An Experimental Analysis of the Use of Different Storage Technologies on a Relational DBMS

Francisco D. B. S. Praciano, Italo C. Abreu, Javam C. Machado

Laboratório de Sistemas e Bancos de Dados
Universidade Federal do Ceará, Fortaleza, Brazil
{daniel.praciano, italo.abreu, javam.machado}@lsbd.ufc.br

Abstract. Traditional Database Management Systems (DBMSs) are built with the premise that magnetic disks such as hard disks drives (HDDs) store the data. Recently, several alternatives to HDDs have emerged, such as the solid-state drives (SSDs) based on non-volatile memory (NVM) technology such as 3D XPoint and the new generations of dynamic random access memories (DRAMs). Different characteristics of these storage technologies may impact the performance of DBMSs. In this work, we analyze the performance of a DBMS using three storage technologies as data locations: HDD, SSD NVM, and DRAM, as well as a hybrid way combining all three. To do this, we use two workloads, analytical and transactional, and we observe throughput as well as latency. After, we discuss the reasons for the results obtained for each type of storage. We also show that the query processing can benefit from the different characteristics of each storage technology to perform faster queries and, finally, we analyze the benefits of using a hybrid storage system.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems; H.3.2 [**Information Storage and Retrieval**]: Information Storage

Keywords: databases, storage technologies, hybrid storage, performance evaluation

1. INTRODUCTION

As one of the most impactful elements of computer applications' performance, memory technologies are in constant improvement. Recently, persistent storage technology took a significant step with new non-volatile memory (NVM) technology in the market [Hady et al. 2017; Lepers et al. 2019; Mironov et al. 2019]. That brings up the question of how database management systems (DBMSs) can take advantage of the new features and particular characteristics of these new technologies [Wu et al. 2019]. Most of these systems considered the available technologies' features, capabilities, and limitations when developed, which reduces DBMSs' ability to follow technology advances.

NVM technologies fill the gap between Dynamic Random Access Memory (DRAM) and flash-based Solid-State Drive (SSD), introducing a memory capable of working both as primary and secondary memory [Kourtis et al. 2019]. Current technology may not be on the same level as main memory speed, which is understandable since the DRAM technology speed has not stopped in time; however, in this scenario, the performance of persistent storage technologies and volatile memories are nearing. This situation opens the horizon to different system configurations that can take advantage of these new technologies.

Some studies have addressed the question of how to take advantage of these new technologies in DBMS scenario in order to obtain systems that have a better overall performance [Liu and Salem 2013; Höppner et al. 2014; Brayner and Monteiro 2016; Wu et al. 2019; Kourtis et al. 2019; Didona et al. 2020]. On the one hand, some of these works have addressed this problem by proposing new techniques

This research was partially supported by CAPES (under grant number 1782887) - Brazil and LSBD/UFC.

for the DBMS core. For example, [Liu and Salem 2013] proposed a new buffer mechanism that is aware of the underlying hardware. On the other hand, several works have analyzed the influence of these new storage technologies without extreme changes in the DBMS core, as can be seen in [Brayner and Monteiro 2016] in which the authors have used SSD as storage technology to analyze the impact that a simple change from HDD to SSD may have on a DBMS. Our work follows this approach of conducting a study without making changes to the DBMS core. In addition, we also set up a hybrid approach to use multiple storage technologies at the same time.

This work is an extension of the paper [Praciano et al. 2019] and it aims to empirically evaluate the impact of the storage technology evolution in DBMS's performance and to study how DBMSs can be used in conjunction with other storage technologies on a hybrid approach. We measure the throughput and latency of PostgreSQL, a relational DBMS, running with the following storage technologies: HDD, SSD NVM, and DRAM. Our investigation discusses the effects through the following questions: 1) What are the impacts of storage technologies on a relational DBMS's performance? 2) Given the data and workload's features, how to build a hybrid storage model using different storage technologies that provides better performance? 3) What would be the cost-benefit of this hybrid storage?

The article is organized as follows: Section 2 discusses the current storage technologies available and their features. In Section 3, we study the performance impact of using different storage technologies separately on both analytical and transactional workloads. In Section 4, we show how to build a possible hybrid system using a transactional workload and analyze both the performance and cost-benefit of it, which may heavily improve a database system performance. In Section 5, we discuss recent works related to the subjects covered in this article. Lastly, in Section 6, we present our final remarks and indicate possible future works.

2. STORAGE TECHNOLOGIES

Hard disk drive (HDD) is the most common storage technology used by relational database management systems. These DBMSs often adopt several premises assuming that HDDs would store data; for example, most DBMSs implement a block-oriented storage interface [Ramakrishnan and Gehrke 2002]. In recent years, new storage technologies were proposed, such as solid-state drives (SSDs) based both on flash and new non-volatile memory (NVM). Figure 1 shows these technologies' components. The absence of mechanical parts in SSDs, such as the spindle or the actuator arm present on HDDs, provides persistent storage with higher access rates than current HDDs, while its cost per stored byte has been decreasing consistently every year [Shah et al. 2008; McCallum 2017].

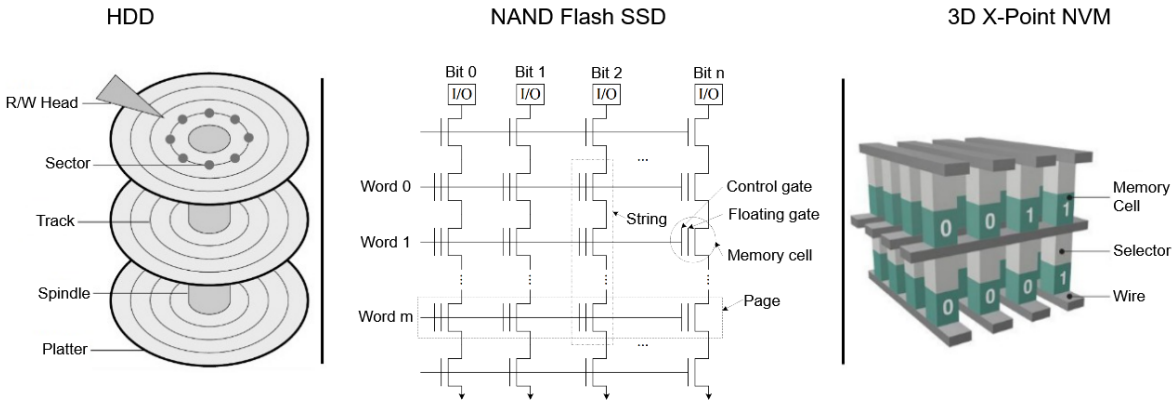

Fig. 1. Working diagram of HDD, NAND Flash SSD and 3D XPoint NVM, respectively. Modified from [Zukowski 2009; Kuo et al. 2011; Kelion et al. 2015].

SSD is a persistent storage technology that stores data in electric circuits. Most SSDs used as secondary memories use NAND flash memory cells organized into blocks composed of pages (Figure 1). The cells can be Single-level (SLC), which can store only one bit, or Multi-level (MLC), which may store two or more bits. These cells have a lifetime, computed by the number of rewrites, and may start causing errors once they are close to ending; consequently, the device loses its maximum capacity over the years. The access latency of both read and write operations is asymmetric because the read function is page addressed, while the write one is block addressed. HDDs are much slower than SSDs, which in turn are still orders of magnitude slower than DRAMs. However, in energy consumption, SSDs are more efficient than DRAMs.

