Technical Perspective for Sherman: A Write-Optimized Distributed B+Tree Index on Disaggregated Memory

Tim Kraska
MIT
kraska@mit.edu

Separation of compute and storage has become the defacto standard for cloud database systems. First proposed in 2007 for database systems [2], it is now widely adopted by all major cloud providers such as Amazon Redshift, Google BigQuery, and Snowflake. Separation of compute and storage adds enormous value for the customer. Users can scale storage independently of compute, which enables them to only pay for what they really uses. Consider a scenario in which data grows linearly over time, but most queries only access the last month of data, which remains relatively stable. Without the separation of compute and storage, the user would gradually be forced to significantly increase the database cluster capacity. In contrast, modern cloud database systems allow scaling the storage separately from compute; the compute cluster stays the same over time, whereas the data is stored on cheap cloud storage services, like Amazon S3.

However, current cloud database systems still tightly couple compute and memory. While it might feel unnatural to disaggregate compute and memory as the CPU needs access to data, a disaggregation would certainly have advantages. Servers used for database workloads tend to have large amounts of memory, so they can cache as much data as possible, to avoid the relatively slow access to storage services and keep all intermediate results in memory. The latter is of the utmost importance as queries, which spill to disk, often fall off the performance cliff and are orders of magnitude slower than their in-memory counterparts. Yet, keeping the most relevant data, intermediate results, and meta-data in memory is a non-trivial and often costly problem. Memory is expensive and the right machine type, which offers enough memory and compute, is hard to determine upfront. Moreover, workloads are rarely static. For example, a database system that mainly serves dashboarding queries might not require a lot of memory, except when it has to process ad-hoc data exploration queries.

If it were possible to scale memory independently from compute, it would be feasible to dynamically adjust the amount of memory based on the workload. It would further enable better resource utilization. Consider a dynamic workload regarding the number of queries but with very strict response time requirements, which can only be met, if data is kept in-memory. In this case, the separation of compute and memory would enable to scale the compute with the number of queries while keeping all the data constantly in-memory. This design principle is already used by services such as Google, which keeps the entire web-index in-memory.

In some ways, the disaggregation of compute, storage, and memory is the natural evolution of the currently prevalent separation of compute and storage. However, only recently, with the advancement of RDMA and low-latency network standards, this evolution actually becomes possible. In 2016 [1], we proposed the Network-Attached-Memory Database (NAM-DB) as a first prototype to explore the design of a system with separation of compute, storage, and memory. As part of the same project, we also explored the design of the first RDMA-based B-Tree index [3], which only uses one-sided RDMA messages and no RPC calls. Compute nodes do not require any CPU cycles to access the memory on the memory nodes, which is an important design-principle to avoid additional overheadh with the dissaggregation of compute and memory. However, while achieving good read performance, our design required several message round-trips for writes.

The Sherman paper addresses this limitation in a very elegant and novel way. It uses a lock-free search with versions to resolve read-write conflicts and exclusive locks to resolve write-write conflicts. It further cleverly uses the in-order delivery property of modern RDMA NICs to issue simultaneous commands and reduce round-trips further. I found it most interesting how the authors use the on-chip memory of modern RDMA NICs, which allows the elimination of PCIe transactions at the receiver-side and provides extremely high throughput.

Overall, I am convinced that this paper is an important stepping stone to achieving the full vision of the separation of compute, storage, and memory for the next generation of cloud database systems.

1. REFERENCES

[1] C. Binnig et al. The end of slow networks: It's time for a redesign. *Proc. VLDB Endow.*, 2016.
[2] M. Brantner et al. Building a database on S3. In *SIGMOD*, 2008.
[3] T. Ziegler et al. Designing distributed tree-based index structures for fast rdma-capable networks. In *SIGMOD*, 2019.