Bw-Tree

Josef Schmeißer

January 9, 2018

Table of contents

1. Fundamentals

2. Tree Structure

3. Evaluation

4. Further Reading

Fundamentals

```
bool compare_and_swap(int * ptr, int & expected, int desired) {
    int oldValue;
    atomic {
        oldValue = *ptr;
        if (oldValue == expected) {
            *ptr = desired;
            return true;
        }
    }
    expected = oldValue;
    return false;
}
```

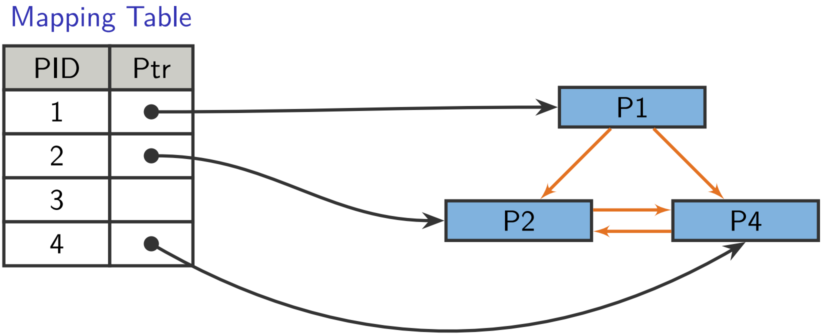
Figure: Semantics of the CAS instruction.

Features

Main features:

- Lock-free data structure
- Mapping Table, which maps Page Identifiers ($PIDs$) to pointers
- Mapping Table entries can be atomically altered via CAS
- B$^{\text{link}}$-Tree like side links (important for split and merge)

Architecture



Mapping Table

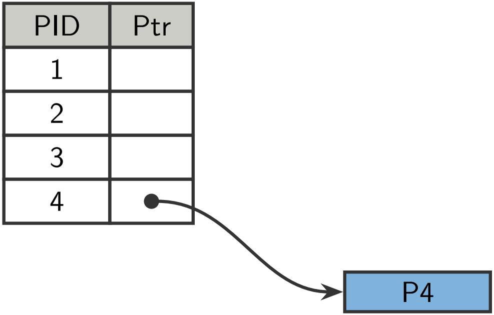
PID	Ptr
1	●
2	●
3	
4	●

→ PID Reference

→ Memory Pointer

Delta Updates

Mapping Table

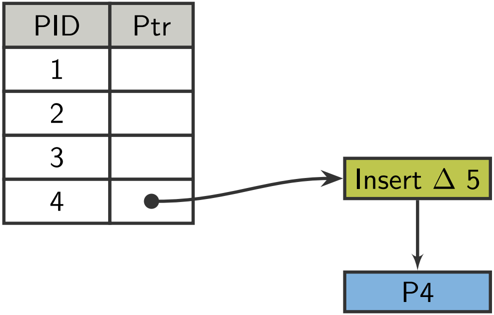


PID	Ptr
1	
2	
3	
4	●

- Immutable base page

Delta Updates

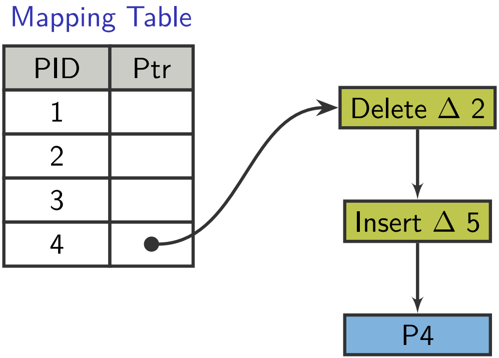
Mapping Table



PID	Ptr
1	
2	
3	
4	●

- Immutable base page
- Perform updates to logical pages through delta records

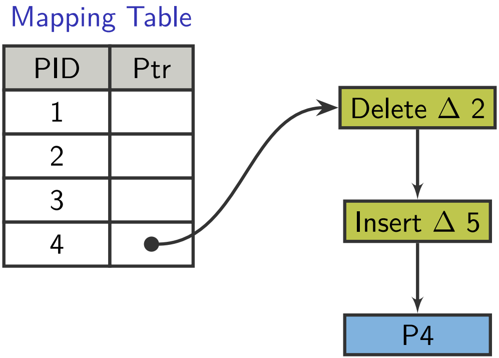
Delta Updates


Mapping Table

PID	Ptr
1	
2	
3	
4	●

- Immutable base page
- Perform updates to logical pages through delta records
- Delta records are chained in a singly linked list

Delta Updates

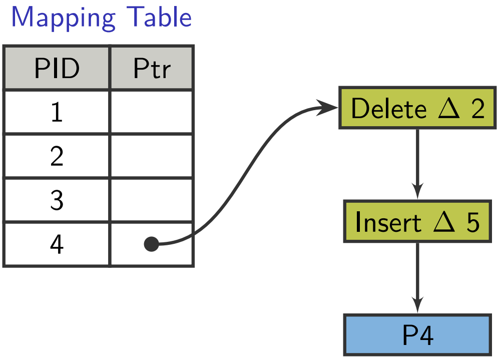


Mapping Table

PID	Ptr
1	
2	
3	
4	●

- Immutable base page
- Perform updates to logical pages through delta records
- Delta records are chained in a singly linked list
- Install updates atomically via CAS

