COP5615: Distributed Operating System Principles, Fall 2018

Project 2

Submitted by:

Abhinav Pratik, 1019-4316, pratikabhinav@ufl.edu

Anubhav Jha, 4339-3979, ajha1@ufl.edu

- ## Problem Definition

Gossip Algorithm for information propagation

The Gossip algorithm involves the following:

1. Starting: A participant(actor) it told/sent a rumour(fact) by the main process
2. Step: Each actor selects a random neighbour and tells it the rumour
3. Termination: Each actor keeps track of rumours and how many times it has heard the rumour. It stops transmitting once it has heard the rumour 10 times (10 is arbitrary, you can play with other numbers or other stopping criteria).

Push-Sum algorithm for sum computation

1. 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
2. Starting: Ask one of the actors to start from the main process.
3. Receive: Messages sent and received are pairs of the form (s;w). Upon receive, an actor should add received pair to its own corresponding values. Upon receive, each actor selects a random neighbour and sends it a message.
4. 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.
5. Sum estimate: At any given moment of time, the sum estimate is s/w where s and w are the current values of an actor.
6. Termination: If an actors ratio s/w did not change more than (10 ^ -10) 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. As part of this project you have to experiment with various topologies. The topology determines who is considered a neighbour in the above algorithms.

1. Full Network: Every actor is a neighbour of all other actors. That is, every actor can talk directly to any other actor.
2. 3D Grid: Actors form a 3D grid. The actors can only talk to the grid neighbours.
3. Random 2D Grid: Actors are randomly position at x, y coordinates on a [0-1.0]X[0-1.0] square. Two actors are connected if they are within .1 distance to other actors.
4. Sphere: Actors are arranged in a sphere. That is, each actor has 4 neighbours (similar to the 2D grid) but both directions are closed to form circles.

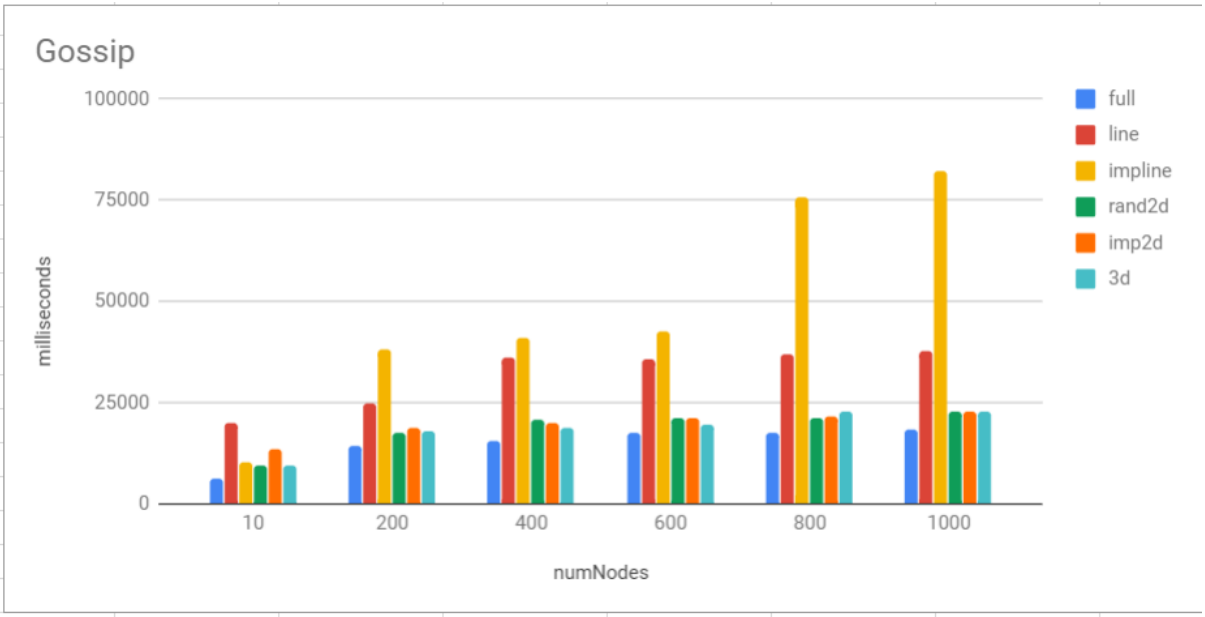
5. Line: Actors are arranged in a line. Each actor has only 2 neighbours (one left and one right, unless you are the first or last actor).
6. Imperfect Line: Line arrangement but one random other neighbour is selected from the list of all actors.

