Evaluating Persistent Memory Range Indexes: Part Two

Yuliang He, **Duo Lu**, Kaisong Huang, Tianzheng Wang - *Simon Fraser University*

SFU SIMON FRASER UNIVERSITY

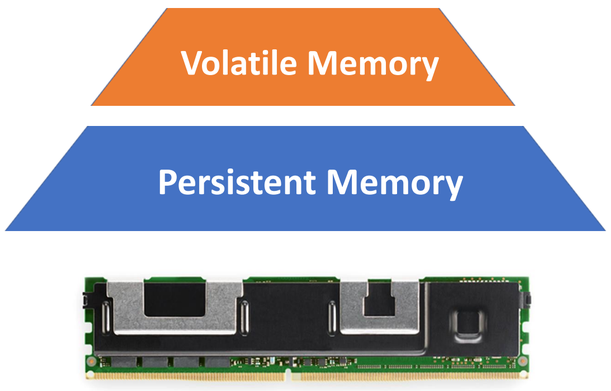
GitHub Repo: `https://github.com/sfu-dis/pibench-ep2`

What? Benchmark and evaluate Optane-era **Persistent Memory (PM)** range **indexes**

Why? Unclear performance they achieve on **real PM** hardware (Intel Optane DCPMM)

How? Utilize **PiBench*** to experiment on **eight** range indexes under various workloads

Persistent Memory (PM)



Key features:

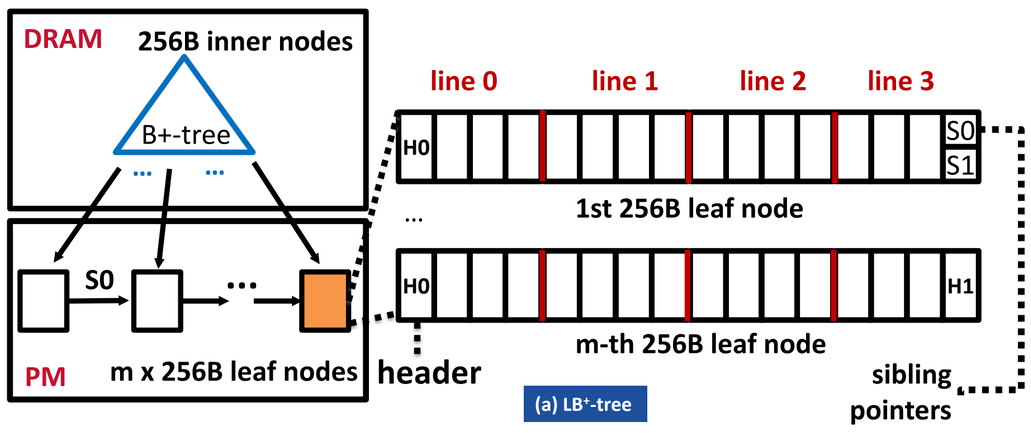
- Byte-addressability
- Near DRAM latency
- Non-volatile
- Large capacity
- Cheaper than DRAM