NVM Express (NVMe) [NVM Express 2019] is a standard protocol for host-to-controller interfaces that allows host software to communicate with non-volatile memory through a PCI Express (PCIe) bus. It has become the industry standard for PCIe SSDs. Its development aimed to improve latency and takes benefit from parallelism. It has a shorter I/O data path and simplified software stack compared with SATA's AHCI standard protocol, which hard drives use to connect and communicate with host systems. The NVMe standard may support up to 65.535 I/O queues with $64K$ entries each, although manufacturers often implement much less. Figure 2 shows the NVMe command processing steps. First, the host inserts a command to a submission queue (1), then it tells the controller that there is a new command (2). The controller then fetches command from the queue (3) and processes it (4). Then, the controller writes the completion entry, including the last fetched command (5), and communicates it to the host by an interrupt (6). Finally, the host processes the entry (7) and writes a doorbell to release completion entry (8).

The NVMe SSD used in this work, Intel Optane, is based on a technology called 3D XPoint [Hady et al. 2017]. It is a three-dimensional structure that allows direct access to storage cells. Similar to DRAM, each memory cell stores one bit. It is structured in a matrix, in which rows are wordlines, and columns are digitlines. This way, data is addressed by byte, which is not possible with NAND technology. Besides that, these memory cells do not degrade per write cycle as NAND cells do. This structure is in Figure 1. 3D XPoint SSDs bring significant improvements in both performance and cell density than flash-based SSDs and are examples of non-volatile memory potential. We believe that DBMSs must be ready to adapt to the evolution of this type of technology.

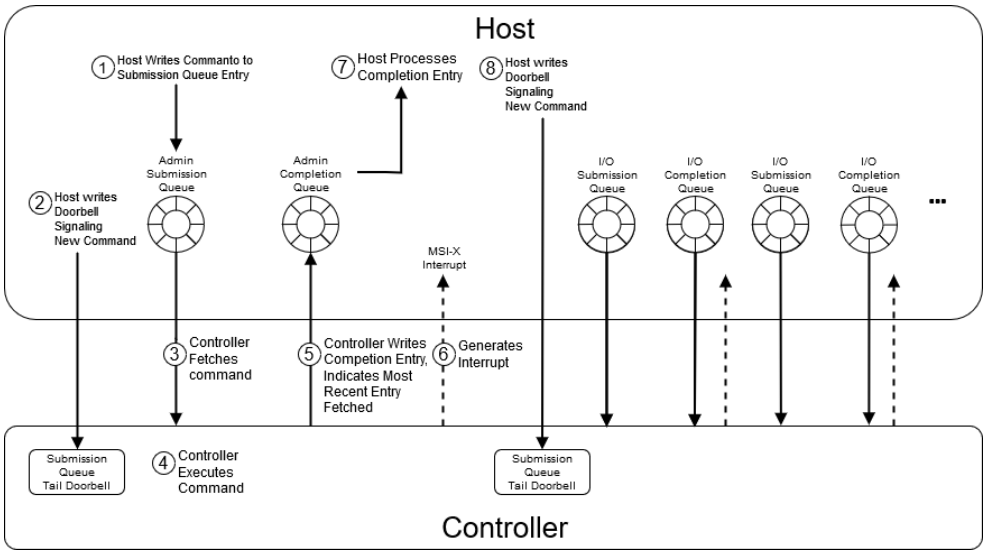

Fig. 2. NVMe command processing. Modified from NVMe base specification.

Not only non-volatile memory technologies are improving, but the evolution of DRAM has made it feasible to build DBMSs for main memory-resident databases, which has become good alternatives to many applications. In this type of system, data organization takes advantage of the byte addressing provided by this technology. However, a security copy of the database and a log must be stored in slower but persistent storage technologies and must be kept relatively up-to-date. In-memory databases are faster in terms of query processing but are more expensive per byte, and energy consumption efficiency is also a concern.

3. SINGLE STORAGE

This work investigates initially, through the execution of both analytical and transactional workloads, the performance measured by latency and throughput of a relational DBMS when using different storage technologies. The descriptions of these two types of workloads are in Section 3.1. In our experiments, we used PostgreSQL, which implements a blocks-oriented interface in its storage module. Hence, for our in-memory DBMS experiments, we employed a ramfs file-system that allows the block-addressing of main memory. Thereby, we can evaluate DRAM side by side with other block-oriented technologies. The usage of ramfs can impact the DRAM's performance, but it serves for comparative analysis.

The experiments were executed with a machine with three storage technologies, an HDD, an SSD NVM, and a DRAM according to Table I, with an Intel Xeon E5-2609 v3 1.9 GHz 15M Cache 6 cores and a GNU/Linux Kernel 4.15.0. Moreover, we chose PostgreSQL version 11.1 as DBMS. To obtain the results, we performed ten independent executions of each experiment on each storage technology and then calculated the metric average. It is worth mentioning that, in each storage technology' evaluation, we made sure to eliminate the presence of I/O in other technologies.

Table I. Storage systems' technical features.

Features	**HDD 7.2K RPM**	**SSD NVM**	**DRAM**
Manufacturer	Dell	Intel	Smart
Capacity (GB)	1000	375	24
Latency (ms)	$8,3$	$0,01$	$5 \cdot 10^{-7}$
Bandwidth (GB/s)	1,2	2,4	17
Price/bit (R$/GB)	1,99	20,27	43,75

PostgreSQL organizes the caching subsystem in two layers, the internal one directly controlled by the DBMS, and another one managed by the operating system (OS). When PostgreSQL reads a page from disk, it first goes to the OS cache and then to its buffer. A similar process occurs in writing, i.e., dirty data is flushed to the OS cache, and then, the OS writes into the disk. This caching subsystem impacts the results by improving the HDD performance with cache hits and creating an additional overhead that can reduce DRAM's performance. To avoid this scenario, we modified our setup to ensure that the OS cache does not affect the technologies' performance. As a result, it is as if the OS cache did not exist in this setup. Before presenting the experimental evaluation results, we briefly explain the differences between the two workloads used.

3.1 Analytical vs Transactional Workloads

Two types of workloads, analytical and transactional, represent well the two main ways that database applications perform access to data. They have different characteristics that make the queries carried out on each type differently [Rayner et al. 2014]. In the analytical or Online Analytical Processing (OLAP) workload [Chaudhuri and Dayal 1997], queries design focuses on providing information from

analyzing a plethora of historical data. As a result, these queries take longer to perform, and most of their operations is from readings to aggregation calculations. Furthermore, these queries need to be executed efficiently because their response time must be as optimized as possible, supporting a possible interactive use. Among several benchmarks proposed to model this type of work pattern, the TPC-DS [TPC 2020] rises as an evolution of TPC-H, an important benchmark commonly used to represent OLAP workloads.

On the other hand, the transactional or Online Transaction Processing (OLTP) workload is a cornerstone in the development and evaluation of databases due to its great importance in the use of these systems [Chen et al. 2012]. OLTP is characterized by transactions that often take a short time to complete. Besides, several of these transactions can run simultaneously. Hence an important performance metric is the number of transactions that are executed per second, namely throughput. For this type of workload, the considered standard benchmark is the TPC-C [TPC 2010]; thus, we opted for using its OLTP-Bench implementation [Difallah et al. 2013].

