Letter from the Rising Star Award Winner

I am honored to have received the 2019 IEEE TCDE Early Career Award "for contributions to main-memory indexing and database architectures for NVM". Let me use the opportunity of this letter to describe three open, interrelated problems in this area that I consider both interesting and important.

Is the current dominance of LSM trees over B-tree justified?

For decades, virtually all database systems relied on B-trees for indexing (with hashing being a distant second). Most modern NoSQL, NewSQL, and cloud database systems, in contrast, primarily rely on Log-Structured Merge-trees (LSM) as their main data structure. B-trees and LSMs differ in terms of many different dimensions: in-place vs. out-of-place writes, eager writes vs. background merges, favoring reads vs. writes, etc. I therefore wonder: Have B-trees become obsolete? Are LSMs just a fad? Is it possible to design a data structure that combines the best properties of the two approaches?

Do we need a new class of database systems for flash arrays?

In the past 7 years, main memory capacities have stagnated. The first commercially-available version of byte-addressable non-volatile memory ("Intel Optane DC Persistent Memory") turned out to be as expensive as DRAM, but significantly slower. Flash, on the other hand, has become much cheaper during this time frame and is now $20\times$ cheaper than DRAM per byte. Furthermore, flash has become much faster, and it is now possible to directly attach a dozen or more devices to a single server, which results in a theoretical aggregated bandwidth close to DRAM. Neither traditional disk-based, nor modern in-memory or NVM-based database systems are capable of exploiting such extremely fast flash devices. This raises the question of whether a new system design is needed and how it would differ from existing approaches.

How to exploit hardware fluidity in the cloud?

When developing high-performance database systems, most of us implicitly assume that the hardware is fixed and optimize for a particular configuration. Given how most organizations procure hardware, this a reasonable approach. In the cloud, however, because it is easy to migrate to a different instance with potentially very different underlying properties, hardware should not be thought of as fixed. After all, users care about performance and cost, not about which kind of instances their service runs on. Therefore, cloud-native database systems could autonomously optimize the hardware configuration they run on. This requires an economical, literally cost-based approach that takes actual market prices into account.

Viktor Leis
Friedrich-Schiller-Universität Jena