Solid-State Disks: How Do They Change the DBMS Game?

Angelo Brayner[1], Mario A. Nascimento[2]

[1] University of Fortaleza, Brazil
[2] University of Alberta, Canada
brayner@unifor.br, mario.nascimento@ualberta.ca

It is well know that the speed of processors has increased exponentially whereas the number of inputs/outputs per second (IOPS) afforded by hard-disk drivers (HDDs) has only increased marginally. A set of new storage media, called Solid State Disk (SSD), has emerged as a promising solution to decrease the difference between an HDD's data access time and the time that processors can consume data [Boboila and Desnoyers 2011]. Regarding storage capacity, SSDs still have a capacity inferior to their HDD counterparts. However, in the current pace of SSD technology development, we can expect that in the near future SSD storage capacity might be similar to that provided by HDDs. For that reason, we can also expect that SSDs be used in large scale within database management systems (DBMSs). However, the *modus operandi* of SSDs are rather different when compared to HDDs, which motivates revisiting many well-established techniques and algorithms in the core of a DBMS; all originally oriented towards HDDs. This tutorial focuses exactly on such issues, aiming at bringing this new trend, SSD-based DBMS, and its implications to the attention of the Brazilian database community.

A typical SSD is a computer chip which can be electrically reprogrammed and erased. An SSD stores data in an array of floating-gate transistors, called cells. Bits are represented by means of the voltage level in a cell. A cell with high voltage level represents a bit 1 (default state), whereas a low voltage level represents a bit 0. Sets of cells form data pages, data pages are organized in blocks and, finally, sets of blocks are stores within chips. There are three operations which can be executed on a flash device: read, erase and program [Kim and Koh 2004]. A *read* operation may randomly occur anywhere in a flash device. An *erase* operation is applied to a whole block, i.e., a set of pages, setting all bits to 1. A *program* operation sets a bit to 0. It is important to note that a program operation can only be executed on a "clean" (free) block, which is a block with all bits set to 1. Since SSDs have their lifetime determined by the number of write operations, a technique called *wear leveling* is applied to prolong its useful life. The key goal is to evenly spread out write operations across the storage area of the medium.

One of the most evident feature of SSD technology is the absence of mechanical parts in their assembly, only semi-conductors (chips) are used. Due to such a feature, SSDs presents a few interesting characteristics, for example:

(1) Low energy footprint. This is because, to perform read/write operations, there are only logical gates (circuitry) are involved;

(2) Low random access time. SSDs allows random access at least a few orders of magnitude faster than hard disk drives (HDDs);

(3) High random IOPS rates. Since SSDs have no mechanical moving parts, there is no mechanical seek time or latency to overcome.

(4) Asymmetry of read and write execution time. As a consequence of the technology used in SSDs, a read operation may be 1-2 orders of magnitude faster than a write operation.

From a database perspective, the first three characteristics are directly beneficial to existing database systems. On the other hand, the last characteristic, read/write asymmetry, poses new challenges to database technology, since write-intensive components (e.g., query processing and recovery components) of a database system (DBMS) may hinder a SSD-based DBMS performance, given they were designed for a symmetric rather than an asymmetric I/O (sub-)system.

In this tutorial we will discuss a number of core DBMS techniques and algorithms and will highlight recent research, e.g., [Tsirogiannis et al. 2009; Graefe et al. 2010; Fang et al. 2011; Sarwat et al. 2011; Koltsidas and Viglas 2011] that has addressed them in the context of this HDs/SSDs change of paradigm, thus highlighting opportunities for relevant and practical research, for instance:

—Indexing. It is known that conventional B+-trees may have high write-operation rates due to so-called "node splits". Some techniques to overcome such a problem will be discussed.
—Join Processing. Classical join physical operators require temporary result storage, i.e., they may increase write-operation rates during query processing. New join physical operators which reduce the number write-operations will be analyzed.
—Query optimization. Conventional query optimization techniques aim at reducing the number of disk pages transferred between disk and buffer area. While reading data stored in SSDs is not a problem anymore, writing pages into SSDs may negatively impact the DBMS's IOPS rate. In this sense, new ideas, in particular new query-execution cost models will be discussed.
—Caching. An SSD-aware database buffer management policy may reduce the impact of write operations on SSD-based DBMSs, hence SSD-aware buffer replacement policies will be discussed as well.
—Logging. Methods to delay log recording (e.g., a "batch" logging) could help to alleviate the write overhead inherent to SSDs. Other research thread to be discussed in this topic is the use of different log file structures, which may take advantage of the SSD behaviour. New commit processing techniques adapted to SSDs will also be discussed.

Acknowledgements: M.A. Nascimento has been partially supported by NSERC, Canada, and is also a visiting Professor at Federal University of Ceará, Brazil and Ludwig-Maximilians-Universität München, Germany

REFERENCES

Boboila, S. and Desnoyers, P. Performance models of flash-based solid-state drives for real workloads. In *Proc. of the IEEE 27th Symposium on Mass Storage Systems and Technologies*, A. Brinkmann and D. Pease (Eds.). IEEE Computer Society, pp. 1–6, 2011.

Fang, R., Hsiao, H.-I., He, B., Mohan, C., and Wang, Y. High performance database logging using storage class memory. In *Proc. of the 27th IEEE Intl. Conf. on Data Engineering*, S. Abiteboul, K. Böhm, C. Koch, and K.-L. Tan (Eds.). IEEE Computer Society, pp. 1221–1231, 2011.

Graefe, G., Harizopoulos, S., Kuno, H. A., Shah, M. A., Tsirogiannis, D., and Wiener, J. L. Designing database operators for flash-enabled memory hierarchies. *IEEE Data Eng. Bull.* 33 (4): 21–27, 2010.

Kim, K. and Koh, G.-H. Future memory technology including emerging new memories. In *Proc. of the 24th International Conference on Microelectronics*. Vol. 1. IEEE, pp. 377–384, 2004.

Koltsidas, I. and Viglas, S. Data management over flash memory. In *Proc. of the 2011 ACM SIGMOD Intl. Conf. on Management of Data*, T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis (Eds.). ACM, pp. 1209–1212, 2011.

Sarwat, M., Mokbel, M. F., Zhou, X., and Nath, S. Fast: A generic framework for flash-aware spatial trees. In *Proc. of the 12th Symp. on Advances on Spatial and Temporal Databases*, D. Pfoser, Y. Tao, K. Mouratidis, M. A. Nascimento, M. F. Mokbel, S. Shekhar, and Y. Huang (Eds.). Lecture Notes in Computer Science, vol. 6849. Springer, pp. 149–167, 2011.

Tsirogiannis, D., Harizopoulos, S., Shah, M. A., Wiener, J. L., and Graefe, G. Query processing techniques for solid state drives. In *Proc. of the 2009 ACM SIGMOD Intl. Conf. on Management of Data*, U. Çetintemel, S. B. Zdonik, D. Kossmann, and N. Tatbul (Eds.). ACM, pp. 59–72, 2009.