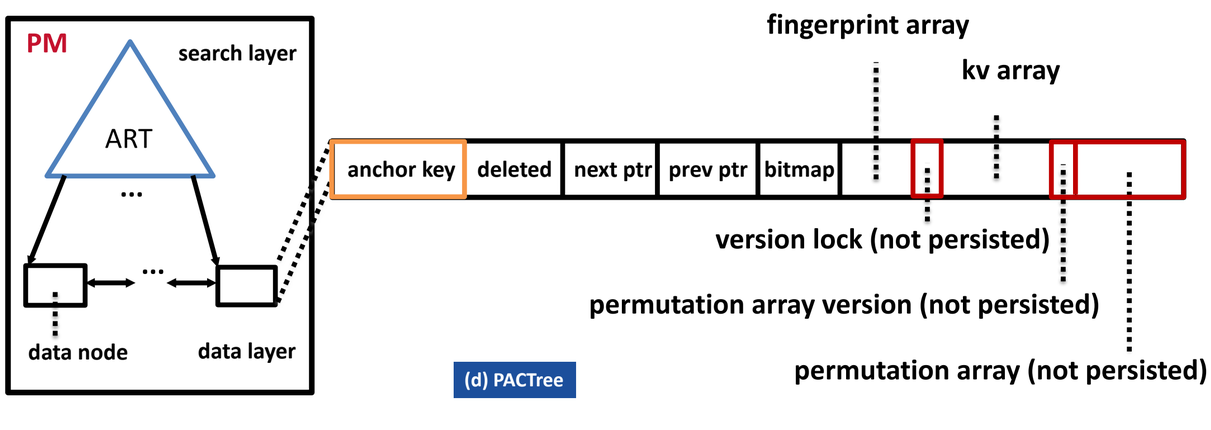
PM Partners

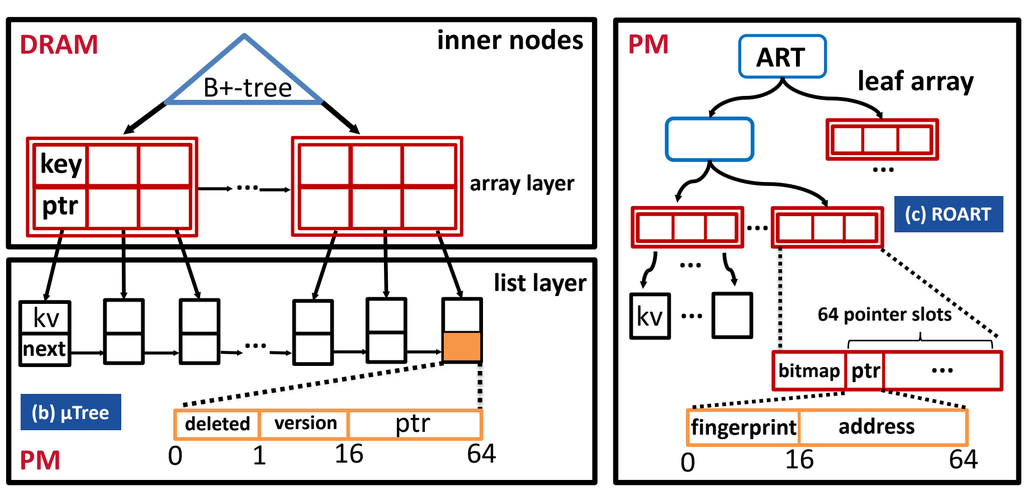
Hardware & Software Configuration

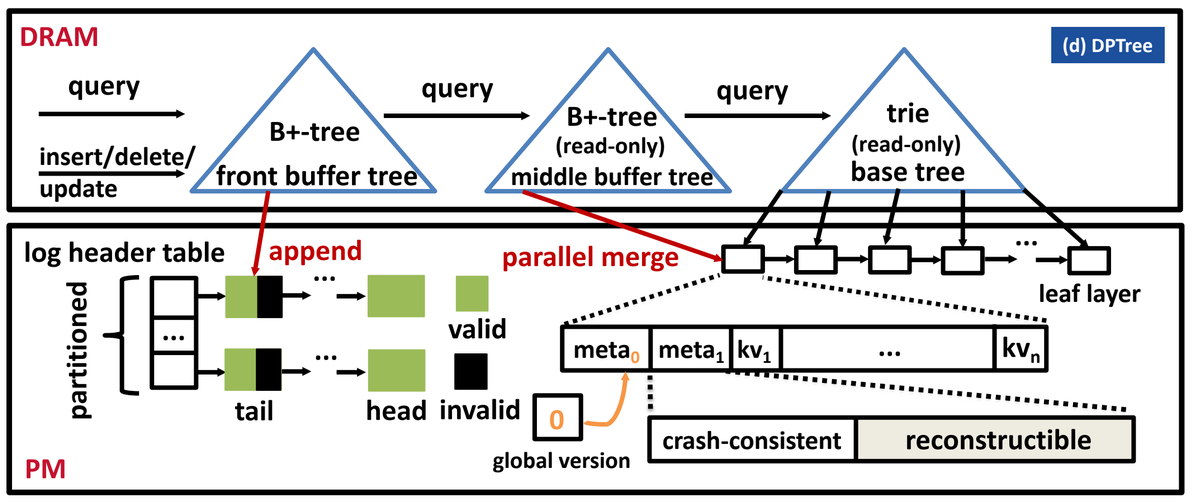
- 2 x 20-core (2-socket, 80-hyperthread) Intel Xeon Gold 6242R clocked at 3.10 GHz, 12 x 32GB DRAM (384GB), **12 x 128GB DCPMM** (1.5TB)
- Arch Linux kernel 5.14.9, GCC 11.1, `glibc` 2.34
- Allocators: `jemalloc` for DRAM, PMDK for PM
- **PiBench**: PM indexes benchmark framework
  - 8-byte key-value pair
  - Preload 1M keys
  - 10s of operations each run

