Comp9154 23T2

Z5211414

Jin-Ao Olson Zhang

Assignment 1 report

This assignment aim to implement a data structure that maintains a large set of non-negative numbers but need to considerate some points

1. a number is either a member of the set or it is not a member - there is no notion of duplicated copies of the same number
2. require a solution that works for an arbitrary number of processes
3. The amount of memory for the data structure is bounded, so that it can contain at most N elements. Any process attempting to insert a new number when there are already N members should be blocked until deletions reduce the number of elements to be less than N.
4. allow multiple processes to be performing the member operation at the same time

First of all , I choose to use a

1. For performance reasons, we want the calls to member, delete and print sorted to be as efficient as possible. Assume that the insert calls are expected to be less frequent, and that it is acceptable for these operations to be slower (run in time O(N)).
2. implement the data structure using an array that is maintained in sorted order, and to implement the member operation using a binary search algorithm.
3. if there are many deleted entries, binary search will operate less efficiently, and space is being “wasted”. Therefore, an insert operation should contribute to cleaning up the deleted entries and “compactify” the region of the array that it modifies, so that it forms a contiguous block of actual members of the set.

To complete all the requirement and solved the question, I need to start with the basic logic of how concurrent works in my code,since I should allow multiple processes at the same time,for example, conflicts will happen when different thread access “insert” and “delete” or “print\_sorted” at the same time,machine can not decide the new data provided to “print\_sorted” thread that contains the deleted data or not at that moment, which will cause a conflict.

I provided a read-write lock in my code to manage the access to the data structure and used to prevent multiple access conflicts. In detailed,write lock use to ensure only one thread can access “insert”,“delete”,“cleanup” operations at a time, and read lock is used to ensure multiple thread can access “member” “print\_sorted” operations.

**The benefit of use read-write lock is the data structure is always in a consistent state and the operations always return correct results. The liveness property guaranteed by this solution is that every operation eventually completes. 安全性，需要补充或者重写**

**Java Code**

The Java concurrency that I used for implement the specification are the “ReentrantReadWriteLock” class, this class have the “readLock” and “writeLock” methods that can ensure multiple access and single access of threads to solved concurrent problem。

“readLock” method acquires the read lock if the write lock is not held by another thread, and otherwise blocks until the write lock is released.

“writeLock” method acquires the write lock if neither the read nor write lock is held by another thread, and otherwise blocks until both locks are released.

For the complexity,

**insert(x : int):** Find the space that use to contain the new inserted element takes O(N)

**delete(x : int):** use of binary search to find the element to delete, takes O(log(N)).

**member(x : int) :** use of binary search to find the element , takes O(log(N)).

**print sorted():**  by iterate the array print out elements that are not marked as deleted,take O(N).

**cleanup():** delete elements in array that are not marked as deleted by iterate the array,take O(N).

**Promela Code:**