- ## Program Structure

The whole program is contained in 5 files which are placed under Project2/lib/

1. main.ex – is the entry point of the whole program.
2. gossipImplementation.ex – holds the implementation for the gossip algorithm
3. pushsumImplementation.ex – holds the implemention for the push-sum algorithm
4. functions.ex – contains helper functions that were used in the project.
5. topologies.ex – has a list of network topologies that are being used in the project.

- ## Graph for Gossip Algorithm



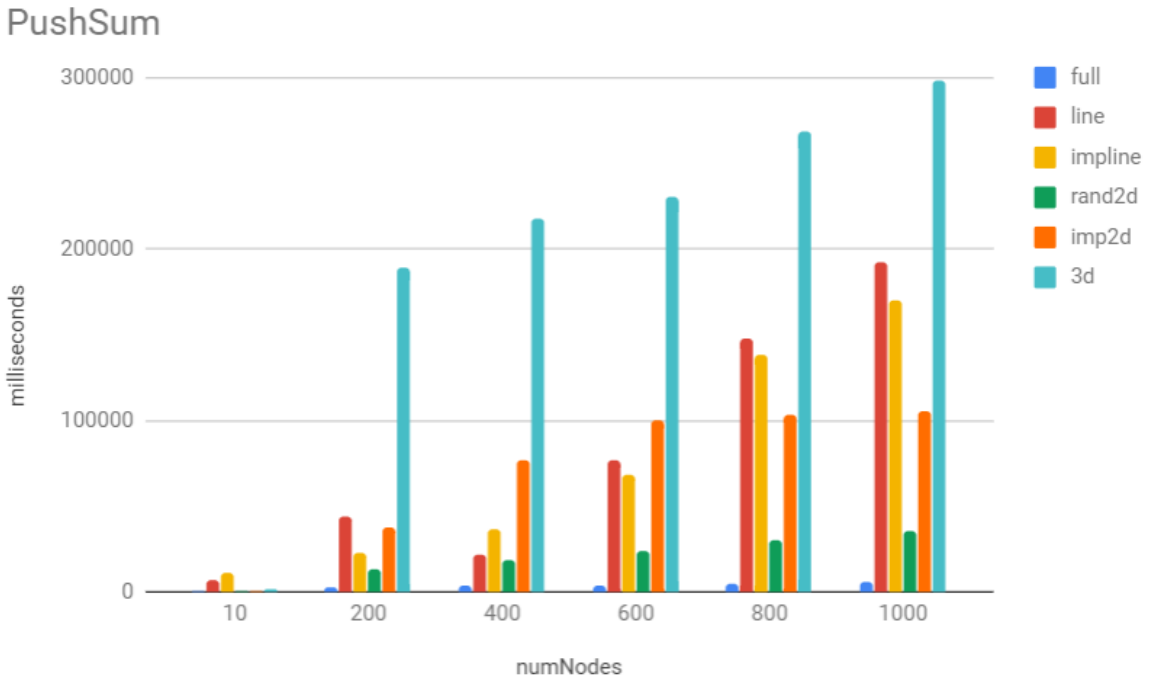
The y-axis denotes time taken in milliseconds for the algorithm to converge and the x axis denotes the number of nodes the algorithm was executed for.

Full < (Rand2D, 3D, Imp2D) < Line < ImpLine

Interesting findings:

1. 3D, Rand2D and Imp2D have running times that are very close to each other. And hence it is hard to distinguish between them.
2. For the same network size and topology, multiple runs of Gossip yields different convergence time.
3. Overall, Full network always performed the best yielding lowest convergence time, whereas the worst convergence time was a Line network (either Line or ImpLine) depending on the network size.

- Graph for PushSum



The y-axis denotes time taken in milliseconds for the algorithm to converge and the x axis denotes the number of nodes the algorithm was executed for.

Full < Rand2D < Imp2D < ImpLine < Line < 3D

Interesting findings:

1. As the number of nodes were increased, a clear distinction was observed between the times of the various networks.
2. Full network was always the fastest and 3D was always the slowest to give the output.
3. For the same network size and topology, multiple runs of push-sum yields different convergence time.
4. Sometimes, the push-sum algorithm had to be re-run multiple times to get an output due to node failure.