TECHNISCHE UNIVERSITÄT DRESDEN

SAP®

Dresden Database Systems Group

Benchmarking Persistent Range Indexes

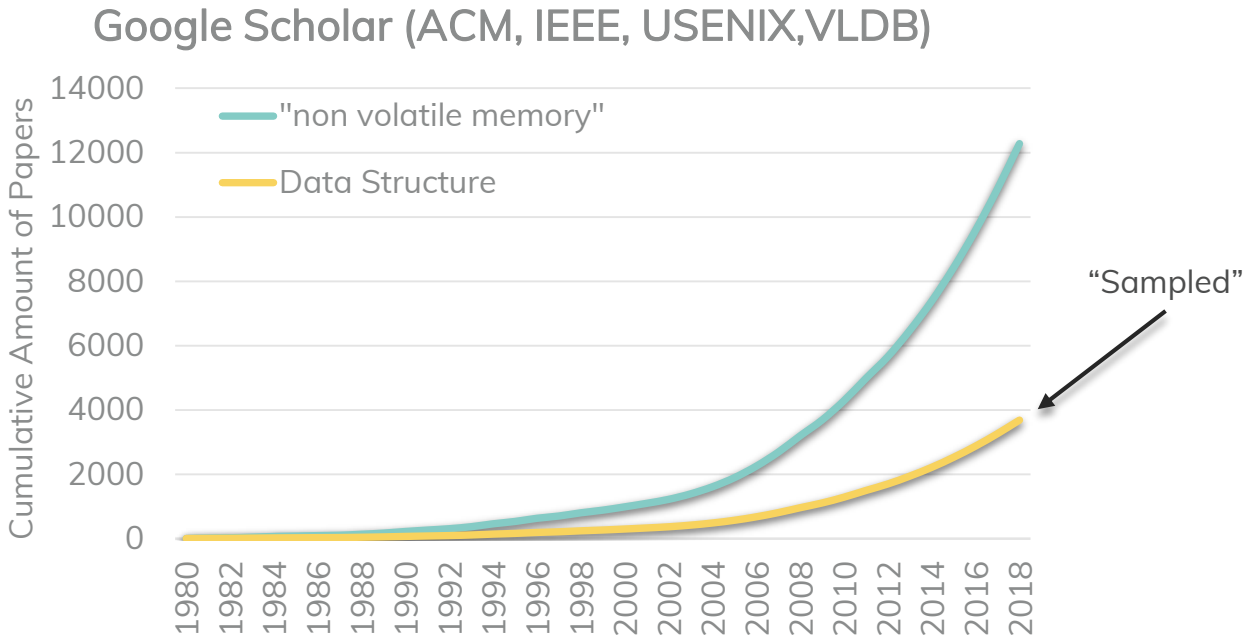
And lessons learned

Lucas Lersch
22.07.2019

Context of this talk

Dresden Database Systems Group

Many proposed persistent data structures in the past years



Most work so far

Dresden Database Systems Group

Hardware was not widely available

- DRAM (tmpfs, ramdisk)
- DRAM-based emulation (higher latencies): Intel PMEP[1], HP Quartz[2], pcmsim[3], etc.
- Flash-backed DRAM (expensive)

Different programming models

Different benchmark methodologies

"Evaluating Persistent Memory based Range Indexes"

- Joint work with
  - Xiangpeng Hao and Tianzheng Wang (Simon Fraser University, Canada)
  - Ismail Oukid (SAP SE, Germany)
  - Thomas Willhalm (Intel, Germany)

[1] Dulloor, S.R., Kumar, S., Keshavamurthy, A., Lantz, P., Reddy, D., Sankaran, R. and Jackson, J. System software for persistent memory. EuroSys'14
[2] Volos, H., Magalhaes, G., Cherkasova, L. and Li, J. Quartz: A lightweight performance emulator for persistent memory software. Middleware'15
[3] https://code.google.com/archive/p/pcmsim/

Work overview

Dresden Database Systems Group

Pick representatives that cover the design space

Index	Architecture	Node structure	Concurrency
wBTree[1]	PM-only	Unsorted; indirection sorted array	Single-threaded
NV-Tree[2]	PM-only (optionally hybrid)	Unsorted leaves; inconsistent inner nodes	Locking
BzTree[3]	PM-only	Partially sorted leaves; sorted inner nodes	Lock-free (PMwCAS[5])
FPTree[4]	Hybrid DRAM+PM	Unsorted leaves; sorted inner nodes	Selective (HTM+locking)

Unified implementation using PMDK

Benchmark on real persistent memory

- Intel Optane DC Persistent Memory Modules (DCPMM)

