# Contention and Space Management in B-Trees

Christoph Rotte

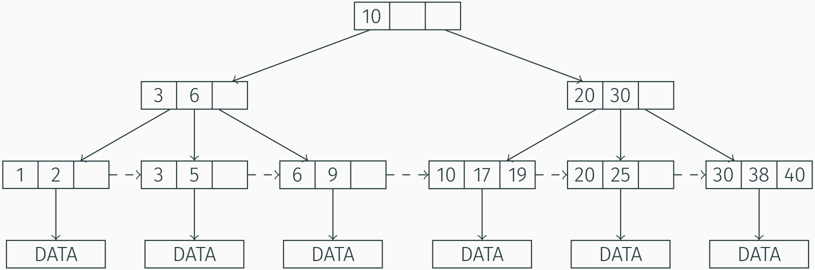
22 November 2021

Seminar: Implementation Techniques for Main Memory Database Systems

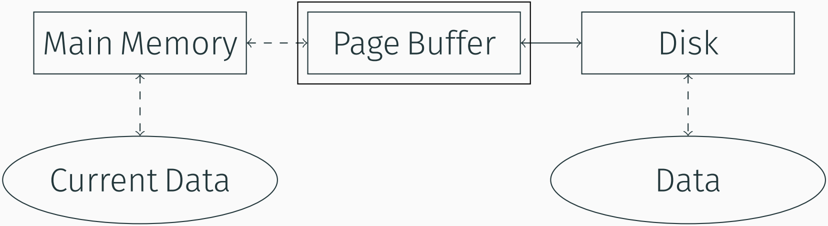
# Contention and Space Management in B-Trees

- Paper by Adnan Alhomssi and Viktor Leis (2021)
- Two techniques counteracting contention and page evictions (in B-Tree environments)

