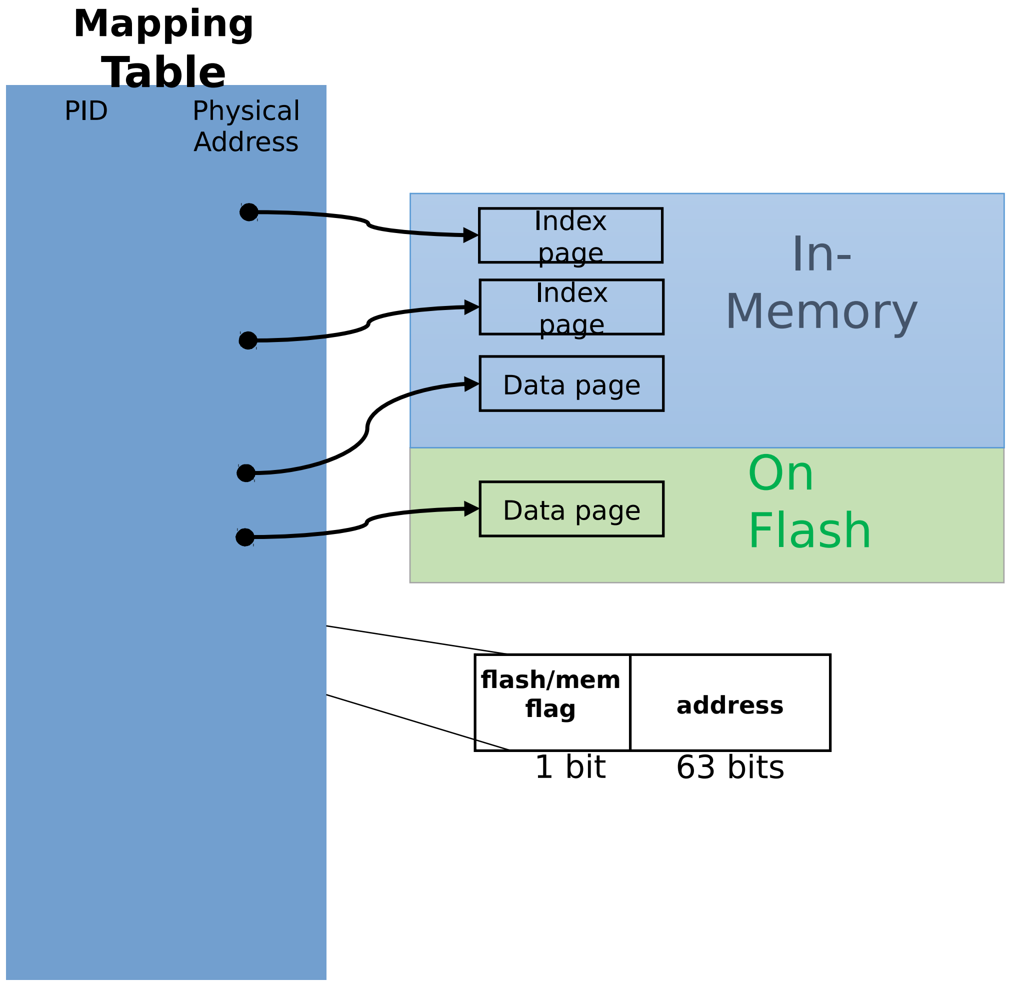
BW-Tree Discussant

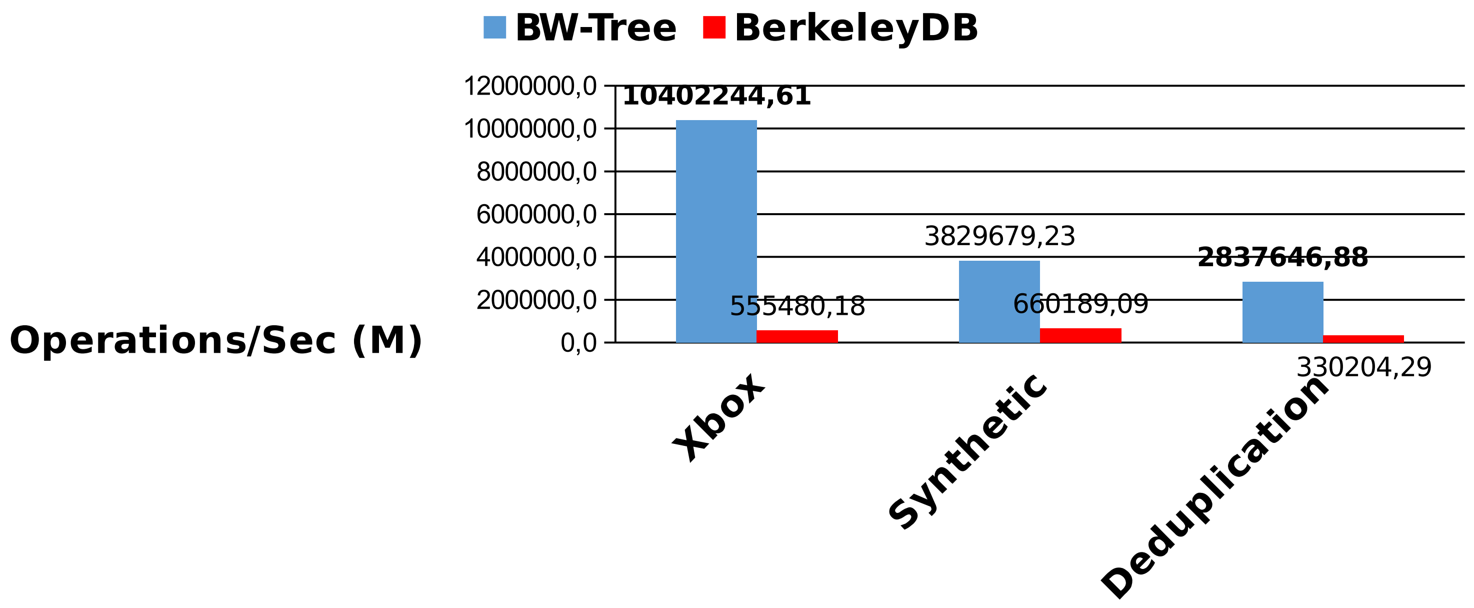


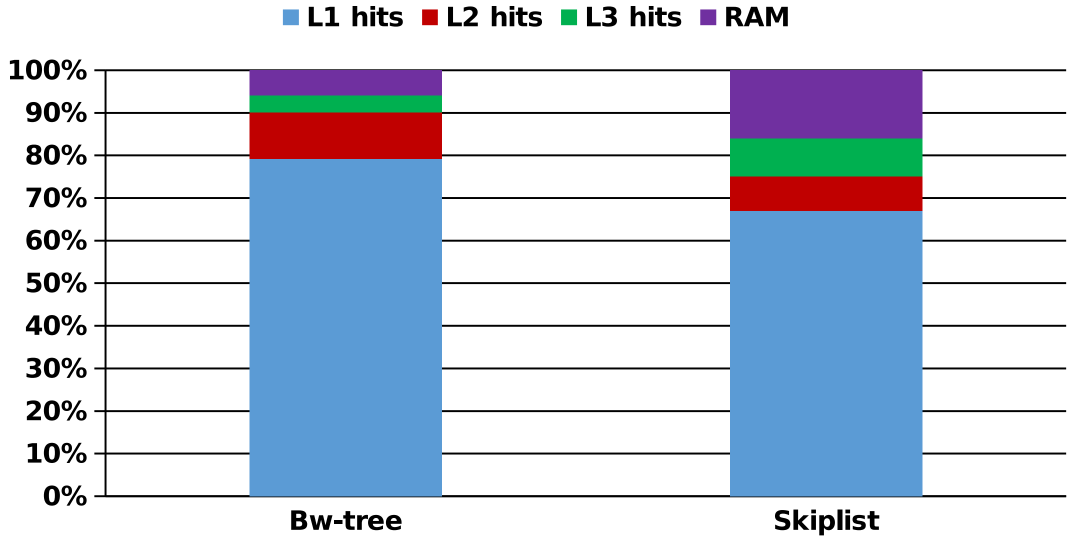
Andrew Osgood
CS2270

Quick Overview of the BW-Tree

- Flash Storage
- Latch-Free Threading
- “Delta” updates

Hard to argue against the numbers





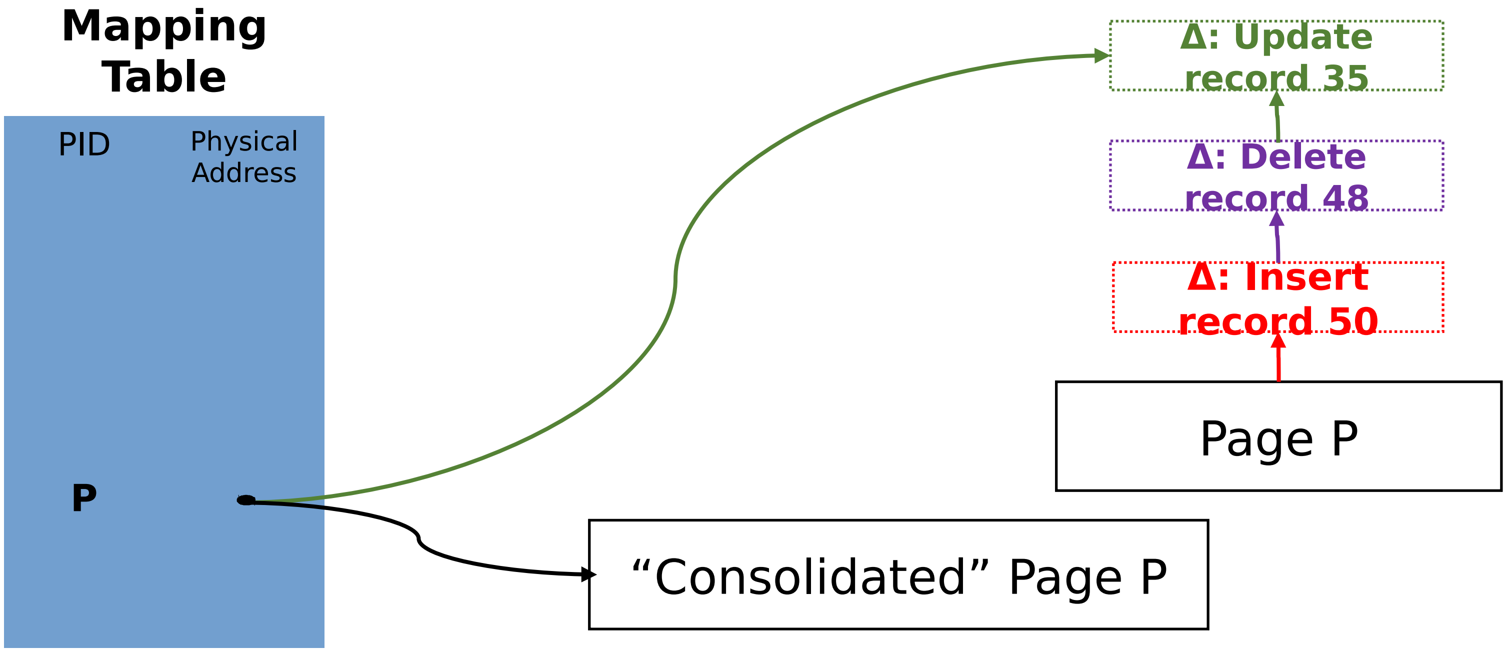
	Bw-Tree	Skiplist
Synthetic workload	3.83M Ops/Sec	1.02 M Ops/Sec

Considerations for Flash Storage

- Fast, effectively no seeks (nothing comparable to a disk)
- **However :**
  - erase-cycles prevent in-place updates
  - Erasing data after a while will cause storage to become compromised/corrupted

BW-Tree's response