[1] S. Chen and Q. Jin. Persistent B+-trees in non-volatile main memory. VLDB'15
[2] J. Yang, Q. Wei, C. Wang, C. Chen, K. Yong, and B. He. NV-Tree: A consistent and workload-adaptive tree structure for non-volatile memory. IEEE TC'15
[3] J. Arulraj, J. Levandoski, U. F. Minhas, and P.-A. Larson. BzTree: A high-performance latch-free range index for non-volatile memory. VLDB'18
[4] I. Oukid, J. Lasperas, A. Nica, T. Willhalm, and W. Lehner. FPTree: a hybrid SCM-DRAM persistent and concurrent B-tree for storage class memory. SIGMOD'16
[5] T. Wang, J. Levandoski, and P. Larson. Easy lock-free indexing in non-volatile memory. ICDE'18

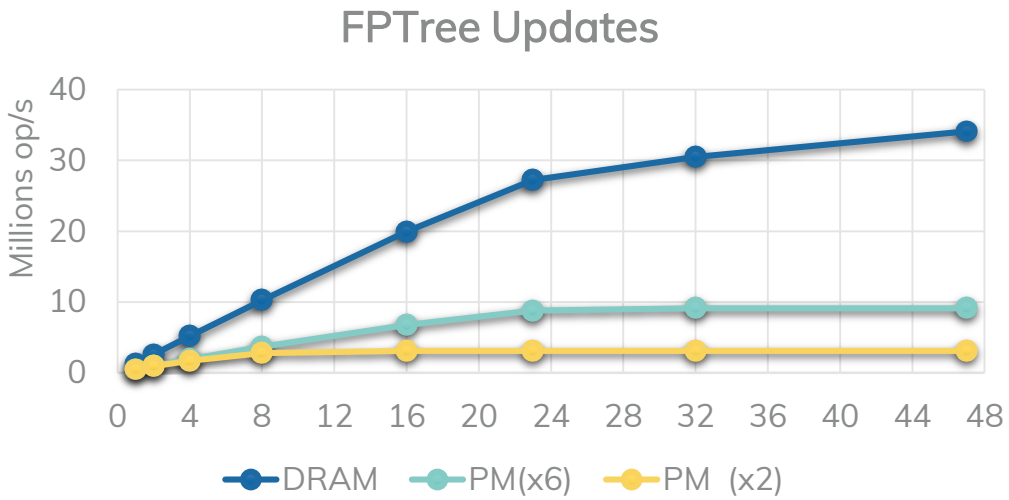
Lesson #1

Dresden Database Systems Group

"Read and write as less bytes as possible per operation"

Obvious, but…

- Most applications hit other bottlenecks (network, NUMA, etc.) before saturating DRAM bandwidth
- These applications will saturate the DCPMM bandwidth much sooner



Lesson #1 (cont.)

Dresden Database Systems Group

Bandwidth is the villain, not latency

- After latency is hidden, we still hit the bandwidth barrier[1]

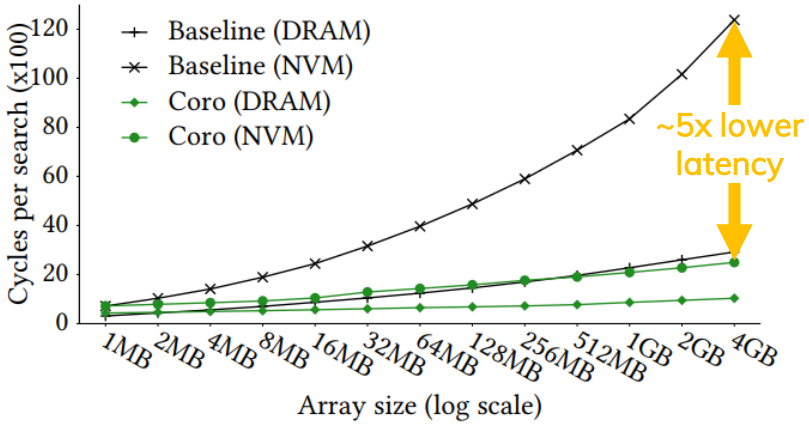


Figure 1: Binary search performance on DRAM vs NVM.