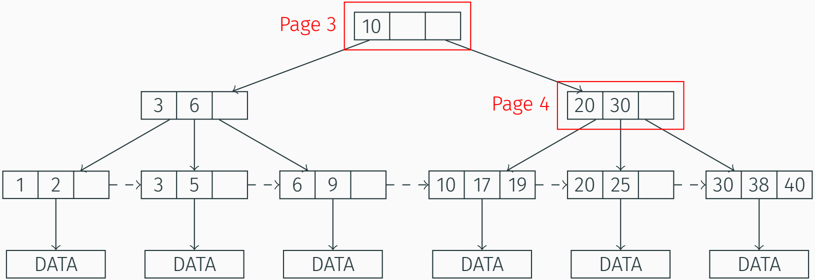
# B-Trees in Databases



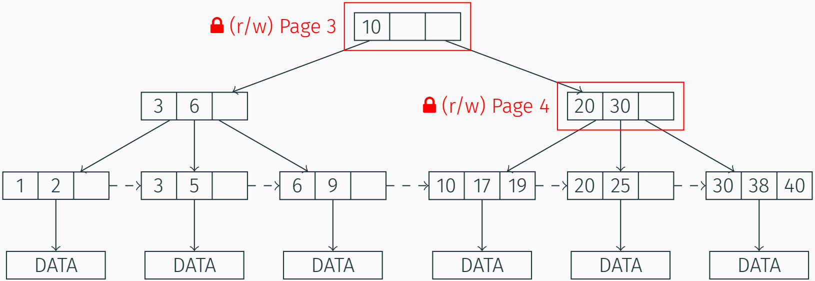
# B-Trees in Databases



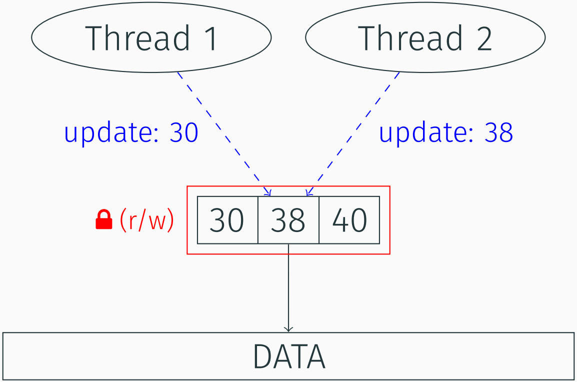
# B-Trees in Databases



# B-Trees in Databases



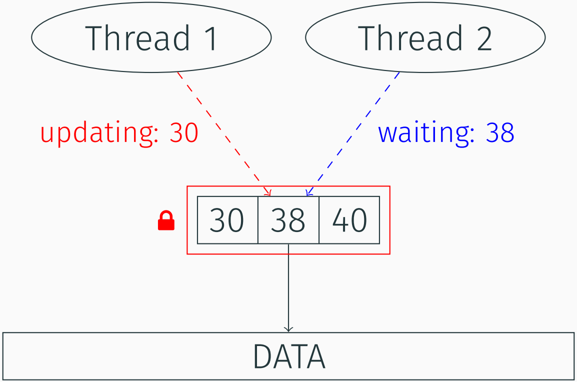
# Node Contention



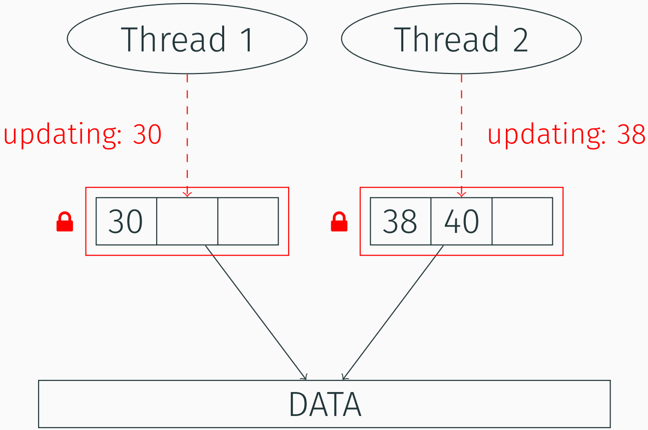
# Contention Split

```
procedure POST_UPDATE ($page, update\_index, waited$)
    $last\_update \leftarrow page.last\_index$
    $r \leftarrow random(0.0, 1.0)$
    if $r < sample\_prob$ then
        Update $update\_count, last\_index, wait\_count$ on $page$
    end if
    if $r < period\_prob$ then  # $period\_prob < sample\_prob$
        if $page.wait\_times \approx page.update\_times$ then
            Split $page.node$ at mid($update\_index$, $last\_update$)
            Reset $update\_count, last\_index, wait\_count$ on $page$
        end if
    end if
end procedure
```

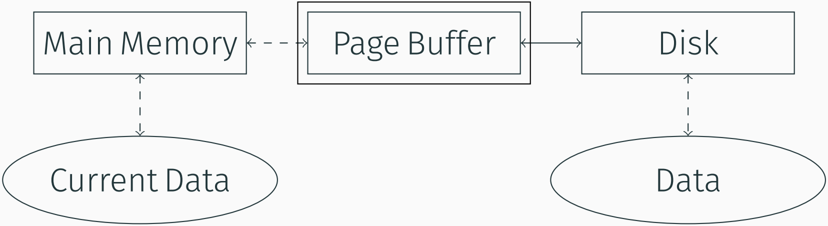
# Contention Split



# Contention Split



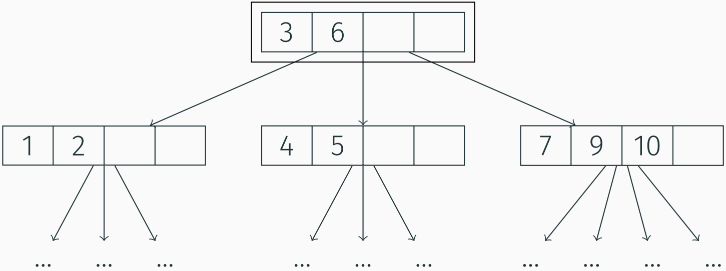
# Page Evictions



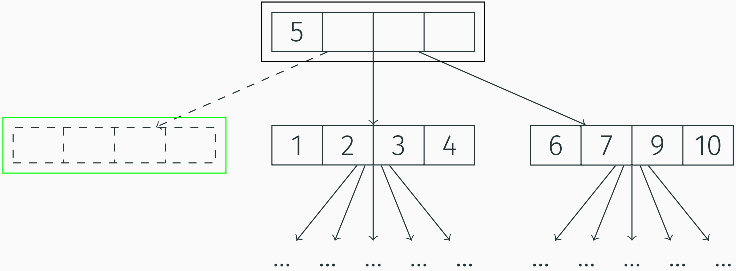
# X-Merge

```
procedure PRE_EVICTION ($requested\_id$)
    $node \leftarrow random\_inner\_node()$
    if not $is\_qualified(node)$ then
        return
    end if
    $start\_index \leftarrow random\_index(node)$
    $i \leftarrow 0$
    $space \leftarrow 0$
    while $i < max\_nodes$ and $space < node\_size$ do
        $space \leftarrow space + node.child[start\_index + i].free\_space$
        $i \leftarrow i + 1$
    end while
    if $space >= node\_size$ then
        $merge\_children(start\_index, start\_index + i)$
        $load\_page(node.child[start\_index], requested\_id)$
    end if
end procedure
```