TPC-DS is an advancement of the TPC-H benchmark, which specifically models a data warehouse that is part of a decision support system for a retail product supplier [Barata et al. 2015]. The retailer sells products through three channels: Store, Catalog, and Internet, as demonstrated in Figure 3. This one shows the multiple snowflake schema implemented by TPC-DS, a hybrid approach between a 3-Normal Form (3NF) and a bright star schema [Nambiar and Poess 2006]. For this reason, TPC-DS has a more complex structure in comparison to TPC-H. For instance, TPC-H has a 3NF schema composed of only eight relations, whereas TPC-DS consists of multiple snowflake schema with 24 tables. Therefore, it is a more significant challenge imposed by the TPC-DS for DBMSs to perform query processing.

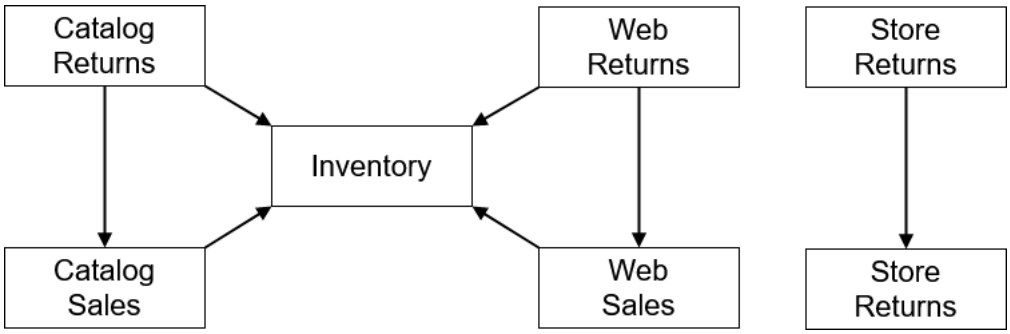

Fig. 3. TPC-DS schema.

The TPC-DS queries consist of 99 distinct SQL queries distributed in four types: pure reporting, pure ad-hoc, iterative OLAP, and extraction or data mining queries. This way, OLAP queries are covered as much as possible; consequently, the benchmark is considered representative for OLAP workloads.

TPC-C is an industry-standard benchmark for transactional workload. It models a system that processes order operations of a wholesale supplier where warehouses store its items' stock. Figure 4 shows this benchmark's nine relations, and the edges demonstrate each relationship between them. The supplier sells $100K$ items distributed in a defined number of warehouses, each with ten sales districts, and each district serves $3K$ customers. The number of warehouses is an input parameter that defines the supplier's size, so it is possible to add more warehouses and sales districts, depending on the company growth. Therefore, it is possible to have a more complex database as one wants by increasing the number of warehouses.

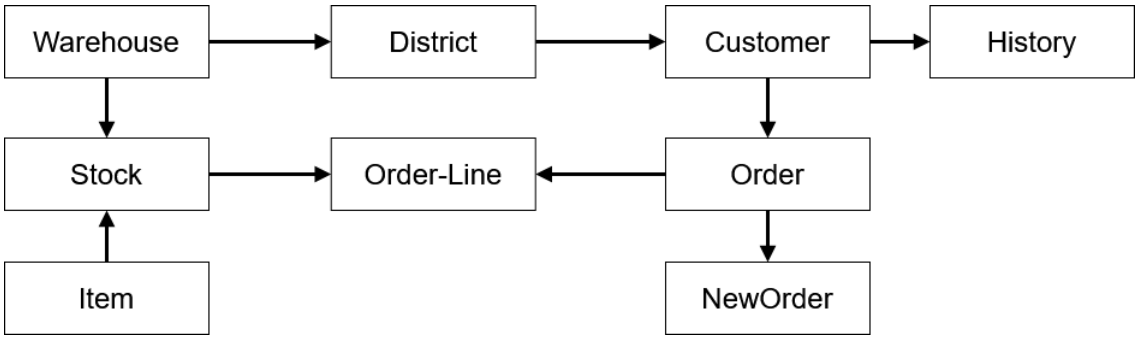

Fig. 4. TPC-C schema.

There are five types of transactions, described in Table II [Transaction Processing Performance Council (TPC) 2010]. A mixture of these transactions is executed simultaneously according to the percentages determined for each one of them. The performance metric, throughput, is defined by the number of New-Order transactions performed. In short, Table II describes what I/O operations the transactions do and the frequency they occur. For example, the New-Order transaction does both read and write operations and has a middleweight.

Table II. TPC-C's types of transactions and its associated role and I/O characteristic.

Name	Role	I/O Characteristic
New-Order (NO)	receive a new order from a customer	read-write, mid-weight
Payment (P)	update the customers' balance to record a payment	read-write, light-weight
Order-Status (OS)	retrieve the status of customers' most recent order	read-only, mid-weight
Delivery (D)	deliver orders asynchronously	read-write
Stock-Level (SL)	return the status of the warehouses' inventory	read-only

3.2 Overall Performance Impact

Before performing the experiments with PostgreSQL, we initially used `dd` and `fio`, tools available in Linux systems, to indicate the performance differences among the technologies without yet using a complex system like a DBMS. Figure 5 presents the variation in the data transfer rate among technologies when we use `dd`, a tool that allows reading or writing files on a specific technology. The figure shows the rate reached to read and write a 1GB file on each. We observe that taking reference as the transfer rate of HDD, SSD NVM has a rate five times higher, while DRAM has a rate fifteen times higher.

Likewise, we used `fio` to obtain more details on the possible ways to perform the following I/O operations: random reading (RR), random writing (RW), sequential reading (SR), and sequential writing (SW). Figure 6 shows the throughput (number of input and output operations per second - IOPS) of these operations on each device.

The small throughput achieved by the HDD's random operations draws attention, an expected result because HDDs have mechanical components, which results in a significant impact. On the other hand, the device's performance is highlighted in sequential operations, in which it almost matches the others. Lastly, the constant evolution of SSDs allows their throughput to approach that achieved by the main memory.

To answer the first point raised at the beginning of this work, we first run the standard TPC-C workload (i.e., 45-43-4-4-4 percentage of each transaction) with a warm-up time equals 5 seconds, a scale factor equals 50. The results are presented in Figures 7 and 8.

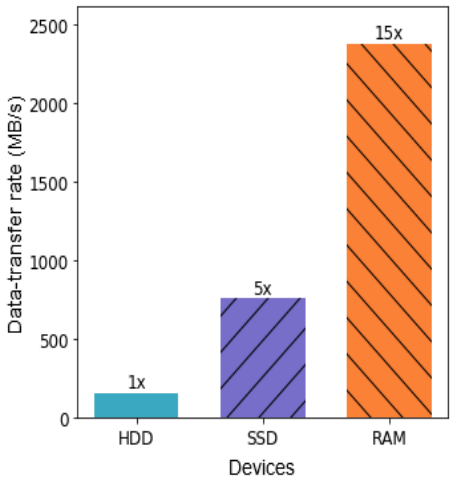

Fig. 5. Data transfer rate using `dd`.

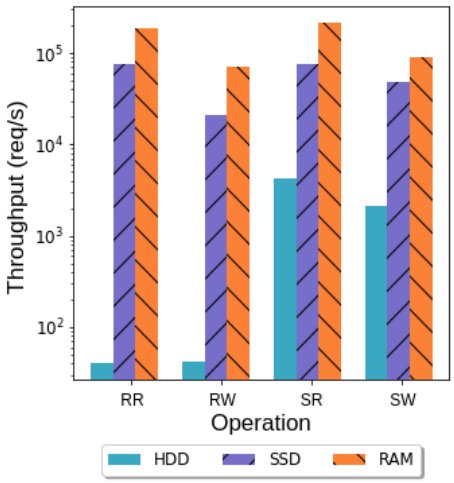

Fig. 6. Technologies' throughput using `fio`.

