A Transaction-Friendly Binary Search Tree

Tyler CRAIN Vincent GRAMOLI Michel RAYNAL
{tyler.crain|raynal}@irisa.fr
{vincent.gramoli}@epfl.ch

TRANSFORM
Theoretical Foundations of Transactional Memory

TM Theory Workshop 2011

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
IRISA
UNIVERSITÉ DE RENNES 1
INRIA RENNES

ASAP team, IRISA, Rennes, France & Distributed Programming Laboratory, EPFL, Lausanne, Switzerland

Existing Data Structures

Preserving structural invariants

- **Balanced Binary Trees**
  - Rotations keep the tree balanced
- **Skiplist**
  - Node levels follow some fixed distribution
- **Hashtable**
  - Bucket size must not exceed some threshold

Using data structures with TM

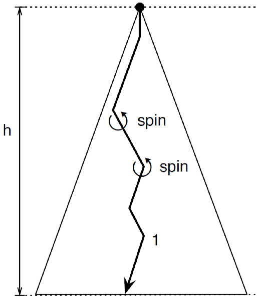
Simple!

- Copy-paste into transactions (more or less)
- \+ Easy to program/use
- \+ The TM system ensures safety
- \- Are there disadvantages?

Concurrent Data Structures

Balanced Binary Tree

- Specially designed for concurrency
- Hand-over-hand locking

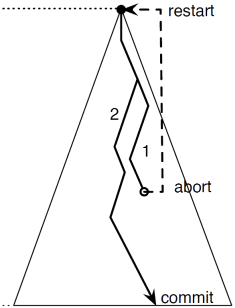


- *O*(log *n*)

Data Structures in TM (So far)

Balanced Binary Tree

- (Mostly) Unmodified from their original versions
- Not designed for concurrency or transactions
  - Could lead to unnecessary conflicts and aborts.



- $\Omega(r \log n)$

$\Omega(r \log n)$

What is r?

- The number of restarts
  - Depends on the contention of the workload
  - Depends on the conflicts between transactions

$\Rightarrow$ r depends on n

Aborts and wasted work

- Still $O(\log n)$ operations?

Update	0%	10%	20%	30%	40%	50%
AVL tree	29	415	711	1008	1981	2081
Sun red-black tree	31	573	965	1108	1484	1545

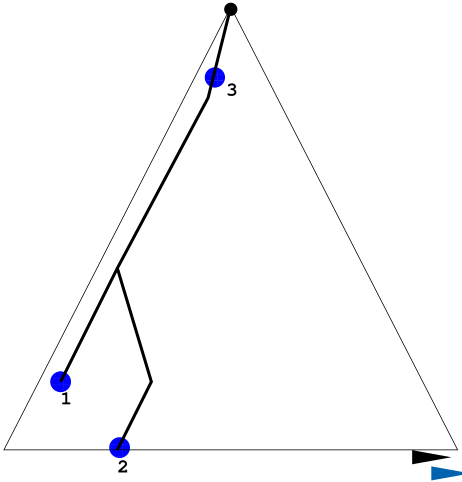
Table: Maximum #reads/op in 2^{12} sized trees

- Can we relax some invariants in order to reduce conflicts?