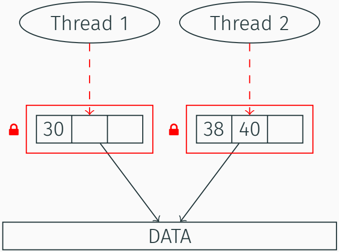
# X-Merge



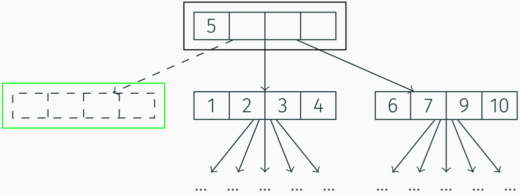
# X-Merge



# Contention Split vs. X-Merge

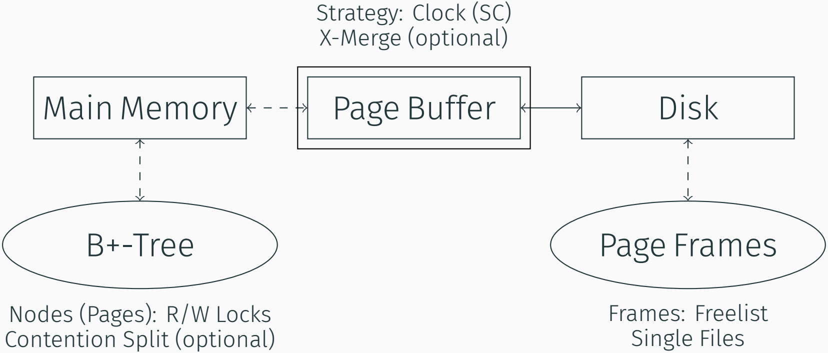


Contention Split



X-Merge

# Implementation


Strategy: Clock (SC)
X-Merge (optional)

Main Memory

Page Buffer

Disk

B+-Tree

Page Frames

Nodes (Pages): R/W Locks
Contention Split (optional)

Frames: Freelist
Single Files

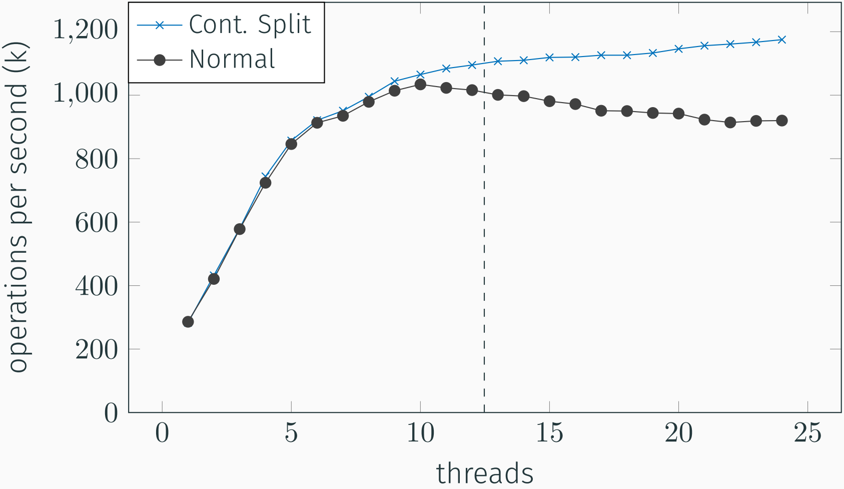
# Yahoo Cloud Serving Benchmark (YCSB)

- Different workloads (with different distributions)
- Possible operations: insert, delete, read, write, update, scan
- Loading phase ↔ operation phase
- Entries: key → tuple ($10 \cdot 100$ bytes)
- YCSB-cpp: implementation of YCSB in C++

# Evaluation Setup | Contention Split

- AMD Ryzen 5 2600X (12 threads) | Samsung SSD 860 EVO
- Page size: 4 KiB
- **Buffer**: holds all pages in memory
- 10M loaded entries + 100M operations
- **Workload**: 20% reads, 80% updates
- **Distribution**: Zipfian
- **Parameters**: $sample\_prob$ = 0.5% | $period\_prob$ = 0.05%

