Fast Concurrent AVL Trees

Erik Henriksson

December 16th, 2013

Outline

Project

- Related work

Complexity in an AVL Tree

Idea

Algorithm

- Rebalancing

Results

Project

Implement a parallel implementation of an AVL Tree which is as scalable as possible in the timeframe of this semester.

- The tree will have the functions `contains`, `insert`, `delete`, ~~succ~~ and ~~pred~~.
- Show benchmarks of scalability

Related work

- G. Bronson, Casper, Chafi and Olukotun: A Practical Concurrent Binary Search Tree (2009)
- H. Kim, Cameron and Graham: Lock-Free Red-Black Trees Using CAS (2011)
- Maurice Herlihy and Nir Shavit: The Art of Multiprocessor Programming (2008)

Fast Concurrent AVL Trees

- B. Wicht: Binary Trees Implementations Comparison for Multicore Programming (2012)

- Adelson-Velskii, G.; E. M. Landis: An algorithm for the organization of information (1962)

Complexity in an AVL Tree

- `find`: O(log n) operations
- `modify`: O(1) operations
- `rebalance`: O(1) operations

Idea

- `contains` (`find`): O(log n) operations
- `insert` (`find` + `modify` + `rebalance`): O(log n) operations
- `remove` (`find` + `modify`): O(log n) operations

Âť It seems that the most expensive operation is `find`, so lets optimize our concurrent algorithm for that.

Algorithm

- We want fast `find`
- We dont care that much about scalability of the others

Lets organize our algorithm so that the tree is always consistent for find, effectively re-using the find algorithm for the sequential case. This operation is wait-free without helping and scales very well.

Rebalancing

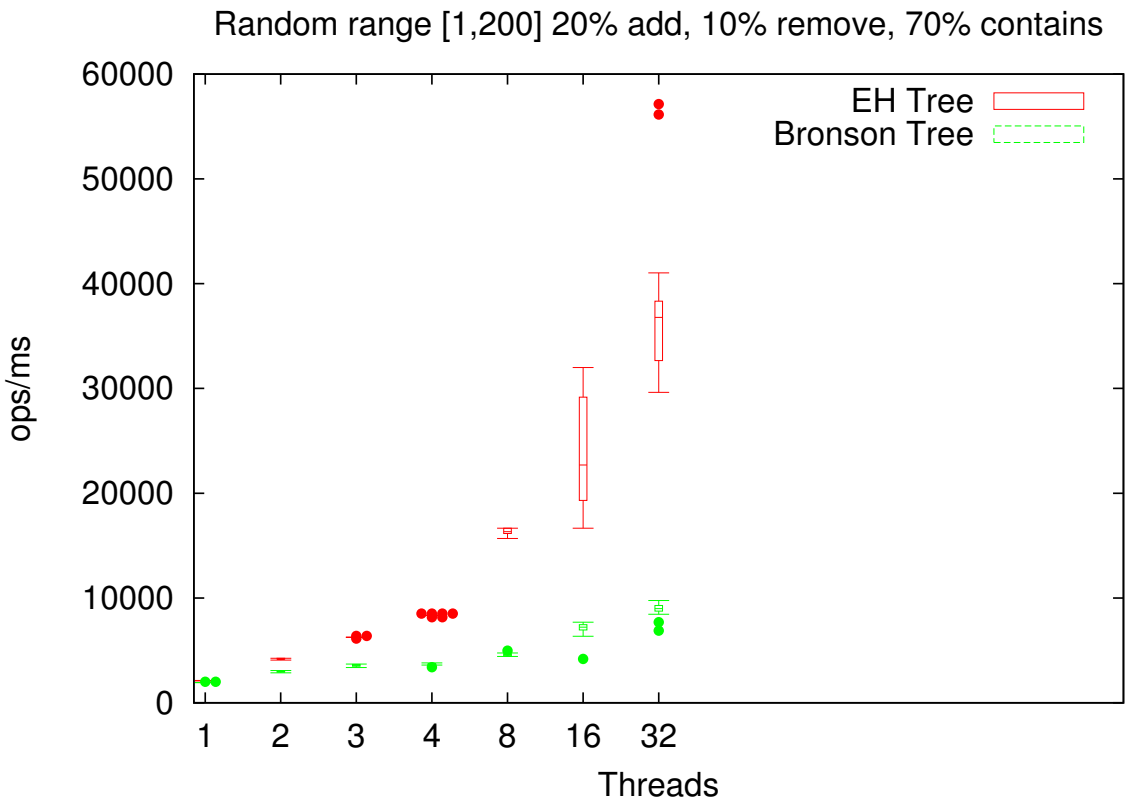
- We need to always have a consistent tree
- For simplicity, I use a global lock on rebalancing. This can be improved with using more fine-grained locking, but that is beyond the scope of this project.
- The operation is based on only change pointers in the tree, which does not invalidate any concurrent `find` operations.
- We need to allocate one node to maintain this.