IRISA ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

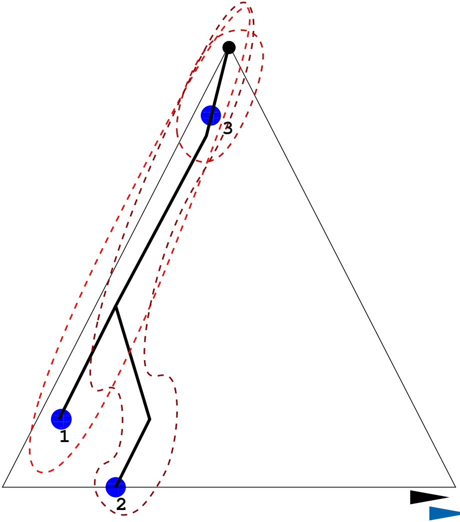
Example

- 3 operations
- 1 → *insert*, 2 → *delete*, 3 → *contains*



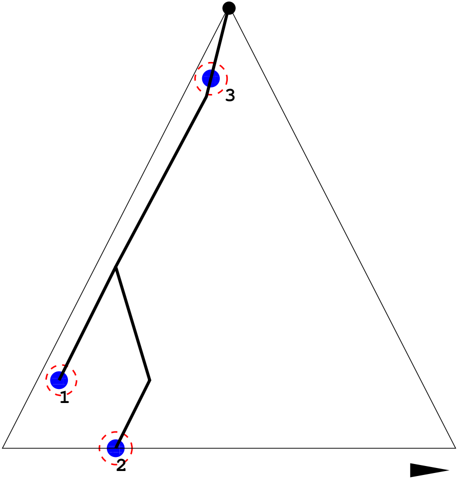
Where they can conflict

- Along their entire path



Goal

- Minimize conflicts



Rotations

Correctness & Conflicts

- Rotations are not required for correctness
- There will be concurrent insertions/deletions
  - Concurrent insertions/deletions might have conflicting rotations
  - They might cancel each other out
  - A later insert/deletion might invalidate these rotations
- Idea: relax the balance requirement in order to allow more concurrency

Rotations cont.

Solution

- Separate rotations from insert/delete operations
- Perform rotations in their own thread
- Each rotation is a single transaction

Bougé L., Gabarro J., Messeguer X., Schabanel N., Height-relaxed AVL rebalancing: A unified, fine-grained approach to concurrent dictionaries. Tech Report RR1998-18, INRIA, 1998

Deletions

Reducing conflicts further

- A delete operation can still modify the tree structure
- A successor must be found to replace the node being deleted

Deletions cont.

Solution

- Logical deletions
  - Each node has a *deleted* boolean flag
  - Initialized as *false*
  - Set to *true* on deletion
- Allows concurrent operations to traverse the node being deleted without conflicting

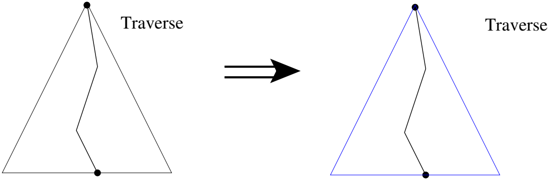
Removals

- Logically deleted nodes must be removed from the tree
  - Done in a separate thread
  - Only nodes with 1 or 0 children are removed

Bronson N., Casper J., Chafi H., Olukotun K., A Practical Concurrent Binary Search Tree, PPoPP '10