Search

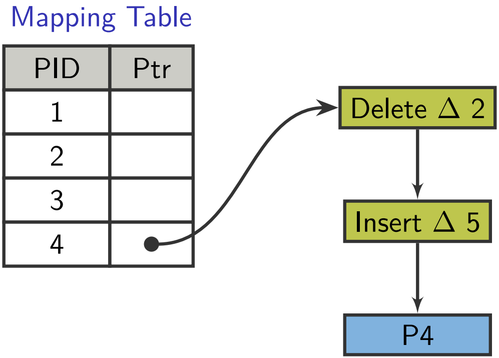


Mapping Table

PID	Ptr
1	
2	
3	
4	●

- Traverse the tree as usual

Search



Mapping Table

PID	Ptr
1	
2	
3	
4	●

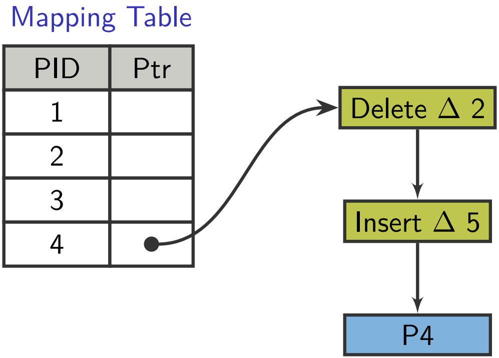
Delete Δ 2

Insert Δ 5

P4

- Traverse the tree as usual
- Inspect each record of the delta chain, and stop at the first occurrence

Search

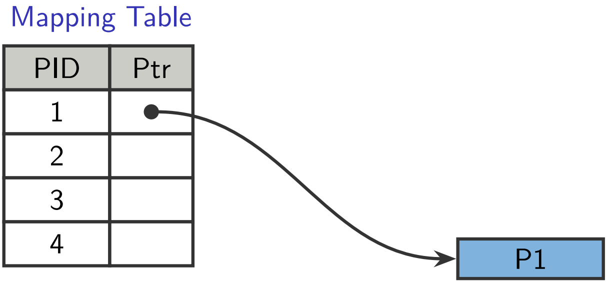


Mapping Table

PID	Ptr
1	
2	
3	
4	●

- Traverse the tree as usual
- Inspect each record of the delta chain, and stop at the first occurrence
- Perform a binary search on the base page if the search drops through

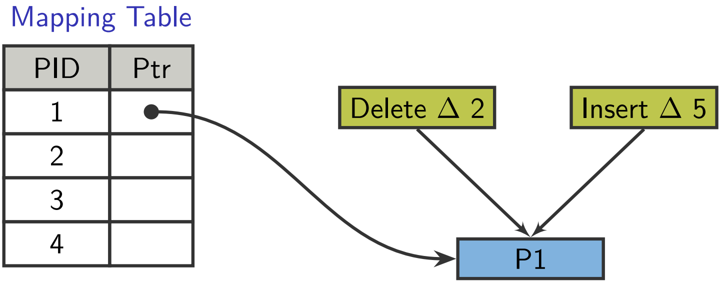
Conflicts



Mapping Table

PID	Ptr
1	●
2	
3	
4	

Conflicts



Mapping Table

PID	Ptr
1	●
2	
3	
4	

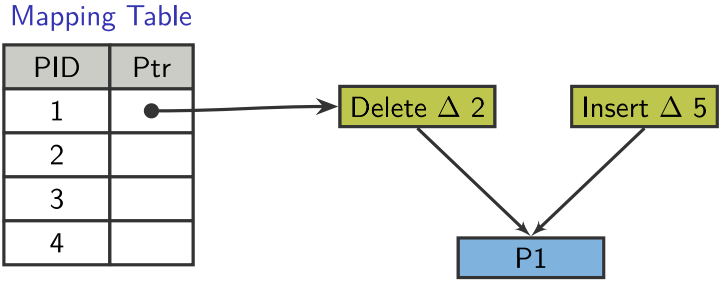
Delete Δ 2

Insert Δ 5

P1

- Multiple threads may try to install an update to the same page simultaneously

Conflicts



Mapping Table

PID	Ptr
1	●
2	
3	
4	

Delete Δ 2

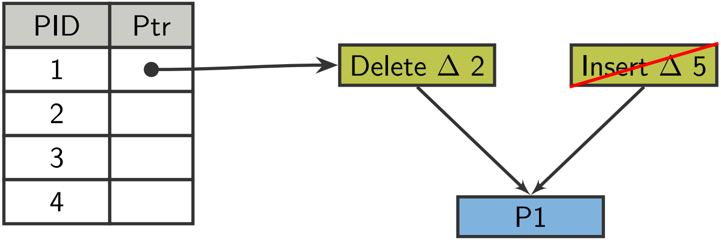
Insert Δ 5

P1

- Multiple threads may try to install an update to the same page simultaneously
- The atomic CAS ensures that only one thread succeeds

Conflicts

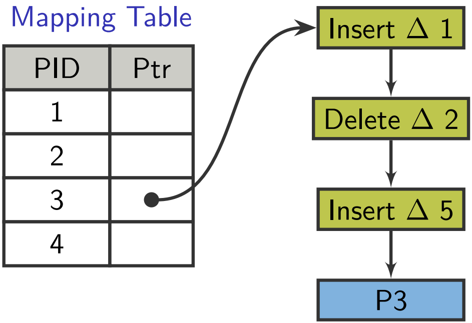
Mapping Table



PID	Ptr
1	●
2	
3	
4	

- Multiple threads may try to install an update to the same page simultaneously
- The atomic CAS ensures that only one thread succeeds
- Slower threads may retry

Consolidation



Mapping Table

PID	Ptr
1	
2	
3	●
4	

Insert Δ 1

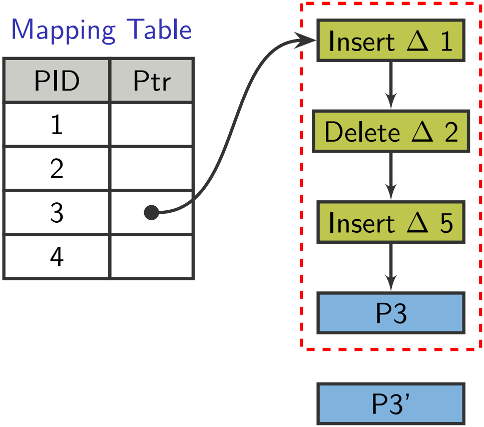
Delete Δ 2

Insert Δ 5

P3

Constantly appending deltas leads to ever-expanding chains.

