Page Size Selection for OLTP Databases on SSD Storage

Ilia Petrov, Todor Ivanov, Alejandro Buchmann

Databases and Distributed Systems Group, Department of Computer Science,
Technische Universität Darmstadt

{petrov | ivanov | buchmann}@dvs.tu-darmstadt.de

Abstract. *Flash SSDs are a technology that has the potential of changing the database architecture and principles. We reevaluate present trend of growing database page sizes considering its validity for SSD-based database storage. Our major findings are: (a) on Flash storage this trend is reverted and best OLTP performance can be attained with smaller page sizes; (b) DBMS with smaller page sizes require less buffer space while providing the same performance; (c) due to the lower response times we report better CPU utilization for small page sizes.*

1. Introduction

Over past two decades the standard page size used by database systems has been continuously growing. This is viewed as an established trend by the database community [ScAG03, GrGr97, Grae08]. Larger page sizes and hence larger database buffers compensate for the poor IO properties of magnetic hard drives, minimizing the so called access gap. Over the last two decades the accesses per second and the transfer rate improved only about 10 times, while capacities grew by a factor of 1000 [Grae08,GrGr97]. Hard disks as storage technology have reached their physical limits hence no significant technological improvement can be expected. Flash Solid-State Disks (SSDs), on the other hand, are a disruptive technology, that has the potential of changing the established principles of DBMS architecture. In comparison (Table 1), SSDs exhibit low latency and very high random throughput (accesses per second or simply IOPS–Input/Output Operations Per Second) especially for small block sizes.

Please consider the following introductory example: while the typical hard drive's random throughput remains constant for block sizes between 4KB and 32KB, the SSD performance ranges significantly (up to seven times) within the same block range (Figure 3, Table 1) – the smaller the block size the higher the random throughput. *This fact influences significantly the OLTP database performance on SSD storage*. In the present paper we explore this hypothesis.

Table 1. Comparison of enterprise HDDs, SSDs

Device	Seq. Read [MB/s] (128K)	Seq. Write [MB/s](128K)	Rand. Read [ms] (4 KB)	Rand. Write [ms] (4 KB)	Rand. Read [ms] (16 KB)	Rand. Write [ms] (16KB)	Read IOPS (4 KB)	Write IOPS (4 KB)	Read IOPS (16 KB)	Write IOPS (16 KB)	Price [€/GB]
E. HDD	160	160	3.2	3.5	3.3	3.4	291	287	288	285	2.5
E. SSD	250	180	0.161	0.125	0.294	0.377	35 510	5 953	12 743	3 665	10

The contributions of the present work can be shortly summarized as follows:

(a) SSD storage characteristics revert the trend of increasing page sizes of database systems. We claim that for OLTP databases a *smaller 4KB page size is better choice* than a larger one, e.g. 16 KB.

(b) *Smaller block sizes relax the demand for essential buffer space*. Larger buffers can be used to additionally improve performance by buffering more data or for providing space for maintenance operations such as index rebuilding etc.

(c) We claim that all database systems (not only several commercial ones) should support *multiple dynamically configurable* block sizes. And the "default" block size should be smaller (in the range of 4KB to 8KB) since it influences the database catalogue.

(d) Last but not least *higher CPU utilization can be observed for OLTP databases*, which are typically IO-Bound environments. This is a result of the lower response times for small block operations. This increased CPU demand is a natural fit for the multi-core CPU trend.

The present paper is structured as follows: we continue by examining the IO properties of Flash SSDs and describing the system under test. Next we investigate the database page size influence on the performance of an OLTP database. We use TPC-C as a standard OLTP workload. Last but not least we summarize our findings.

1.1 Related Work

There is a large body of research on the properties of NAND SSDs [ChKZ09, APW+08], the design of Flash Translation Layer. Research influence of Flash SSDs in the database field [LeMP09, LMP+08] reflects primarily logging [LeMo07], indexing [YBH+09], page organization for analytical loads and its influence on joins [SHWG08, DoPa09]. There are new algorithms and data structures emerging. They address issues such as indices, page formats, logging and log record formats [NaKa07, LeMo07, SHWG08, YBH+09]. [Grae08] outlines the influence of SSDs on 5-Minute-Rule and discusses the influence of flash properties the node utility metric and on the page size of an B-Tree database storage. [Grae08] proposes an optimal page size of 2KB. A detailed analysis of the database page size influence on performance does not exist.

