Empirical Evaluation of Indexing on Modern DBMS

Shawn Zhong, Yusen Liu, Libin Zhou

Outline

Survey on Index Techniques

- Tree-Based Indexing
- Hash-Based Indexing

Evaluation

- Experiment Setup
- Scalability Evaluation
- Hash vs. B+ Tree

Tree-Based Indexing

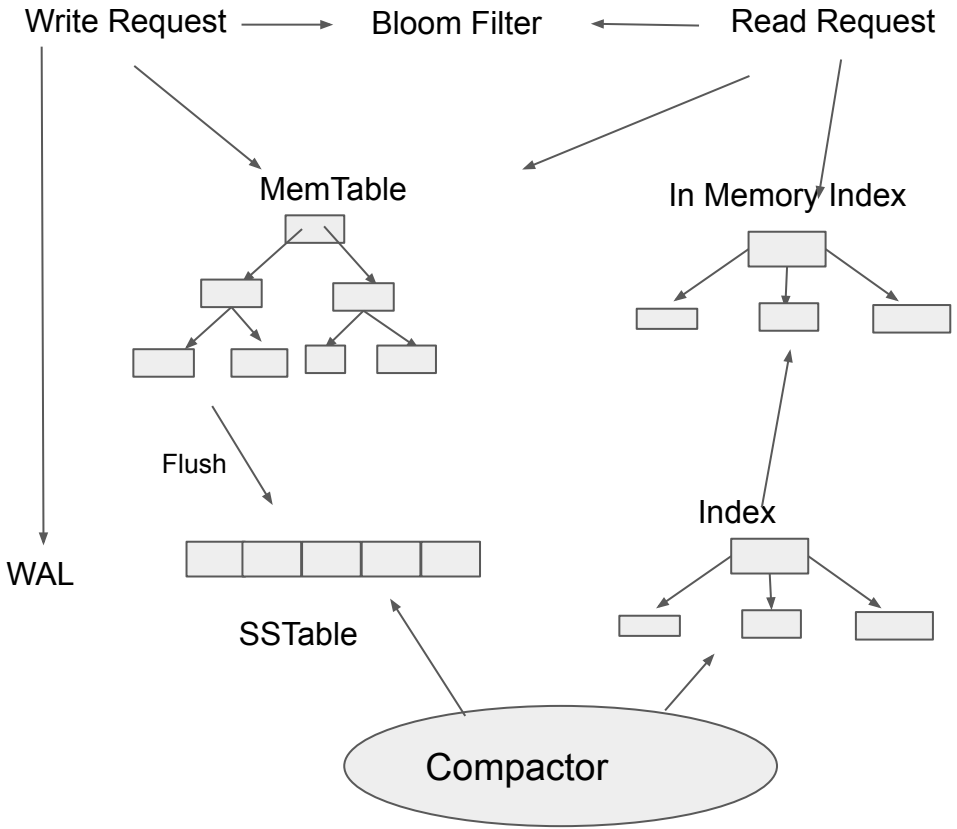
- B Tree
- B+ Tree
- B-link Tree
- B* Tree
- Log-Structured Merge Tree
- Mass Tree