- “Delta” updates
  - Solve some problems: cause “delta chains”

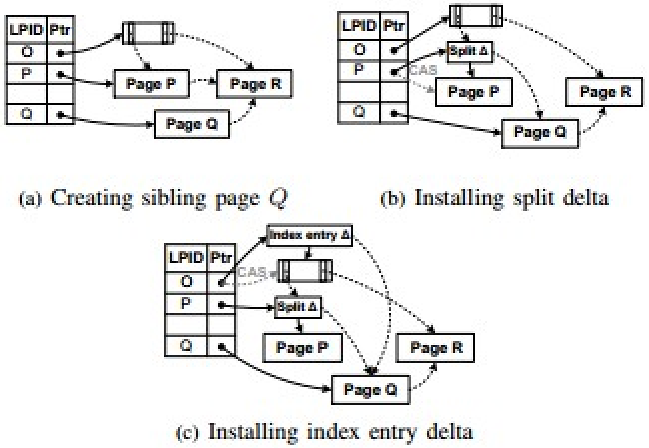


  - Pages then consolidated:

    takes time/unaddressed multi-threading issues

Structure Modification Operations(SMOs)

- Seen in typical B-Tree
  - Nodes merging/splitting
  - But Latch-Free for BW-Tree
- Problems
  - Merging parent problem
  - SMO stack for threads



(a) Creating sibling page Q

(b) Installing split delta

(c) Installing index entry delta

Other remarks

- Range scans :
  - “So before delivering a record from our vector, we check whether an update has affected the yet unreturned subrange” [1]
  - necessitates previous checks
- EPOCH Concurrency not explained:
  - Addressed in full paper but only alluded to here
- Assumption of Logical Concurrency control, yet presented as an autonomous unit
- Non-contiguous reads

Questions ?

References

- 1. J. J. Levandoski, D. B. Lomet, and S. Sengupta. “The Bw-Tree: A B-tree for New Hardware Platforms.” In *ICDE*, 2013.

- 2. J. J. Levandoski, S. Sengupta. “The Bw-Tree: A Latch-Free B-Tree for Log Structured Flash Storage.” In *Bullentin of the Techincal Committee on Data Engineering,* 2013

- All figures and images taken from these papers and the slides:
  - *The Bw-Tree: A B-tree for New Hardware Platforms*

    by J. Levandoski, D.B. Lomet, S. Sengupta