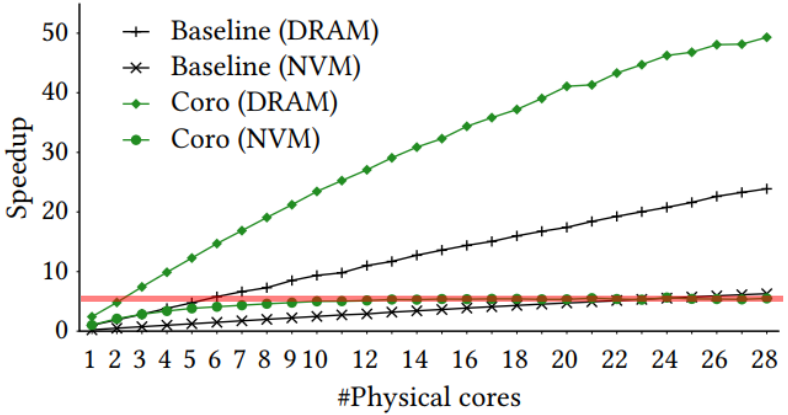


Figure 2: Scalability of binary search for a 2 GB sorted array.

[1] Psaropoulos, G., Oukid, I., Legler, T., May, N. and Ailamaki, A. Bridging the latency gap between NVM and DRAM for latency-bound operations. DaMoN'19

Lesson #2

Dresden Database Systems Group

"Carefully consider your use case before deciding for a setup"

DCPMM comes in different sizes

- 128 GB (~700 $), 256 GB (~2500 $), 512 GB (~7800 $)

Questions to ask:

- How much persistent memory do you need?
- How many modules do you need?

Example:

4x 128 GB DCPMM	1x 512 GB DCPMM
512 GB	512 GB
~2800 $	~7800 $
Higher bandwidth (4-channels)	Lower bandwidth (1-channel)
Occupies 4 slots	Occupies 1 slot (more slots for DRAM)

Lesson #2 (cont.)

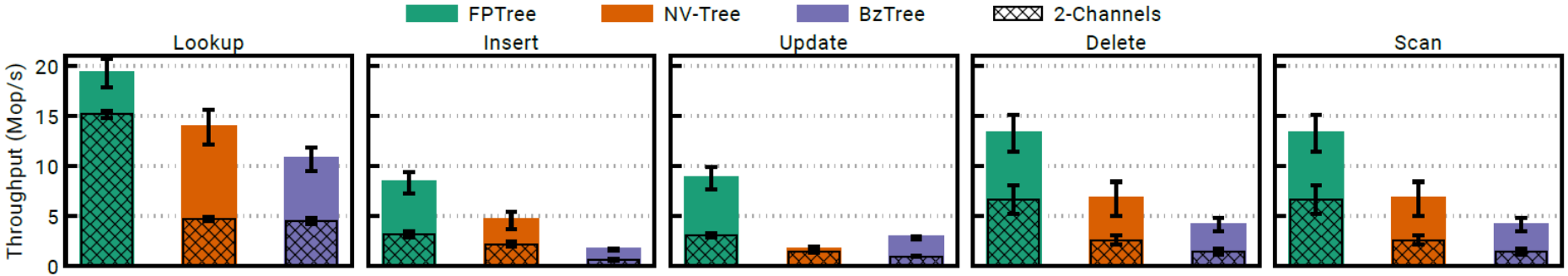
Dresden Database Systems Group

Not all data structures present the same gains

- Do not buy more DCPMM and expect (linear) improvements out-of-the-box

Workload:

- 23 threads
- Uniform distribution
- 2-channels vs. 6-channels



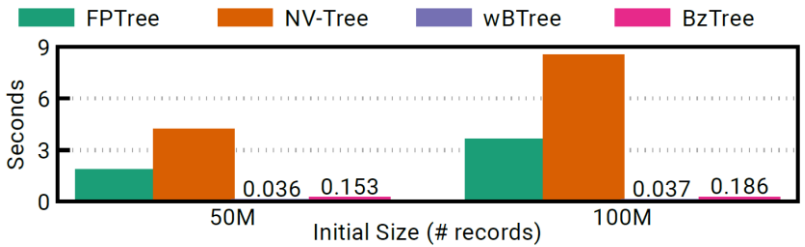
Lesson #3

Dresden Database Systems Group

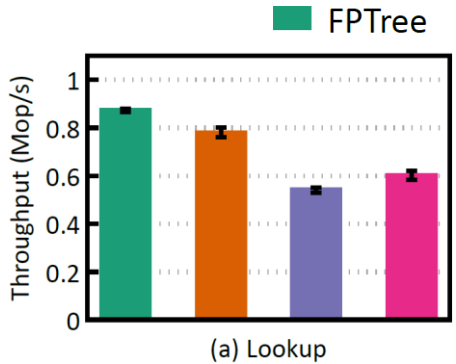
"For the time being, PM should coexist peacefully with DRAM"