Results

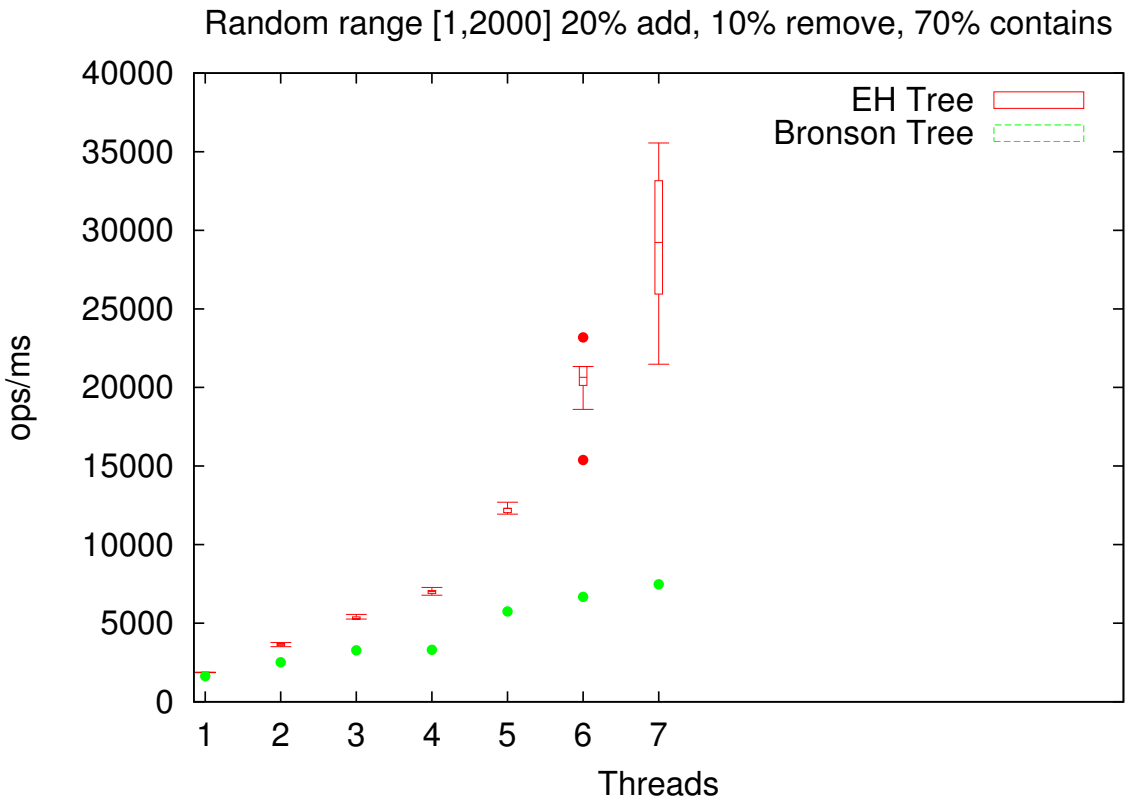
Setup

- Using the provided 32 core machine
- c++ compiler flags: `--std=c++11 -pthread -Wall`
- Used gcc 4.7.2 to compile
- 20 runs for each benchmark
- Bronson tree implementation from B. Wicht