Consolidation



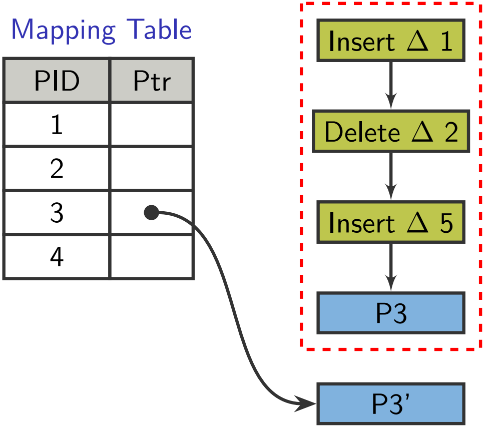
Mapping Table

PID	Ptr
1	
2	
3	●
4	

Constantly appending deltas leads to ever-expanding chains. Solution:

1. Consolidate the logical page by creating a new base page

Consolidation



Mapping Table

PID	Ptr
1	
2	
3	●
4	

Insert Δ 1

Delete Δ 2

Insert Δ 5

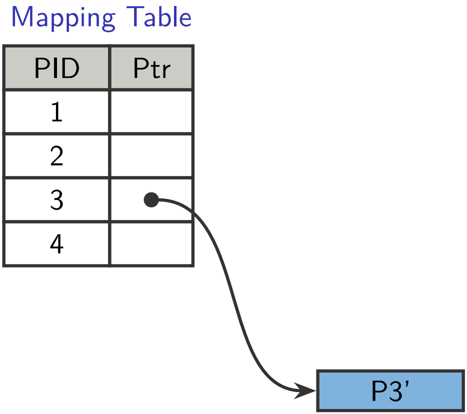
P3

P3’

Constantly appending deltas leads to ever-expanding chains. Solution:

1. Consolidate the logical page by creating a new base page
2. Install the new base page with an atomic CAS

Consolidation



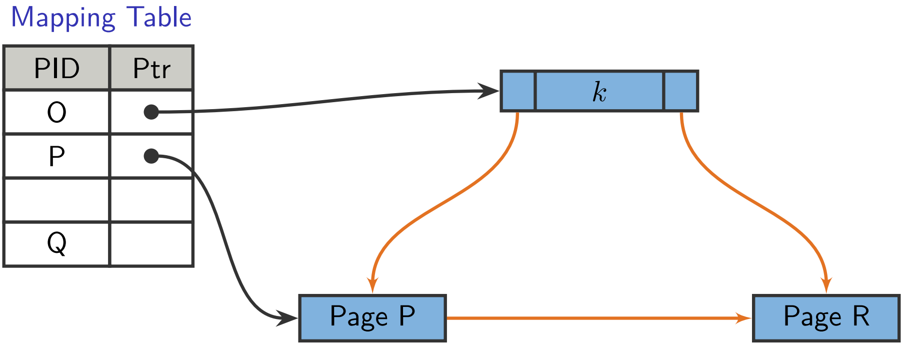
Mapping Table

PID	Ptr
1	
2	
3	●
4	

Constantly appending deltas leads to ever-expanding chains. Solution:

1. Consolidate the logical page by creating a new base page
2. Install the new base page with an atomic CAS
3. Reclaim the memory of the old logical page, once it is no longer used