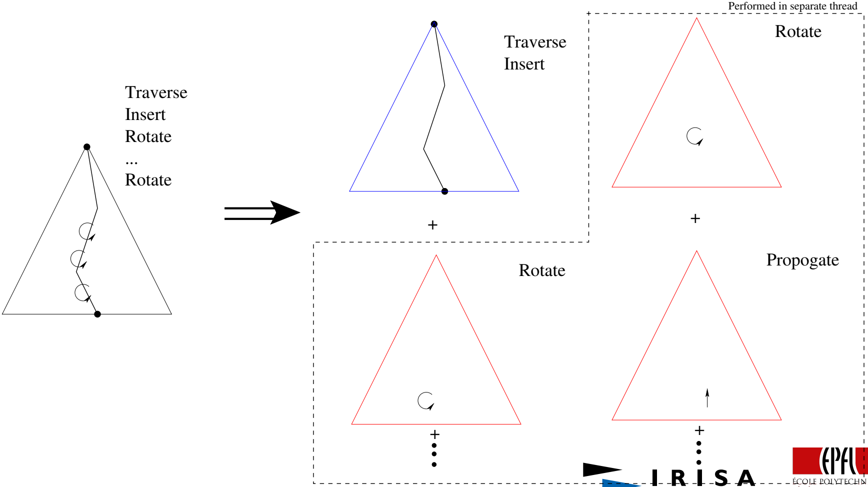
Contains

- Each diagram is a single transaction



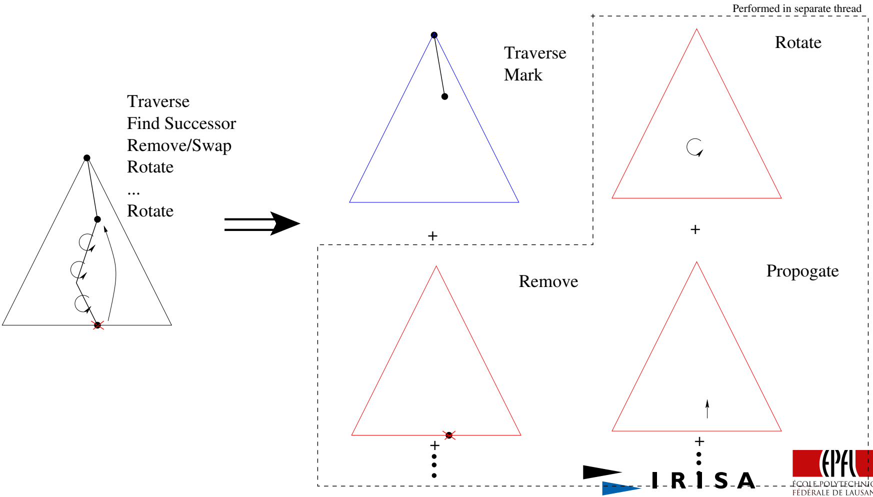
Insert

- Each diagram is a single transaction



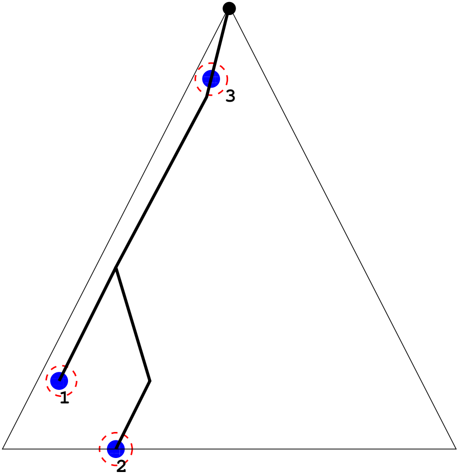
Delete

- Each diagram is a single transaction



Now we have

Abstract transaction conflicts



Impact on read size

Update	0%	10%	20%	30%	40%	50%
AVL tree	29	415	711	1008	1981	2081
Sun red-black tree	31	573	965	1108	1484	1545
Tx-friendly tree	29	75	123	120	144	180

Table: Maximum #reads/op in 2^{12} sized trees

Conclusion

Benefits of a Transaction Friendly Data Structure

- Improved Performance
- No difference to the programmer using the tree as a library
- Uses normal transactional reads/writes
  - Compatible with many TMs
  - Tested on TinySTM and $\mathcal{E}$-STM
  - Independent of TM specifications
  - Tested using CTL/ETL, different contention managers

IRISA ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Reusability

Move operation

Algorithm 3 Move operation

```
move(old_key, new_key)_p:
  transaction {
    ret ← false
    if ¬contains(new_key) then
      if v ← delete(old_key) then
        insert(new_key, v)
        ret ← true
  } // current transaction tries to commit
  return ret
```

IRISA

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Future Work

Other structures

- Transaction friendly skip list
- Transaction friendly hash table
- Transaction friendly . . .

- There’s more?

TM Optimizations

Optional

- Certain TMs give mechanisms for improved performance at the cost of safety
  - Early-release
  - $\mathcal{E}$-STM
  - View transactions
  - Unit reads

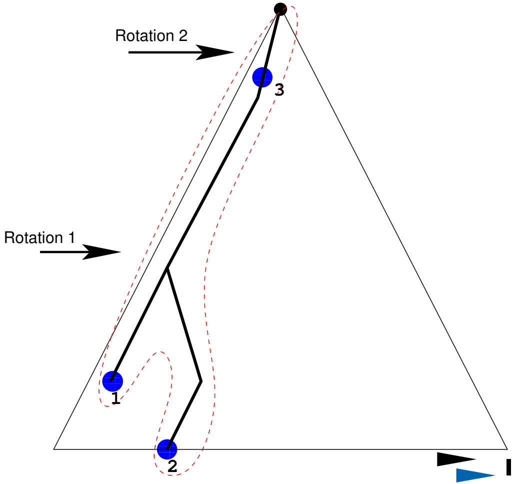
Unit Reads

- Returns the latest value written by a committed transaction
- Does not add the location to the read set or perform validation

- How can unit reads be used to improve performance of the algorithm?

