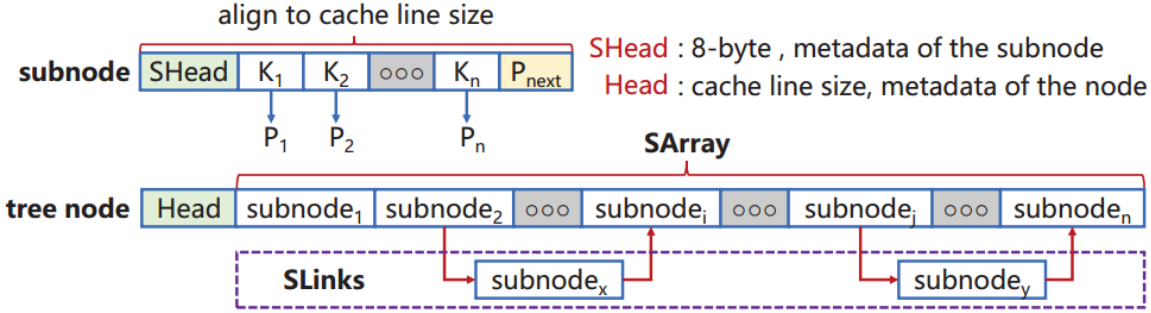
WOBTree: a write-optimized B+-tree for non-volatile memory

Haitao WANG, Zhanhuai LI, Xiao ZHANG, Xiaonan ZHAO, Song JIANG

Frontiers of Computer Science, DOI: 10.1007/s11704-020-0228-1

Problems & Ideas

- Problems of B+-tree on non-volatile memory (NVM)
  - Atomic granularity mismatch between CPU cache (usually 64B cache line) and NVM (usually 8 bytes failure-atomicity) can introduce write amplification and compromise data consistency of B+-trees
  - To ensure data consistency, a conventional B+-tree needs to flush half of the whole tree node on average even only one key-value pair is inserted or deleted, which increases write amplification and degrades write performance
- Ideas: A Write-Optimized B+-Tree for Non-Volatile Memory
  - Minimize the update granularity from a tree node to a smaller subnode
  - Carefully arranges the write operations in subnodes to ensure crash consistency and reduce write amplification



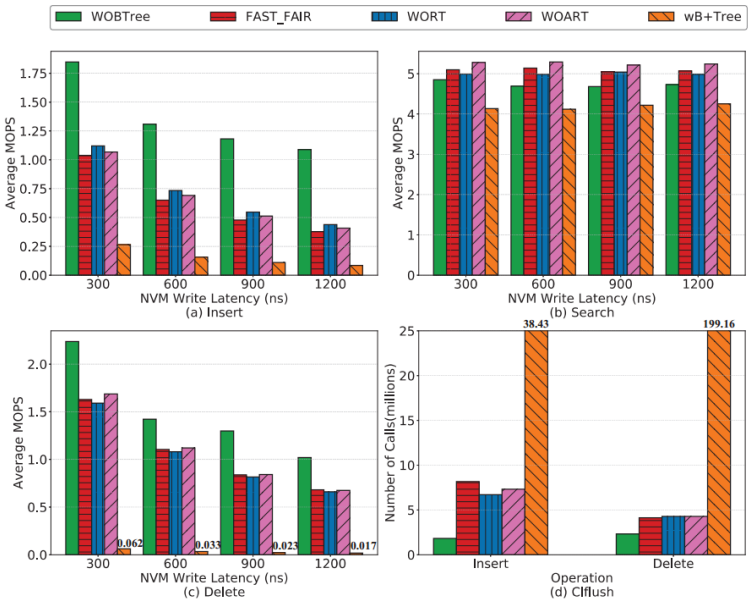
Main Contributions

- **WOBTree significantly reduce the write amplification**

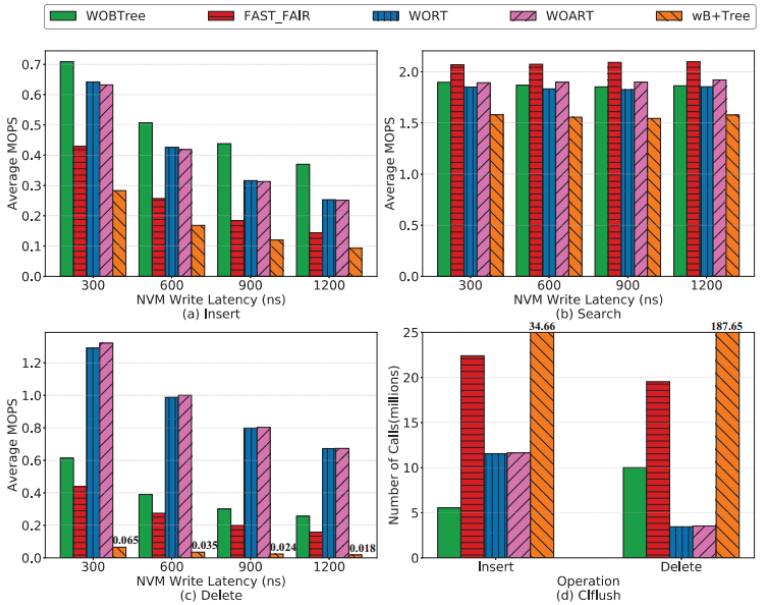
  - FAST_FAIR and wB+Tree are two B+-Tree variants on NVM
  - WORT and WOART are two radix tree variants on NVM

workload	WOBTree	FAST_FAIR	WORT	WOART	wB+Tree
Zipfian Insert	1.46	6.54	5.37	5.85	30.75
Zipfian Delete	1.85	3.31	3.42	3.42	159.33
Uniform Insert	4.43	17.92	9.24	9.31	27.72
Uniform Delete	7.99	15.63	2.76	2.82	150.11

- **WOBTree largely improve the write performance (Million Operations Per Second)**



Performance comparison under the Zipfian workload.



Performance comparison under the Uniform workload.