Log-Structured Merge Tree

Idea: Log structured file system

Pro: Good write

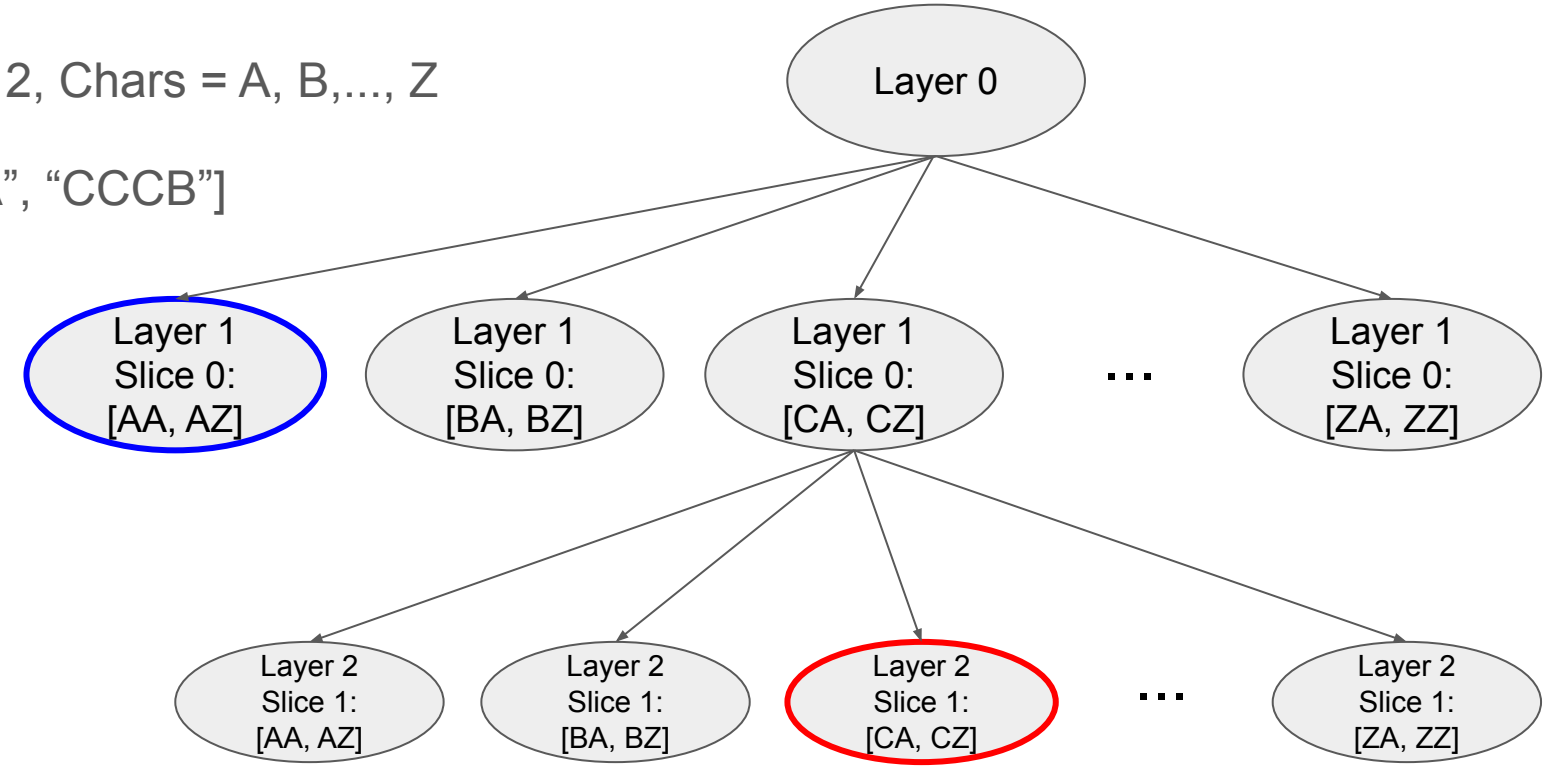
Con: Slower read



Trie

Slice size = 2, Chars = A, B,..., Z

Keys = [“AA”, “CCCB”]



Mass Tree

- Trie with B+ trees as nodes
- Variable-length keys
- Deals with long key length
- Divide-and-conquer

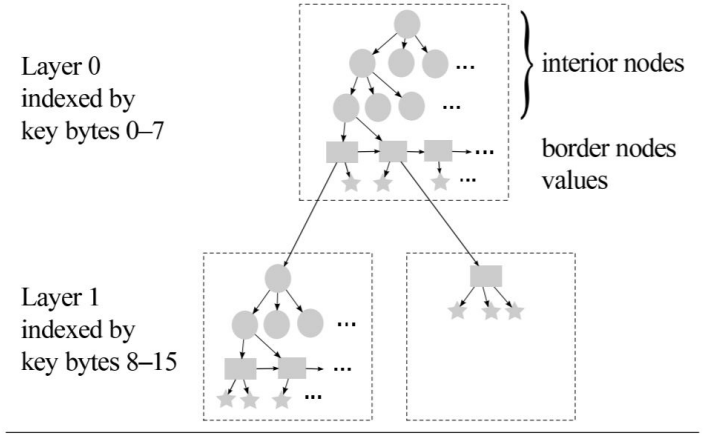


Figure 1. Masstree structure: layers of B^+-trees form a trie.

https://pdos.csail.mit.edu/papers/masstree:eurosys12.pdf

Hash Index

Build Phase:

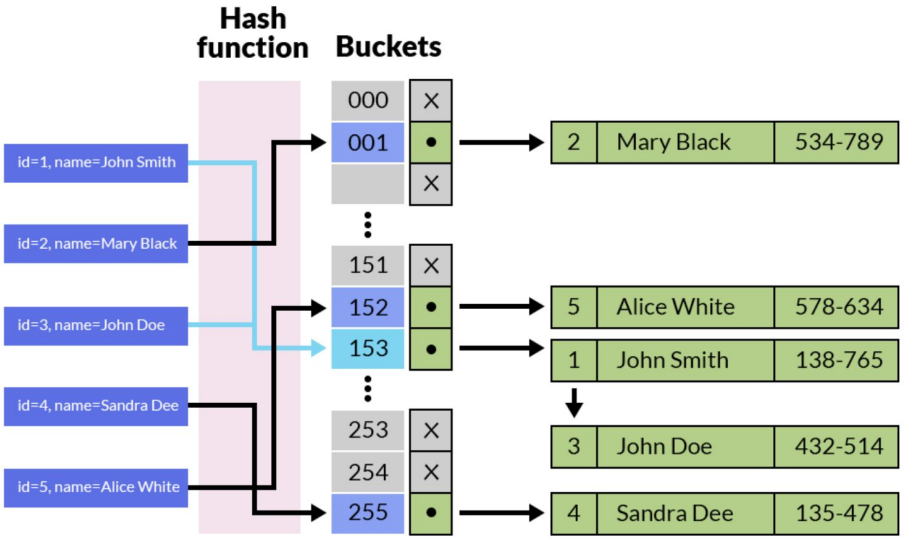
For each tuple t in smaller relation R1,

Hash build (t);

Probe Phase:

For each tuple q in larger relation R2,

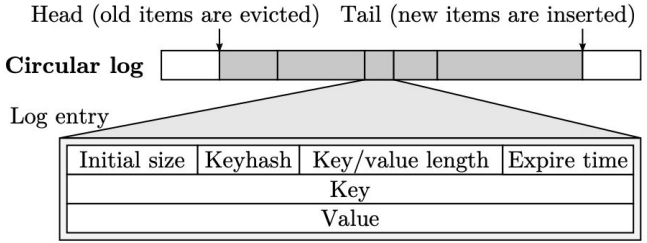
Probe (q);



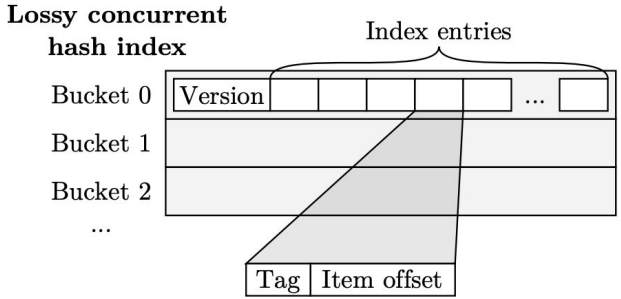
https://stackoverflow.com/questions/29436034/hash-index-vs-inverted-index

Hash Index used in MICA

- MICA: A scalable in-memory key-value store
- Goal: supports intensive write
- Circular Log - for memory allocation
  - FIFO: Evict oldest item at head of log when space's not enough (LRU as well)
- Lossy Concurrent Hash Index
  - Evict oldest entry from the set-associative table when a bucket is full



https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-lim.pdf



https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-lim.pdf

Evaluations

Database: DBx1000 (https://github.com/YSL-1997/DBx1000)

Workload:

TPC-C (New Orders and Payment) and YCSB

Configuration:

