COP 5615 Distributed Operating System Principles
Project 2
Date - 10/07/2014

TEAM MEMBERS –

Mugdha Mugdha (54219168)
Palak Shah (55510961)

CODING ENVIRONMENT –

Scala Version – 2.11.2
Akka Version – 2.3.6
SBT version -0.13.6

INSTRUCTIONS TO COMPILE AND RUN –

We assume that the scala environment and sbt are set up on the machine on which this code will be tested.

1) To run Project 2, go to the project folder and enter the sbt console

   $ cd Project2_54219168_55510961
   $ cd Project2
   $ sbt
   - publishLocal
   - runMain gossip.project2 <number of nodes> <topology> <algorithm>

2) To run Project 2_Bonus , go to the project folder and enter the sbt console

   $ cd Project2_54219168_55510961
   $ cd Project2_Bonus
   $ sbt
   - publishLocal
   - runMain gossip.project2_bunus <number of nodes> <topology> <algorithm>

(If Project2_Bonus gives binding error, it is because of a hidden copy of application.conf might be there in its resources folder. This error can be removed by replacing the application.conf of Project2_Bonus with the application.conf of Project2)

<number of nodes> is the total number of nodes in the system
<topology> can be any one of the following – “line”, “full”, “2D” or “imp2D”
<algorithm> can be “gossip” or “push-sum”

INTRODUCTION –

In this project, we present an implementation of Gossip algorithm on various topologies. We also calculate aggregate using the push-sum algorithm discussed in class. For this project, we have implemented four different types of topologies – Line Topology, 2D Grid topology, Imperfect 2D Grid – which is like a 2D Grid with one random neighbor - and Fully Connected topology. For the bonus part, we have handled the case where random nodes in the system fail one by one.
In this report, we present our observations about the performance of both these algorithms on all 4 topologies. We also present our observations on the effect of node failures and failure handling algorithm for all the topologies.

We have created two separate sbt projects -

1) In Project2, we have implemented gossip and push-sum algorithms on all the four topologies, viz line, 2D, imperfect2D and full. The Gossip algorithm converges when all the nodes get the message at least once. The push-sum algorithm converges when one of the nodes gets a nearly constant value of aggregate
2) In Project2_Bonus, we have handled the case where random nodes fail one after the other. In this case, the Gossip algorithm converges when all the living nodes get the message at least once. The push-sum algorithm converges when one of the nodes gets a nearly constant value of aggregate

CODE STRUCTURE –

Both the projects are made up of 3 scala classes –

1) project2 /project2_bonus – This scala object contains the main class that creates the Actor System and starts all the nodes in the system.
2) nodeActor – This class represents the nodes in the topology. In this class, we create a topology, and implement gossip and push-sum algorithms over that topology.
3) bossActor – This class was created so we can handle node failures and terminate gracefully. This class also measures the execution time of the algorithm.

Please see comments in the code for a detailed description of what each class does.

BUILDING THE TOPOLOGY –

For every node, we created a list that holds name of its neighbouring actors.

For a line topology, every actor has only two neighbours. For 2D grid, a node has at most four neighbours. For imperfect 2D grid, the node has atmost four neighbours in the gris and one random neighbor. For Full topology, every node is a neighbor of every other node.

The neighbor list is initialized only at the beginning. Once it is ready, the node selects a random neighbor from the list to send a message.

ROUNDS –

In Gossip algorithm, a Round is defined as periodic execution of certain part of the code. In every round, every node that has received a message atleast once, will send the message to one random neighbour. I our code, we have used the akka scheduler function to start rounds at regular intervals. The interval is set to 1 second, so that we can observe the convergence clearly. In Gossip-based algorithms, convergence depends on the number of rounds. So, execution time will be approximately (1 second) * (number of rounds)

CONVERGENCE –

We have to decide when to stop the gossip-based algorithms in order to prevent redundancy and network overload. In this project we have specified definite rules about when to stop each of the two algorithms. We say that the algorithm has converged when it finishes the task it was supposed to do.

GOSSIP –

The Gossip algorithm is used to convey a single message to all the nodes in a topology. It converges when all the nodes get the message at least once. To avoid overloading of nodes, we make sure that if a node receives the message 10 times, it stops transmitting.

PUSH-SUM –

The push-sum algorithm is used to calculate the average of values of each node. It converges when the average calculated by any one node does not change by 10^-10. This value is reported as the average of all values of nodes.

GRAPHS AND ANALYSIS –

For this project, we have made graphs showing the performance of both algorithms running on each topology. Since, execution time depends on number of rounds, we have made graphs of number of nodes vs number of rounds.
All the values obtained are stored in the graphdata.docx sheet attached along with the project.

