Letter from the Special Issue Editor

Relational database systems were designed assuming data would be stored on hard disks (HDD) which, at the time, was the only realistic choice. Over the years database systems have been optimized for HDD storage. Solid state disks (SSD) based on flash storage have now become an attractive alternative, offering much higher performance albeit at higher prise per GB. Simply replacing HDDs by faster SSDs can produce impressive performance improvements but a database system optimized for HDDs may not fully exploit the capabilities of SSDs. SSDs have different characteristics and capabilities than HDDs. For example, on SSDs writes are much more expensive than reads. So how should database systems adapt to fully exploit the capabilities of flash SSDs while also avoiding some of their limitations? The papers in this special issue explore different aspects of this challenge.

The first three papers explore ways to exploit the substantial internal parallelism of a flash SSD. The paper by Roh et al on MPSearch proposes a way to speed up individual B-tree operations by exploiting the fact that a single device can process multiple I/O request concurrently. The key idea is to submit a batch of read or write requests at once instead of one at a time. Their experimental results show significant speed-up on range searches and inserts.

In the second paper, Fan et al propose a striped layout of pages such that pages from different stripes are accessed through different internal channels of the SSD. This layout makes it possible to scan different stripes in parallel. They then show how this can be exploited to speed up scans, aggregation, joins, and sorts.

Smart SSDs with a programmable processor inside the device are on the horizon. This makes it possible to push some processing down into the device, leverage its internal parallelism, and reduce the amount of data returned to the host. The paper by Park et al explores the opportunities and challenges arising from this functionality. Their results show significant speedups for highly selective queries.

Flash SSDs do not perform updates in place; a modified page is always written to a new physical page. This property has primarily been seen as a challenge to overcome but this too provides opportunities for improvement. A flash device essentially implements shadow paging - after a write, the old copy of the page still remains. The paper by Kang et al proposes to take advantage of this property by delegating the responsibility for ensuring transactional atomicity to the device. Their experimental results show greatly reduced transaction latency compared with the traditional approaches currently used by SQLite.

SSDs are increasingly being used as a persistent second-tier cache, that is, as an extension of the buffer pool. This raises the question of how to most efficiently manage such a two-level cache, which is the issue explored in the paper by Liu and Salem. An easy solution is to manage the two caches completely independently of each other but, as shown in the paper, this is not best strategy. They introduce an integrated strategy and experimentally show that this improves the overall cache performance.

I would like to extend a heartfelt thanks to the authors for their contributions to this issue.

Happy reading! I trust that you will find the papers as interesting as I did.

Per-Ake Larson
Microsoft Corporation