Figure 7 shows that the throughput of the underlying storage device can speed up PostgreSQL transaction processing by up to 24 times. Only the exchange of HDD for SSD already brings a seven times improvement in DBMSs' throughput. This effect is also seen in the drop in transaction latency, as shown in Figure 8. This improvement is mainly because TPC-C transactions execute random input and output operations, which are an outstanding feature of transactional workloads. Consequently, performance is impacted by the reasons given above when using `fio`.

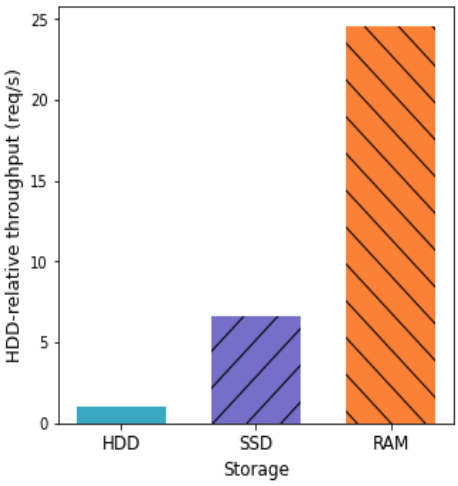

Fig. 7. HDD-relative throughput when running TPC-C.

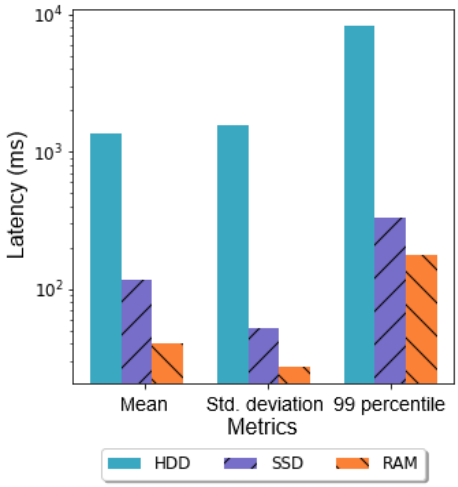

Fig. 8. Accumulated latency when running TPC-C.

Now let us consider the same question for OLAP workloads using TPC-DS. Again, we applied a warm-up time of 5 seconds, and a scale factor equals six. The chosen scale factor was due to the limitation imposed by the amount of RAM available on the machine used to perform these experiments, in this case, 24 GB. Even choosing a scale factor of six, five out of the 99 TPC-DS queries did not run because the RAM space was exhausted while they were executing on the RAM-stored database. The following are not in the result obtained: Q4, Q11, Q22, Q30, Q78. That is 94 of the 99 TPC-DS queries executed successfully. Figure 9 shows the results obtained in this experiment.

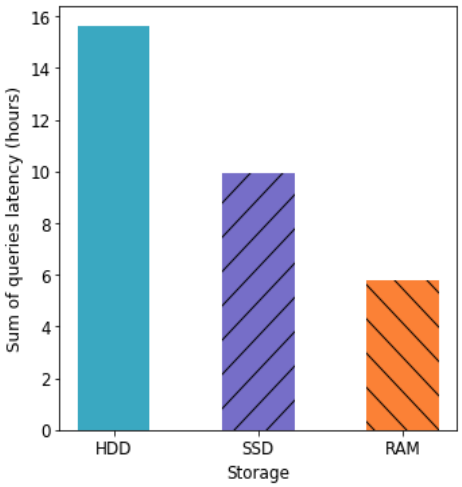

Fig. 9. Overall latency by running TPC-DS workload.

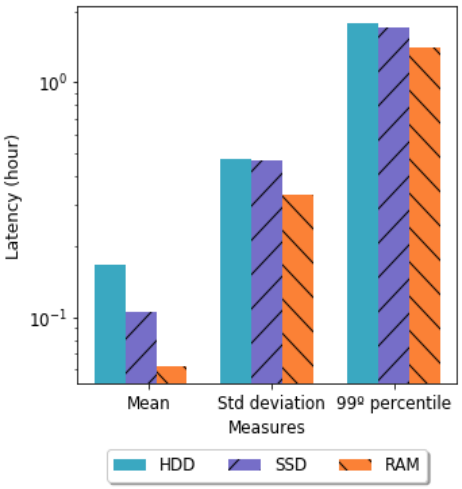

Fig. 10. Accumulated latency by running TPC-DS workload.

Figure 9 shows the sum, in hours, of the latency for the 94 queries executed on each of the technologies separately. As expected, the sum of the latency in HDD was higher than in SSD NVM, which, in turn, was higher than in RAM. Due to the speed difference between these types of technologies, this is an expected result. The overall latency reduction was approximately 6 and 10 hours in SSD NVM and RAM, respectively, in comparison to HDD's one. This improvement in the latency is directly related to the difference in the time of the technologies' I/O operations presented in Figures 5 and 6. Moreover, it is essential to note that OLAP queries involve a large number of I/O operations due to the characteristics of these queries; for instance, data aggregation is widespread.

Figure 10 details the performance impact using the metrics of mean, standard deviation, and 99-percentile of device latency. The first observation is that this result reflects the one from Figure 9. However, the relative differences between the performance seen in the mean do not show for the other two metrics. Note that the bars are closer together in standard deviation and 99-percentile metrics. As we will see in the next section, the latency of running some queries on the SSD or RAM nears or equals the latency of execution on the HDD.

Lastly, from the results, we can argue that changing the storage technology, without changes in DBMS's architecture (e.g., without changing blocks-oriented per bytes-oriented storage interface), is enough to bring gain to these systems' performance when dealing with analytical or transactional workloads.

3.3 Performance of Query Processing

In the previous section, we showed an analysis of DBMS's overall performance results. It was not possible to analyze queries execution locally. We evaluate the performance benefit for TPC-DS queries to answer the first question raised in this work, considering queries processed separately. We show that just replacing the underlying storage technology leads to faster running times for the most part.

Figure 11 shows the queries' speed up (in log scale) achieved when SSD or RAM replaces the HDD. In Figure 11, we chose to show only the first 50 queries. Considering the removed TPC-DS queries - Q04, Q11, Q22, Q30, Q78 - the horizontal axis range from 1 to 54.

At first glance, it is already possible to notice that RAM shows better improvement than SSD, even if the latter also shows an improvement compared to the HDD. Nevertheless, first, we only compare