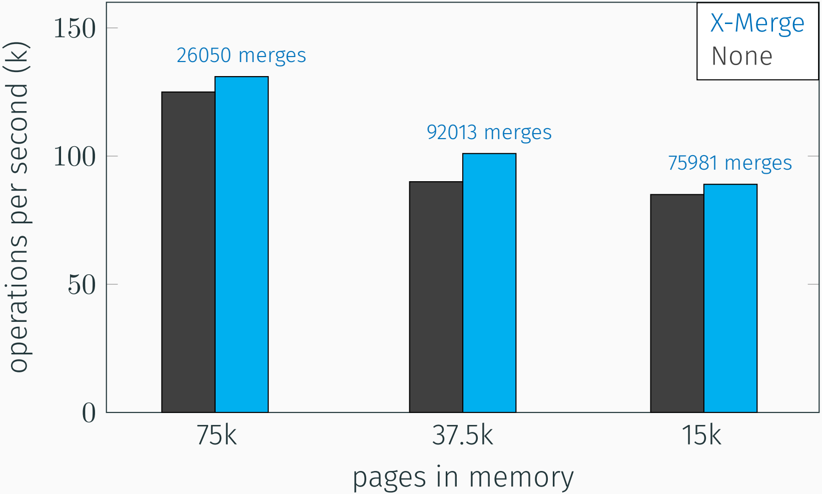
# Evaluation | Contention Split



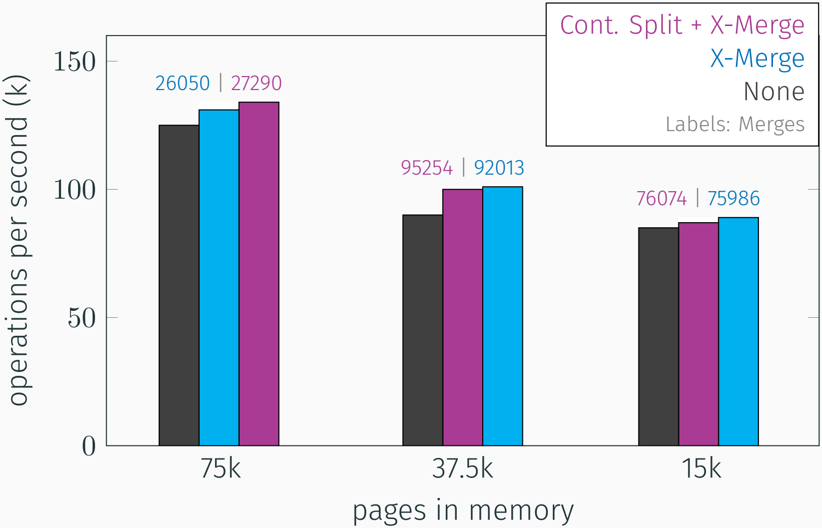
# Evaluation Setup | X-Merge

- AMD Ryzen 5 2600X (12 threads) | Samsung SSD 860 EVO
- Page size: 4 KiB
- **Buffer**: limited memory
- 10M loaded entries + 100M operations
- **Workload**: 30% reads, 60% updates, 10% inserts
- **Distribution**: Zipfian
- **Parameter**: $max\_nodes$ = 5
- Threads: 10

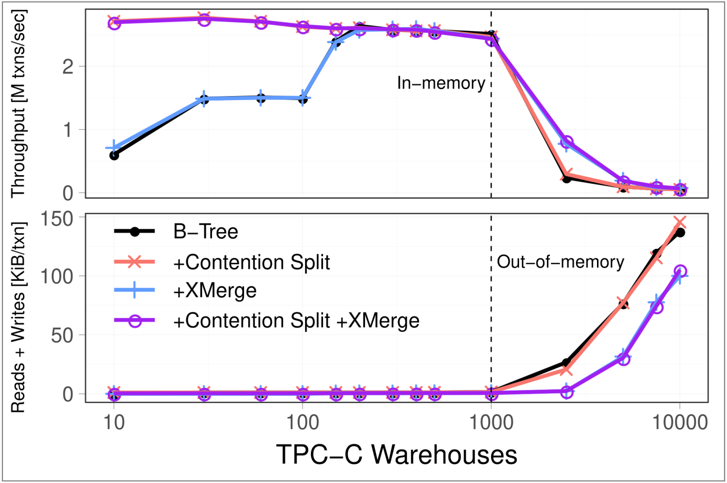
# Evaluation | X-Merge



# Evaluation | Combined

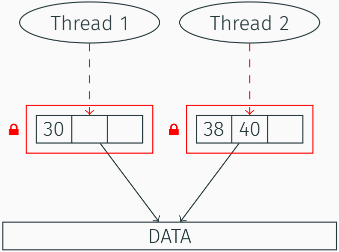


# Contention Split and X-Merge in Leanstore

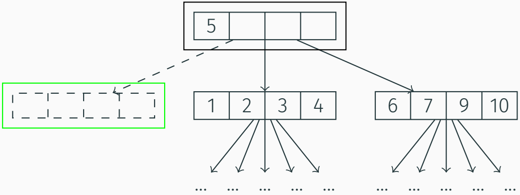


Leanstore: TPC-C | buffer=240GiB | workers=120 (Alhomssi & Leis)

# Conclusion



Contention Split



X-Merge