**Implementation (Java)**

Two variations of linked list are implemented.

1. *Linked List with Lazy Synchronization* – only locks on the appropriate window when found (as opposed to hand-over-hand method). A marked bit is added to the node as to not impede search progression.
2. *Skip List with Lazy Synchronization* – same as (1) with added functionality. Node levels are added to help skip sections of the list in order to speed up lookup time.

**Verification of correctness**

1 & 2:

* Node values are strictly increasing (i.e. not non-decreasing, so no duplicate values exist)
* numSuccessfulAdds – numSuccessfulRemoves = lengthOfList

2:

* Node values in the list at each level are strictly increasing
* Leveli+1 ⊆ Leveli

**Analysis (Java)**

System throughput is measured in number of operations performed per millisecond (ops/ms). Threads are varied incrementally from 1 to 32 by powers of 2. Total operations per thread is 100,000. Search/Add/Remove distributions are: 90/9/1 (90LX), 70/20/10 (70LX), 0/50/50 (0LX). Tests are executed on a TACC machines (16-core system). Below are the results.

From the above graph, we can infer that linked list and skip list performs similarly for key space = 100 where there are more search operations but the skip list’s performance decreases after number of threads = 16. As the number of modify operations increases with decrease in search operations, the curve of both linked list and skip list increases but are lower than the extensive search operations; again, for the skip list the performance decreases after 16 threads. For extensive modify operations (add and remove) the linked list is almost linear and the skip list increases up to 8 threads and then decreases. The reason being that the window of operation is not uniformly distributed but has contention due to over lapping windows as a result of smaller key space.

The linked list’s performance is lower than the skip list for all the cases and the curves keeps increasing with the number of threads. Whereas, the skip lists have a better performance for all scenarios but decreases after 16 thread operations. As the key space is larger now (1000) the window for operations are spaced out and can perform more operations with less overlapping windows.

From the above graph we notice that the curve for skip list have a better performance compared to the linked list which are almost linear (small increment). The skip list now has a larger key space of 10,000 and have fewer overlapping windows compared to key space of 1000. The curve for skip list increases up to 16 threads and then decreases for 32 threads due to the load on the cores.

Comparing the performance of linked-list to skip-list in all cases shows skip-list performing better. In theory, skip-list should perform better on larger key spaces, which can be observed from the above graphs.

**Future Work**

* Tweak the number of levels in skip-list to see if there is significant performance impact.
* Find a more efficient locking mechanism for skip-list.