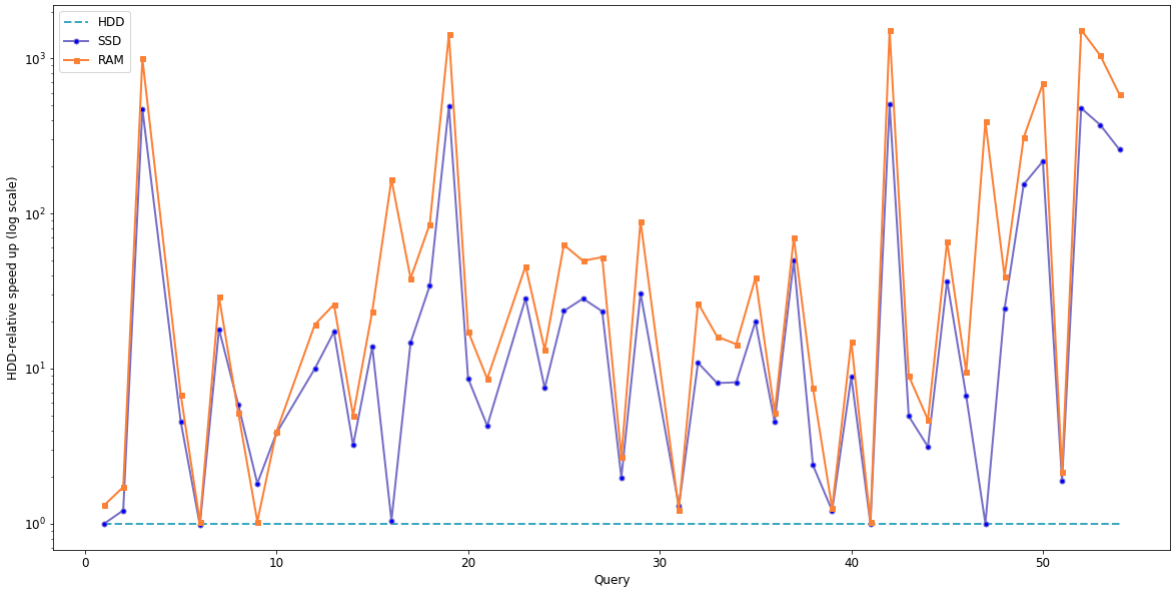

Fig. 11. Speed up of technologies on TPC-DS queries.

RAM to HDD. All of the queries were faster in RAM than in HDD, even though the speed up was not high in three of them, queries $Q06, Q09, Q41$. These queries had an improvement factor of less than 1.10. With further investigation, we could discover that they had the characteristics of correlated sub-query and common sub-expression.

In counterpart, queries $Q19, Q42, Q52$ are the ones with the most improvement. For those queries, the speed-up factor is approximately 1500. When executed on HDD, they run in seconds but last only milliseconds when executed on RAM. Moreover, it is essential to note that a common feature of these three queries is that they are interactive and have the structure of a simple star-join query.

All queries were faster in SSD than in HDD, except query $Q06$, which was slower in SSD. The difference in this query latency is only 5% between execution in SSD and HDD. In addition to this, queries $Q01, Q16, Q47$ did not present an improvement factor greater than 1.01. As happened in RAM, these queries also have the characteristic of correlated sub-queries. Finally, the same queries, $Q19, Q42, Q52$, showed the most significant improvement factor in RAM, also presented this improvement in SSD.

Now comparing RAM to SSD, 47 out of 50 queries were faster in RAM than in SSD. Only queries $Q08, Q09, Q31$ have a speed up lower in RAM. When analyzing these queries, we observed that their latency is at most 1 minute. Even more, the most significant difference in latency occurs at the query $Q07$ and lasts only 12 seconds. That is due to the overhead that `tempfs` causes in using RAM as block-oriented. Thus, we can conclude that the RAM was better than the SSD, except for one query since a difference of 12 seconds in 60 seconds is equivalent to a 20% improvement.

Summarizing all the results, we can argue that query processing can benefit from replacing storage technology even though the DBMS's architecture is not adapted to take advantage of each technology's best characteristics. In particular, we observed a reduction of approximately 60% in the accumulated query latency when replacing HDD with RAM. On the other hand, replacing HDD with SSD brought an improvement of 35% approximately.

4. HYBRID STORAGE

In this section, we strive to construct a hybrid storage system so that its performance is almost as high as that of a DRAM-only system at the same time that the associate cost is not as prohibitive. To do this, we opt to use the OLTP workload since this type of workload has a more widespread access pattern across relations, making it more challenging to create an approach that uses all three storage technologies simultaneously.

4.1 How to Build

In hybrid storage, we design a database placement considering the three different storage technologies. We applied two heuristics, 4.1 and 4.2 below, to drive which technology would store each relation, both of which followed a uniform distribution, that is, each storage technology received three relations. The rationale for using a uniform distribution is to bring balance to the amount of data and workload assigned to each device. This way we distribute the work as much as possible among the storage technologies.

We now propose and briefly analyze two heuristics that we use for generating five configurations (H1 - H5) based on TPC-C workload. The first heuristic is very simple.

Heuristic 4.1. Relations should be distributed among storage technologies according to their size so that each technology stores a balanced amount of data.

In the first heuristic, the criterion used as a deciding factor is the relation size. We measure the size of each relation and, then, distribute them to balance the amount of data stored in each storage technology. The upside of this approach is that, being simple, it can be easily automated. The downside is that it does not consider the characteristics of the workload, which can cause a storage technology to become overloaded with I/O operations. To tackle with this problem, we propose the second heuristic.

Heuristic 4.2. Relations should be distributed among storage technologies according to the number of I/O operations they receive in a given workload so that each technology performs a balanced number of I/O operations.

In the second heuristic, we analyze the workload's I/O operations pattern and, then, distribute the relations so that these operations performed by the workload are distributed in a balanced way among storage technologies. The rationale for this heuristic is that, considering the characteristics of the workload, a mapping can be performed in order to optimize the execution of I/O operations. For example, considering that a relation will only be read and never modified in a given workload, it may be interesting that this relation is stored in a storage technology that has a good reading rate, such as RAM.

As previously mentioned, we apply these heuristics to the TPC-C workload that consists of nine relations and five types of transactions in order to generate five configurations (H1 - H5). Using first heuristic, we measure the size of the nine TPC-C's relations and allocate them among storage technologies. Thus, two configurations H1 and H2, shown in Figure 12, were generated as simple as possible.

In counterpart, it was not straightforward to generate the configurations based on second heuristic. To do this, we analyze all types of TPC-C transactions, shown in Table II, to distribute the I/O operations executed by these transactions among storage technologies. The distribution of the transaction I/O operations among relations is given in Table III.

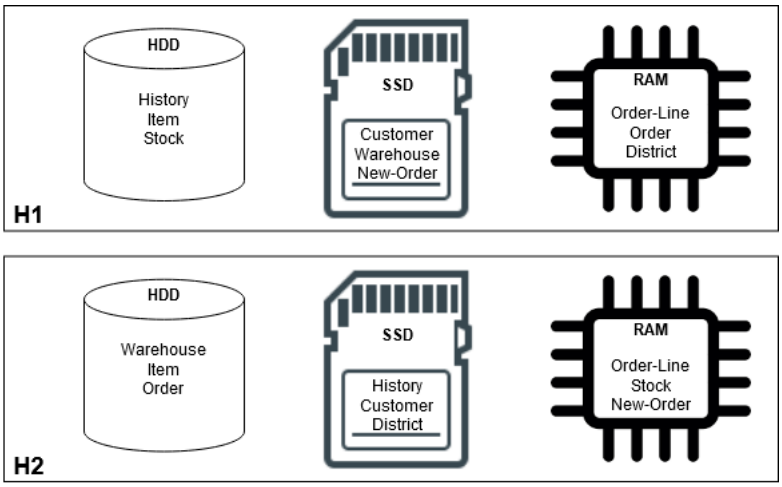

Fig. 12. H1 and H2 configurations generated based on Heuristic 4.1.

Table III. TPC-C's relations and its I/O operations executed by each TPC-C transaction.

Relation	SELECT	INSERT	UPDATE	DELETE
Warehouse	NO, P		P	
District	NO, P, SL		NO, P	
Customer	NO, P, OS, D		P, D	
Stock	NO, SL		NO	
Item	NO			
Order	OS, D	NO	D	
New-Order	D	NO		D
Order-Line	OS, D, SL	NO	D	
History		P		

Table III shows which query's type (or equivalently, I/O operations) a given relation can receive from TPC-C transactions. For example, Warehouse can receive SELECT queries from NO (New-Order) and P (Payment) transactions and UPDATE queries only from P (Payment). That is, all entry values in this table represent TPC-C transactions that were presented in Table II. After analyzing this table, we generate the configurations H3, H4 and H5 based on second heuristic. In Figure 13, we show these configurations.