2. Enterprise Flash SSDs

The performance Flash SSDs is characterized through: low latency (Table 3); very high random throughput (Figure 1); acceptable sequential performance (Figure 2); low power consumption. In the following we extend on these points.

(a) asymmetric read/write performance – the read performance is significantly better than the write performance – up to an order of magnitude (Figure 1, Figure 2). This is due to the internal organization of the NAND memory, which comprises two types of structures: pages and blocks. A page (typically 4/2 KB) is a read and write unit. Pages are grouped into blocks of 32/128 pages (128/512KB). NAND memories support three operations: read, write, erase. Reads and writes are performed on page-level, while erases are performed on block level. Before each write, the whole block containing the page must be erased, which is a time-consuming operation. The respective latencies are:

read-55µs; write 500µs; erase 900µs. Writes should be evenly spread across the whole volume (longevity, wear-leveling). Hence no write in-place as on HDDs.

(b) excellent random read throughput (IOPS) – especially for small block sizes. Small random reads are up to hundred times faster than on an HDD (Table 1). The good small block performance (4KB, 8KB) affects the present assumptions of generally larger database page sizes.

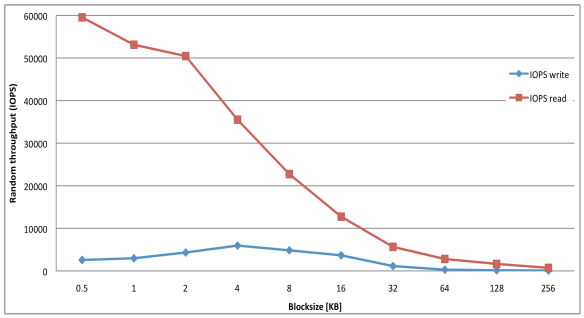


Figure 1: Random throughput (IOPS)

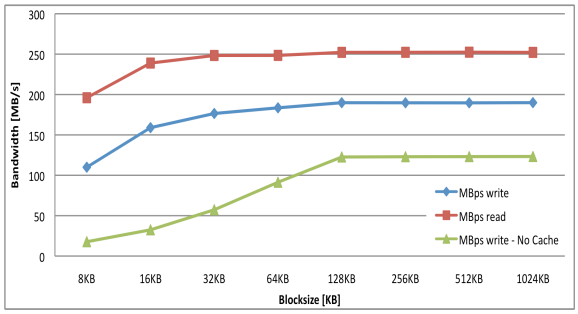


Figure 2: Sequential throughput (MB/s)

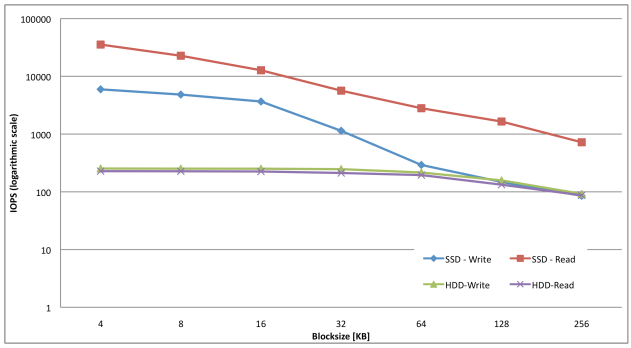
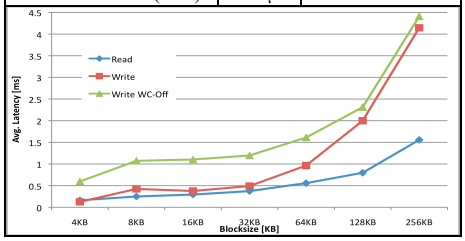


Figure 3: Random throughput HDD, SSD

Table 2. Avg/Max latency (4KB)

Sequential Read (SR)	53 µs	max 12.29 ms
Sequential Write (SW)	59 µs	max 94.82 ms
Random Read (RR)	167 µs	max 12.41 ms
Random Write (RW)	125 µs	max 100.68 ms



(b) acceptable random write throughput – small random writes are 5x to 10x slower than reads (Figure 1). Nonetheless, the random write throughput is an order of magnitude better than that of an HDD (Table 1). Unfortunately it changes over time.