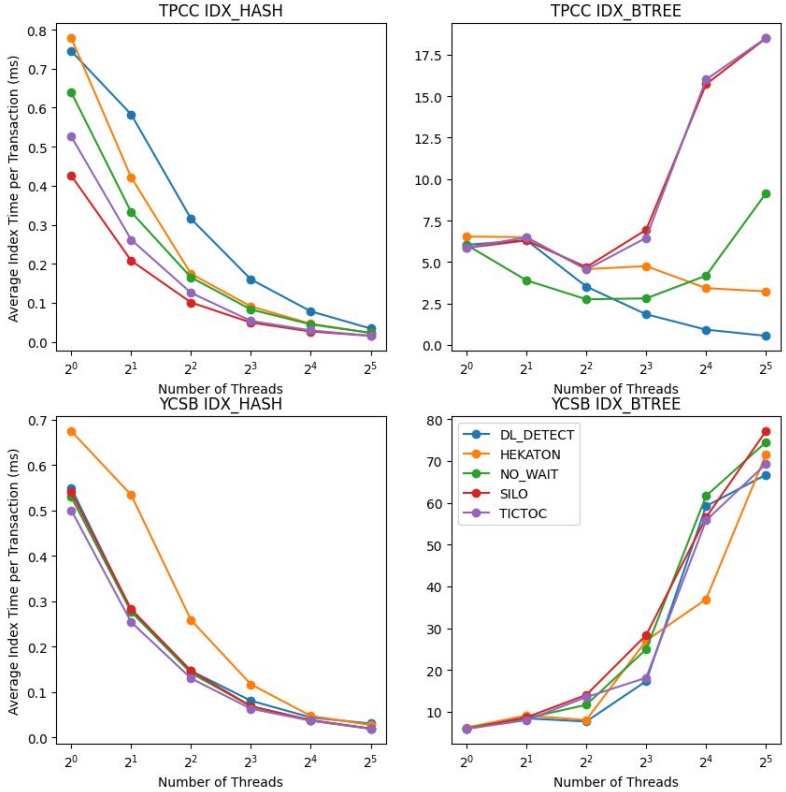
1. Number of threads: 1-128
2. Index types: Hash, B-Tree (implemented), MassTree (WIP)
3. CC Algorithms: DL_Detection, SILO, No-Wait, TicToc, Hekaton

Performance of indexing algo. under various CC

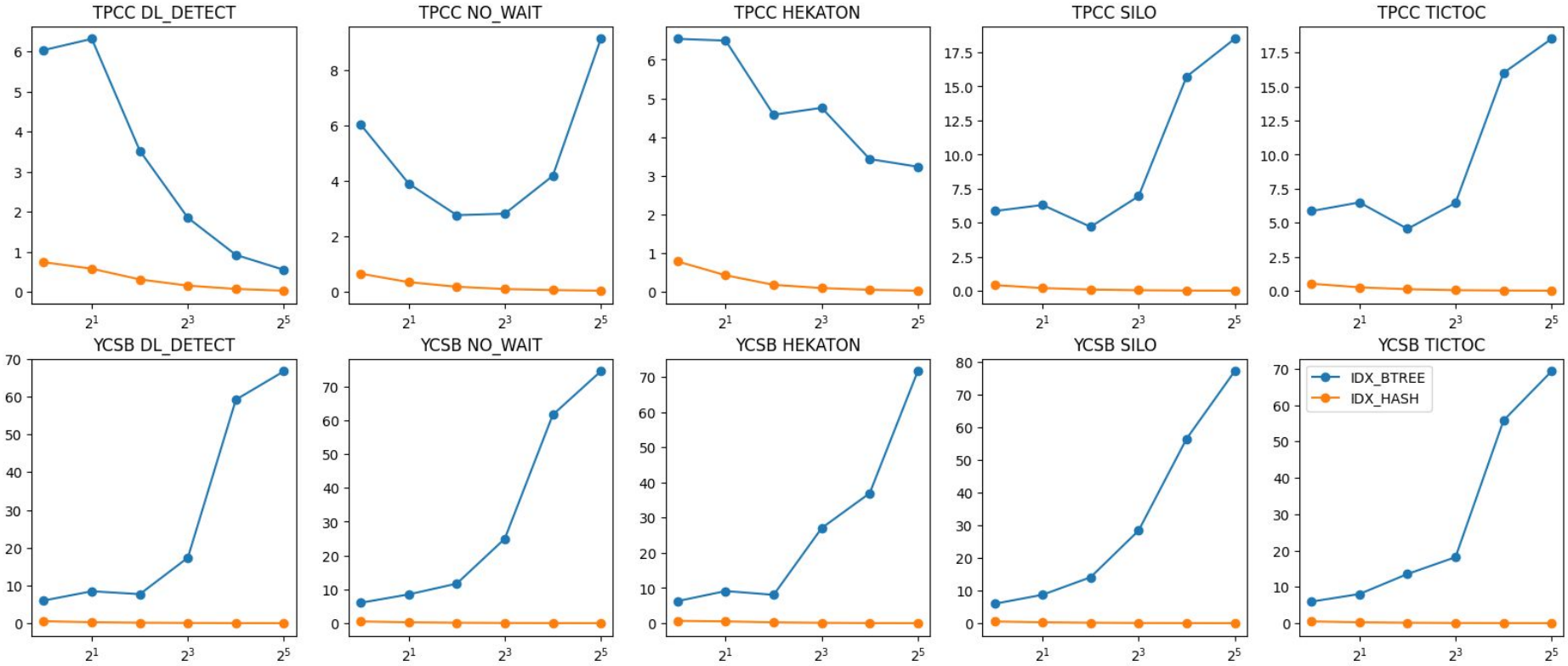
Concurrency Control Algorithms:

- DL_Detection: 2PL with central monitor of wait-for graph
- NO_WAIT: Non-preemptive 2PL
- SILO: State of art OCC
- Hekaton: Microsoft in memory database using OCC
- TicToc: Time traveling OCC

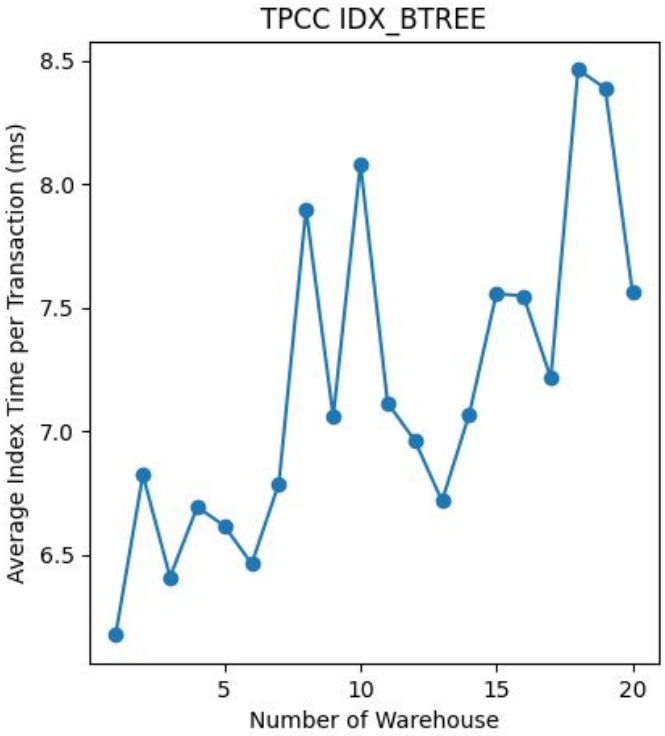
Scalability Evaluation

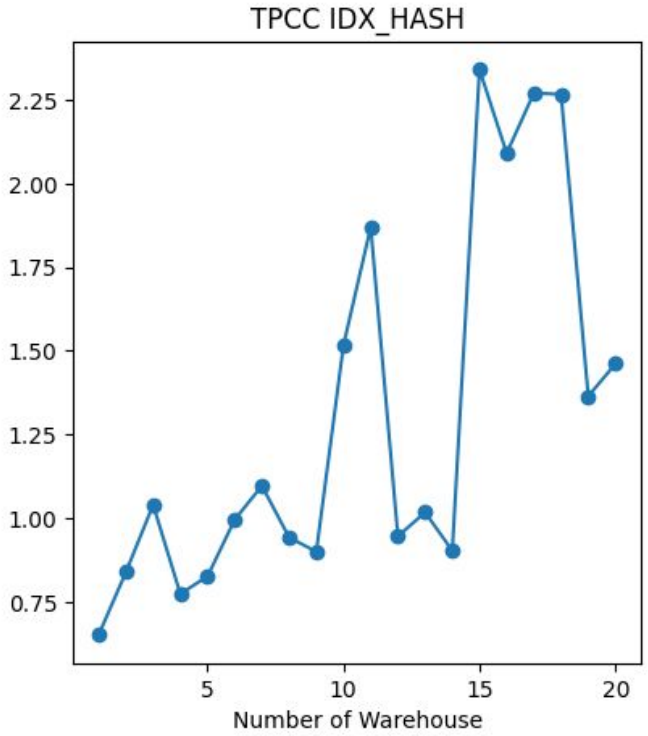


Hash vs. B+ Tree

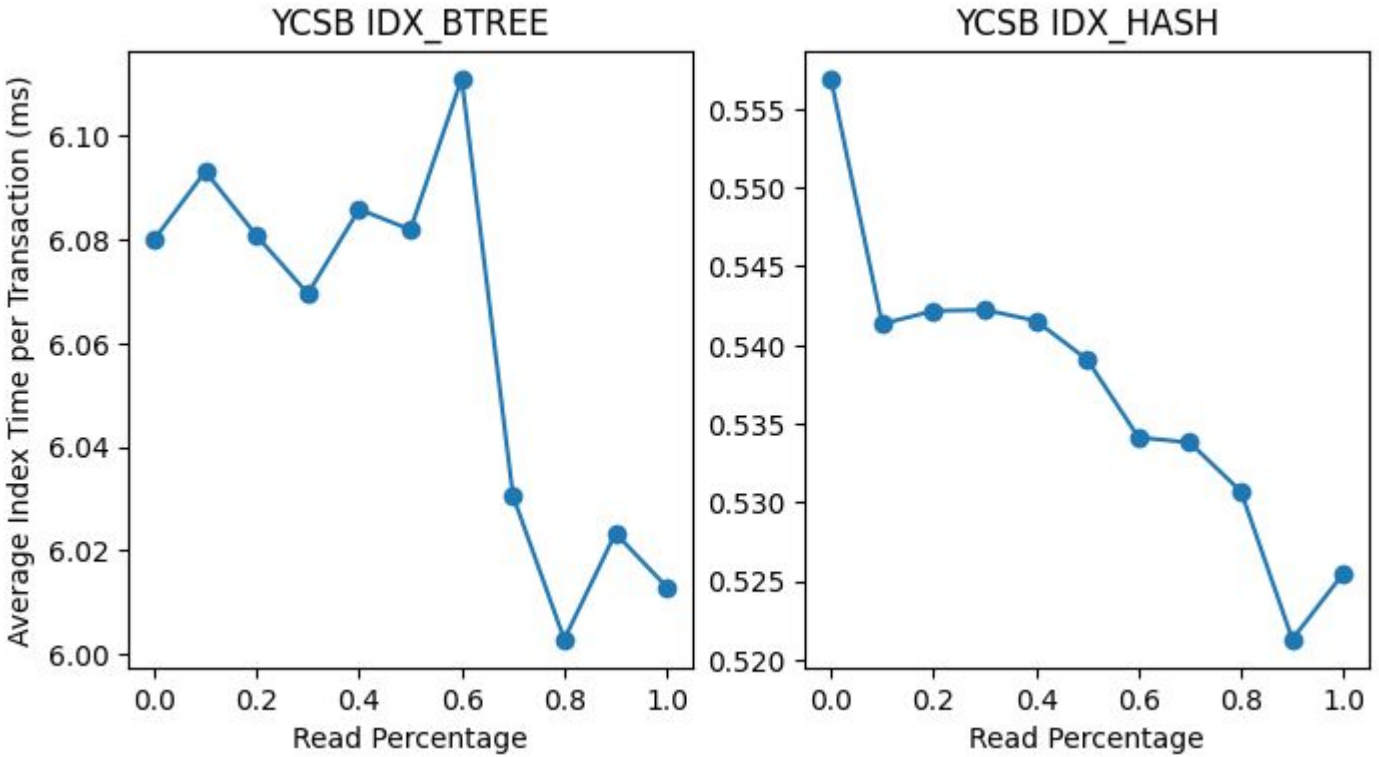