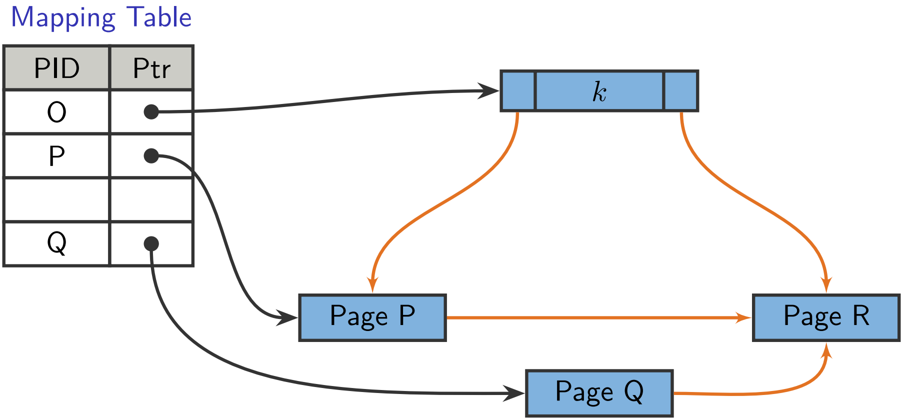
Node Split



Mapping Table

PID	Ptr
O	●
P	●
Q	

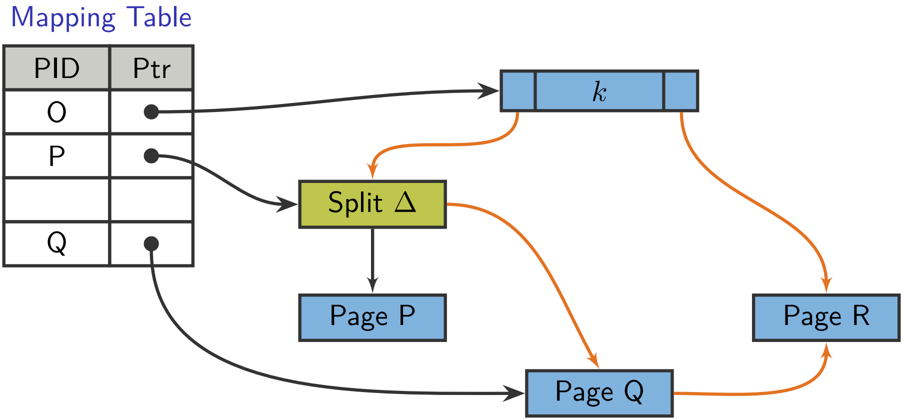
Node Split



Mapping Table

PID	Ptr
O	●
P	●
Q	●

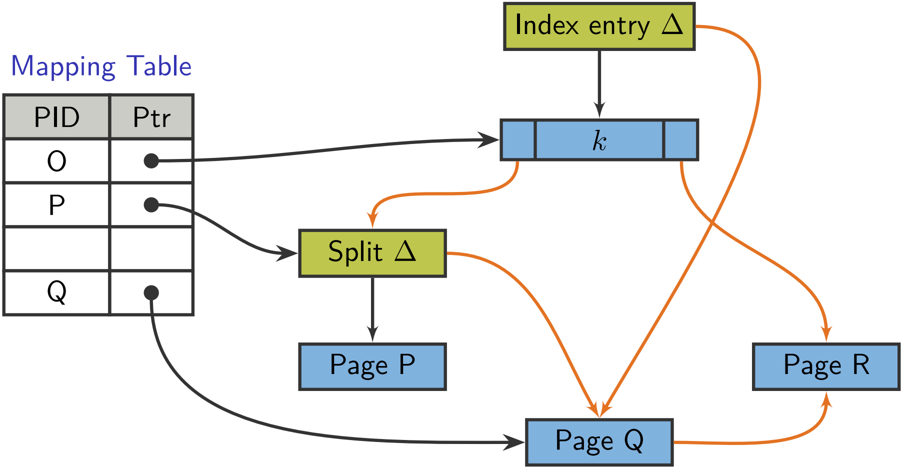
Node Split



Mapping Table

PID	Ptr
O	●
P	●
Q	●

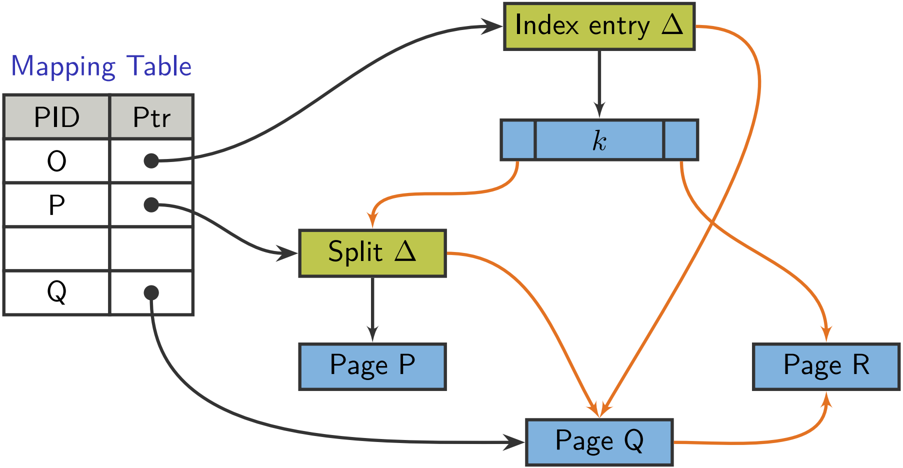
Node Split



Mapping Table

PID	Ptr
O	●
P	●
Q	●

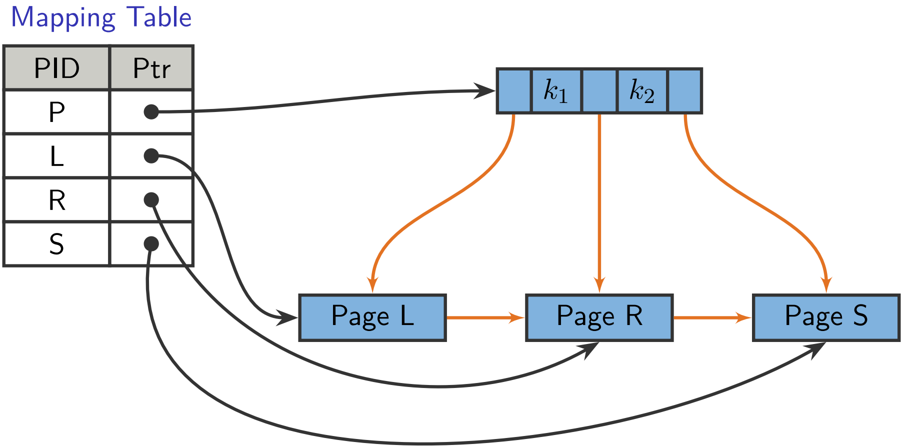
Node Split



Mapping Table

PID	Ptr
O	●
P	●
Q	●

