Hardware-aware Database Systems: A New Era for Database Technology is Coming - Vision Paper

Angelo Brayner, Jose Maria Monteiro Filho

[1]Universidade Federal do Ceara (UFC) Fortaleza – CE – Brazil
{brayner,monteiro}@dc.ufc.br

Abstract. *A new era of computers with solid state memory (SSDs) chips providing petabytes of storage area is coming. Nonetheless database systems (DBSs) were designed presupposing that databases are persisted in hard disk drives (HDDs) for storing databases. Over the years, DBS components have been optimized based on performance characteristics of HDDs. Simply replacing HDDs by faster SSDs may not fully exploit the capabilities of SSDs. In this paper, we present and defend the idea of hardware-aware database systems in order to make database systems able to fully exploit the advantages provided by new hardware, such as solid state memories.*

1. The Present

From a computer hardware perspective, we are witnessing nowadays the existence of two movements towards perhaps a singularity point in computer science. First, the computer industry is moving towards the construction in large scale of chips with hundred of cores in order to increase on-chip parallelism. Thus, in a near future we may have several-chips machines, each of which with hundreds of cores.

In parallel to the development of several-core chips, a new type of non-volatile memory is emerging, the so called solid state memory or solid state drive (SSD). Examples of solid state memories are Flash Memory, Phase Change Memory (PCM), Memristor and Non-Volatile RAM (NV-RAM), among others. The most evident characteristic of SSDs is the nonexistence of mechanical parts.

SSDs present distinct characteristics and capabilities from HDDs. IOPS rates supported by SSDs may be over 10^2 times greater than 15K RPM HDDs. Write operations on SSDs are much more expensive w.r.t. execution time and energy consumption than read operations, phenomenon called read/write asymmetry. A read operation may be up to 3 times faster and consume up to 8 times less energy than a write operation [Park et al. 2011]. The number of physical write operations on SSDs may be far larger than the logical operations, since SSDs internally run two processes (wear leveling and garbage collection) to minimize the impact of read/write asymmetry [Chen et al. 2009]. The lifetime of an SSD is determined by the number of write operations on it. Finally, SSDs present low rates of energy consumption.

Database systems (DBSs) were designed based upon two premises. The first one is the usage of magnetic disks for storing databases. The second premise is that distributed DBS could scale beyond what a single-node DBMS can support. However, the latter premise only holds for a small number of CPU cores in a node. Yu *et al.* present in [Yu et al. 2014] evidences that many-core chips require a completely redesigned DBS architecture that is built from ground up and is tightly coupled with the hardware. In other words, in order to fully exploit the parallelism supported many-core machines database systems should be aware of upcoming CPU architectures. The scope of this work is the former premise.

In this sense, this work bring clues that near future database systems should also be redesigned in order to take into account the emerging solid state memory technology. For that, we have investigated and compared the behavior of three well-known database systems running on SSD and HDD. Our strategy was to evaluate database system performance in two different scenarios. To simulate an OLAP scenario, we used the TPC-H benchmark. For reproducing a typical OLTP environment, TPC-E has been utilized.

2. The Solid State Memory Landscape

From all types of SSDs, flash-based SSD (or flash memory) and NV-RAM have already reached a commercialization stage. An NV-RAM device is composed of three different modules: DRAM (Dynamic-RAM), flash memory and an UPS (Uninterrupted Power Supply) [Vetter et al. 2012]. Thus, in NV-RAM device, the DRAM module is used as a cache memory and the flash memory is responsible for data persistence. Whenever occurs a power failure, the UPS component ensures the necessary power to retain data in DRAM module, while those data are flushed from DRAM to flash memory [Narayanan and Hodson 2012].

A flash-based SSD is a computer chip, which can be electrically reprogrammed and erased. Bits 0 or 1 are stored in floating-gate transistors, called cells. Typically, a cell with a voltage level higher than 5v means a bit 0. On the other hand, a voltage less than 5v represents a bit 1, which is the default state for any flash-based SSD.

Generally, a flash-based SSD is composed of host interface, internal processor, SDRAM buffer, flash controller and several flash memory packages [Park et al. 2012]. The main goal of the flash memory internal processor is to execute the request queue control and the Flash Translation Layer (FTL). FTL is responsible for running three critical processes: wear leveling, garbage collection and address mapping. A flash memory package is composed of several chips (or dies). Each chip presents several planes, where each plan contains a set of blocks. Each block is divided into pages. A page size may vary from 2KB up to 16KB. Most SSDs have blocks of 64, 128 or 256 pages [Dirik and Jacob. 2009].

