Enhancing Energy Efficiency of Database Applications Using SSDs

Speaker: Felix Martin Schuhknecht Opponent: Frederic Raber

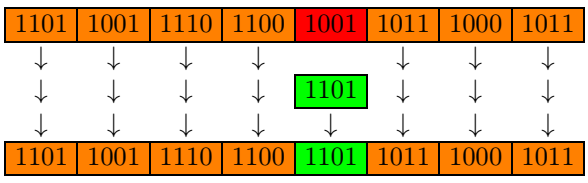
Motivation

Today, flash based disks like so called "Solid State Drives (SSDs)" enjoy greater popularity than ever. The reason for this is, that in the last few years the main problems of SSDs, namely the small capacity and the high price are decreasing rapidly and the trend shows, that these attributes converge against the qualities of HDDs. Therefore, it is necessary to think about the usability of these devices inside of database environments and to analyze the properties and characteristics of them compared to standard HDDs.

Essentially, SSDs are known for providing high performance data access while having a very small power consumption at the same time. Since these properties are very positive features, a main purpose of the papers is to analyze whether and under which circumstances they hold. To do so, they first provide an insight into the architectural details of flash based SSDs compared to HDDs. After that the authors evaluate solid state drives in terms of power consumption, access performance and energy efficiency in various experiments both for pure file reading/writing as well as inside of a real database management system.

Architectural Differences

To be able to compare SSDs to standard HDDs, a comparison of the hardware architecture of both device types has to be performed. SSDs differ from hard disk drives in various aspects. Usually, HDDs have a build in flash based cache which buffers frequently accessed data, whereas SSDs have none. Also, an important difference and drawback of solid state drives is the so called limited "write endurance". This means, that each block can bear only a limited number of write cycles. HDDs don't suffer from this problem. Connected to this issue is the difference of block size. Usually, SSD blocks are much larger than blocks of hard disk drives. This is not a problem itself but is connected to another specific behavior of SSDs. Since only transitions from 1 to 0 are possible, a page update can not be done trivially. To update a page, an unused block has to be erased (to set all bits to 1) and the update page as well as the untouched old pages are copied to the new block. Since this has to be performed frequently, the previously mentioned large block size can have significant influence on the writing performance. Also, a system is needed to manage the allocation of blocks since only an uniform distribution of block usage guarantees a long lifetime of the disk (write endurance problem). This system, which separates a good SSD from a bad one, is called the *Flash Translation Layer (FTL)*. Unfortunately, the behavior of the *FTL* is unknown for most of the disks.



SSD Page Update Problem: The Update has to be propagated to a new block

Experimental Setup

In the following experiments, the authors evaluated three different SSDs against three HDDs, whereas each device type was separated into a low-end, mid-range and high-end disk.

In the experiments, three different access patterns are used. In the sequential access pattern, multiple consecutively stored blocks are accessed sequentially. Random access means that multiple blocks at any position of the disk are accessed in arbitrary order. Since this is not optimal for most of the disks, an optimization is also considered, called random sorted access. Here, the randomly accessed blocks are accessed in the order in which they are physically placed on the disk. This optimization is called *List Prefetch* and often used inside of DBMSs to reduce the problems of HDDs during random access.

Direct I/O Experiments

The following experiments are performed in the so called "Direct I/O" mode, in which pure file reading and writing is performed and as much buffers and caches as possible are disabled to obtain clean and undisturbed results.

Energy Cost

In the first experiment, the authors evaluated the energy cost of SSDs compared to standard HDDs with respect to the three different access patterns for pure file reading. This evaluation lead to two important observations. First of all, the overall energy consumption is much higher for HDDs than for SSDs. This can be explained by the higher power consumption of the hardware itself (i.e. the moving parts) and also by the longer run-time of the operations. The second observation is, that for SSDs the access pattern has nearly no effect on the energy consumption of the disks, whereas for HDDs, sequential access has the cheapest energy cost and random access the highest. This differentiation can be explained with focus on power consumption and on run-time. For HDDs, both run-time and power consumption is directly connected to the amount of head movement that has to be performed during the operation.

Rule of Thumb: Random Access vs. Sequential Access

Connected to access time of hard disk drives is the following popular "rule of thumb": If more than 2% of a file is accessed, it pays off to read the entire file sequentially. To understand the problem, the following formula calculates the access time of b blocks which are stored on an HDD

$$t_d(b) \approx t_{\text{SIO}} + t_{S_{\text{avg}}} + \frac{t_{\text{rev}}}{2} + \frac{b}{F_d}$$

where $t_{S_{\text{avg}}}$ is the average seek time of the disk head to find a block and t_{rev} is the rotational delay of the platters, until the searched block is at the head position. These two factors are exactly those, that lead to the huge difference in performance of sequential and random access, since for each randomly accessed block, these delays occur.

