Concurrent AVL Tree Operations on GPUs

Abhishek Yadav
(abhi92914@gmail.com)

IIT Madras

Abstract

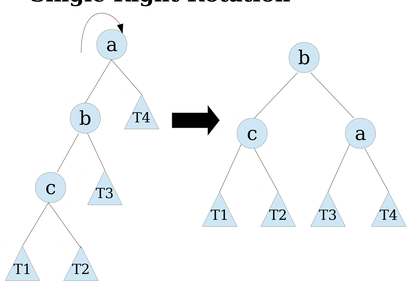
- Self-Balancing Binary Search trees support O(h) time search, min, insert, delete, etc. operations where **h** is the height of the tree
- AVL Trees are used to implement interfaces such as map, set etc, which are used in applications that require frequent lookups and updates concurrently.
- General Purpose GPUs consist of thousands of cores optimized for parallel processing hence preferable for concurrent operations over multi-core CPUs
- Goal is to implement concurrent search and update operations on AVL trees to run on GPGPUs

Introduction

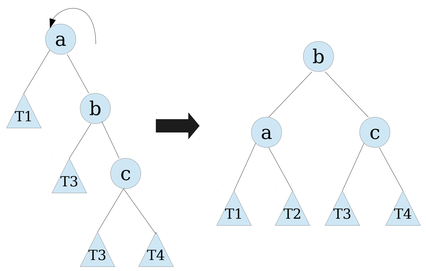
- Each node of the AVL tree satisfies **height-invariant** or **balance factor** property, i.e. difference between the heights of the left and right subtrees can be at most 1.
- Search and update operations are performed the same way as in standard Binary Search Trees except update operations require *rotation* operations to balance the tree

Update operations traverse up to the root from the place of update and perform one of the rotations wherever required

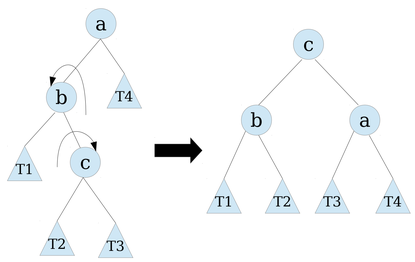
Single Right Rotation



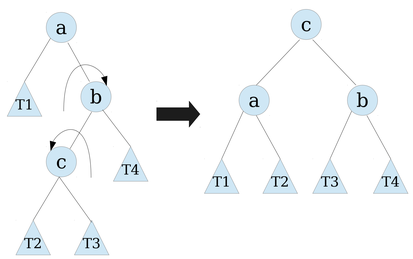
Single Left Rotation



Left-Right Rotation



Right-Left Rotation



Motivation and Related Work

Challenges:

- Process of restoring ***tree invariant*** becomes bottleneck in concurrent implementations due to mutating operations
- Rotations require locks to guarantee the atomicity of their change and keep other mutating operations from affecting the balance conditions of nodes that will be rotated
- Concurrent deletion of nodes with both children may lead to locking arbitrary number of nodes

Concurrent Search:

- Search operations advance by using self-compatible ρ-locks
- Multiple search processes can ρ-lock a node

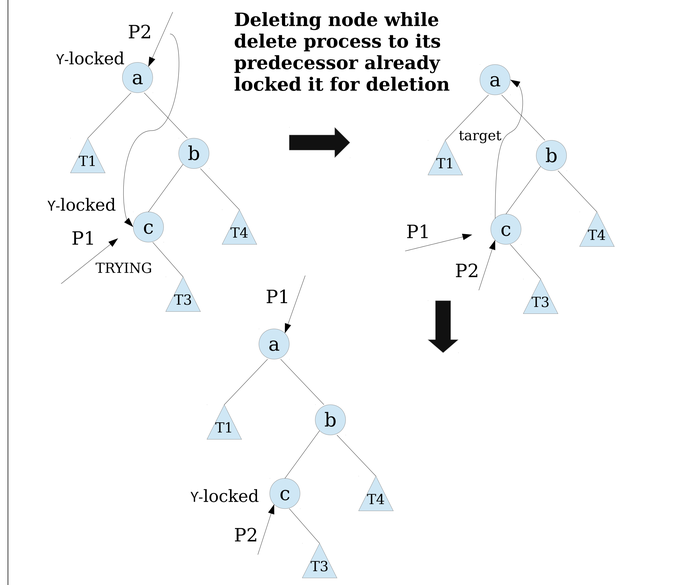
Concurrent Delete:

- Self-incompatible β-locks are used to ensure that a node be deleted by just one delete process
- Self-incompatible ϒ-locks are used to lock the node before physically deleting it from the tree and prevents other insert/delete processes from accessing the node
- When a node **n** is ϒ-locked by a delete process which wants to delete its predecessor, **n**.*target* is set to point to predecessor node and **n** is marked *invalid*
- Self-incompatible ξ-locks, incompatible with ρ-locks as well, are used to lock a node **n** when key of its successor gets copied to it due to delete request to **n**

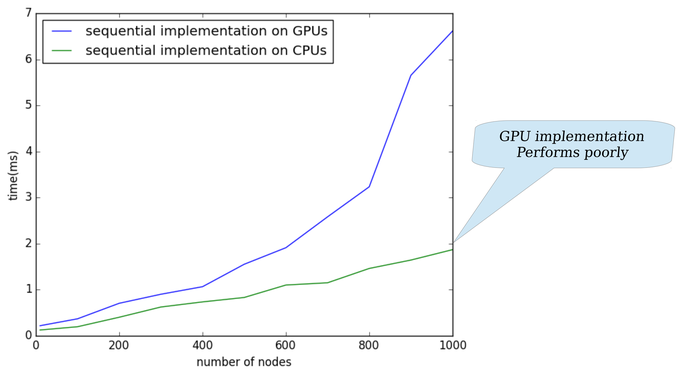
Concurrent Insert:

- Insert process β-locks the node **n** and checks for insertion
- It ξ-locks the father of **n** to exclude descending processes from accessing **n** and makes insertion
- Uses α-locks in steps, traverses till the root to rebalance the tree

Example



Result



References

- Zhang Yin and Xu Zhuoqun. *Concurrent Manipulation of Expanded AVL Trees*. Journal of Computer Science and Technology, 1998,V13(4): 325-336
- Mulralidhar Medidi and Narsingh Deo. *Parallel Dictionaries Using AVL Treess. JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 49, 146155 (1998) ARTICLE NO. PC981432*
- Carla Schlatter Ellis. *Concurrent Search and Insertion in AVL Trees*. IEEE Transactions on Computers, September 1980
- Dana Drachsler, Martin Vechev, Eran Yahav. *Practical Concurrent Binary Search Trees via Logical Ordering*. Proceedings of the 19th ACM SIGPLAN symposium on Principles and practice of parallel programming
- Joaquim Gabarró and Xavier Messeguer. *A Unified Approach to Concurrent and Parallel Algorithms on Balanced Data Structures*. Computer Science Society, 1997 Proceedings.