Robust concurrency control in main-memory DBMS: What main memory giveth, the application taketh away

Ryan Johnson, Kangnyeon Kim, Tianzheng Wang
University of Toronto
ryan.johnson,knkim,tzwang@cs.utoronto.ca

Ippokratis Pandis
Cloudera
ippokratis@cloudera.com

Modern systems with large main memories and massively parallel processors have inspired many new high-performance OLTP systems [2, 3, 4, 7], often referred to as "main memory DBMS" (MMDBMS). These systems leverage spacious main memory to fit the whole working set in DRAM with streamlined, memory-friendly data structures; further, optimizations for multicore and multi-socket hardware allow a much higher level of parallelism compared to conventional database systems. With disk overheads and delays removed, transaction latencies drop precipitously and worker threads can usually execute transactions to completion without interruption. The result is a welcome reduction in contention and less pressure on whatever concurrency control (CC) scheme might be in place.

Meanwhile, database workloads are evolving to become increasingly heterogeneous, blending the gap between transaction and analytical processing. This trend is at least partly enabled by the improved concurrency and reduced contention offered by MMDBMS. Mixed workloads have two significant impacts on CC, however. First, the read/write ratio increases from 2:1 (e.g. TPC-C) to 10:1 or higher [1, 5], usually by increasing the number of reads as the number of writes remains stable. Second, workloads frequently include some fraction of large transactions that are read-mostly rather than read-only—a trend reflected in the TPC-E [6] benchmark. Unfortunately, both of these workload properties mean an increase in effective concurrency control footprints, and increased pressure on the CC scheme. As is usually the case, it appears that our workloads stand ready to absorb any and all concurrency gains the MMDBMS has to offer.

In this talk, we argue that the shift to heterogenous workloads means effective and robust CC schemes will become increasingly important for main-memory DBMS going forward. A growing body of research shows that the CC schemes currently in vogue with MMDBMS are not robust under contention, particularly when short write-intensive transactions coexist with longer read-mostly transactions. For example, the two most common families of approaches can be loosely classified as two-phase locking (2PL) and optimistic concurrency control (OCC). 2PL is common in traditional disk-oriented systems, and is often criticized because of high overheads, its policy of blocking transactions (leading to deadlocks and other scheduling problems), and a tendency to "lock up" (performance crash) once the aggregate transactional footprint grows too large (a state quickly attained when large transactions enter the system). OCC, on the other hand, never blocks readers—and may not even block writers—thus avoiding most scheduling issues. Although they differ in details, the rising generation of MMDMBS almost universally adopts a form of OCC that is effectively single-versioned, with read footprint validation at pre-commit [4, 7]. This type of approach suffers badly in high-parallelism systems [8] because transactions must abort if any portion of their read footprint is overwritten before they commit. Some systems [2, 3] sidestep the issue entirely by adopting a single-threaded transaction execution model, but that introduces a different set problems for mixed workloads.

In light of the weaknesses in 2PL and the common flavors of OCC, we next highlight several alternative approaches to concurrency control. Some are lesser-known (and worth taking more seriously); others are imperfect or still in progress, but promising (and good candidates for further refinement); and we round out the discussion with a few approaches that are new, unproven, and perhaps even a little crazy (but worth exploring because they are so different they—or something else just as wacky—might just work).

Finally, we close with a discussion of low-level issues (latching, thread scheduling, etc.) and design decisions—particularly at the system architecture level—that strongly influence the system's ability to provide robust and effective CC. The form of logging used, the storage management architecture, and scheduling policies for worker threads can impose drastic constraints on which forms of CC can be implemented at all, let alone efficiently. We examine several existing systems and show how their choice of CC is largely dictated by their system architecture—for better or for worse—and that it can be difficult or impossible to adopt a different CC scheme without significant changes to the rest of the system. The point is not that such design choices should be avoided, but rather that they should be made only with a full awareness of the consequences for concurrency control. Time permitting, we will report on some early progress in designing a MMDBMS from the ground up to support efficient concurrency control, and how the resulting architecture does not necessarily sacrifice performance in other areas.

References

[1] S. Chen, A. Ailamaki, M. Athanassoulis, P. B. Gibbons, R. Johnson, I. Pandis, and R. Stoica. TPC-E vs. TPC-C: Characterizing the new TPC-E benchmark via an I/O comparison study. *SIGMOD Record*, 39, 2010.

[2] R. Kallman, H. Kimura, J. Natkins, A. Pavlo, A. Rasin, S. Zdonik, E. P. C. Jones, S. Madden, M. Stonebraker, Y. Zhang, J. Hugg, and D. J. Abadi. H-Store: a high-performance, distributed main memory transaction processing system. *PVLDB*, 2008.

[3] A. Kemper and T. Neumann. HyPer – a hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In *ICDE*, 2011.
[4] P.-A. Larson, S. Blanas, C. Diaconu, C. Freedman, J. M. Patel, and M. Zwilling. High-performance concurrency control mechanisms for main-memory databases. *PVLDB*, 5(4), 2011.
[5] P. Tözün, I. Pandis, C. Kaynak, D. Jevdjic, and A. Ailamaki. From A to E: Analyzing TPC's OLTP benchmarks - the obsolete, the ubiquitous, the unexplored. In *EDBT*, 2013.
[6] TPC. TPC benchmark E standard specification, revision 1.12.0, 2010. Available at http://www.tpc.org/tpce.
[7] S. Tu, W. Zheng, E. Kohler, B. Liskov, and S. Madden. Speedy transactions in multicore in-memory databases. In *SOSP*, 2013.
[8] X. Yu, G. Bezerra, A. Pavlo, S. Devadas, and M. Stonebraker. Staring into the abyss: An evaluation of concurrency control with one thousand cores. Technical report, MIT CSAIL, 2014.