Exploring a continuum between main-memory and disk-oriented OLTP systems

Caetano Sauer
TU Kaiserslautern, Germany
csauer@cs.uni-kl.de

1. PROBLEM

In-memory OLTP databases re-emerged 10 years ago as a rebuild-from-scratch solution to a pressing problem: a 25-year-old software architecture was still the foundation of systems running on modern hardware [21]. A typical design approach in these systems is to eliminate large, complex software components by restricting the behavior of transactions. For example, buffer management is not needed if all data fits in memory, concurrency control is eliminated if transactions run serially, and redo logging is not required if durability is provided with synchronous replication [9].

However, eliminating large components also means giving up on important features and introducing limitations. Therefore, successful approaches have been introduced to, among other goals, offload cold portions of a growing dataset [2, 3, 5, 19]; support efficient recovery from logs [14, 24]; and improve parallelism by either tuning partitions [4, 17] or re-introducing concurrency control [11, 12, 16]. Looking further ahead, new challenges such as the advent of non-volatile memories [1, 10, 13, 18, 23] are likely to introduce even more techniques to the state of the art.

Given this context, one must be careful to not go "full circle" and end up with a system design that is actually more complex than the legacy system that was initially replaced.

Focusing on storage management issues, our discussion aims to take a step back and re-evaluate traditional designs in some aspects that could benefit modern designs. While in-memory systems are able to deliver much higher transaction throughput than their legacy counterparts, that usually comes with a cost for recovery and reliability. In this aspect, traditional designs like ARIES [15] and its variants still have the upper hand, since they support a wider range of failures and enable more efficient, fine-granular recovery. Furthermore, they solve most of the problems addressed by the approaches cited above, albeit with a cost in performance. Therefore, an evolution coming from the opposite side—improving the legacy design for in-memory performance—might be a valuable contribution towards achieving the best of both worlds.

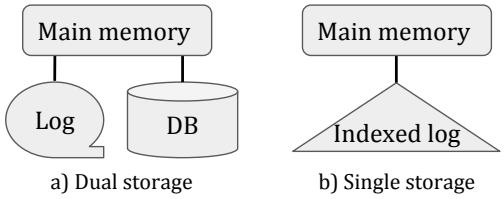

Figure 1: Dual- vs. single-storage approaches

2. SOLUTION

Our current line of research identifies the *duality of storage* as the main problem with legacy architectures. As illustrated on the left side of Fig. 1, database architectures usually maintain a history of changes in a log, in order to support fast commits with a no-force approach [8] as well as provide redundancy for recovery. Additionally, in-memory data structures are mapped to pages on a database file, which requires a disk-oriented layout and careful propagation from a buffer pool.

We propose a *single-storage* approach, shown on the right side of Fig. 1. While the idea itself is not new [20, 22], we propose a novel, generalized organization that is aimed at matching the performance of in-memory databases while supporting efficient, on-demand recovery. The key idea is to completely decouple in-memory data structures from any on-disk representation, providing persistence solely through logging. Unlike a dual-storage approach, however, the log is indexed and reorganized incrementally to enable fast access to persistent data.

The new design, which is currently being built in a prototype system, offers substantial benefits for recovery and reliability. It inherits the on-demand, incremental recovery capabilities of previous work on instant recovery [6] as well as pointer swizzling techniques [7]. However, it takes these ideas much further. The single indexed log not only decouples in-memory from on-disk representations—thus enabling in-memory performance—but also offers a generalized design perspective that essentially blurs distinctions between a database page and a log record, a backup and a replica, physical and logical logging, etc. This opens the opportunity for generalized system designs that, instead of being either specialized or one-size-fits-all, provide a common design template and code libraries that embrace specialization and make trade-offs explicit.

This research hopes to debunk the idea that in-memory and disk-based architectures are opposites separated by a sharp border—rather, they are part of a continuum of design choices built upon orthogonal principles.

Unpublished manuscript – submitted to CIDR 2017

3. REFERENCES

[1] J. Coburn, T. Bunker, M. Schwarz, R. Gupta, and S. Swanson. From ARIES to MARS: transaction support for next-generation, solid-state drives. In *Proc. SOSP*, pages 197–212, 2013.

[2] J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. B. Zdonik. Anti-caching: A new approach to database management system architecture. *PVLDB*, 6(14):1942–1953, 2013.

