Distributed Computing: Spring 2024
Programming Assignment 1: Implementing Vector Clocks
Submission Date: 16th February 2024, 9:00 pm

Goal: The goal of this assignment is to implement Vector-clocks and Singhal-Kshemkalyani optimization on a Distributed System. Implement both these algorithms for vector clocks in C++ using ZeroMQ / MPI. Then, you have to compare the overheads incurred with messages stored and exchanged.

Details. Suppose a system of n nodes (processes) connected to each other in the form of graph topology is given as input. These nodes communicate with their neighbors (in the graph) through messages. You can implement these nodes as processes/threads in C++. For the the communication between the threads you can use ZeroMQ/MPI in C++. The details of MPI can be seen here: https://www.mpi-forum.org/ while ZeroMQ can be seen here: https://zeromq.org/languages/cplusplus/.

As studied in the class, each process executes three events: internal, message send, and message received. To simulate these events, you can assume the following: each process creates and executes internal and message send events with a delay that is exponentially distributed with inter-event time λ ms. Assume that the ratio of internal to message send events on each process is α (which, for instance, can be 1.5). When a process p_i has to send a message, it randomly chooses a neighbor, say p_j and then sends p_j a message. The application terminates when each process sent a total of m messages.

In this setting, implement the normal Vector-clocks and Singhal-Kshemkalyani's optimization of Vector-clocks. Then demonstrate the savings in message communication obtained by using the optimization.

Input: The input to the program will be a file, named **inp-params.txt**, consisting of all the parameters described above and the graph topology. The first line of the input file will contain the space separated four parameters as: $n\ \lambda\ \alpha\ m$. For example, $n = 15, \lambda = 5, \alpha = 1.5, m = 40$
The second line onwards will contain the graph topology in the form of an adjacency list. For instance, if n is 3, a sample topology showing a complete graph is as follows:
1 2 3
2 1 3
3 1 2
The example input file for $n = 3$ looks like this:

```
3 5 1.5 40
1 2 3
2 1 3
3 1 2
```

Output: Your program output should demonstrate the strong consistency property of vector-clocks: If two events x and y have timestamps vh and vk, respectively, then

$$x \rightarrow y \Leftrightarrow vh < vk \tag{1}$$

$$x \parallel y \Leftrightarrow vh \parallel vk \tag{2}$$

To demonstrate this, you have to output the contents of all the events onto a common logfile with two time-stamps: real time-stamps and vector time-stamps. Since you will be executing all these programs on the same machine or in a LAN environment like IITH, you can assume that the clocks of the machines are synchronized.

The contents of the log file should be as follows for both the algorithms:

Process1 executes internal event e11 at 10:00, vc: [1 0 0 0]
Process2 executes internal event e21 at 10:01, vc: [0 1 0 0]
Process3 sends message m31 to process2 at 10:02, vc: [0 0 1 0]
Process3 executes internal event e32 at 10:03, vc: [0 0 2 0]
Process2 executes internal event e22 at 10:04, vc: [0 2 0 0]
Process2 receives m31 from process3 at 10:05, vc: [0 3 1 0]

.
.
.

The output should demonstrate Eqn(1) and Eqn(2). **In addition to this, you have to output space utilized by each process for storing the vector clocks and sending messages by both the algorithms.**

Report: You have to submit a report for this assignment. As mentioned earlier, this report should contain a comparison of the performance of vector-clocks and Singhal-Kshemkalyani optimization. You must run both these algorithms multiple times to compare the performances and display the result in the form of a graph.

You run both these algorithms varying the number of processes from 10 to 15 in the increments of 1 while keeping other parameters the same. You can assume any topology for creating the graph with each of these processes. Assume that m is 50 for all the executions.

You measure the average number of vector-clock entries sent in each message. Clearly, this number is going to be fixed for the normal vector-clock. But for Singhal-Kshemkalyani optimization, this number should be lesser.

The graph in the report will be as follows: the x-axis will vary the number of threads from 10 to 15 in increments of 1 (as explained above), while the y-axis will show the average number of entries in each message sent. Finally, you must also give an analysis of the results while explaining any anomalies observed.

Deliverables: You have to submit the following:

- The source files containing the actual program to execute. The normal vector clock implementation should be named as VC-<rollno>.cpp and Singhal Kshemkalyani's optimization as SK-<rollno>.cpp.
- A readme.txt that explains how to execute the program.
- The report as explained above.
- A inp-params.txt as input file.

Zip all the files and name them as ProgAssn1-<rollno>.zip. Please follow the naming convention strictly. Otherwise, your submission will not be evaluated. Then upload the zip on the google classroom page of this course. Submit it by the above mentioned deadline.

Note: Please follow all the instructions given for the assignment strictly (input-file format, src file names, output format, reports, etc.); otherwise, the TA may not evaluate your assignment.

Evaluation Criteria: The following will be the criteria for evaluation:

1. Design: The program design along with the report as described above will carry 50 marks.
2. Execution: The program execution will carry 40 marks.
3. Code Indentation and Documentation: The indentation and documentation of the code will carry 10 marks.

Late Submission Penalty: For each day after the deadline, your submission will be penalized by 10 marks until a maximum of 6 days beyond which your submission will not be considered.

Plagiarism Policy: If we find a case of plagiarism in your assignment (i.e. copying of code from each other, in part or whole), you will be awarded zero marks. Note that we will not distinguish between a person who has copied, or has allowed his/her code to be copied; both will be equally awarded zero marks for the submission. Follow below link for more information about plagiarism policy: https://cse.iith.ac.in/academics/plagiarism-policy.html