

Concurrent Red-Black Trees

Franck van Breugel

DisCoVeri Group, York University, Toronto

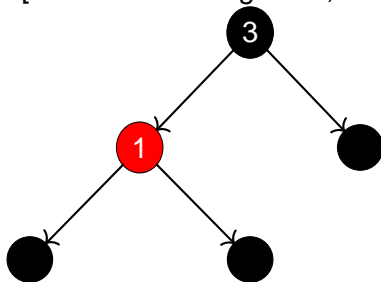
February 26, 2010

Red-Black Tree

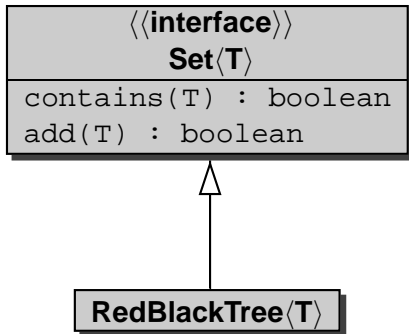
A red-black tree is a binary search tree the nodes of which are coloured either red or black and

- the root is black,
- every leaf is black,
- if a node is red, then both its children are black,
- for every node, every path from that node to a leaf contains the same number of black nodes.

[Bayer, 1972] and [Guibas and Sedgewick, 1978]



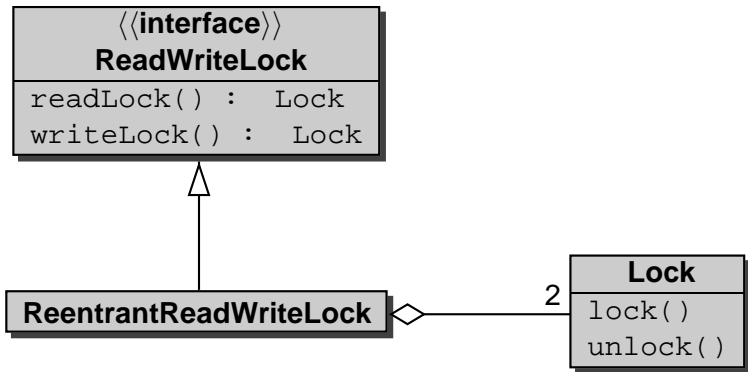
Three Implementations



The Monitor Solution

```
1 package monitor;  
2  
3 public class RedBlackTree<T extends Comparable<T>>  
4     implements Set<T>  
5 {  
6     public synchronized boolean contains(T element)  
7     {  
8         ...  
9     }  
10  
11    public synchronized boolean add(T element)  
12    {  
13        ...  
14    }  
15 }
```

The Readers-Writers Solution



The Readers-Writers Solution

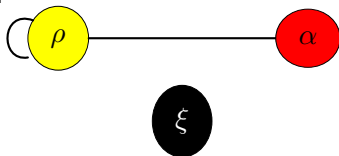
```
1 private ReadWriteLock lock;  
2  
3 public RedBlackTree()  
4 {  
5     this.lock = new ReentrantReadWriteLock();  
6     ...  
7 }  
8  
9 public boolean contains(T element)  
10 {  
11     this.lock.getReadLock().lock();  
12     ...  
13     this.lock.getReadLock().unlock();  
14 }  
15 ...
```

Locking Nodes

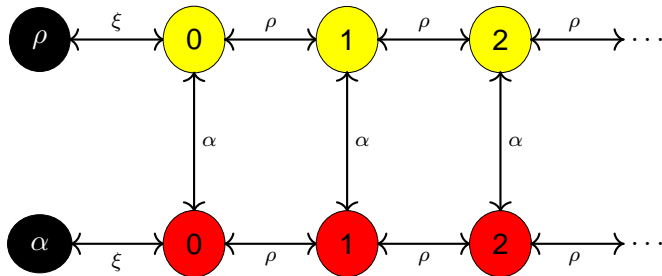
Processes lock the nodes of the red-black tree in three different ways:

- ρ -lock: lock to read
- α -lock: lock to exclude writers
- ξ -lock: exclusive lock

Although a node can be locked by multiple processes, there are some restrictions.



