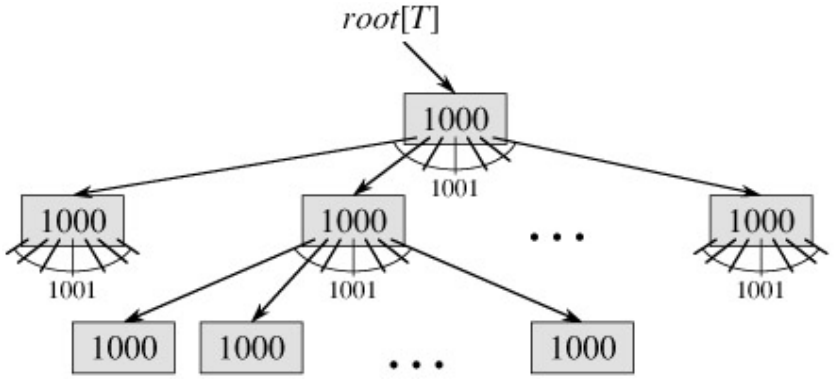
CONCURRENT B-TREE Verification

COSC6490A

Liping Han

B-Tree

- Data items are stored in leaf nodes
- Each Internal node include size-1 keys and size downlinks to its children, $\lceil M/2 \rceil \leq size \leq M$
- Within each node keys are in ascending order
- All leaf nodes have same depth



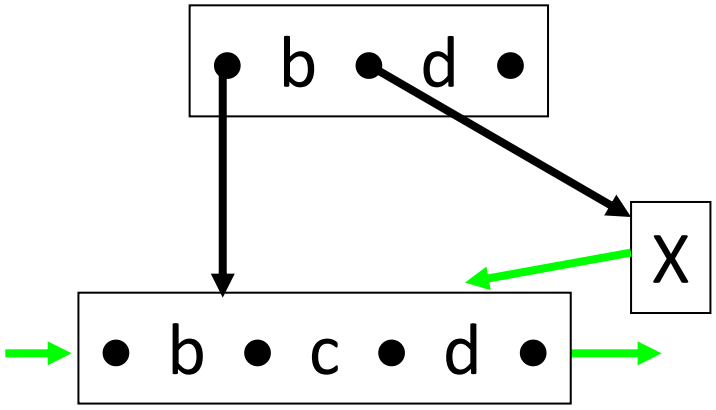
1 node,
1000 keys

1001 nodes,
1,001,000 keys

1,002,001 nodes,
1,002,001,000 keys

Variants for Concurrent

- Each internal node has a rightlink to the node right to it at the same level
- Empty node add an outlink to the node where its content is merged to



Lock Granularity

- Lock on nodes
- General case lock on one node
- Early-lock-releasing

Algorithms – Phase 1

- Locate

  …

  read.lock

  findpath

  read.unlock

  …

Algorithms – Phase 2

- Action on leaf node
  - Search: read lock
  - Insert: write lock
  - Delete: write lock

Algorithms – Phase 3

- Normalization: two phase split and merge
  - Half-split/half-merge: not involve parent
  - Add-link/remove-link: take place on node of level above half-split/half-merge

Monitor Implementation

```
public class monitorBTree implements BTree {
…
    public synchronized boolean delete(int key) {
    …}
    public synchronized boolean search(int key) {
    …}
    public synchronized boolean insert(int key) {
    …}
…
}
```

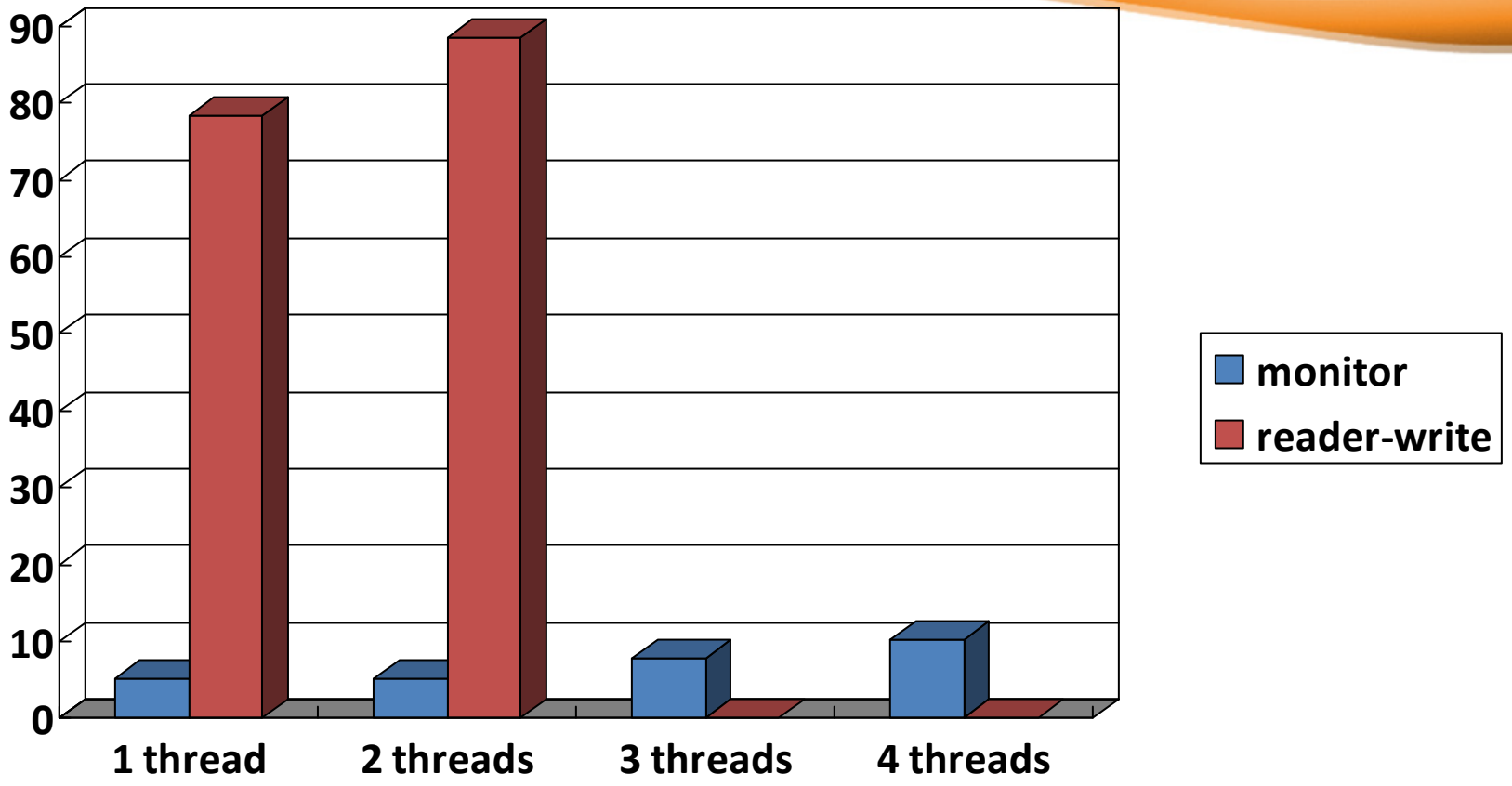
Reader-Writer Solution

```
public class Node {
    public ReentrantReadWriteLock rwLock;
    public Node(int t, int key){
        …
        this.rwLock = new ReentrantReadWriteLock();
        …
    }
}
```