4.2 Cost-benefit Ratio

The replacement of HDD for RAM brings a significant improvement in PostgreSQL performance when executing a transactional workload; however, this incurs an increase in cost related to this change. Even though main memory technologies have evolved over the years and have improved their cost-benefit, there is still a relevant difference in these storage technologies' prices. Today, a DRAM costs, per GB, more than 200 times an HDD, and around 4 times an NVM SSD. In other words, for DRAM, one pays that much more than for an HDD, but, in return, it is possible to get a device that is more than 15 times faster. If one chooses an SSD, it is also possible to get a third of this speed for a quarter of that price. With the advent of new NVM technologies, such as the 3D XPoint used in this work, this gap has narrowed, but it is still remarkable.

Figure 14 shows the relationship between the cost of maintaining the database and the throughput obtained, whether using a single technology or performing one of the various hybrid forms described above. Note the hybrid configurations H3 and H5. With a cost two times lower and maintaining a

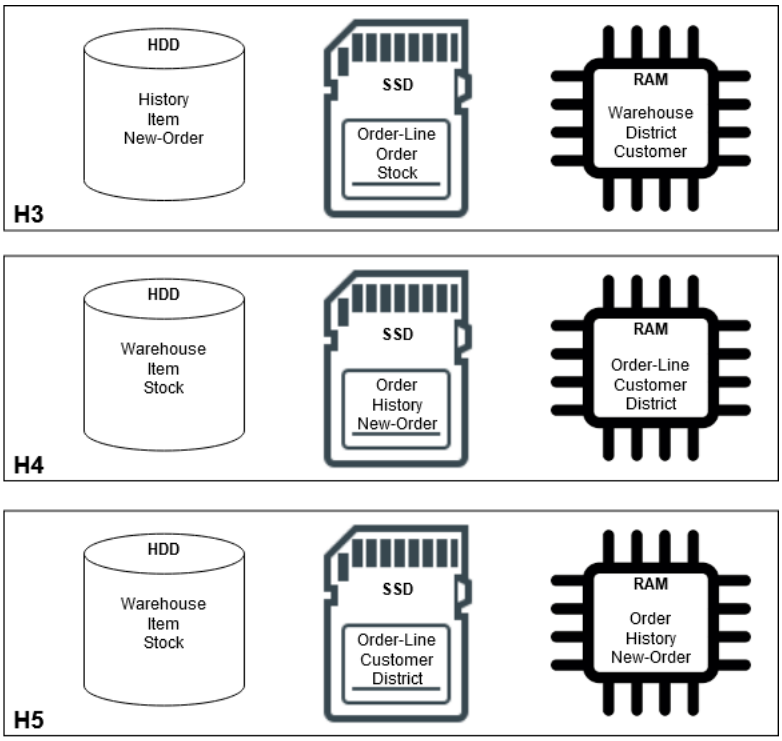

Fig. 13. H3, H4 and H5 configurations generated based on Heuristic 4.2.

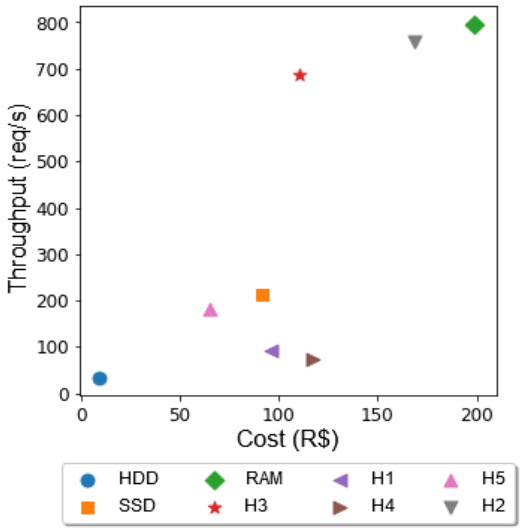

Fig. 14. Hybrid template, HDD, SSD and RAM's cost-benefit

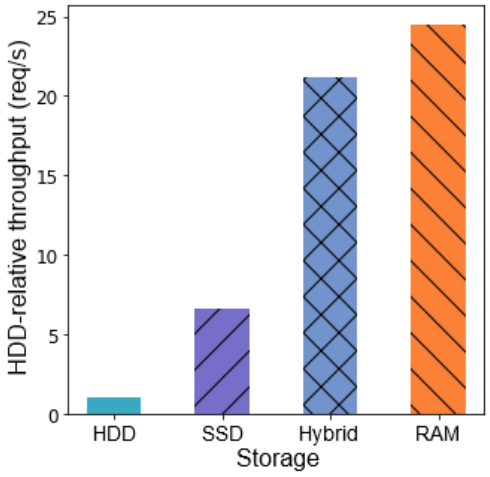

Fig. 15. HDD-relative throughput when running TPC-C considering the best hybrid model (H1).

throughput very close to the SSD, the H5 configuration presents a better cost-benefit relation than the SSD, thus being a viable alternative. The H3 configuration, on the other hand, obtained the most significant gain in the cost-benefit ratio since it presented a performance similar to that of RAM at a lower cost. Based on this configuration, the cost-benefit ratio of hybrid storage would be 6, while

RAM's would be 4.

Figure 15 shows the throughput of hybrid configuration, in the case configuration H3, compared to HDD, SSD and RAM. We can notice that the throughput achieved by that configuration, which obtained the best cost-benefit ratio when compared to other setups (H1-H5), has an improvement of around 20 times when compared to the throughput rate of the HDD. In this configuration, the TPC-C's Warehouse, District, and Customer relations were allocated in RAM, while the Order-Line, Order, and Stock, on the SSD and, finally, the History, Item, and New-Order, on the HDD, as seen in Figure 13. Therefore, we can say that building a hybrid storage model is possible considering the workload and still obtain excellent performance.

5. RELATED WORK

Many works have studied the impacts of using different technologies for storing data in DBMSs. With a detailed experimental analysis, the paper [Xu et al. 2015] presents a characterization of the performance of SSDs that use NVMe standard. They compared an NVMe's I/O stack with a SATA SSD's one and identified factors that provide performance improvement in NVMe-based SSDs. Using a tool called *blktrace*, the authors also measure the time spent in different phases of the NVMe stack after an I/O request until its completion. They also perform an experimental analysis of the performance of relational and non-relational DBMSs when using NVMe SSDs, SATA SSDs and HDDs, and a tmpfs file system on DRAM as upper limit reference. They use the MySQL DBMS with a TPC-C workload and evaluate CPU utilization for: SSD drive, RAID 0 of four SSD drives, NVMe drive, and tmpfs on DRAM. They run YCSB workloads on Cassandra and MongoDB and evaluate the throughput and latency of the NVMe SSD compared to the single SATA SSD and four SATA SSDs. Their experiments show improvements in user-level activity, I/O wait time, throughput, and latency when using NVMe-based SSD even against a RAID 0 of four SATA SSDs in the majority of workloads tested, and similar to the RAID 0 in others.