Three operations can be executed on a flash device: read, erase and program. A *read* operation may randomly occur anywhere in a page within a flash device. An *erase* operation has the functionality of setting to 1 all bits within a block. Thus, the erase operations is block addressable, i.e., it can not be executed on a single page. A *program* operation sets a bit to 0. A program operation can only be executed on a block with all bits set to 1 ("clean" block). The program operation is page addressable. The higher the number of write operations on a flash memory, the shorter its lifetime is. In order to avoid that some blocks die much earlier that the others, the wear levelling process is employed to evenly spread write operations out across the storage area.

Another critical process internally running in a flash-based device is the garbage collection, whose main goal is to set all bits of stale blocks to 1. Thus, the amount of erase operations is reduced. However, the combination of these two mechanisms has a negative side-effect, since it makes the number of physical write operations on a flash-based SSD far larger than the amount logical write operations (write amplification phenomenon). The mapping address process is responsible for managing the address mapping table, a structure stored in SDRAM buffer, which is responsible to map addresses from SDD to main memory and vice versa.

There are four different levels of parallelism supported by flash-based SSDs. The first level, denoted channel-level parallelism, arises from the fact that the FTL communicates with flash packages through multiple channels, which can be accessed simultaneously. A given channel may be shared by multiple packages (Figure 1), supporting this way the package-level parallelism. Consequently, packages connected by a given channel can be accessed in parallel. The third parallelism level supports simultaneous access to chips within a package.

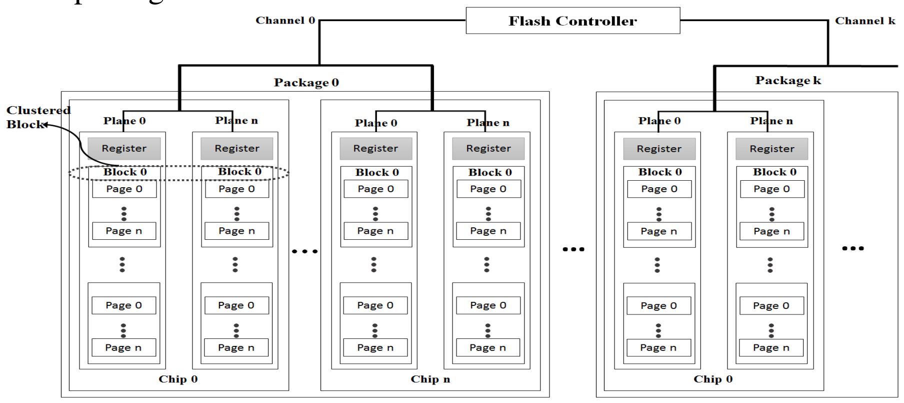

Figure 1. SSD internal architecture.

The last parallelism level, denoted plane-level, assures that the same operation (read, write or erase) can be executed simultaneously on multiple planes of a given die. Moreover, FTL stores data of a given file in a logical unit, called clustered block [Kim et al. 2012]. A clustered block contains blocks belonging to different planes of a chip. The blocks of a clustered block have the same logical plane addresses. Thus, several blocks of different planes can be accessed simultaneously. Figure 1 illustrates a clustered block composed of blocks with address 0 in planes 0 and 1 of chip 0.

3. The Clues To SSD-Aware Database Systems

In order to achieve our goal, we run experiments on three major relational DBSs. The idea was to investigate through a ”looking glass” the behavior of the evaluated DBSs on different storage hardware, more specifically on SSD and on hard disk. Due to legal reasons, instead of the DBS’s commercial names, we use the alias *Database A*, *B* and *C*.

The experiments have been carried out in two Intel Intel Pentium Quad Core 1.83 GHz server machines with 4-Gbytes RAM memory. In one server, there was an 120 GB Corsair Force 2.5’ SSD attached to the server through a SATA II interface. The other server had a Samsung SATA II HDD (502HI) with 500 GB (7,200 rpm). The TPC-H (scale factor 10) and TPC-E benchmarks have been used as test bed. Thus, we could simulate OLAP and OLTP workloads. Before starting a test run, we shut down and started up the DBS in order to reduce the influence of the DBS buffer manager in the results.

The experiments with TPC-H aimed at to observe the ability of classical query processing framework to take full advantages of SSD features. Thus, each database system (A,B and C) has been used to submit the TPC-H workload to the TPC-H database. For stressing the random access time in SSD and HDD, the 22 queries belonging to the

TPC-H workload have been executed once, ten times, twenty times, thirty times, forty times and fifty times.

Figure 2 brings the results of experiments using TPC-H. Observe that the difference between the response time for 50 executions (of the whole set of TPC-H queries) using SSD and HDD in *Database A* was of only 3 sec. In the case of *Database B*, that difference is of 2 sec (see Figure 2). For *Database C*, the difference is of approximately 3 sec. As already mentioned, the TPC-H is comprised of database queries. Furthermore, SSDs provide random access time up to four orders of magnitude faster than HDDs. Accordingly, it is reasonable to make the following assumption: *the response time of queries executed on databases stored in an SSD should be some orders of magnitude less than the response time of queries on databases stored in an HDD*. Nevertheless, the results depicted in Figure 2 reveal that the response time of queries executed on a database stored in an SSD is not even one order of magnitude less than the response time of queries on a database stored in an HDD. It is important to emphasize that there is no update operation in the TPC-H workload. Thus, for such a workload, the database system uses its query engine in full power.