(c) very good sequential read/write transfer. It is commonly assumed that HDDs are better for sequential operations. The newer generations of SSDs perform significantly better (Figure 2, Table 1).

(d) Command Queuing (CQ) allows several IO requests to be executed in parallel. It enables database systems to successfully use asynchronous paged I/O in OLTP environments where traditional blocked I/O cannot be used. CQ is very beneficial for small random reads, where doubling the queue depth (up to eight commands) doubles the throughput while keeping the latency almost constant below 0.28 ms (for 8KB block size). The reason for this improvement is the better utilization the internal SSD parallelism and request interleaving capabilities. CQ has less of an effect on random write, where the performance is only marginally better. There are significant benefits of CQ for sequential read or sequential write because it translates to read ahead or write-back. In these cases increasing the queue depth (up to 32 commands) increases the data transfer rate up to saturation while keeping the latency almost constant below 0.3 ms (64KB block size).

3. Experimental Setup

We investigated our hypotheses by performing TPC-C experiments on a testbed comprising a MySQL database and SSD storage described below. The used benchmark - DBT2 [DBT10] is an open source TPC-C implementation [TPCC10] on top of MySQL version 5.1.44 with innoDB 1.06. DBT2 is instrumented according to the TPC-C specification (Section 4.2.2 of the TPC-C specification [TPCC10]), i.e. to use 20 database connections and 10 terminals per warehouse. The standard MySQL codebase uses a static page size of 16KB. In addition we reconfigured and recompiled it for a page size of 4KB. For all experiments the block size (of the file system) and the database page size are configured to be identical. The TPC-C tests were performed on both versions for different CPU and RAM configurations. We measure the average response time and the average throughput in new order transactions per minute (NoTPM) by varying the number of warehouses, to increase the load on the system. The dataset contains 800 warehouses amounting to approx. 100GB. As defined in Section 4.2 of the TPC-C specification [TPCC10] we preserve the specification defined ratio of connections and terminals (and hence transactions) per warehouse – therefore the only way to increase the load on the system is to increase the dataset in terms of warehouses(Figure 4). (This not only loads the IO-subsystem, it also increases the memory demand of the buffer manager.) The used server is a SUN Fire x4440 with four quad-core AMD processors, 64GB RAM and enterprise-level RAID controllers with 512MB cache with 8 SSDs running under Windows 2008 Server R2. Depending on the type of experiment the resources are limited.

We measured the performance of the IO-system initially with IOMeter [Iome10] and validated the results with Oracle Orion [Orio10]. Table 3 reports the sequential and random throughput as well as sequential and random latency for different block sizes.

Table 3: Performance of SSD storage. (a) top-left Sequential throughput and latency; (b) top-right random throughput and latency

BlockSize	Seq. Read MB/s	Seq.Write MB/s	Avg. Read Latency[ms]	Avg.Write Latency[ms]
8KB	391	547	0.179	0.111
16KB	672	781	0.200	0.127
32KB	972	897	0.283	0.176
64KB	1341	792	0.323	0.255
128KB	1350	848	0.460	0.426
256KB	1349	881	0.606	0.669
512KB	1350	891	0.742	1.160

BlockSize	Rand. Read IOPS	Rand. Write IOPS	Avg. R.Read Latency [ms]	Avg. R.Write Latency [ms]
4KB	51164	23661	0.276	0.106
8KB	46156	21372	0.330	0.116
16KB	36448	15893	0.430	0.135
32KB	26176	12065	0.512	0.171
64KB	16623	7542	0.651	0.242
128KB	9674	3274	0.728	0.359

Based on the figures in Table 3 and general database experience we can draw the following conclusions: (i) In contrast to HDD storage SSDs storage can offer high sequential bandwidth and high random throughput at the same time. For database systems this requires database support for multiple concurrent page-sizes to harness the random and sequential performance. (ii) performance is asymmetric (better reads than writes). (iii) Sequential throughput is naturally better for larger block sizes, while the random throughput is better for small block sizes. Please notice the performance difference between 4KB and 16KB blocksize. OLTP databases should use small page sizes to harness the random performance. (iv) The lower the block size the lower the random latency (Table 3 (b)). (v) extensive use of command queuing (asynchronous I/O) improves the performance significantly. For OLTP databases the use of blocked IO

can be successfully replaced by asynchronous paged IO due to the good command queuing behavior.