Another work [Brayner and Monteiro 2016] investigates the performance of three commercial DBMSs using both HDD and SSD as a storage technology, arguing that, in the future, these systems should be aware of the underlying hardware. The evaluation uses the TPC-H and TPC-E workloads and SATA SSDs. In their experiments, the authors show that instead of significantly outperform HDDs, SSDs perform only a couple of orders of magnitude better. In that work, they argue that if DBMSs do not consider the asymmetry of write and read operations in write-intensive workloads, the SSD performance is compromised.

In [Liu and Salem 2013], the authors describe a cost-aware buffer pool management algorithm for DBMSs with hybrid storage consisted of SSD and HDD that recognizes that page access patterns change when the page moves between the SSD and the HDD. In MySQL InnoDB, they also implement an automatic manager that chooses which tables are stored on SSD to work along with the buffer manager. An HDD stores all pages, and an SSD stores a copy, working as an intermediary buffer. When the DBMS must read a page, it checks if it is in the buffer; otherwise, it looks for it in the SSD and loads it in the buffer. The authors conduct the experiments with TPC-C workloads to compare their approach with standard techniques and other works. They compare their buffer manager with the standard LRU based one implemented in InnoDB.

The work [Höppner et al. 2014], in turn, proposes a DRAM approach, with a hybrid memory device, consisting of a random memory and a flash component, and HDD for columnar in-memory DBMSs to avoid scaling-out. Their study discusses the impact of Storage Class Memory (SCM) on in-memory DBMSs, using a hybrid memory device to emulate SCM. This approach divides the columns into three categories based on hit frequency, namely hot, warm, and cold. A memory hierarchy is structured to store hot columns in faster devices, going down to cold columns in slower technologies. Columns labels must be frequently updated when the application access pattern changes. For experimental

evaluation, the authors use the columnal in-memory DBMS SAP HANA and the TPC-C workload. They argue that their approach avoids scale-out for a relatively small loss in throughput.

In [Wu et al. 2019], the authors study the use of 3D XPoint Intel Optane SSD as a caching layer between the main memory and the storage flash-based SSD. In their experiments, the 3D XPoint SSD does not perform aways better than the flash-based one. Their experiments indicate that the flash-based SSD is more proper to deal with higher I/O concurrency, as it has internal parallelism. The Optane does not show improvement with high request scale workloads, for example, the TPC-H workload. They argue that the Intel Optane SSD performs better when used as a cashing layer in a system alongside a flash-based SSD, so they propose a caching strategy.

The work [Didona et al. 2020] studies the elements that impact a persistent tree data structure performance on SSDs when doing benchmarks. It also discusses many characteristics of SSDs that may affect the overall performance of a workload. They identify seven of these elements that are often overlooked in empirical evaluations, sometimes leading to biased and inaccurate conclusions. These elements are related to many aspects, as the drive's internal routines and initial internal state, the dataset size in comparison to the drive capacity, and the maintenance routines of the persistent tree data structure. They run experiments using two database systems: RocksDB, which implements an LSM-tree data structure, and WiredTiger, that implements a B+Tree. The paper describes general advice on how to avoid each of these pitfalls, varying from controlling the drive's pre-condition before every experiment to perform evaluations of different dataset sizes. One of the elements discussed is the storage technology impact. They run experiments comparing the performance of three SSDs, two flash-based SSDs, and a 3D-XPoint one, when the system variables are the same, using two datasets.

The focus of the study described in [Eisenman et al. 2018] is reducing the cost of a system without losing too much performance through the partial replacement of the faster but expensive DRAM memory for the slower but cheaper block addressed NVM. The paper describes an implementation of a system on top of MyRocks that optimizes NVM's use as second-layer cash for key-value databases. The approach does a series of modifications to improve performance, as partitioning the index structure and storing in NVM for releasing DRAM space. Also, the approach sets up 6KB blocks so they are not spread in two physical pages when compressed and go further, doing zero-paging when a block cannot be stored in the page's remaining free space. Both of these strategies reduce bandwidth consumption, which improves performance. The experimental analysis compares its implementation against systems with only one DRAM.

Comparing our work with all of those mentioned above, we can highlight two points that are addressed in this work and have not been addressed previously: i) we present and discuss the results of an experimental analysis of a relational DBMS that stores data in three storage technologies separately, one of them is 3D XPoint technology, using two workloads: TPC-C and TPC-DS; ii) we provide two heuristics to combine three storage technologies at the same time to build a hybrid system. These main differences were pointed out and summarized in Table IV.

Table IV. Summary of related works.

Work	Technologies	Hybrid System	Workload
[Liu and Salem 2013]	HDD, SSD	HDD, SSD (as buffer)	TPC-C
[Höppner et al. 2014]	HDD, SSD, DRAM	HDD, Hybrid Device, DRAM	TPC-C
[Xu et al. 2015]	HDD, SSD, DRAM	-	TPC-C, YCSB
[Brayner and Monteiro 2016]	HDD, SSD	-	TPC-H, TPC-E
[Eisenman et al. 2018]	3D XPoint, DRAM	3D XPoint, DRAM	Custom, LinkBench
[Wu et al. 2019]	SSD, 3D XPoint	SSD, 3D XPoint (as cache)	TPC-H
[Didona et al. 2020]	SSD, 3D XPoint	-	Custom
This work	HDD, 3D XPoint, DRAM	HDD, 3D XPoint, DRAM	TPC-C, TPC-DS

6. CONCLUSION AND FUTURE WORK

This paper presents an experimental analysis of the influence that different storage technologies can have on the performance of a relational DBMS, PostgreSQL, when used as the underlying hardware. In addition, we also considered the building of hybrid approach that uses different storage technologies combined. Based on the experimental results, we discuss three questions raised in this work.

In the first question, we are interested in the impacts the replacement of storage technologies can have on DBMSs' performance. The rapid evolution of storage technologies brings significant improvement for DBMSs, both on throughput and latency. Notably, we show that just the direct exchange of HDD for DRAM had a positive impact, up to 24 times in our experiments, on the throughput of a transactional workload. Furthermore, the latency of an analytical workload has been reduced considerably. Specifically, it was reduced by approximately 10 hours, from 16 hours on the HDD to 6 hours on DRAM in our experiments. Deepening the analysis for each query, we could notice that some queries can be a benefit of storage technology exchange and we observed an improvement factor in these queries that reached a maximum of 1500 considering an analytical workload.

After, in the second one, we are interested in finding ways to build a hybrid approach that combines different technologies simultaneously. To answer this question, we proposed two heuristics to distribute the data among storage technologies. The deciding factor of the first heuristic is only relations' size, while we also considered the quantity of reading and writing operations over those relations in the second one. Using TPC-C workload, we generated five configurations (H1 - H5) based on these two heuristics. We used these configurations to set up our hybrid system to assess their cost-benefit.

Lastly, the third question addresses the feasability of these hybrid approaches taking into account their cost-benefit. Thus, as cost of RAM or NVM technologies may be prohibitive, we showed that it is possible to build hybrid storage that improves the cost-benefit, up to 2 times in our experiment, compared to system stored only on RAM.

As for future works, one possible way is to deepen the discussion by studying ways to make DBMSs exploit the new storage technologies potential, for instance, using some machine learning models to learn the parameters according to underlying storage technologies characteristics. Also, to deepen the study of hybrid storage database systems for developing autonomous solutions to take advantage of this type of configuration, as for choosing the best data organization in the hybrid format, dividing relations of complex schemas efficiently for maximizing throughput and latency to both analytical and transactional workloads.

