A Storage Engine for Modern Hardware

Viktor Leis
Figure 1. Breakdown of instruction count for various DBMS components for the New Order transaction from TPC-C. The top of the bar-graph is the original Shore performance with a main memory resident database and no thread contention. The bottom dashed line is the useful work, measured by executing the transaction on a no-overhead kernel.

- disk-based systems are hopeless
- emergence of in-memory DBMS:
  - radically new architecture
  - <100K instructions/tx
  - no good support for large data sets
Storage Trends

[CIDR 2020]

GB/$ [log scale]

disk

flash

DRAM

year
LeanStore Features

- high-performance storage engine for OLTP
- RocksDB-like C++ interface
- highly scalable on multi-cores CPUs
- optimized for directly-attached NVMe arrays
- B-tree indexes with row-wise storage
- logging, checkpointing, and recovery
- concurrency control implementation in progress
- page-based storage (4 KB)
- pointer swizzling: page reference can be pointer or page id
- lightweight two-stage page replacement algorithm (random + FIFO)
Indexing

• almost textbook B⁺-trees
• keys and values have variable size
• keys are prefix-compressed
• cache optimization: first four key bytes are in slot
• not quite as fast as the best in-memory structures but robust
(Optimistic) Latching

- each page has
  - version: `atomic<u64>`
  - mutex: `pthread_rwlock`
- page access modes:
  - optimistic
  - shared
  - exclusive
- B-tree is synchronized using Optimistic Lock Coupling
- buffer managers don’t need memory reclamation (!)
• Optimistic Lock Coupling solves read contention
• B-trees can have unnecessary write contention, solved by Contention Split:
  • use probabilistic per-page access counters to detect contention
  • split if contention is detected
• B-trees have low space utilization, solved by XMerge:
  • before page is evicted from buffer pool, check its and its neighboring fill factors
  • merge to save space
• lightweight in-memory logging approaches are not really feasible for out-of-memory workloads
• ARIES has many nice features:
  • arbitrarily large transactions
  • fuzzy and cheap checkpoints
  • fast recovery
• but standard ARIES is inefficient and not scalable on multi-core CPUs
• physiological WAL with read+undo info
• scalable distributed per-thread logging
  • WAL on PMem: low-latency commits with remote flush avoidance
  • WAL on SSD: high throughput with group commit
• continuous fuzzy checkpoints
• multi-threaded recovery
In-Memory TPC-C Performance (64 Core AMD Rome)

The graph shows the TPC-C performance (transactions per second) as a function of the number of threads, which is equivalent to the number of warehouses. The performance increases as the number of threads (warehouses) increases, indicating a positive scalability trend.
Out-Of-Memory TPC-C Performance (10 GB Buffer Pool, 7 × PCIe 3 SSDs)

TPC-C perf. [txn/s] vs. Data Size [GB]

IO [GB/s] vs. Data Size [GB]

- Total
- Read
- Write
Conclusions

- old techniques optimized for modern hardware
- made lots of progress
- currently a research prototype
- still many interesting technical challenges ahead (stay tuned)

http://leanstore.io

Figure 1. Breakdown of instruction count for various DBMS components for the New Order transaction from TPC-C. The top of the bar-graph is the original Shore performance with a main memory resident database and no thread contention. The bottom dashed line is the useful work, measured by executing the transaction on a no-overhead kernel.
• [SIGMOD 2008]: OLTP through the looking glass, and what we found there, Harizopoulos et al.
• [ICDE 2018]: LeanStore: In-Memory Data Management Beyond Main Memory, Leis et al.
• [DEBULL 2019]: Optimistic Lock Coupling: A Scalable and Efficient General-Purpose Synchronization Method, Leis et al.
• [Damon 2020]: Scalable and robust latches for database systems, Böttcher et al.
• [SIGMOD 2020]: Rethinking Logging, Checkpoints, and Recovery for High-Performance Storage Engines, Haubenschild et al.
• [CIDR 2020]: Exploiting Directly-Attached NVMe Arrays in DBMS, Haas at al.
• [CIDR 2021]: Contention and Space Management for B-Trees, Alhomssi at al.