To demonstrate and to prove the problem, the authors measured the performance of the different access patterns with respect to a different amount data read. The results clearly confirm the rule of thumb for HDDs. For SSDs, the results of the experiment look different. As already expected, the difference between random and sequential access is much smaller for SSDs than for HDDs. For the mid-range solid-state drive, it pays off to read the entire file sequentially if more than 83% are accessed. This result supports the previous assumptions: The differences of access is much smaller for flash based drives than for hard disk drives, since the seek-time and the rotational delay don't occur. Nevertheless, there is a small difference in performance for SSDs. Unfortunately, the authors don't explain the details behind this phenomenon.

Power Consumption

In the following experiments, all six disks are evaluated against each other and all three access patterns are tested. The authors start with the analyze of the pure power consumption (measured in Watt). Different observations can be made by evaluating the results of the experiment. First of all, the raw power consumption of HDDs is by a factor of three to four higher than for SSDs, no matter which access type is used. Connected with this, it is interesting to see that the access pattern has basically no influence on the power consumption, neither for solid-state drives nor for hard disk drives. The explanation for this behavior can be made by looking at the hardware again. The motor of the spindle inside the HDDs is the main factor of the overall power consumption. This does not exist for SSDs and therefore, the power consumption is lower. Unfortunately, the authors don't give an explanation for the constant power consumption of HDDs under different access types. Since more arm movement has to be performed for random access than more sequential access, one would expect a higher power consumption in this case. It seems as if the power consumption for the head movement is negligible compared to the one caused by the spindle motor, but this is not concerned in the paper in detail. Also, reading or writing has basically no effect on the power consumption since in both cases, the hardware has to perform the same movements. If we compare the disks to each other, we can observe that the differences in power consumption are overall much smaller for SSDs than for HDDs. The explanation for this is, that for hard disk drives, the class of the disk is highly connected to the rotation speed and the capacity (which is connected to the number of platters to rotate). For SSDs, the class is mainly influenced by the internal firmware and the *FTL*.

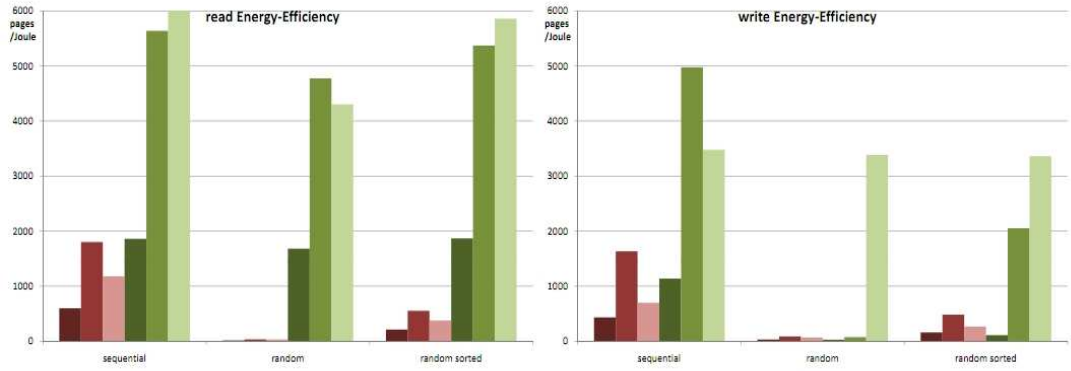
Performance

Next, the performance, measured in processed pages per second of the disks is evaluated in the same setup as before. First of all, sequential reading and writing is comparable for HDDs and SSDs. This is totally different for random access, where reading is the worst-case scenario for hard disk drives because of the heavy head movement, that has to be performed. This is were the SSDs can show their strength. Sorting the random access decreases the gap between HDDs and SSDs, nevertheless the latter ones are still dominant. For writing, the situation is different. Here, also some of the SSDs have huge problems, because of the shadow block copying mechanism described above. Only the high-end SSD can solve this problem. How the manufacturer is doing this is unfortunately unknown. This experiment clearly shows, that SSDs have strong potential in decreasing the random access problem. Nevertheless, also SSDs can suffer from a low random write rate.