Reader-Write Solution

```
public class RWBTree implements BTree {
    private ReentrantReadWriteLock treeLock;
    public RWBTree(int k){
        …
        this.treeLock = new ReentrantReadWriteLock();
        …
    }
}
```

Performance



Properties

- Synchronization

  insert(1) || delete(1) || search(1)
- Deadlock free
- No uncaught exceptions

States Explode Quickly

- B-Tree instance has read-lock and/or write-lock in case of root read/write
- Each Node instance has read-lock and/or write-lock
- Search thread places a read lock on each of the nodes it accesses
- Insert thread places a read lock on each of the nodes it accesses except the final leaf node and a write lock on the leaf
- Same condition for Delete thread
- Potential Split/Merge places write locks on the nodes being modified

States Explode Quickly

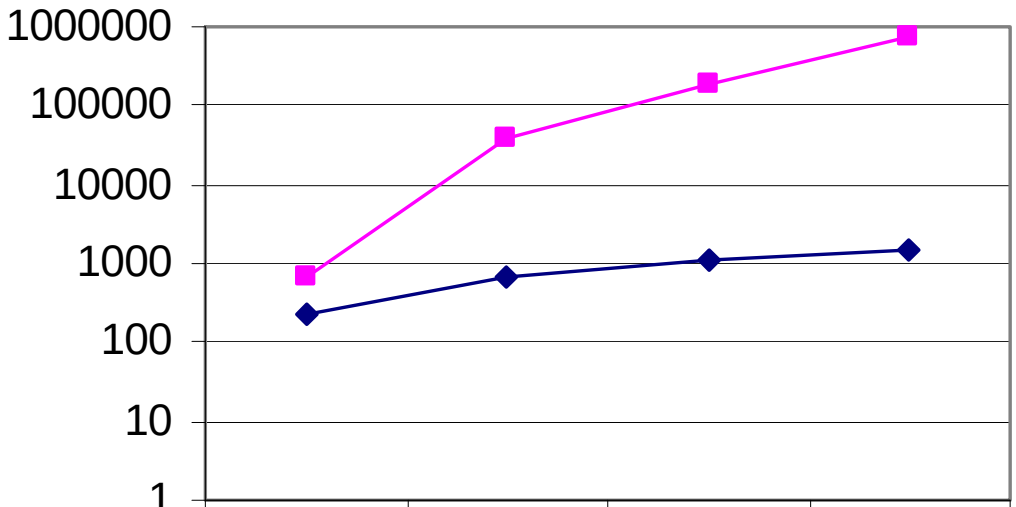
- 1 inserter, 1 searcher, 1 deleter, each threads do 1 operation on initially empty tree

  states: new=254580, visited=426950, backtracked=681529

- 1 inserter, 1 searcher, 1 deleter, each threads do 2 operations on initially empty tree

  states: new=1720955, visited=3008107, backtracked=4729061

States Explode Quickly



	1 op per thread	2 ops per thread	3 ops per thread	4 ops per thread
Monitor	221	680	1061	1454
Reader-Writer	665	37795	187468	761203

Papers

- Lehman, P. L. and Yao, s. B. 1981. Efficient locking for concurrent operations on B-trees. *ACM Trans. Database Syst.* 6, 4 (Dec. 1981), 650-670. DOI= http://doi.acm.org.ezproxy.library.yorku.ca/10.1145/319628.319663
- Lanin, V. and Shasha, D. 1986. A symmetric concurrent B-tree algorithm. In *Proceedings of 1986 ACM Fall Joint Computer Conference* (Dallas, Texas, United States). IEEE Computer Society Press, Los Alamitos, CA, 380-389.