Concurrent Data Structures
Lab Assignment 1

Parosh Aziz Abdulla
Sarbojit Das

November 25, 2021

Deadline: 2021–12–07.

Please, submit your programs and instructions to run them.

Task 1 Fine-grained synchronization Implement a concurrent list-based set with Fine-Grained synchronization in a file named "FineList.cpp".

Task 2 Optimistic synchronization Implement a concurrent list-based set with Optimistic synchronization in a file named "OptimisticList.cpp".

Task 3 Experiment Performs the following experiments:

- Create n worker threads, where $n = 2, 4, 8, , 16, 32, 64$ threads
- For each n, let each worker thread perform various operations `add`(), `rmv`(), and `ctn`(). The type of the operations are selected randomly, and the input values are selected randomly from the set $\{0, 1, \ldots, 7\}$. Measure the throughput of the system in 10 seconds, i.e., the total number of operations completed by the threads in 10 seconds. Perform the experiments for all three algorithms (including the coarse-grained algorithm from the first lab).
- Repeat the previous experiments when the the input values are selected randomly from the set $\{0, 1, \ldots, 1023\}$.
- Repeat all the above sets of experiments, now increasing the percentage of the `add()` operations. For each value $i = 10, 50$, and 90, conduct the above experiments such that i% of the operations are `cnt()`. From the remaining set, 90% should be `add`, and 10% should `rmv()`. For instance, for $i = 60$, we have 60% `ctn()`, 36% `add()`, and 4% `rmv()`.

Depict the results in tables or (preferably) curves, where the x-axis is the number of threads, and the y-axis is the throughput.

Task 4 Multiset A multiset is a generalization of a set where a given element may occur multiple times in the multiset. An example of a multiset (over the set of integers) is [2; 2; 2; 3; 7; 7]. Here, the number of occurrences of 2, 3, and 7 is 3, 1, and 7, respectively. A multiset can also be viewed as a special case of a stack or a queue, where the order of the elements is not relevant. For this assignment you are tasked to implement a concurrent multiset that supports the following functions:

- `add(x)`: Adds element x to the set and returns true
- `rmv(x)`: If possible, removes (an instance of) element x from the set and returns true, otherwise false
- `cnt(x)`: Returns the multiplicity of element x(how many instances of x there are in the multiset)

Implement a concurrent multiset using Fine-Grained synchronization. Create 16 worker threads, each does 100 operations on multiset. Mix various operations `add()`, `rmv()`, and `cnt()`. Identify linearization policy and check if it is consistent with non-concurrent multiset ADT. You can create a monitor thread exactly like the first assignment and communicate. Create a shared sequence of operations. Whenever a worker does some operation, put it in the shared sequence at the linearization point. The monitor reads the sequence and do the operations on non-concurrent multiset. Here you can find c++ multiset: std::multiset.