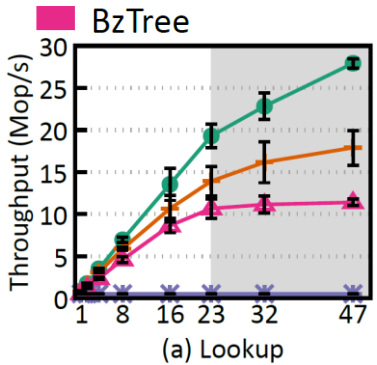
Many proposals for purely persistent data structures

Instant recovery time



But lower performance

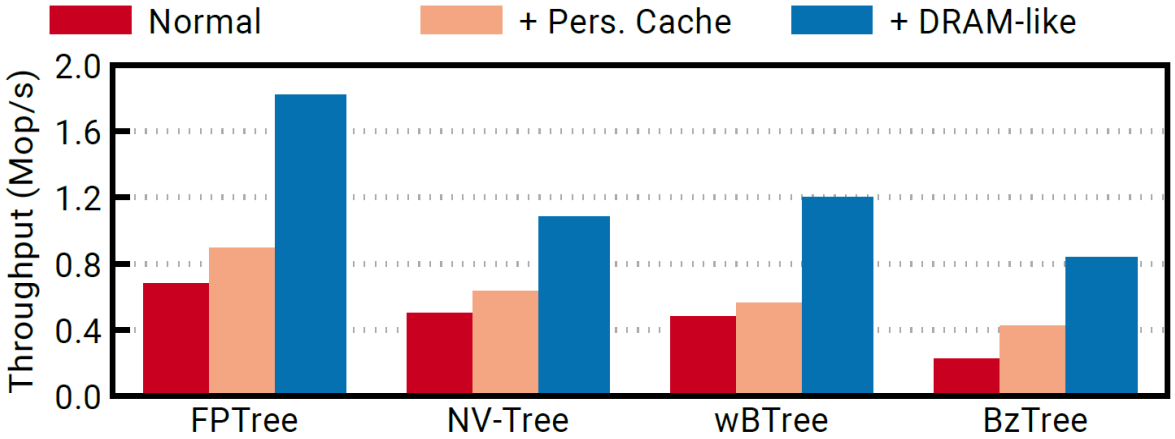




PM Wishlist

Workload

- Single thread
- 50% lookups, 25% inserts, 25% updates
- Uniform key distribution

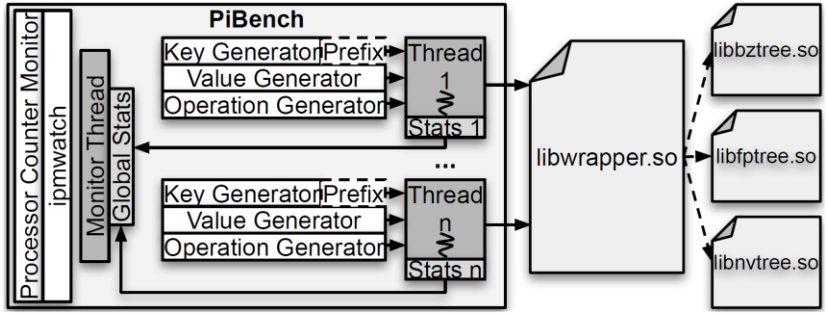


PiBench Framework

Dresden Database Systems Group

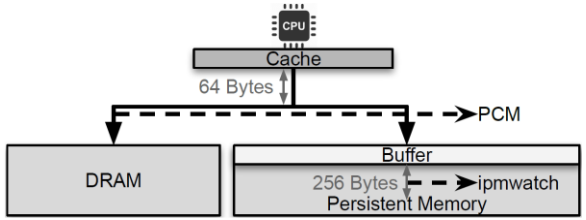
Collect metrics within a specified time-window

- Better understanding of runtime behavior



Integrate hardware analysis tools

- Processor Counter Monitor[1]
- ipmwatch



No assumptions regarding persistent memory management

- Not only PMDK

Available at: https://github.com/wangtzh/pibench

Low effort benchmark

[1] https://github.com/opcm/pcm

Conclusion

Dresden Database Systems Group

Lesson #1: ***"Read and write as less Bytes as possible per operation"***

Lesson #2: ***"Carefully consider your use case before deciding for a setup"***

Lesson #3: ***"For the time being, PM should coexist peacefully with DRAM"***

Wishlist:

- Persistent caches
- Better performance (better bandwidth)
- Open analysis tools

Thank you