Energy Efficiency

Both power consumption and processing performance influence the overall energy efficiency of a device. This efficiency, measured in processed pages per Joule is evaluated in the following experiment, which clearly shows that energy efficiency is way higher for SSDs than for HDDs in nearly all situations. The overall processing time is smaller for solid state drives and at the same time, the power consumption is lower. These two facts lead to the superiority of SSDs over HDDs in terms of energy efficiency.

To explain the previous results, we can make the following conclusion. For HDDs the main factors that influence performance, power consumption and thus energy efficiency are hardware specifications like rotation speed and capacity of the disk. The faster the spindle is rotating, the higher is the processing performance and at the same time the power consumption. The more platters are inside the disk, the more power is drained by the motor. For SSDs, this is not the case since the software, which is running inside of it highly influences the performance. This also means that it is much harder to estimate the behavior of a SSD than for an HDD.



Energy Efficiency of HDDs (red bars) compared to SSDs (green bars)

XTC DBMS Experiments

The previous experiments where performed in the *Direct I/O* mode, which consists of pure file reading and writing. Since real database management systems are much more complex, it is necessary to evaluate the disks inside of such an environment. For example, a real DBMS contains different kind of data like index structures or log files with different access patterns and the physical layout of the data can vary. Different query types can occur and many buffers and caches can have an influence on the performance of the system. Therefore, the following experiments are performed using the XTC DBMS.

XTC Architecture

To understand the behavior of the system, we will give a small introduction at this point. The DBMS stores XML tree structured data. Each node of the tree is identified by an SPLID (Stable Path Labeling IDentifier), an hierarchically formed number. The schema of the data set is stored inside of path class references. The actual data is stored in the following manner on the disk: A B*-tree is used as an index structured based on the SPLIDs for a fast lookup. The data itself is stored on disk in deep-first order, where the SPLIDs are stored together with the attribute names and the actual content. This way of storing the data is called the *full* representation. The authors also use two kind of compression techniques to reduce the amount of data and thus the I/O the disks have to handle. The first compression is called *prefix-compressed (pc)* representation and applies difference coding on the SPLIDs. This reduces the size of the *full* representation by 25%, because the hierarchically build node identifiers compress very well. The second compression is called *elementless* representation, where references to the path classes are stored instead of the attributes themselves. This saves 5% of data.

Optimizations

During their experiments, the authors have to deal with many optimization techniques that are running in the background and which try to increase the processing speed. Many of them come in the form of caches and buffers, that try to keep frequently accessed data inside of main memory to avoid heavy disk access. Unfortunately, it is hard to evaluate results if uncontrollable optimizations are performed during the tests, especially when disk performance is measured. The authors demonstrate this problem by investigating random read access performance for HDDs and SSDs with cold and hot caches. For empty buffers, the random access performance for SSDs is, as expected, much higher than for HDDs, no matter how large the read dataset is. In contrast to that, for a full buffer the access performance for both disk types is the same if not more data is read than the buffer can hold. This is because all data is read from main memory instead of reading it from disk. This also shows, that a lot of optimizations are performed to reduce the random access problem of standard HDDs. Some of them might be no more necessary, if disks are used that do not suffer from this problem.

To have valuable and explainable results, the authors tried to disable as much caches as possible. Only the DBMS cache and the internal HDD cache are still activated.

XTC Experiment: Read Performance

In the experiment, all three dataset representations are used and the size of the underlying dataset is varied from 10MB to 10GB. The queries, which are uniformly chosen from an XML benchmark (TPoX), are divided into I/O bound and CPU bound queries. In the first run, 1000 transactions are performed on the dataset. First, we investigate the I/O bound queries. For a small dataset (10MB) SSD and HDD performance is basically the same. The reason for this is that most of the used data can be stored inside of the internal HDD cache. If the dataset grows, the differences between the performance of SSDs and HDDs also grow. As expected, SSD perform better by a factor of two to three. For the CPU bound queries, the situation is different. Here, the gap between SSDs and HDDs is much smaller, since now the disk is no more the bottleneck. Interesting is the behavior of the system, when 10000 transactions are performed. In this case, the difference in performance between the two disk types is smaller than before. The reason for this is again an applied optimization. The more transaction are performed, the more parts of the index structure are loaded into the DBMS buffer. This reduces the random access, that has to be performed on the disk and thus closes the gap a little bit.