REFERENCES

Barata, M., Bernardino, J., and Furtado, P. An overview of decision support benchmarks: Tpc-ds, TPC-H and SSB. In *New Contributions in Information Systems and Technologies - Volume 1 [WorldCIST'15, Azores, Portugal, April 1-3, 2015]*, Á. Rocha, A. M. R. Correia, S. Costanzo, and L. P. Reis (Eds.). Advances in Intelligent Systems and Computing, vol. 353. Springer, Portugal, pp. 619–628, 2015.

Brayner, A. and Monteiro, J. M. Hardware-aware database systems: A new era for database technology is coming - vision paper. In *31º Simpósio Brasileiro de Banco de Dados, SBBD 2016, Salvador, Bahia, Brazil, October 4-7, 2016*, J. C. Machado (Ed.). SBC, Brazil, pp. 187–192, 2016.

Chaudhuri, S. and Dayal, U. An overview of data warehousing and olap technology. *SIGMOD Rec.* 26 (1): 65–74, Mar., 1997.

Chen, Y., Raab, F., and Katz, R. H. From TPC-C to big data benchmarks: A functional workload model. In *Specifying Big Data Benchmarks - First Workshop, WBDB 2012, San Jose, CA, USA, May 8-9, 2012, and Second Workshop, WBDB 2012, Pune, India, December 17-18, 2012, Revised Selected Papers*, T. Rabl, M. Poess, C. K. Baru, and H. Jacobsen (Eds.). Lecture Notes in Computer Science, vol. 8163. Springer, India, pp. 28–43, 2012.

Didona, D., Ioannou, N., Stoica, R., and Kourtis, K. Toward a better understanding and evaluation of tree structures on flash ssds. *CoRR* vol. abs/2006.04658, pp. 1–15, 2020.

Difallah, D. E., Pavlo, A., Curino, C., and Cudre-Mauroux, P. Oltp-bench: An extensible testbed for benchmarking relational databases. *Proc. VLDB Endow.* 7 (4): 277–288, Dec., 2013.

Eisenman, A., Gardner, D., AbdelRahman, I., Axboe, J., Dong, S., Hazelwood, K., Petersen, C., Cidon, A., and Katti, S. Reducing dram footprint with nvm in facebook. In *Proceedings of the Thirteenth EuroSys Conference*. EuroSys '18. Association for Computing Machinery, New York, NY, USA, 2018.

Hady, F. T., Foong, A. P., Veal, B., and Williams, D. Platform storage performance with 3d xpoint technology. *Proceedings of the IEEE* 105 (9): 1822–1833, 2017.

Höppner, B., Waizy, A., and Rauhe, H. An approach for hybrid-memory scaling columnar in-memory databases. In *International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures - ADMS 2014, Hangzhou, China, September 1, 2014*, R. Bordawekar, T. Lahiri, B. Gedik, and C. A. Lang (Eds.). VLDB Endowment, China, pp. 64–73, 2014.

Kelion, L., BBC, Intel, and Micron. 3d xpoint technology. `https://www.bbc.com/news/technology-33675734`, 2015. Accessed: 2019-07-15.

Kourtis, K., Ioannou, N., and Koltsidas, I. Reaping the performance of fast NVM storage with udepot. In *17th USENIX Conference on File and Storage Technologies, FAST 2019, Boston, MA, February 25-28, 2019*, A. Merchant and H. Weatherspoon (Eds.). USENIX Association, USA, pp. 1–15, 2019.

Kuo, T.-W., Huang, P.-C., Chang, Y.-H., Ko, C.-L., and Hsueh, C.-W. An efficient fault detection algorithm for nand flash memory. *SIGAPP Appl. Comput. Rev.* 11 (2): 8–16, Mar., 2011.

Lepers, B., Balmau, O., Gupta, K., and Zwaenepoel, W. Kvell: the design and implementation of a fast persistent key-value store. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles, SOSP 2019, Huntsville, ON, Canada, October 27-30, 2019*, T. Brecht and C. Williamson (Eds.). ACM, Canada, pp. 447–461, 2019.

Liu, X. and Salem, K. Hybrid storage management for database systems. *Proc. VLDB Endow.* 6 (8): 541–552, June, 2013.

McCallum, J. C. Memory prices (1957-2017). `https://jcmit.net/memoryprice.htm`, 2017. Accessed: 2020-10-28.

Mironov, V., Chernykh, I. G., Kulikov, I. M., Moskovsky, A. A., Epifanovsky, E., and Kudryavtsev, A. Performance evaluation of the intel optane DC memory with scientific benchmarks. In *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing, MCHPC@SC 2019, Denver, CO, USA, November 18, 2019*. IEEE, USA, pp. 1–6, 2019.

Nambiar, R. O. and Poess, M. The making of TPC-DS. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, U. Dayal, K. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y. Kim (Eds.). ACM, Korea, pp. 1049–1058, 2006.

NVM Express, I. Nvm-express revision 1_4a base specification © 2007 to 2019 nvm express, inc. all rights reserved. `https://nvmexpress.org/developers/nvme-specification/`, 2019.

Praciano, F. D. B. S., de Sousa, J. F. L., and Machado, J. C. Uma análise experimental da utilização de diferentes tecnologias de armazenamento em um SGBD relacional. In *XXXIV Simpósio Brasileiro de Banco de Dados, SBBD 2019, Fortaleza, CE, Brazil, October 7-10, 2019*. SBC, Brazil, pp. 259–264, 2019.

Ramakrishnan, R. and Gehrke, J. *Database Management Systems*. McGraw-Hill, Inc., USA, 2002.

Rayner, N., Feinberg, D., Pezzini, M., and Edjlali, R. Hybrid transaction/analytical processing will foster opportunities for dramatic business innovation. `https://www.gartner.com/doc/2657815/hybrid-transactionanalyticalprocessing-foster-opportunities`, 2014. Accessed: 2020-10-28.

Shah, M. A., Harizopoulos, S., Wiener, J. L., and Graefe, G. Fast scans and joins using flash drives. In *4th Workshop on Data Management on New Hardware, DaMoN 2008, Vancouver, BC, Canada, June 13, 2008*, Q. Luo and K. A. Ross (Eds.). ACM, Canada, pp. 17–24, 2008.

Transaction Processing Performance Council (TPC). TPC BENCHMARK™C Standard Specification Revision 5.11 © 2010 Transaction Processing Performance Council All Rights Reserved. `http://www.tpc.org/TPC_Documents_Current_Versions/pdf/tpc-c_v5.11.0.pdf`, 2010.

Transaction Processing Performance Council (TPC). TPC BENCHMARK™DS Standard Specification Revision 2.13.0 © 2020 Transaction Processing Performance Council All Rights Reserved. `http://www.tpc.org/TPC_Documents_Current_Versions/pdf/TPC-DS_v2.13.0.pdf`, 2020.

Wu, K., Arpaci-Dusseau, A., Arpaci-Dusseau, R., Sen, R., and Park, K. Exploiting intel optane ssd for microsoft sql server. In *Proceedings of the 15th International Workshop on Data Management on New Hardware*. DaMoN'19. Association for Computing Machinery, New York, NY, USA, 2019.

Xu, Q., Siyamwala, H., Ghosh, M., Suri, T., Awasthi, M., Guz, Z., Shayesteh, A., and Balakrishnan, V. Performance analysis of nvme ssds and their implication on real world databases. In *Proceedings of the 8th ACM International Systems and Storage Conference, SYSTOR 2015, Haifa, Israel, May 26-28, 2015*, D. Naor, G. Heiser, and I. Keidar (Eds.). ACM, Israel, pp. 6:1–6:11, 2015.

Zukowski, M. Balancing vectorized query execution with bandwidth-optimized storage, 2009.