4. Performance Influence of Database Page Sizes

The database research community has widely recognized the trend of growing database page sizes to compensate for the IO characteristics of magnetic disks. Small random accesses (omnipresent in OLTP environments) are a weakness of existing HDDs. For large sequential operations (blocked IO) the HDD efficiency increases since the transfer rate dominates over the positioning costs. Large blocks however lead to larger database buffers. As pointed out by [ScAG03, ScAG03, Grae08] there is a compromise between IO efficiency and buffer size. However a larger page size stands in stark contrast to the characteristics of SSDs.

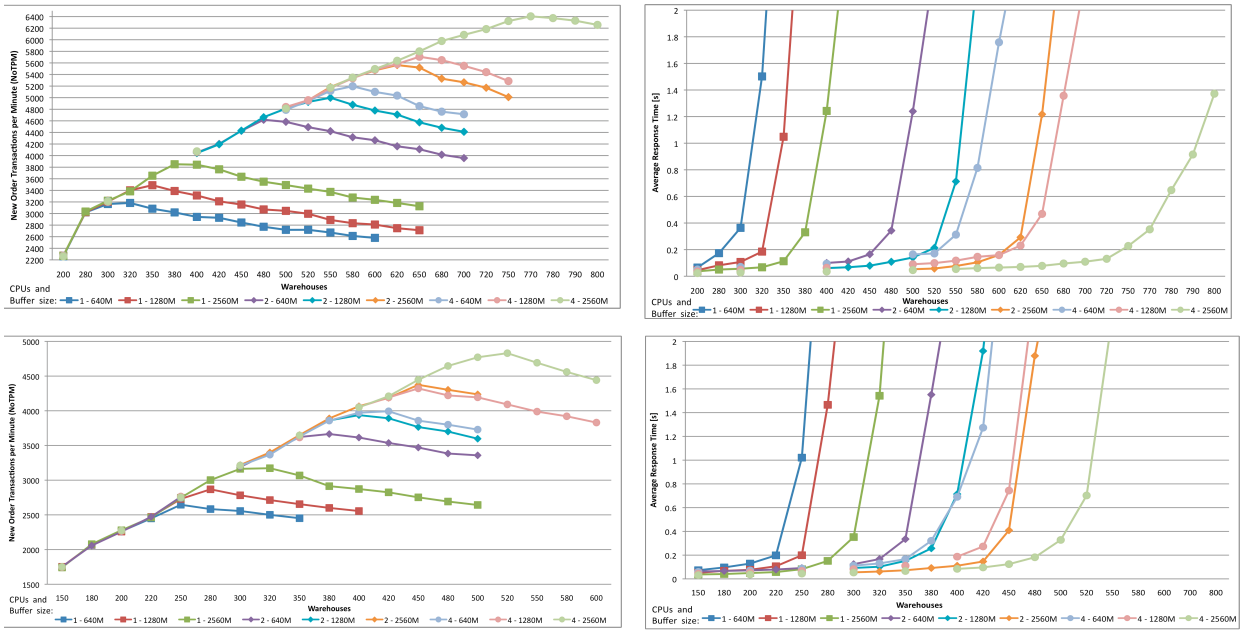


Figure 4 (a) top-left - throughput NoTPM, 4KB; (b) top-right - response times 4KB; (c) bottom-left - throughput NoTPM, 16KB; (d) bottom-right - response times 16KB

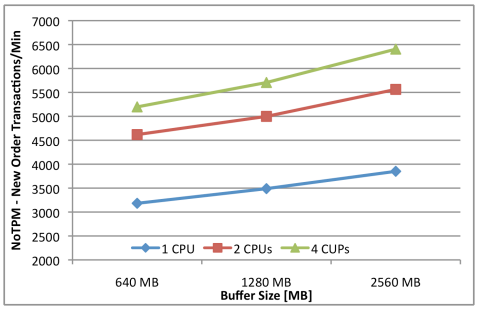
Figure 4 summarizes the test results of the TPC-C experiments for different page sizes, buffer sizes and CPUs. We clearly observe a **30% performance improvement** in transaction throughput (NoTPM) due to the page size performance influence (4KB over 16 KB). This very number can also be derived from the results in Table 3 considering the 75%/25% read/write ratio DBT2 exhibits. The measured transaction throughput improvement can be therefore clearly attributed to the SSDs characteristics. These figures substantiate the claim that SSD storage reverts the trend towards larger page sizes. We claim that depending on the use of indices versus direct table operations the optimal page size is between 2KB and 4KB.