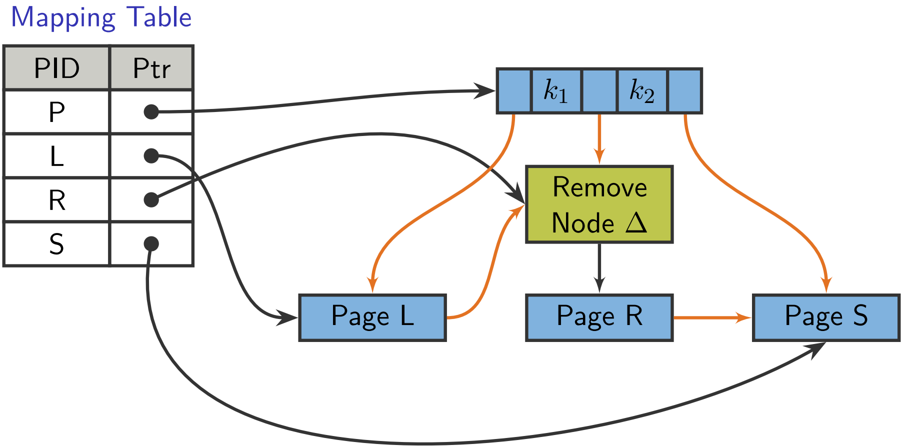
Node Merge



Mapping Table

PID	Ptr
P	●
L	●
R	●
S	●

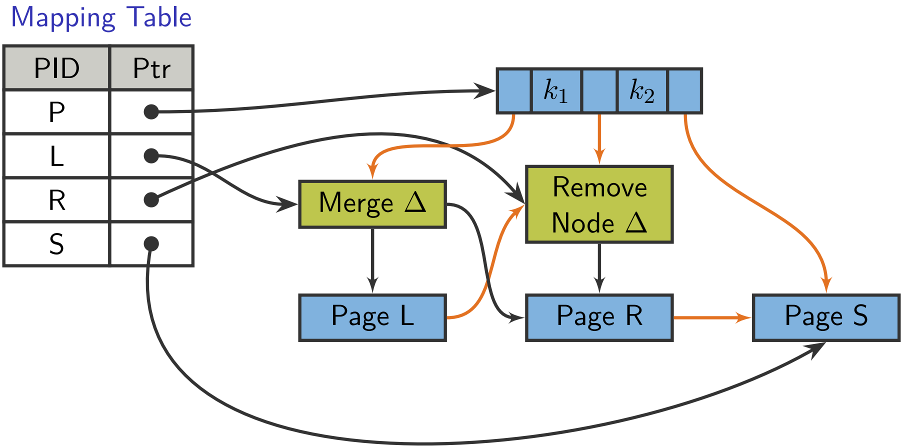
Node Merge



Mapping Table

PID	Ptr
P	●
L	●
R	●
S	●

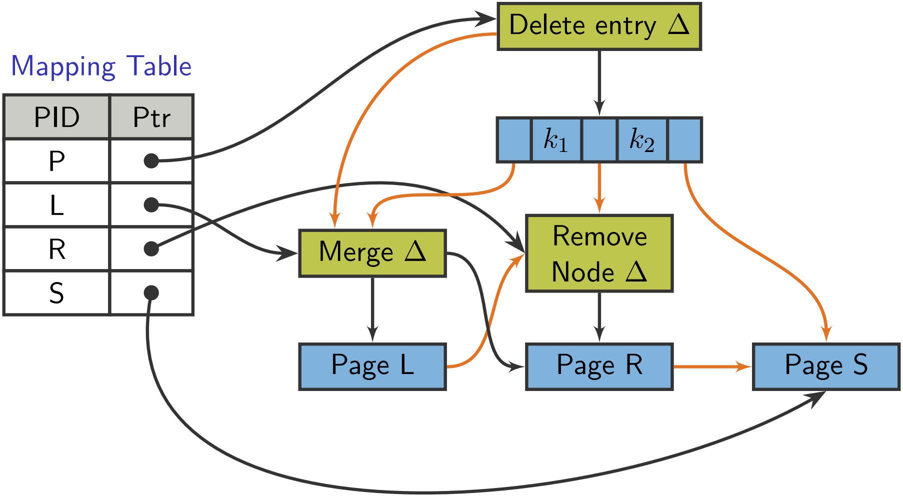
Node Merge



Mapping Table

PID	Ptr
P	●
L	●
R	●
S	●

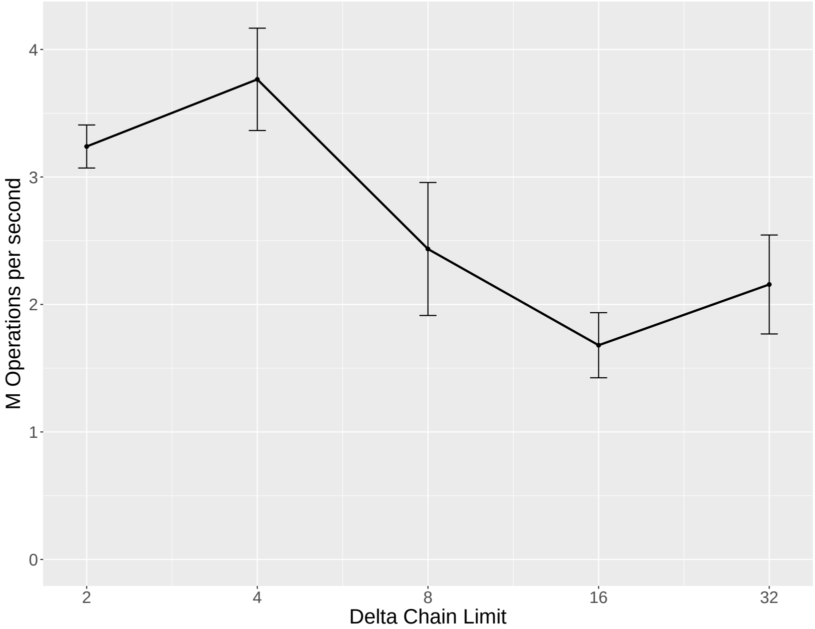
Node Merge



Mapping Table

PID	Ptr
P	●
L	●
R	●
S	●

Optimal Delta Chain Length



Experiment Description

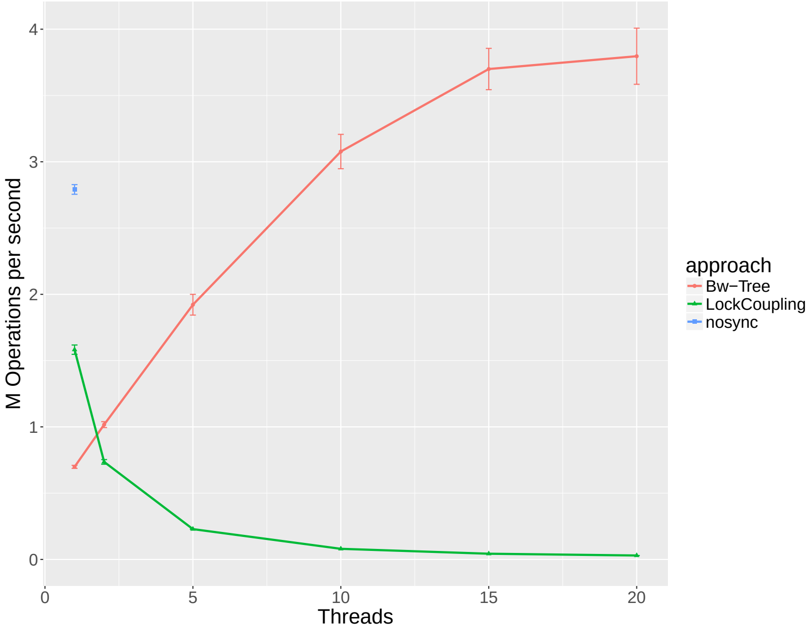
Synthetic workload:

- Integer keys and payload
- Randomly distributed
- Index size: 5M

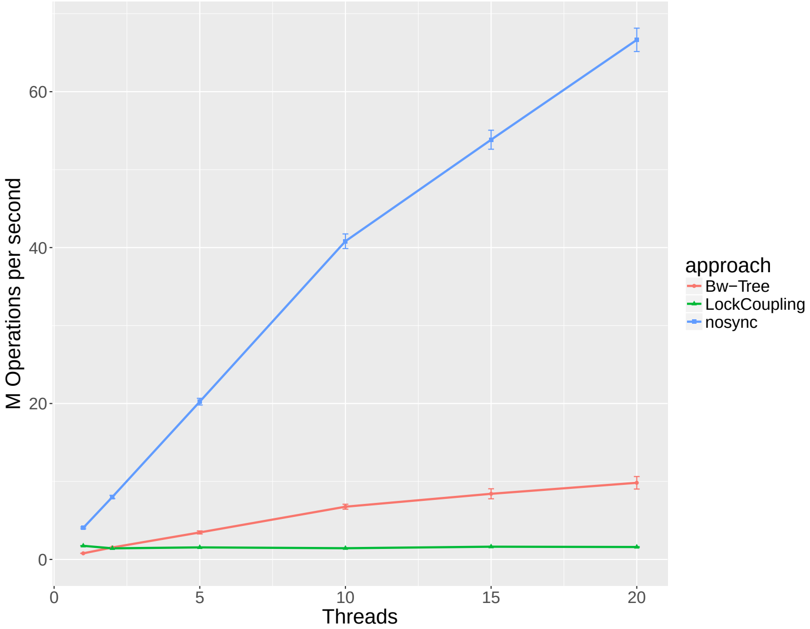
Test System - `atkemper4`:

- Intel® Core™ i9-7900X
- 10 Cores; 20 Threads
- Restricted Transactional Memory

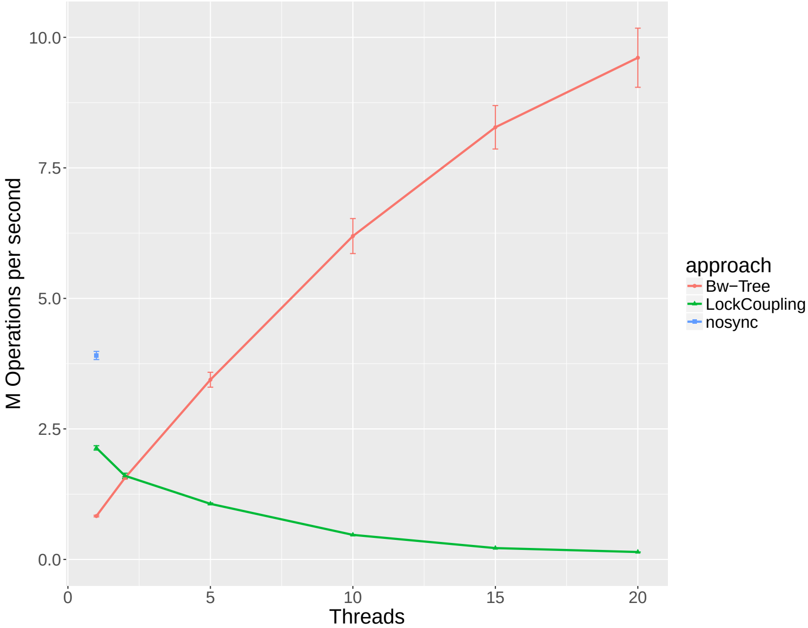
Insert


M Operations per second
0
1
2
3
4
Threads
0
5
10
15
20
approach
Bw−Tree
LockCoupling
nosync

Lookup


M Operations per second
60
40
20
0
0
5
10
15
20
Threads
approach
Bw−Tree
LockCoupling
nosync