State-Of-The-Art Persistent Memory Range Indexes

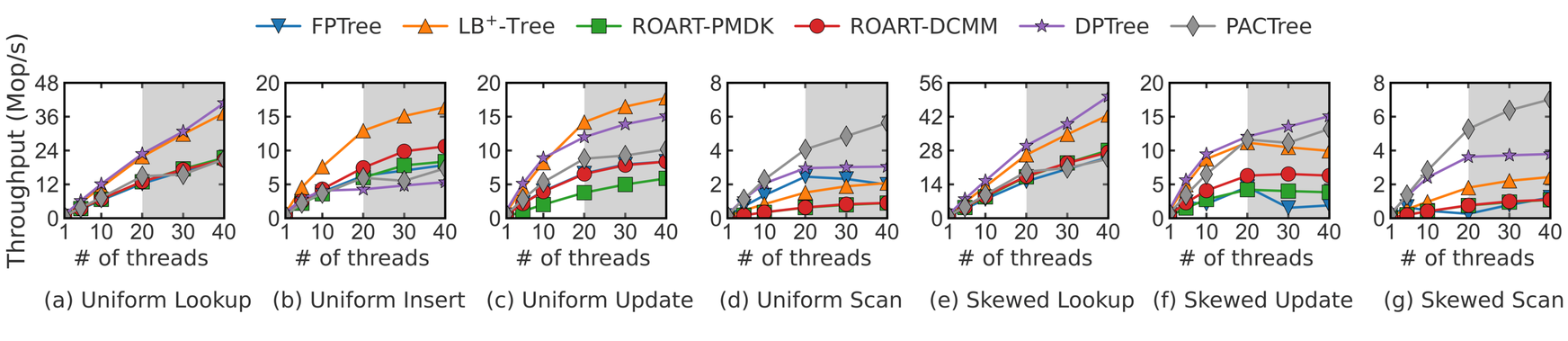


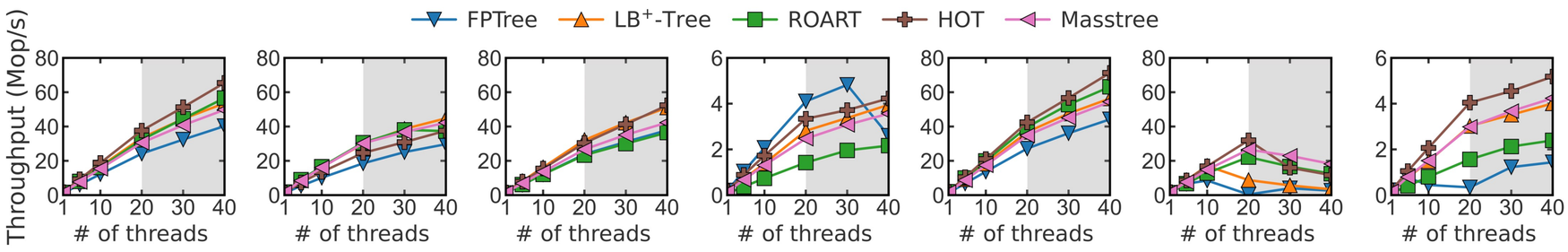






Single/Multi-Threaded Experiment Under Various Workloads





Throughput under uniform (a–d) and skewed (e–g, self-similar with 80% accesses on 20% of keys) distributions

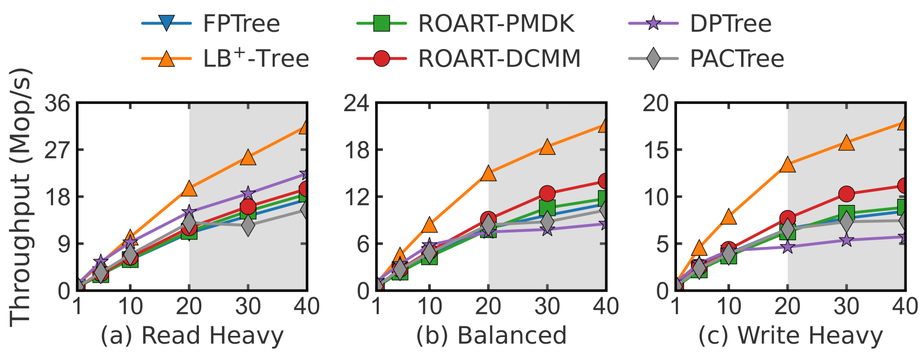
Key takeaway:

- DPTree and LB+-Tree achieve best performance
- PM allocator matters (PMDK vs. ROART customized DCMM)
- Be careful when you use Hardware transactional Memory

Unifying PM and DRAM indexing:

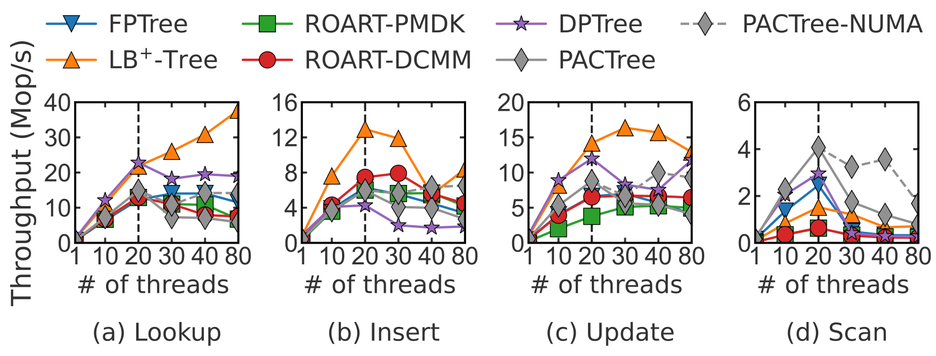
- PM indexes can also be effective for DRAM
- Compare to two representative DRAM-optimized indexes
- Techniques proposed by PM indexes may also apply to DRAM

Mixed Workload



Throughput of mixed workloads (lookups + inserts) under uniform distribution

Impact of NUMA Effect



No index scales well due to additional PM accesses by the directory-based CPU coherence protocol