Table 4 Max. TPC-C throughput (NoTPM) for different buffer sizes and number of CPUs

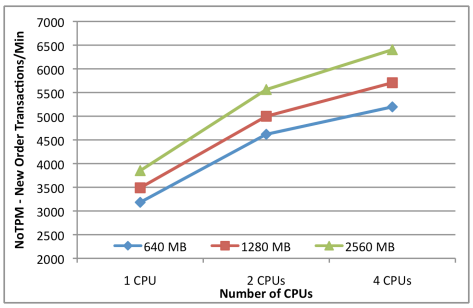
DB Buffer	1 CPU		2 CPUs		4 CPUs	
	P.Size 4KB	P.Size 16KB	P.Size 4KB	P.Size 16KB	P.Size4KB	P.Size 16KB
640MB	3183	2646	4619	3665	5197	3993
1280MB	3489	2868	4999	3937	5707	4322
2560MB	3851	3173	5563	4374	6404	4830

Table 5 Response times (New Order Transactions) in [s] for different buffer sizes and number of CPUs for the maximum throughput values in Figure 4.

DB Buffer	1 CPU		2 CPUs		4 CPUs	
	P.Size 4KB	P.Size 16KB	P.Size 4KB	P.Size 16KB	P.Size4KB	P.Size 16KB
640MB	1.49	1.71	0.92	1.53	0.76	1.23
1280MB	1.06	1.47	0.70	0.70	0.44	0.71
2560MB	0.35	1.56	0.30	0.51	0.34	0.68



(a) different buffer sizes – CPU constant



(a) different CPUs – buffer size constant

Figure 5: Increase in transactional throughput (NoTPM) with respect to CPUs and Buffer Sizes (figures based on Table 4 data, 4KB page size)

Let us now consider Table 4 (which contains the NoTPM maxima from Figure 4) and the relation between buffer size, page size and transaction throughput. Clearly for the same number of CPUs and the same buffer size there is a 30% performance advantage for the 4KB page size over 16KB. Consider the data for four CPUs (Table 4): the NoTPM throughput for 4KB page size and 640MB buffer size is comparable (7% better) than the throughput for 16KB page size and 2560MB buffer size (page size and buffer size are four times larger). The same result is visible in the two CPU data. This observation is even more interesting in light of the TPC-C access skew [HsSY01] due to which comparatively small database buffers can significantly reduce the number of accesses (for instance buffering 10% of the database pages reduces 50% of all accesses). On this basis we can conclude that *smaller page sizes relax the demand for essential buffer space*.

Table 5 shows the average response times for the New Order transaction ([TPCC10]) for the maxima (Table 4). It can be clearly seen that the difference of transaction throughput is reflected at the response times as well. The maxima in transaction throughput for 4KB page size are not only 30% higher they are also achieved at lower response times. This results should not be surprising – it is a direct consequence of the results described in Section 1 (Table 2) and Section 6 (Table 3 (b)).

Last but not least OLTP databases on SSD storage exhibit good CPU utilization due to the lower response times. The transaction throughput and response times in Table 4 and Table 5 improve with the higher number of CPUs. Figure 5 depicts the transactional throughput over the different database buffer sizes (a) and number of CPUs (b). We observe an increase in NoTPM with more CPUs and larger buffer sizes. More importantly it seems that *doubling the CPUs and doubling the buffer size yields the similar increase in the transactional throughput* (CPU increase has slightly stronger effect). Although the curves on Figure 5 (b) are will possibly flatten with 8 CPUs this is still a very interesting observation in an IO-bound environment under test. HDD based

systems will not exhibit such behavior: they will be influenced more by the buffer increase and remain practically unaffected by CPU increase (in the same resource range). Clearly the higher random performance and lower latency on the SSD storage yield better CPU utilization. Hence the demand for more CPU power which leverages the multicore-CPU trend.