Contention Level

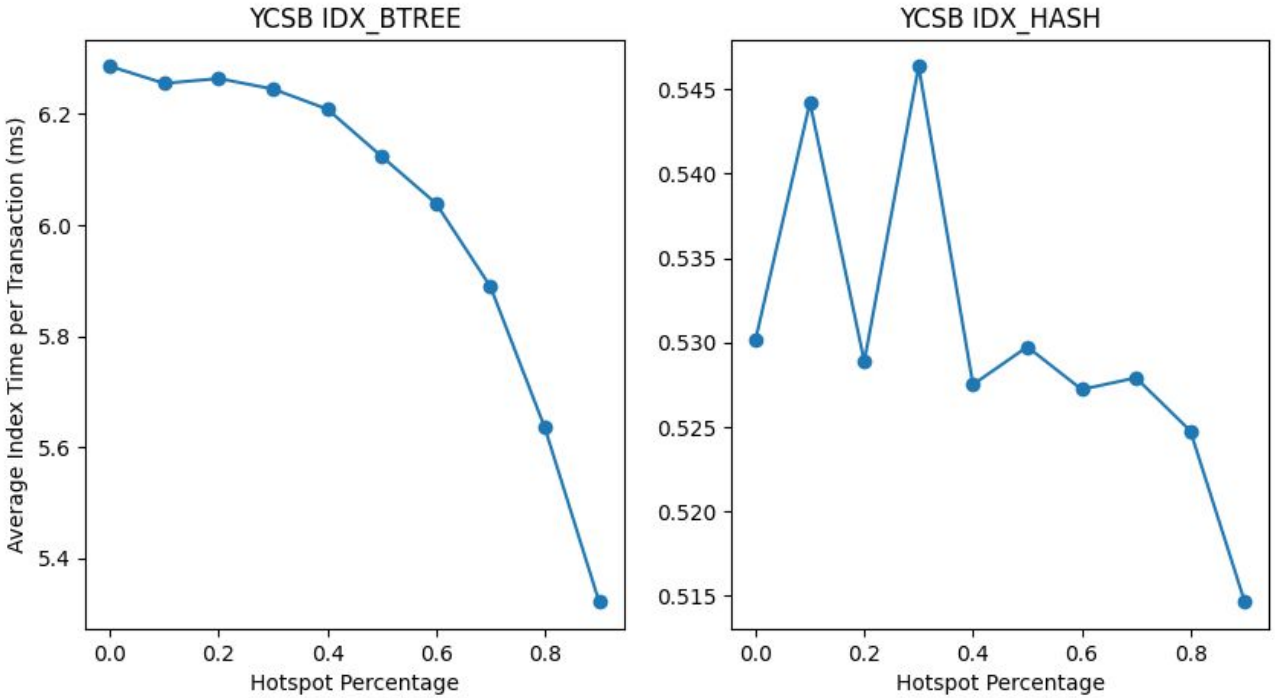




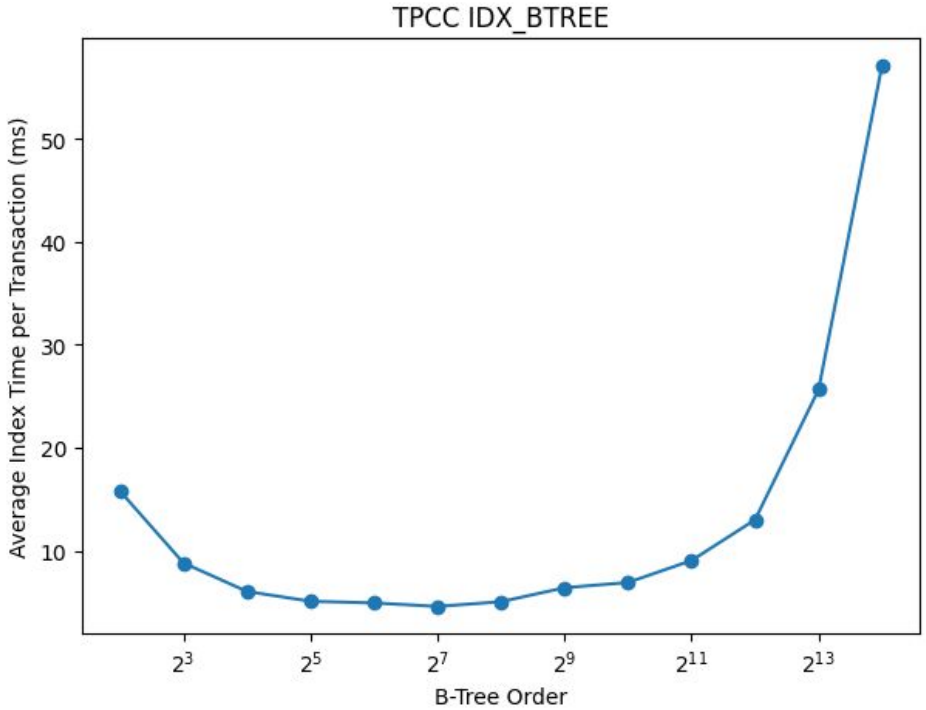
Read/Write Ratio



Hotspot percentage



B+ Tree fanout



Conclusion

Contribution

- We surveyed different indexing techniques in modern DBMS.
- We implemented B+ Tree index in DBx1000.
- We evaluated the performance over different indices, different concurrency control algorithms, and different number of threads with various setups.

Conclusion

Future Work

- Optimize B+ Tree to an extent that it can have comparable performance or even outperform the hash index under certain workloads.
- Evaluate Mass Tree, B-link Tree, Bw Tree, Palm Tree
- Redo our experiments to check whether we can achieve better results.