[3] A. Eldawy, J. J. Levandoski, and P. Larson. Trekking Through Siberia: Managing Cold Data in a Memory-Optimized Database. *PVLDB*, 7(11):931–942, 2014.

[4] A. J. Elmore, V. Arora, R. Taft, A. Pavlo, D. Agrawal, and A. El Abbadi. Squall: Fine-grained live reconfiguration for partitioned main memory databases. In *Proc. SIGMOD*, pages 299–313, 2015.

[5] F. Funke, A. Kemper, and T. Neumann. Compacting transactional data in hybrid OLTP & OLAP databases. *PVLDB*, 5(11):1424–1435, 2012.

[6] G. Graefe, W. Guy, and C. Sauer. *Instant Recovery with Write-Ahead Logging: Page Repair, System Restart, Media Restore, and System Failover, Second Edition*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2016.

[7] G. Graefe, H. Volos, H. Kimura, H. A. Kuno, J. Tucek, M. Lillibridge, and A. C. Veitch. In-memory performance for big data. *PVLDB*, 8(1):37–48, 2014.

[8] T. Härder and A. Reuter. Principles of transaction-oriented database recovery. *ACM Computing Surveys*, 15(4):287–317, 1983.

[9] S. Harizopoulos, D. J. Abadi, S. Madden, and M. Stonebraker. OLTP through the looking glass, and what we found there. In *Proc. SIGMOD*, pages 981–992, 2008.

[10] J. Huang, K. Schwan, and M. K. Qureshi. NVRAM-aware logging in transaction systems. *PVLDB*, 8(4):389–400, 2014.

[11] E. P. C. Jones, D. J. Abadi, and S. Madden. Low overhead concurrency control for partitioned main memory databases. In *Proc. SIGMOD*, pages 603–614, 2010.

[12] P. 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):298–309, 2011.

[13] L. Ma, J. Arulraj, S. Zhao, A. Pavlo, S. R. Dulloor, M. J. Giardino, J. Parkhurst, J. L. Gardner, K. Doshi, and S. B. Zdonik. Larger-than-memory data management on modern storage hardware for in-memory OLTP database systems. In *Proc. DaMoN Workshop*, pages 9:1–9:7, 2016.

[14] N. Malviya, A. Weisberg, S. Madden, and M. Stonebraker. Rethinking main memory OLTP recovery. In *Proc. ICDE*, pages 604–615, 2014.

[15] C. Mohan, D. Haderle, B. Lindsay, H. Pirahesh, and P. Schwarz. ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. *ACM Trans. Database Syst.*, 17(1):94–162, 1992.

[16] T. Neumann, T. Mühlbauer, and A. Kemper. Fast serializable multi-version concurrency control for main-memory database systems. In *Proc. SIGMOD*, pages 677–689, 2015.

[17] A. Pavlo, C. Curino, and S. B. Zdonik. Skew-aware automatic database partitioning in shared-nothing, parallel OLTP systems. In *Proc. SIGMOD*, pages 61–72, 2012.

[18] S. Pelley, T. F. Wenisch, B. T. Gold, and B. Bridge. Storage management in the NVRAM era. *PVLDB*, 7(2):121–132, 2013.

[19] R. Stoica and A. Ailamaki. Enabling efficient OS paging for main-memory OLTP databases. In *Proc. DaMoN Workshop*, page 7, 2013.

[20] M. Stonebraker. The design of the POSTGRES storage system. In *Proc. VLDB*, pages 289–300, 1987.

[21] M. Stonebraker, S. Madden, D. J. Abadi, S. Harizopoulos, N. Hachem, and P. Helland. The End of an Architectural Era (It's Time for a Complete Rewrite). In *Proc. VLDB*, pages 1150–1160, 2007.

[22] H. T. Vo, S. Wang, D. Agrawal, G. Chen, and B. C. Ooi. Logbase: A scalable log-structured database system in the cloud. *PVLDB*, 5(10):1004–1015, 2012.

[23] T. Wang and R. Johnson. Scalable logging through emerging non-volatile memory. *PVLDB*, 7(10):865–876, 2014.

[24] C. Yao, D. Agrawal, G. Chen, B. C. Ooi, and S. Wu. Adaptive logging: Optimizing logging and recovery costs in distributed in-memory databases. In *Proc. SIGMOD*, pages 1119–1134, 2016.