Pelinsu Saraç - 28820

CS 403 PA-1 REPORT

ConSet.py:

This file includes the concurrent set implementation as a class named ConSet. The class has 3 properties: a dictionary to track the state of the set and two semaphores, one used as a mutex, and one used directly as a semaphore.

Other than the properties, the class also has 3 methods: insert, pop and printSet. Insert method accepts an object to insert into the set and pop methods returns an element from the set in the way described in the assignment document. PrintSet basically prints the dictionary that represents the current state of the set.

Synchronization was applied with the previously mentioned mutex and semaphore. The mutex was used to wrap the critical sections of the operations, i.e., when the methods were using the dictionary, to prevent data race. Semaphore was associated with the number of existing elements in the set. Semaphore value is increased after the insertion is completed, as a new element was inserted. Before the pop operation takes place, the pop method tries to decrease the Semaphore value. If it is above zero, the pop method can decrease the value by one and continues with popping. However, if the value is zero, meaning that there are no elements in the set of current state, then the thread using this method is forced to sleep until a new element is inserted and Semaphore value is increased. PrintSet method does not use Semaphore as it does not manipulate the number of elements.

Leader.py:

This file includes a global variable called n to represent the number of nodes, a list of n ConSet instances to represent the mailboxes of the nodes, a barrier and a mutex. Mutex is used to ensure the order of statement printings, as print function in Python is not thread-safe.

Other than the variables, there is also the nodeWork function that implements the leader election algorithm and a main function to manage the threads related to nodes.

nodeWork function starts with coming up with a random number between 0 and n^2. After that, the thread that is running on this function sends this number to mailboxes along with its ID. Finally, in the function, a thread pops the elements in its mailbox one by one and checks the numbers inside to find the maximum. Number of pops will be exactly n, as there are n threads/nodes. Even if the thread starts popping elements before other threads can send their messages, due to the implementation of pop method, thread will wait for the others to insert.

While going through messages, a variable called max_repeating_num is used to track whether the current leader is repeating or not. There are two cases: the leader is unique, or it is not. In the case that the leader is unique, eventually, the real leader would be different from an old, repeating leader and the thread can stop. If not, then in the end, variable keeping the leader number would be same with this variable, and we can understand that the leader is not unique, thus thread would continue with next round.

There is a barrier at the end of while loop of the nodeWork function. While loop is to ensure that the threads continue if they cannot find a leader. The barrier is to ensure that the threads/nodes will continue with the next round, if necessary, at the same time.