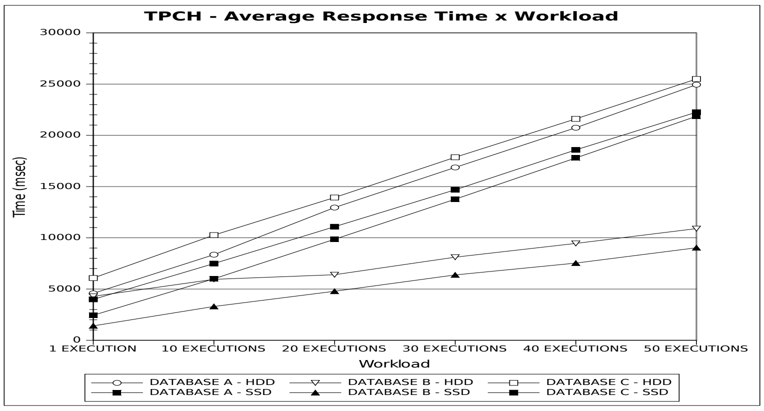

Figure 2. Response Time (rt) for TPC-H using HDD and SSD.

To explain the obtained results we postulate the following hypothesis: *for DBMSs running on SSDs, database query engine reduces the advantages of using SSD w.r..t. HDDs due to their write-intensive characteristics*. The rationale for supporting our hypothesis is the following. Classical query processing techniques have been developed having in mind that databases were stored in HDDs. Thus, the key goal of such techniques is to reduce disk access. Since the cost of executing a read operation is similar to the cost of a write operation in HDDs, those techniques make no difference between read and write operations. Nevertheless, this is not the case of SSDs, where there is a significant asymmetry for executing a read or a write operation. This way, one may conclude that the amount of write operations executed by classical query processing techniques significantly reduce SSD IOPS rates. In other words, SSD-aware query processing techniques should be developed for making DBMS capable to take full advantage of SSD features.

In [Tavares et al. 2013], the authors show that classical buffer allocation policies are not efficient for SSDs. They propose a buffer management policy to keep in buffer write-intensive pages, postponing the moment to write them back to the SSD. By doing so, it is possible to reduce the number of write operations onto SSD. Therefore, new allocation polices for buffer replacement for databases stored in SSDs should be devised.

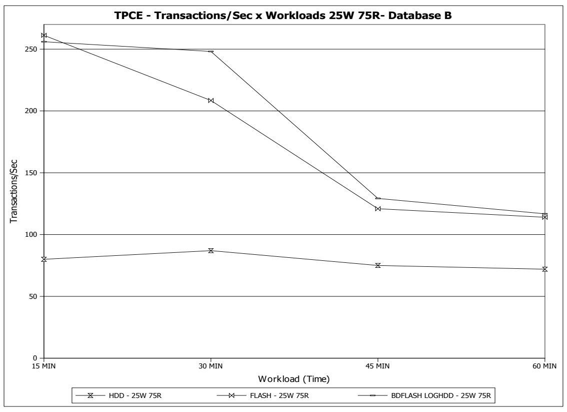

Figure 3. Throughput database B (25W/75R).

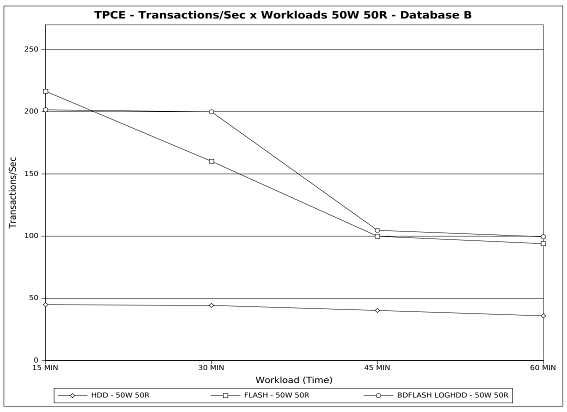

Figure 4. Throughput database B (50W/50R).

For simulating write-intensive workloads, three different workloads of the TPC-E benchmark (with a database with 33 tables, occupying 1 GB and 80 simultaneous users) have been executed. The first workload denoted ”25W-75R”, presents 4 transactions, 1 write-only and 4 read-only. The second workload is composed of 10 transactions, where 5 of them are write-only transactions and the other 5 are read-only, denoted ”50W/50R”, modeling a read-write ratio of 50%-50%. The third workload, denoted ”75W/25R”, has 4 transactions, 3 write-only and 1 read-only. Regarding the experiments with TPC-E, three different scenarios have been considered. In the first scenario, the database and the log file are stored in a flash-based SSD. For the second scenario, the benchmark database and log file were allocated in an HDD. Finally, in the third scenario, the database is stored in SSD, but the database log file is stored in an HDD.