XTC Experiment: Write Performance

In the following, the performance of writing transaction is introduces into the experiments. The authors mix reading and writing transactions in certain proportions from 10% to 30% of writing transactions. The result clearly shows, that the more writing transactions are performed, the smaller is the performance gap between SSDs and HDDs. The reason for this is, that the main problem of older SSDs is the bad writing performance.

Conclusions

Overall, it can be said that solid state drives are superior both in performance and in energy efficiency over HDDs in nearly all cases. Still, the gap between SSDs and HDDs is increasing further and high-end SSDs outperform hard disk drives by far. On the other hand, these disks are still much more expensive and therefore, it is unlikely that companies will replace their entire HDD set by solid state disks at once.

In the next years, we will encounter "hybrid systems", in which SSDs are integrated into existing HDD dominant environments. In such a situation, new challenges arise. For instance, a storage awareness of the DBMS is needed in the first place. This would allow the system to distinguish between the underlying disk types and is the precondition for any further automatic optimization. Current DBMSs don't distinguish between the types of running disks.

Based on that, a smart and automatic data distribution system would be the next step, in which frequently and randomly accessed data like index structures are placed on SSDs while rarely accessed data like archives or log files are stored on HDDs.

As previously mentioned, a lot of optimizations are performed to reduce the problems of HDDs with random access like *List Prefetch*. It might make sense to remove them in SSD dominant environments to save energy and run-time, since many CPU cycles are spend to compute them.

Finally, the highly efficient standby mode might be exploited to enhance the energy efficiency. Restoring an HDD from standby mode can take several seconds and therefore, the disk is set into this mode rarely. In contrast to that, SSDs can wake up from standby mode instantly and so, they can sleep much more often.

The previously mentioned points demonstrate, that a lot of further research has to be done to answer the question how to use SSDs in hybrid database environments efficiently. In non-hybrid environments, SSDs are dominant over HDDs in each aspect today.

Evaluation of the Papers

Positive Aspects

- **Clean Comparison of Disk Types:**

  From their point of view, the work they did in their papers makes sense. Back in 2008/2009, when they started their research on SSDs, the situation was different than today. The early SSDs, especially the cheaper ones, had some significant problems with writing access and the write endurance was also much more limited than today. Therefore, it makes sense to compare SSDs to HDDs in all aspects to clarify the positive and negative aspects of these disks. Especially when it comes to the high prices, it is necessary to be clearly informed about the drawbacks before buying such a drive.

- **Differentiation of Direct I/O Mode and DBMS:**

  Very helpful in understanding the topic is the separation of the Direct I/O mode from a real DBMS. Since DBMSs are very complex and consist of many components that influence the overall behavior of the system, it makes sense to provide experiments for pure file reading and writing. In this simplified environment, the disk specifications can be tested without any disturbance. Based on this results, the disks can be evaluated in a real DBMS. Unfortunately, they performed the experiments in the wrong order, i.e. they tested the hardware first inside of XTC and after that, they tried newer hardware in Direct I/O mode. Nevertheless, the reader of both articles is able to draw valuable conclusions of the two experiments.

Negative Aspects

- **Outdated Hardware**:

  In both of the papers, one main problem can be identified. Since the papers have been published in the year 2010 respectively 2009, the used disks are already a few years old by now. This has a huge impact on the value of their results, if we look at them by today's standards. SSDs have been improved in nearly all aspects drastically in the last years and therefore, the results can not be conveyed to modern hardware. From an up-to-date perspective, (high-end) SSDs outperform HDDs in any case by far. This was not the case a few years ago, where SSDs still had problems in certain situations (random write, write endurance). Therefore, it made sense to investigate the behavior of SSDs, especially compared to HDDs.

- **Hardware Knowledge:**

  Throughout the papers, a lot of experiments are performed that measure the performance of SSDs and HDDs. The results of these tests are evaluated after wards and try to explain the behavior of the disks. Unfortunately, it is often the case that the explanations are too much from an high-level perspective, especially for SSDs, to be able to understand *why* these disks perform like they did. A few left open questions from these papers are: "Why is random access still slower than sequential access for SSDs?" or "Why must the block size of SSDs be so large?".

- **Using XTC DBMS:**

  Unfortunately, they did not use a more popular DBMS like for example PostgreSQL, but an XML store named XTC. In my opinion, the results would have been more valuable if they would have used a relational DBMS and a popular benchmark. There are several reasons imaginable why they used the XTC store. First, XTC was developed at the same university and therefore, promotion of the own system might be a major reason. It might also be the case that their XML store suffered from heavy random access and they tried to reduce it by using solid state drives.