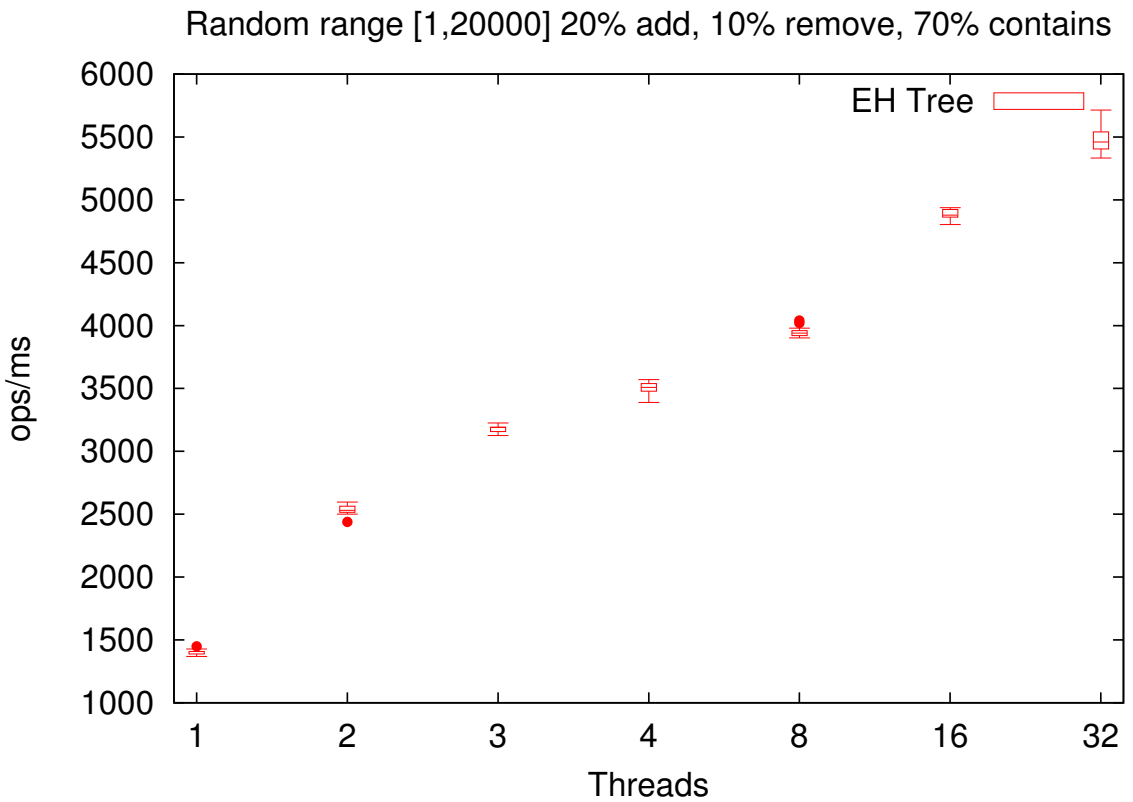
Fast Concurrent AVL Trees



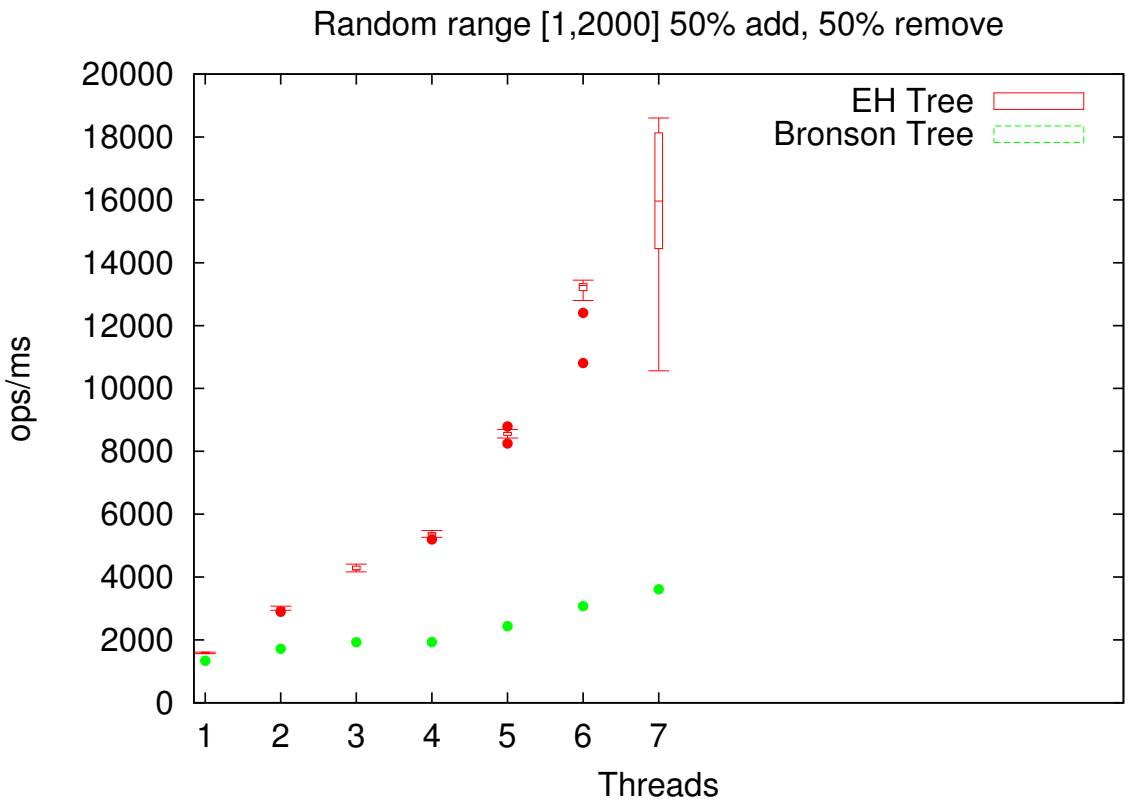
Fast Concurrent AVL Trees



Fast Concurrent AVL Trees



Fast Concurrent AVL Trees



Questions?