Aryan Soman
Jerry Xu
15418 Project Milestone Report
URL: https://github.com/aryansoman/lock-free-AVL-tree

Revised Project Schedule

Fri Dec 2: Add lock-full rebalance tests. (Aryan) Finish rebalance function for lock-full. (Jerry)

Mon Dec 5: Finish search, insert, delete (Jerry) for lock-free + test cases (Aryan).

Fri Dec 9: Write rebalance function + tests for imbalance reduction for lock-free. (Aryan) Write performance tests for the trees. (Jerry)

Mon Dec 12: Run performance tests on trees. (Aryan + Jerry)

Fri Dec 16: Write report. (Aryan + Jerry)

Sun Dec 18: Finish poster. (Aryan + Jerry)

Work Completed So Far

We have implemented insert, delete, and search functionality in a fine-grained lock-full AVL tree, as well as a sequential and concurrent test case for these functions. We have also implemented a method that performs rebalancing on a particular node, but this function is not tested, and also we still need to implement a function that searches for the appropriate nodes to rebalance on. We have no preliminary performance results at this time.

Comparison to Original Goals and Deliverables

We are over a week behind our original optimistic schedule. The literature review took a lot longer than expected as we didn't expect the AVL rebalancing to have to take place separately from the modify operations. Because of this, the operations were also a lot more complicated to implement than expected.

We will still be able to implement our "plan to have" deliverables from the original schedule, but unfortunately we may not have time for any of the "nice to haves", such as testing different types of BST. Another "nice to have" that would be a smaller commitment than this is a coarse-grained locking BST that simply locks the whole structure for each operation, which we could use for a performance reference.

Poster Content

We plan to present graphs of throughput for each implementation when run at various core counts and on various types of workload with regards to the proportion of insert/delete/search operations. We also want to vary the number of rebalancing processes vs the number of modification processes at each processor count we test with.