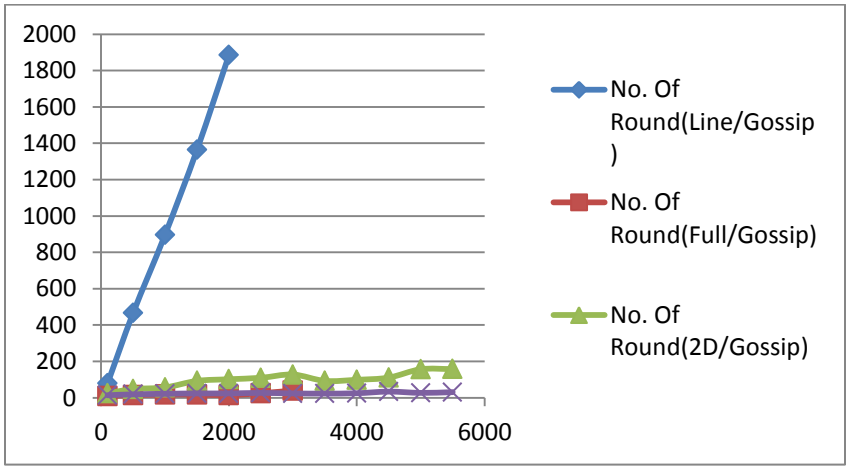

Figure 1 number of nodes vs number of rounds for Gossip

The above graph compares the performance of all four topologies for gossip algorithm.

- For all the graphs, number of rounds increase with increase in the number of nodes. But this increase is linear only for “line” topology. For other topologies, we see that for large increase in number of nodes, the increase in number of rounds is small. This leads us to believe that the relationship between number of nodes and number of rounds may be logarithmic.

- It is clear that “line” topology takes longer to converge than “2D Grid”, “full” and “imperfect 2D Grid” topologies. We already expected “full” topology to perform better than the others. But the performance of “2D grid” “imperfect 2d Grid” topology comes as a surprise. The reason for this could be that ”2D Grid” topology has a stable structure with more neighbors and “imperfect 2d Grid” topology has a perfect balance of randomness and structural stability.

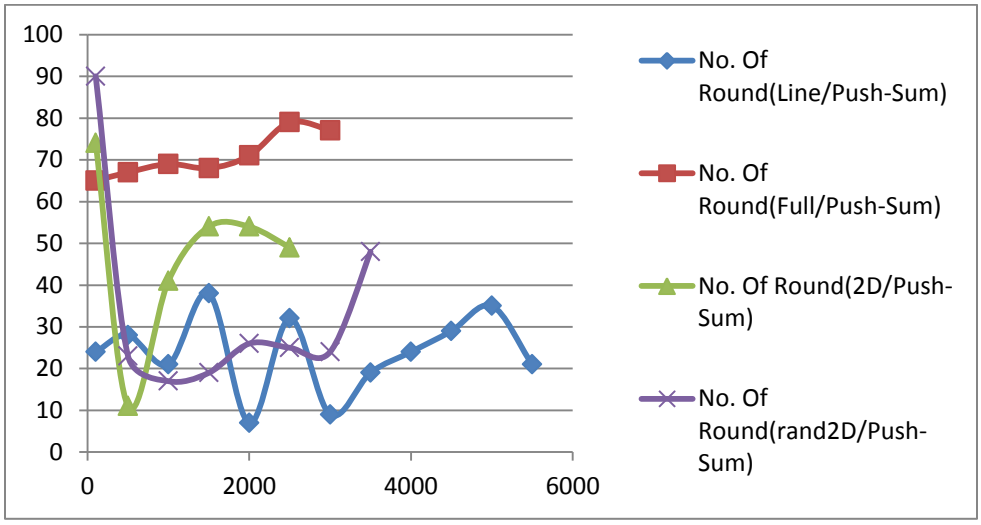

Figure 2 number of nodes vs number of rounds for push-sum

The above graph compares the performance of all four topologies for push-sum algorithm.

- We can see that the convergence time/number of rounds for “full” and “imperfect 2D Grid” topologies is almost constant irrespective of the number of nodes in the system. This can be attributed to the randomness in both these topologies. The more rigid “line” and “2D grid” topologies show highly unpredictable performance results.
- Unlike gossip algorithm, the execution time of push-sum algorithm does not increase with increase in the number of nodes. This can be explained by the fact, that in order to get the aggregate, we need only one node to converge. In gossip, the algorithm converges when all the nodes have received the message.

CONCLUSION –

We conclude this report with the following points –

- Execution time of gossip-based algorithms depends on the number of rounds.
- Topologies with random neighbours converge faster than rigid topologies.
- In gossip algorithm, execution time/ number of rounds increases logarithmically with increase in number of nodes.
- In push-sum, the execution time is constant irrespective of the number of nodes.