Insert + Lookup

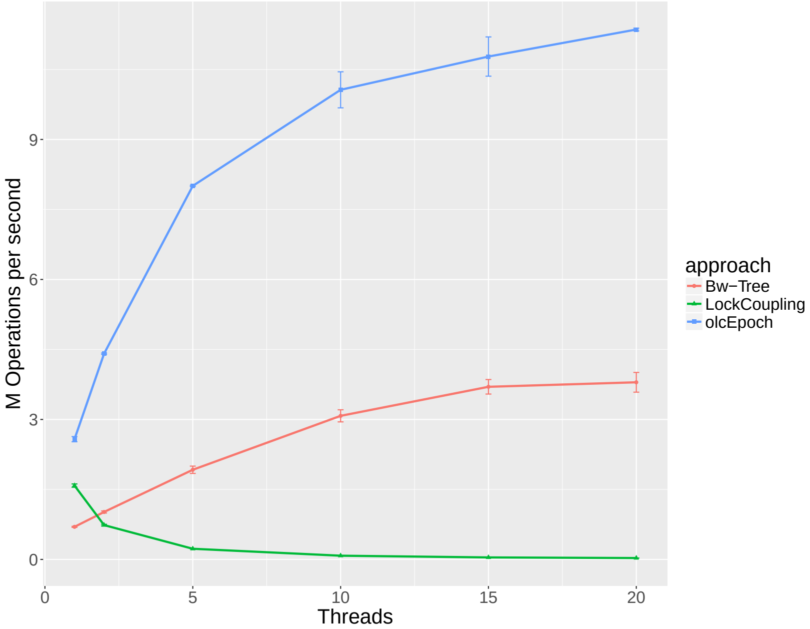


Alternative Approach

Optimistic Lock Coupling:

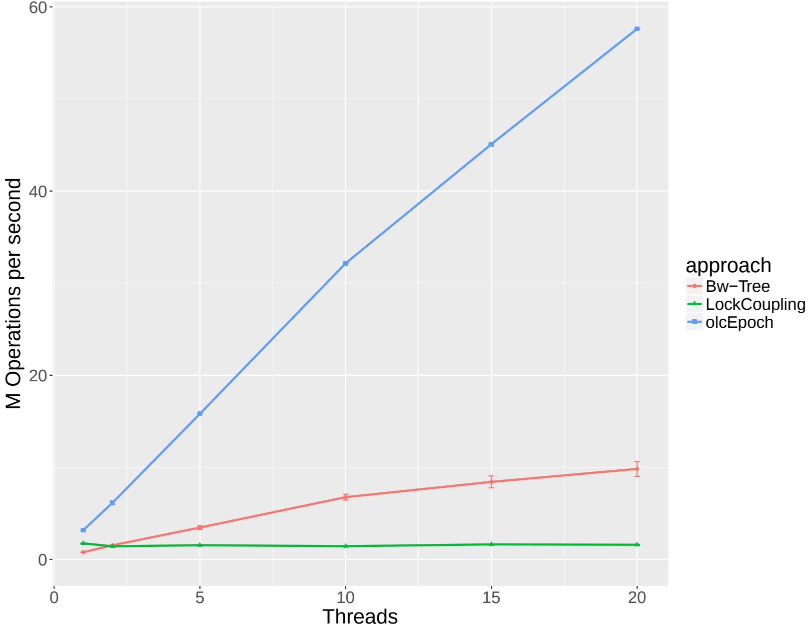
- Versioned write locks
- Writers acquire locks as usual
- Readers traverse the tree optimistically without acquiring any locks
- Validate the version after each page access
- If validation fails, restart

Insert



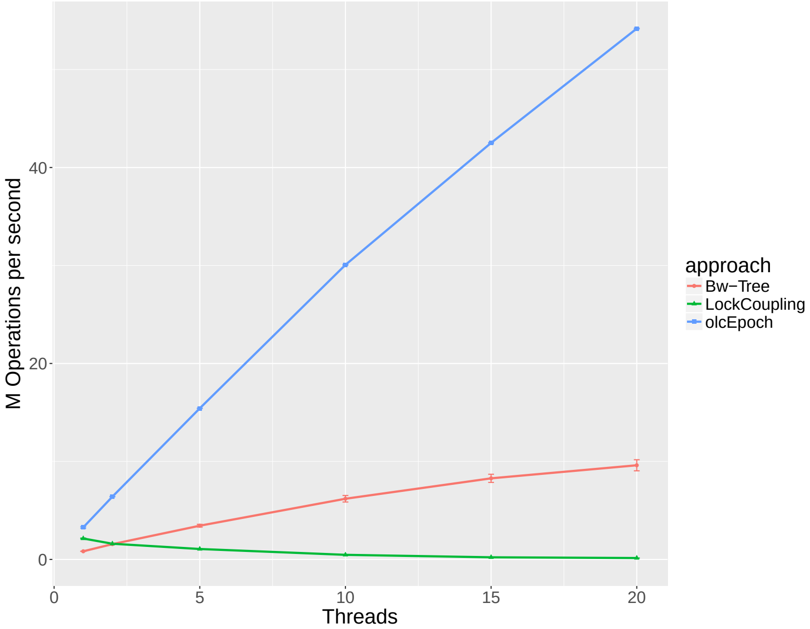
M Operations per second
0
3
6
9
Threads
0
5
10
15
20
approach
Bw−Tree
LockCoupling
olcEpoch

Lookup



M Operations per second
60
40
20
0
0
5
10
15
20
Threads
approach
Bw−Tree
LockCoupling
olcEpoch

