gap between the start and end of the algorithm.(Time of convergence)*

Termination *- The program is terminated when all of its actors receive the rumor message n (10) number of times.*

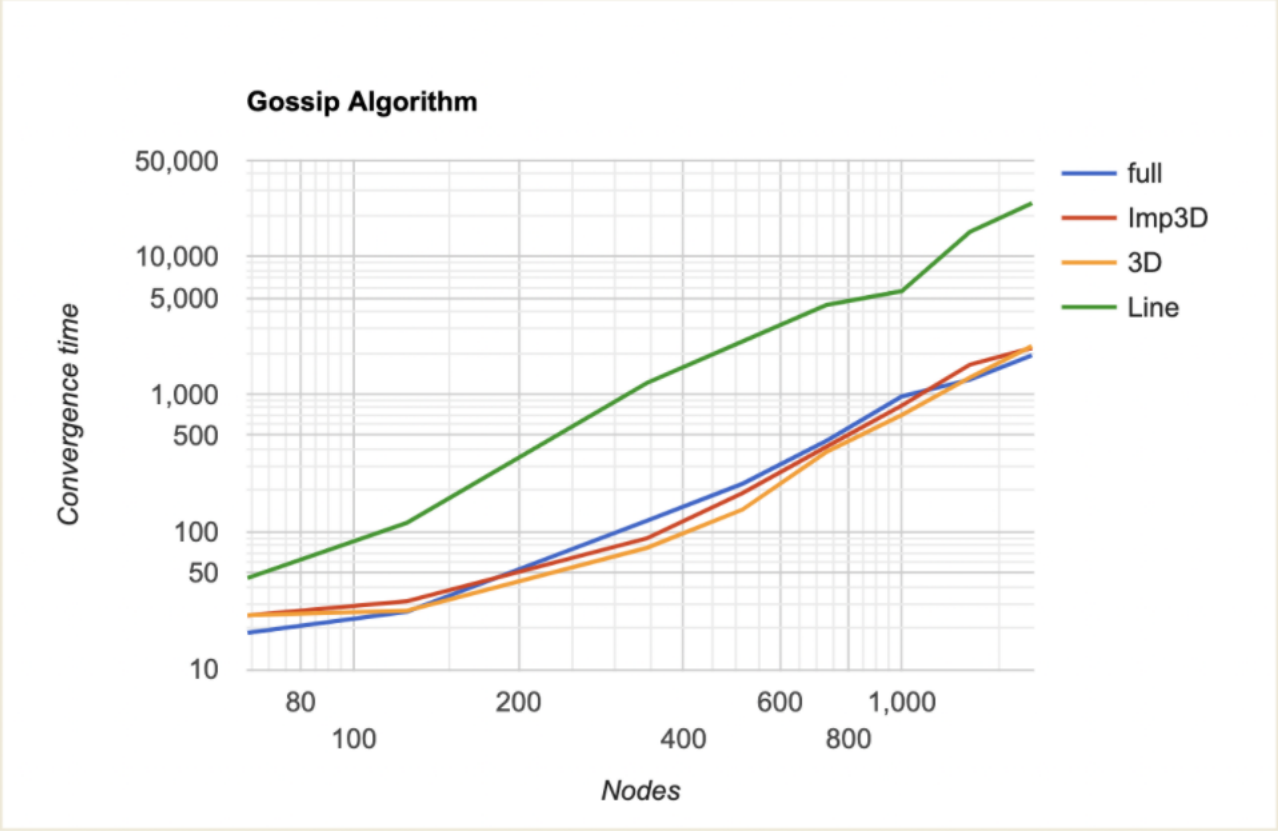
Largest network we managed to run code*: 1200 nodes*

```
kavyagopal@Kavyas-MacBook-Air DOSP-Project1 % dotnet fsi --langversion:preview p2.fsx 3D pus
Number of nodes: 1000
Push Sum Started
Time for convergence: 5658.276000 ms
All 1000 nodes converged to the sum!
kavyagopal@Kavyas-MacBook-Air DOSP-Project1 %
```

Graphs plotting convergence time vs size of the network for different topologies and algorithms

Gossip Algorithm

The graph above plots the convergence time vs the size of the network for all topologies using the gossip algorithm. We have taken nodes that are perfect cubes since it is a prerequisite for 3D and imperfect topologies. We are taking the range of nodes from between 64 and 1728 nodes and the range of convergence time between 0 and 25 seconds.