7. Conclusions

The access gap between memory and HDD has been constantly increasing due to hardware developments. To compensate for the low random performance, page sizes for OLTP database systems have been growing (16KB, 32KB). SSDs however offer superior random performance for smaller block sizes (4KB), which reverts the established trend towards larger page sizes.

In the present paper we proved this hypothesis by performing TPC-C experiments on a database system configured with 4KB and 16KB page sizes on SSD storage. We observe a 30% performance and response time improvement for the smaller block size. In addition we see that on SSD storage databases with smaller page sizes require proportionally less buffer space while offering comparable performance. Last but not least OLTP databases on SSD storage exhibit good CPU utilization due to the lower response times. Since these vary with the page size the smaller the page size the higher the CPU utilization.

SSD storage can offer both good random and sequential throughput depending on the block size. It is therefore important to support multiple concurrent block sizes to accommodate the requirements of different database objects types and access patterns.

Acknowledgement

This work has been partially supported by the DFG project "Flashy-DB".

References

[HsSY01] Hsu, W. W., Smith, A. J., and Young, H. C. 2001. Characteristics of production database workloads and the TPC benchmarks. IBM Syst. J. 40, 3 (Mar. 2001), 781-802.

[LMP+08] Lee, S.-W., B. Moon, C. Park, J.-M. Kim, S.-W.Kim. A Case for Flash Memory SSD in Enterprise Database Applications. In Proc. of the ACM SIGMOD, pp. 1075–1086, 2008.

[Orio10] Oracle Corp. ORION (Oracle I/O Calibration Tool) http://oracle.com/technology/software/tech/orion/index.html

[Iome10] IOMeter project. www.iometer.org.

[ChKZ09] Chen, F., Koufaty, D. A., and Zhang, X. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In Proc. of SIGMETRICS '09 (Seattle, WA, USA, June 15 – 19), 2009

[APW+08] Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., Panigrahy, R. Design tradeoffs for SSD performance. In USENIX08 Boston, MA, June 2008.

[LeMP09] Lee, S., Moon, B., and Park, C. 2009. Advances in flash memory SSD technology for enterprise database applications. In Proc. SIGMOD, Rhode Island, USA, July 2009

[LeMo07] Lee, S. and Moon, B. Design of flash-based DBMS: an in-page logging approach. In Proceedings of the 2007 ACM SIGMOD international Conference on Management of Data (Beijing, China, June 11 - 14, 2007). SIGMOD '07.

[YBH+09] Yinan Li, Bingsheng He, Qiong Luo, Ke Yi, Tree Indexing on Flash Disks, ICDE, pp.1303-1306, 2009 IEEE ICDE, 2009

[SHWG08] Shah, M. A., Harizopoulos, S., Wiener, J. L., and Graefe, G. 2008. Fast scans and joins using flash drives. In Proceedings of DaMoN '08 (Vancouver, Canada, June 13 - 13, 2008).

[DoPa09] Do, J. and Patel, J. M. 2009. Join processing for flash SSDs: remembering past lessons. In Proceedings of the DaMoN '09 (Providence, Rhode Island, 2009).

[NaKa07] Nath, S. and Kansal, A. 2007. FlashDB: dynamic self-tuning database for NAND flash. In Proceedings of the 6th international Conference on information Processing in Sensor Networks (Cambridge, Massachusetts, USA, April, 2007).

[Grae08] Graefe, G. 2008. The Five-minute Rule 20 Years Later: and How Flash Memory Changes the Rules. Queue 6, 4 (Jul. 2008), 40-52.

[ScAG03] Schindler, J., Ailamaki, A., and Ganger, G. R. 2003. Lachesis: robust database storage management based on device-specific performance characteristics. In Proceedings of VLDB (Berlin, Germany, September 09 - 12, 2003)

[DBT10] Database Test Suite. DBT2. http://osdldbt.sourceforge.net/

[GrGr97] Gray, J. and Graefe, G. 1997. The five-minute rule ten years later, and other computer storage rules of thumb. SIGMOD Rec. 26, 4 (Dec. 1997)

[TPCC10] TPC Benchmark C. Standard Specification. Revision 5.11. February 2010 http://www.tpc.org/tpcc/spec/tpcc_current.pdf