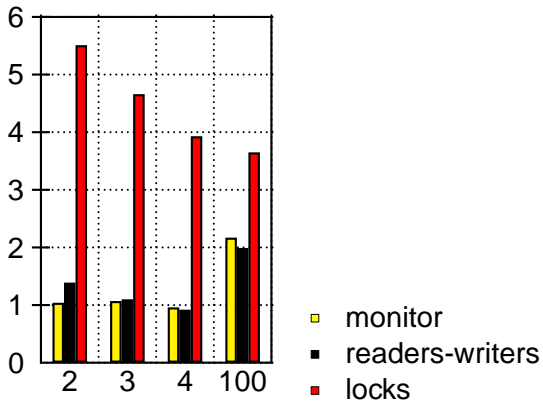
Locking Nodes



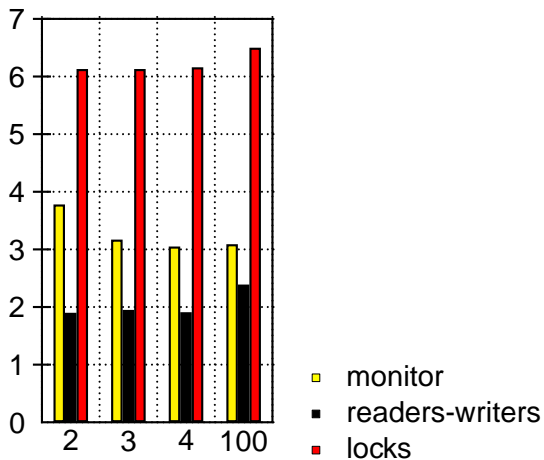
Locking Nodes

```
1 public class Node<T>
2 {
3     private int containers;
4     private int state;
5     private boolean writing;
6
7     public void readLock() { ... }
8     public void readUnlock() { ... }
9     public void writeLock() { ... }
10    public void writeUnock() { ... }
11    public void exclusiveLock() { ... }
12    public void exclusiveUnlock() { ... }
13 }
```

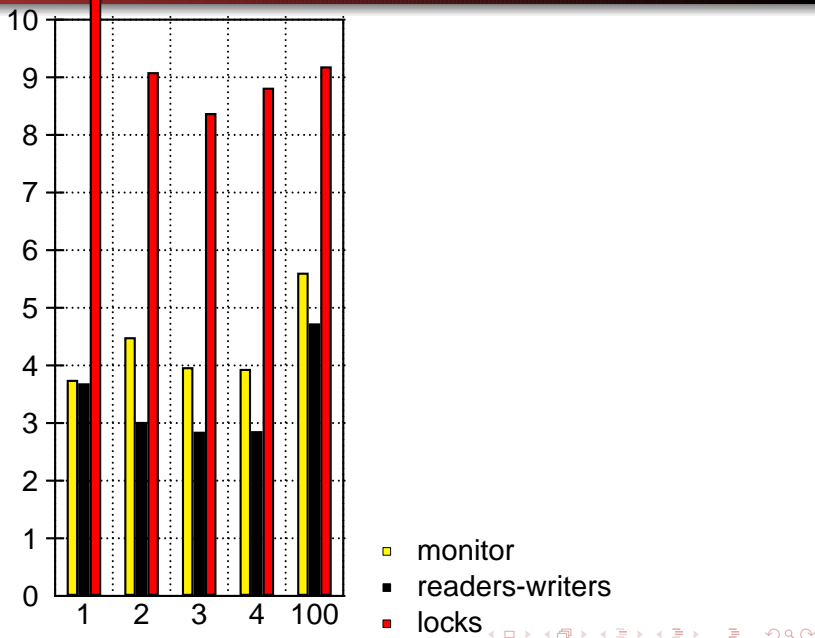
Performance Comparison: add only



Performance Comparison: contains only



Performance Comparison: contains and add



Room for Improvement

- lock only “half a node”
- lock granularity

Plan

- verify some properties, such as deadlock freedom, of all three concurrent implementations by means of Java Pathfinder
- show undesirable behaviour of

```
1 add(3);  
2 add(1);  
3 (add(2) || print(contains(1)))
```

in case no synchronization is used

Challenges

- state space explosion
- native code