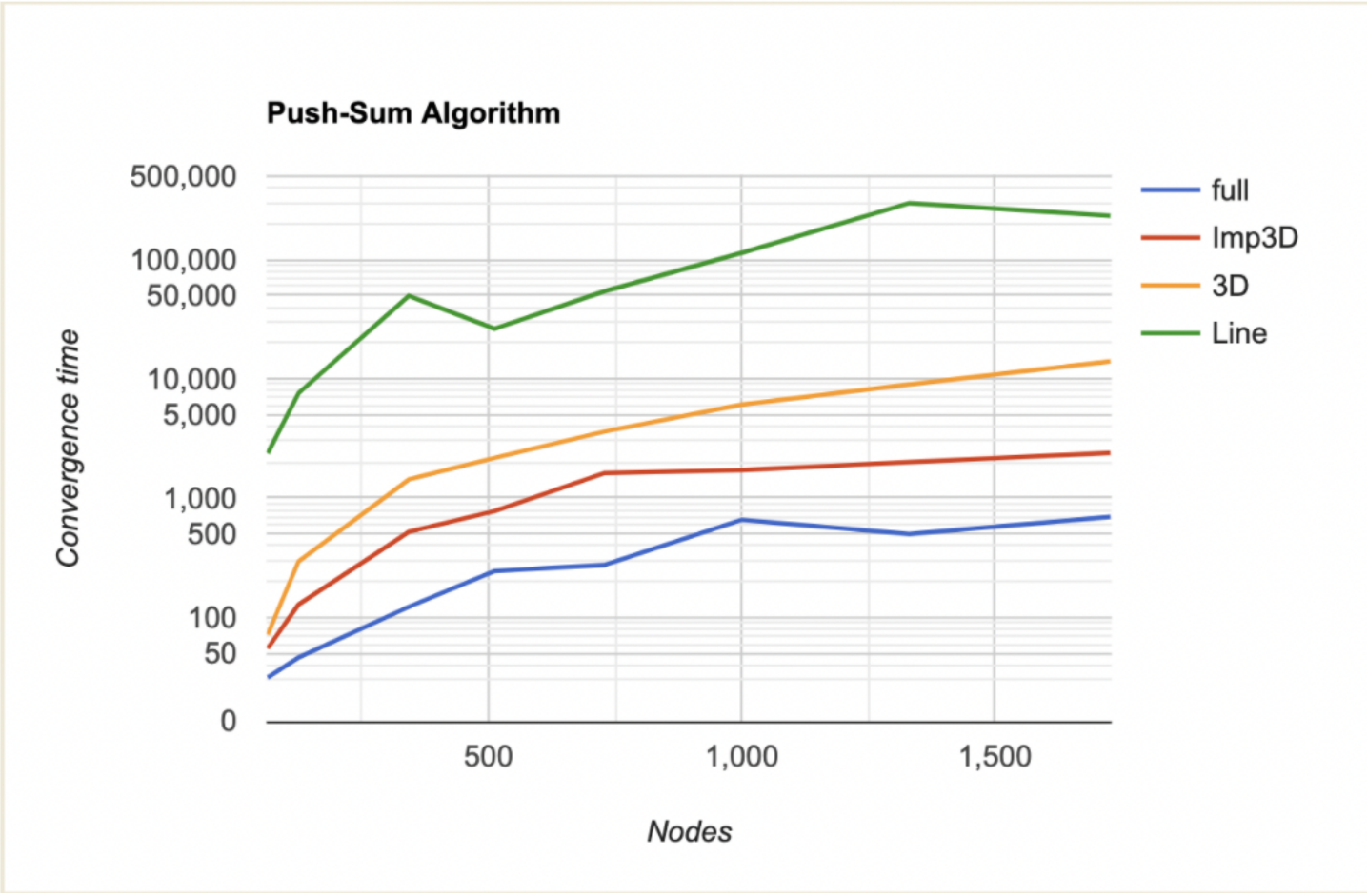
Convergence graphs of different topologies for gossip algorithm on a log scale for large network

Results

Gossip	full	Imp3D	3D	Line
64	14.488	19.721	23.472	35.032
125	22.134	23.7622	28.897	125.375
343	87.219	88.819	91.054	1615.111
512	190.893	181.741	275.839	2949.536
729	446.4632	421.856	365.234	4793.937
1000	1043.261	826.725	733.386	8980.691
1331	1147.435	1575.78	1349.678	15904.594
1728	1900.914	2300.387	2230.808	22297.439

Push-Sum Algorithm

The graph above plots the convergence time vs the size of the network for all topologies using the push-sum algorithm. We have taken nodes that are perfect cubes since it is a prerequisite for 3D and imperfect topologies. We are taking the range of nodes from between 64 and 1728 nodes and the range of convergence time between 0 and 15 seconds.



Results

Push-sum	full	Imp3D	3D	Line
64	31.37	54.978	71.828	2373.324
125	46.076	128.107	295.062	7587.092
343	122.882	522.362	1436.497	49565.317
512	244.979	778.1206	2175.575	26302.329
729	275.456	1623.929	3617.7071	54234.249
1000	659.545	1717.386	6095.785	113833.115
1331	499.962	2009.433	8966.628	297644.916
1728	697.869	2391.651	14050.893	233140.39

Graph Analysis

Full≤ Imperfect3D ≤ 3D ≤ 2D ≤ Line

1. As the number of nodes increases, the convergence time also increases for all the topologies for both the algorithm.

2. Line topology takes the most amount of time to converge for both algorithms. The convergence time increases exponentially with the number of nodes. This is also because each actor sends a message either forward or backward.

3. Full topology takes the least amount of time to converge. This is because every node is connected with every other node and each actor gets messages from multiple neighbors. As the size of the node increases, the size of the neighbor increases.

4. Full topology takes the most amount of time to be built. Although, we are measuring only the convergence time, we noticed that full topologies take more time to build as the number of nodes increases. T

5. he time of convergence for all topologies is lower in posh-sum when compared to gossip.

6. Imperfect 3D takes lesser time to converge compared to 3D since each actor has one extra neighbor. And the time difference is significant as seen in the results tables for both algorithms.

7. Both gossip and push-sum algorithms show similar characteristics for all topologies. The reason is that push-sum is an extension of the gossip algorithm, where the nodes should receive multiple messages for convergence.

4 Challenges

Issue: Initially, we were terminating the actor after it received a message 10 times(or till ratio for push-sum). However, we noticed that some actors were still alive and they weren't receiving any message and hence, the algorithm wasn't converging. After most of the nodes are terminated, there are no actors that are alive to send these actors messages. This situation was mostly seen in line and 3D topology.

Solution: We are handling this by ensuring the actor comes alive and sends messages to any neighbor that is still active. Only when all neighbor of an actor is terminated, the actor will terminate itself. This solves the issue where a neighbor doesn't get messages anymore when most of the actors

are terminated and it also acts as a condition to terminate the actor. When all the neighbors of an actor are terminated, it will not have a need to send messages anymore and can terminate itself.