LB+-Trees: Optimizing Persistent Index Performance on 3DXPoint Memory

Jihang Liu, Shimin Chen*  Lujun Wang

Institute of Computing Technology  Alibaba Group
Chinese Academy of Sciences
Non-Volatile Memory

- Multiple competing technologies
  - PCM, STT-RAM, Memristor, 3DXPoint memory

- 3DXPoint (Intel Optane DC Persistent Memory)
  - 2015, Intel & Micron announced 3DXPoint
  - 2017, Optane SSD products based on 3DXPoint
  - 2019.4, 3DXPoint memory products

- Up to 6TB in a dual-socket server
  - App Direct Mode
  - PMDK to map NVM to virtual address space
Motivation

• 3DXPoint Characteristics
  - 3DXPoint 2-3x slower than DRAM
  - 256B internal data transfer size
  - Different write content: NO impact on performance
  - Persist: can be 10x slower than normal writes
    - CPU cache is volatile
    - Clwb + sfence to flush data to NVM

♫ Our goal: B+-tree on 3DXPoint memory
  - Exploit characteristics of real NVM hardware
  - Focus on insertion performance

3DXPoint performance studies:
  “Initial Experience with 3D XPoint Main Memory”. HardBD & Active workshop, ICDE 2019
  “An Empirical Guide to the Behavior and Use of Scalable Persistent Memory”. FAST 2020
LB+-Tree with 256B Nodes

**3DXPoint**

**leaf node**

- Line 0
- Line 1
- Line 2
- Line 3

- 16B header
- 14 (8B key, 8B val) entries
- 14x1B fingerprints
- lock alt 14bit bitmap

**non-leaf node**

**DRAM**
Insertion Optimization (1)

Entry Moving

Take this opportunity to make empty slots in Line 0
Insertion Optimization (1)

Entry Moving

Line 0 | Line 1 | Line 2 | Line 3
---|---|---|---
9 4 | 1 | 5

Insert 6

9 6 4 | 1 | 5

1 NVM line write

Insert 3

3 9 6 4 | 1 | 5

2 NVM line writes

Insert 7

7 | 3 9 6 4 | 5

Entry moving

1 NVM line write
Insertion Optimization (2)

Logless Node Split

Insert c

Split step 1

Split step 2
Experiments

- **Bulkloading**
  - 70% or 100% full
  - 2 billion (8B key, 8B ptr) entries
  - Over 1/8 NVM capacity

- **Test**
  - Random insertions
  - Dense insertions

- **1.12-2.92x improvements over existing NVM optimized trees**
• LB+-Tree significantly better than skiplist
  □ 1.25—1.83x improvements
More Details in the Paper

- LB+-Tree with multi-256B nodes
- Search, insert, delete algorithms
- Theoretical proof for entry moving benefit
- Extensive performance results
Conclusion

• **NVM is here!**

• **NVM has different characteristics from DRAM**
  - Much larger capacity (up to 6TB for a dual-socket server)
  - 2-3x slower than DRAM
  - Large persist cost

• **LB+-Tree: a promising solution**
  - Similar read performance
  - Much better write performance

https://github.com/schencoding/lbtree
Thank you!