Current Situation

- Rotations can still conflict with concurrent insert/delete/contains operations



IRISA

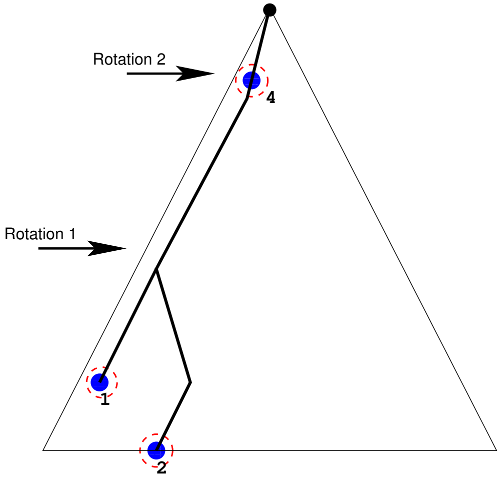
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Solution

- Use unit reads during the tree traversal
- Advantages:
  - Faster traversals (unit reads are cheaper)
  - Avoid during traversal
  - Smaller read set

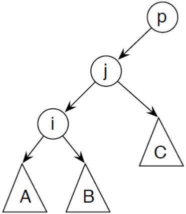
Result

Abstract + Structural Transactions

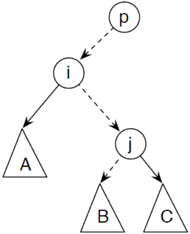


- What about safety?
- Algorithm becomes a bit more complicated to ensure safety

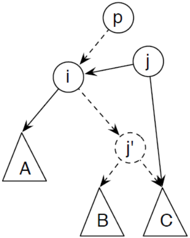
New rotations



(a) Initial tree

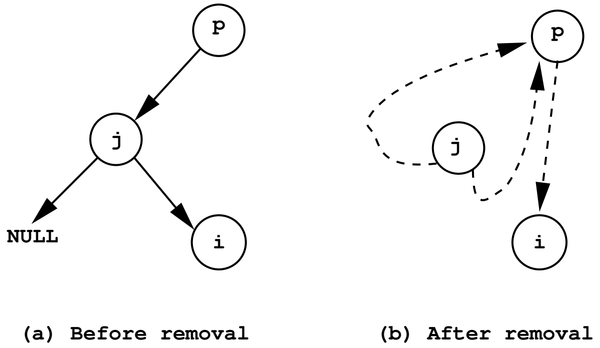


(b) Result of usual right rotation



(c) Result of new right rotation

New removals



Traversals

- Mostly unit reads
- Transactional reads performed at bottom to ensure safety
- Each node has a *removed* flag
    - Used to ensure traversal does not finish on a node that no longer in the tree

Impact on read size

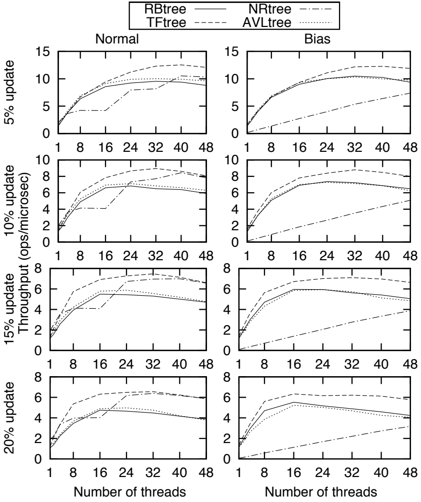
Update	0%	10%	20%	30%	40%	50%
AVL tree	29	415	711	1008	1981	2081
Sun red-black tree	31	573	965	1108	1484	1545
Tx-friendly tree	29	75	123	120	144	180
Unit read tree	2	5	6	13	15	18

Table: Maximum #reads/op in 2^{12} sized trees

IRISA
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- Performance Results: Some graphs from benchmarks

Microbench


RBtree
NRtree
TFtree
AVLtree
Normal
Bias
5% update
10% update
15% update
20% update
Throughput (ops/microsec)
Number of threads
Number of threads

Vacation (STAMP)