For each workload, we have measured the system throughput (tps) for 15 min, 30 min, 45 min and 60 min running the workload’s transactions for 80 users. Due to space restrictions, we present only the simulation results on Database B. Nevertheless, it is important to mention that the other two DBSs presented similar behavior. In Figures 3, 4 and 5, one may observe the negative influence of write operations on workloads running on SSD. In HDD, the throughput has kept almost constant for every TPC-E workload. On the other hand, the experiments with SSD reveal that the higher the write operation rate is, the lower the system throughput is. For instance, augmenting in 50% the number of write operations in the TPC-E workload (see Figutre 5), the throughput decreases up to 45.5%.

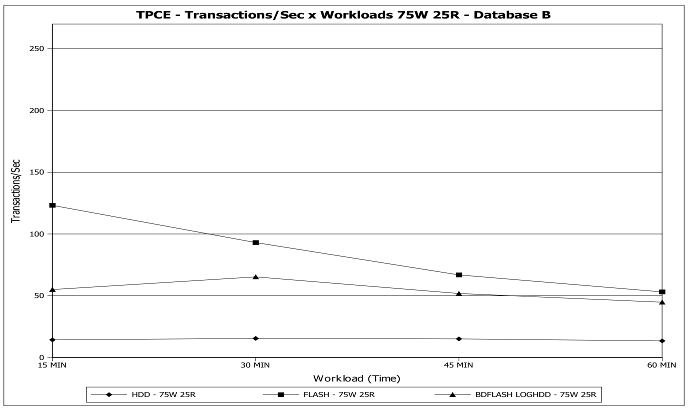

Figure 5. Throughput database B (75W/25R).

4. The Future

Taking all the results and arguments together into account, we may formulate the following observations. First, it is necessary the development of SSD-aware query engine, with features such as new physical operators and query cost model. The new query operators should avoid writing back temporary results (tables) to secondary memory. To illustrate that assertion, consider the join operation $R \bowtie S$, where P_R and P_S represent the size of R and S (in pages). Thus, for computing $R \bowtie S$, the grace hash join operator, for example, requires $P_R + P_S$ write operations to build the partitions of R and S. In the new cost model, the number of pages written onto secondary memory during query plan execution should have a greater weight than the traditional number of pages transferred from HDDs to main memory.

Furthermore, SSD-aware database buffer managers may reduce the amount of write operations on SSDs. Finally, a redesign in classical log mechanism is necessary in order to reduce the amount of write operations onto the log file.

References

Chen, F., Koufaty, D., and Zhang, X. (2009). Understanding intrinsic characteristics and system implications of ash memory based solid state drives. *Proceedings of 11th. International Joint Conference on Measurement and Modeling of Computer Systems - SIGMETRICS09*.

Dirik, C. and Jacob., B. (2009). The performance of pc solid-state disks (ssds) as a function of bandwidth, concurrency, device architecture, and system organization. *ISCA '09*, pages 279–289.

Kim, J., Seo, S., Jung, D., Kim, J.-S., and Huh., J. (2012). Advances in flash memory ssd technology for enterprise database applications. *IEEE Transactions on Computers*, 61:636–649.

Narayanan, D. and Hodson, O. (2012). Whole-system persistence with non-volatile memories. *ASPLOS*, pages 401–410.

Park, S., Kim, Y., Urgaonkar, B., Lee, J., and Seo., E. (2011). A comprehensive study of energy efficiency and performance of flash-based {SSD}. *Journal of Systems Architecture: the EUROMICRO Journal*, 57:354–365.

Park, S.-Y., Seo, E., Shin, J.-Y., Maeng, S., and Lee., J. (2012). Exploiting internal parallelism of flash-based ssds. *IEEE COMPUTER ARCHITECTURE LETTERS*, pages 9–12.

Tavares, J. A., de Aguiar Moraes Filho, J., Brayner, A., and Lustosa, E. (2013). SCM-BP: an intelligent buffer management mechanism for database in storage class memory. *JIDM*, 4(3):374–389.

Vetter, J. S., Marin, G., McCurdy, C., Cira, C., Liu, Z., and Yu, W. (2012). Identifying opportunities for byte-addressable non-volatile memory in extreme-scale scientic applications. *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 945–956.

Yu, X., Bezerra, G., Pavlo, A., Devadas, S., and Stonebraker, M. (2014). Staring into the abyss: An evaluation of concurrency control with one thousand cores. *Proc. VLDB Endow.*, 8(3):209–220.