Insert + Lookup

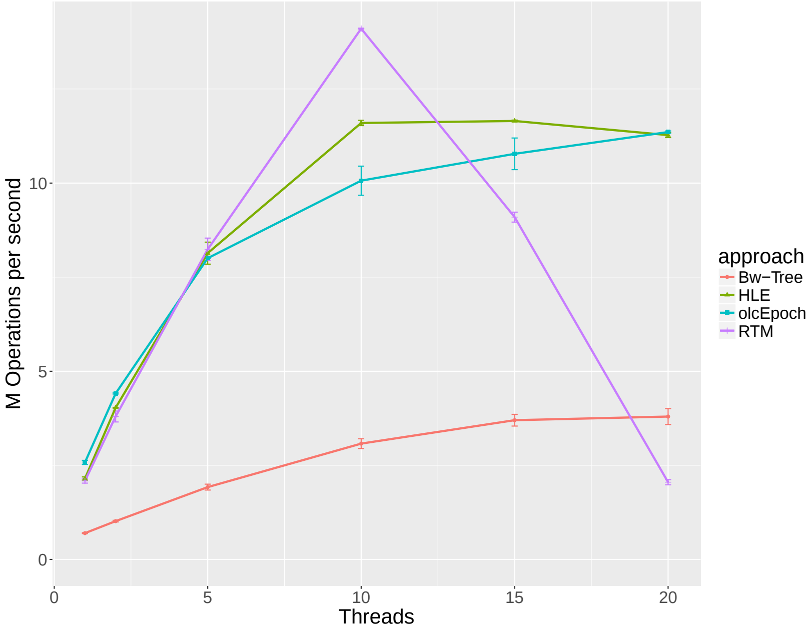


Another Alternative

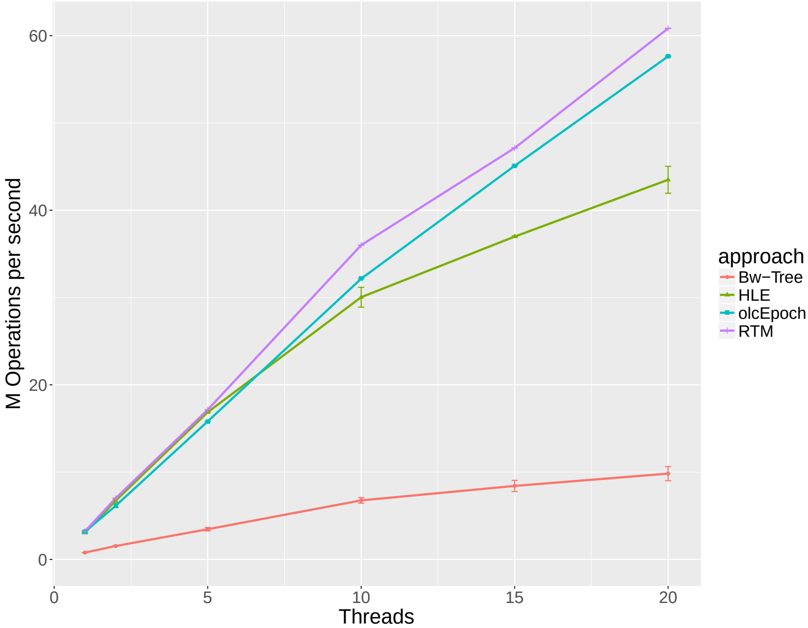
Modern Intel CPUs provide transactional memory support:

- Hardware Lock Elision (HLE)
- Restricted Transactional Memory (RTM)

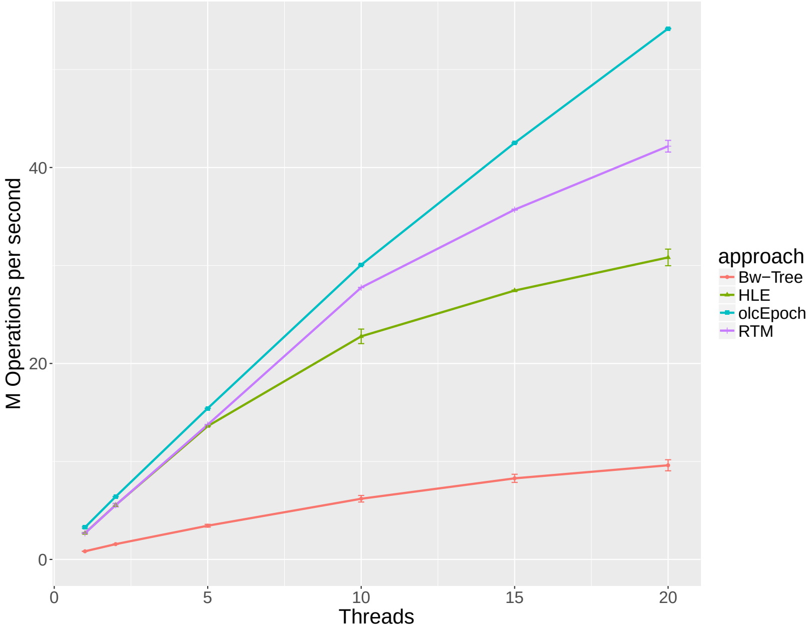
Insert


M Operations per second
0
5
10
0
5
10
15
20
Threads
approach
Bw−Tree
HLE
olcEpoch
RTM

Lookup


M Operations per second
0
20
40
60
0
5
10
15
20
Threads
approach
Bw−Tree
HLE
olcEpoch
RTM

Insert + Lookup



Further Reading

- Justin J. Levandoski, David B. Lomet and Sudipta Sengupta. The Bw-Tree: A B-tree for New Hardware Platforms. IEEE 29th International Conference on Data Engineering (ICDE), 2013.

- Philip L. Lehman and S. Bing Yao. Efficient Locking for Concurrent Operations on B-Trees. ACM Transactions on Database Systems, Vol. 6, No. 4, December 1981, Pages 650-670.

- Viktor Leis, Florian Scheibner, Alfons Kemper and Thomas Neumann. The ART of Practical Synchronization. Twelfth International Workshop on Data Management on New Hardware, 2016.