Operating Systems Laboratory (CS39002)

Spring Semester 2021-2022

Assignment 3: Hands-on experience of using shared-memory

Assignment given on: February 08, 2022

Assignment deadline: February 14, 2022, 11:55 pm

In this assignment you will learn how to use shared memory to communicate between processes.

There will be 1 producer process updating a Facebook friend circle network and 10 consumer (worker) processes calculating shortest paths between people.

Main process - Loads a graph into a *shared memory*. Note that the graph is dynamic - new nodes and new edges will be added later (described below). It then spawns the producer and consumer processes. The graph to be loaded is available at this link. *(facebook_combined)*. For simplicity, assume all edges of the graph to have weight 1.

Producer process - wakes up every 50 seconds, and updates the graph in the following way: A number m in the range [10, 30] is chosen randomly, and m new nodes are added to the graph. For each newly added node, select a number k in the range [1, 20] at random; the new node will connect to k existing nodes. The probability of a new node connecting to an existing node of degree d is proportional to d (in other words, a popular node is more likely to get connections from new nodes). Assume all newly added edges to have weight 1.

Consumer process - These processes wake up once every 30 seconds, and count the current number of nodes in the graph (which is in shared memory). Each consumer process gets mapped to a designated set of nodes. This "mapped set" contains 1/10 of all nodes in the graph. There should be a mapping of the first 1/10 nodes to the consumer-1, the second 1/10 nodes to consumer-2, and so on.

Then each consumer process runs the Djkstra's shortest path algorithms considering all nodes in its mapped set as the source nodes.

A consumer process runs Djkstra's algorithm as stated above and writes (appends) the set of edges depicting the shortest path to each reachable node from the source node (along with the source node) to a file; then it goes to sleep, to wake up 30 seconds later and repeats the process. Note that there should be 10 such files, one per consumer process. Each consumer thread should know, by some mechanism, its mapped set of nodes, and which file it should append outputs to.

Optimization - Note that the producer will add only a limited number of nodes and/or edges to the (much larger) network, which might not change many shortest paths. Design and implement a strategy to optimize the process of re-computing shortest paths. Write a report with the design strategy. This version will run with a "-optimize" flag to your code.

Note: there may arise a situation where the producer updates the graph in the midst of computation of a consumer. Such situations are called 'race conditions'; ignore them for now; in the next assignment, we will learn how to guard against such race conditions.

Submission Guideline:

- Create three programs (in C or CPP), one for the producer, one for the consumers, and one main program which sets up the scenario as described. Also create a Makefile that will compile all programs.
- Include a short report as asked, as a pdf file.
- Zip all the files into Ass3_<groupno>_<roll no. 1>_< roll no. 2>_< roll no. 3>_< roll no. 4>.zip (replace <groupno> and <roll no.> by your group number and roll numbers), and upload it on Moodle.
- You must show the running version of the program(s) to your assigned TA during the lab hours.
- **You cannot use any graph library in the assignment.**
- **[IMPORTANT] Please make only one submission per group.**

Evaluation Guidelines:

Total marks for this assignment: 50.

We will check features of your code (including but not limited to) if the computation is correct, you are creating and using the shared memory correctly, the correct number of processes are created, if the jobs are correctly created, and the consumers are executing those jobs correctly.

Items	**Marks**
Correctness of the main process (loading the initial graph)	5
Correctness of producer (and graph updation)	10
Correctness of consumer (and shortest path computations)	15
Correctness of the setup, shared memory usage	10
Optimization	10
Total	**50**