HPC Software
Assignment 3

Robert Cronin 12313002

February 27, 2017

1 Binary Tree

For this assignment I chose to use an AVL balancing binary tree, whereby in order to be balanced, the heights of the children at each node may differ by at most one[1]. The serial version of this is implemented in `serial.c` and runs through a loop, of default length 1000, adding and deleting random integers between 0 and 999, followed by a rebalance.

2 Using pthreads

To implement pthreads I initially tried just using one lock and only using it when a change needed to be made to the tree. However, this ran into problems whereby when searching the tree it sometimes went down a path into a node that had just been moved or deleted and so eventually returned a segmentation fault.

To overcome this I would need to lock even when searching which would then mean every action needs a lock and is equivalent to the serial code. Instead I created a lock for each node. This meant when searching I would lock the root, find which child I was looking for, lock this child and then unlock the parent and so on. This ensured that the child node would still exist in its position when I searched for it.

When adding, deleting and rebalancing I would ensure to always lock downwards so that if the parent is locked there is no way the child could be moved or deleted by another thread (it could have its child pointers altered but this is not an issue).

[1]https://en.wikipedia.org/wiki/AVL_tree

I also introduced one overall lock called `root_lock` that I would use when looking for the current root node. Anytime I began a search I would lock this to ensure I get access to the node currently at the top of the tree and then lock/unlock down the tree from there[2].

The pthreads version is implemented in `pthreads.c`. It attempts 1000 adds by default, while simulataneously deleting and rebalancing. There is a poisson distributed time gap between each add and delete and a larger gap between each rebalance.

3 Timing

The serial code runs faster than the pthreads code by default but this is only due to the poisson delays in the pthreads code. If these delays are removed the pthreads code runs significantly faster. Although some of these gains are due to running a lot of rebalances/deletes while the tree is relatively empty, there is still a speed up present.

[2]Idea taken from: http://swapnil-pimpale.github.io/lock-free-BST/