Page Size Selection for OLTP Databases on SSD RAID Storage

Ilia Petrov, Robert Gottstein, Todor Ivanov, Daniel Bausch, Alejandro Buchmann

Databases and Distributed Systems Group, Department of Computer Science,
Technische Universität Darmstadt
{petrov, gottstein, ivanov, bausch, buchmann}@dvs.tu-darmstadt.de

Abstract. Flash SSDs are a technology that has the potential of changing the database architecture and principles. We reevaluate the 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.

Categories and Subject Descriptors: H.2.4 [**Database management**]: Systems; H.2.2 [**Database management**]: Physical Design; H.2.7 [**Database management**]: Administration

Keywords: Database Systems, Page Size, Flash Solid State Disks, RAID

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 [Schindler et al. 2003; Gray and Graefe 1997; Graefe 2007]. 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 [Graefe 2007; Gray and Graefe 1997]. 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 4, Table 2, Table I). The seek time dominates the overall response time of hard drives; for smaller block sizes (between 4KB and 32KB) there is a negligible change in the random performance. Regarding SSDs the smaller the block size the higher the random throughput due to the lack of seek time. This fact influences significantly the OLTP database performance on SSD storage. In the present paper we explore this hypothesis in a TPC-C testbed using the MySQL database system.

Table I. 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:

—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.

—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.

—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.

—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 Scope and Topics

There is a large body of research on the properties of NAND SSDs [Chen et al. 2009; Agrawal et al. 2008a], the design of Flash Translation Layer. Research influence of Flash SSDs in the database field [Lee et al. 2009; Lee et al. 2008] reflects primarily logging [Lee and Moon 2007], indexing [Li et al. 2009], page organization for analytical loads and its influence on joins [Shah et al. 2008; Do and Patel 2009]. There are new algorithms and data structures emerging. They address issues such as indices, page formats, logging and log record formats [Nath and Kansal 2007; Lee and Moon 2007; Shah et al. 2008; Li et al. 2009]. [Graefe 2007] 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. [Graefe 2007] 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 AND SSD RAID CONFIGURATIONS

The performance Flash SSDs is characterized through (Table I): asymmetry, very high random throughput; high sequential performance; low latency ; low power consumption. The basic characteristics of the Flash SSDs are well documented [Chen et al. 2009; Agrawal et al. 2008b; Hudlet and Schall 2011]. These can be summarized as follows: (a) asymmetric read/write performance - the read performance is significantly better than the write performance, up to an order of magnitude. This is due to the internal organization of the NAND memory and FTL algorithms. (b) excellent random read throughput (IOPS) - especially for small block sizes. (c) acceptable random write throughput -

small random writes are 5x to 10x slower than the random reads. The random throughput also deteriorates over the time. (d) very good sequential read/write transfer. Although it is still commonly assumed that HDDs have higher sequential throughput, however newer generations of SSDs perform significantly better. (e) IO Parallelism and 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.

Modern database systems manage growing constantly growing data volumes. Using SSDs as primary database storage for large data volume yields special challenges related to producing sizable SSD space: (a) for the time being a single SSD device has a relatively low volume; (b) the performance and reliability bottlenecks of a single device can be avoided by a combination of devices. It seems that the RAID technology may deliver the right answer, however as reported by [He et al. 2010; Petrov et al. 2010] classical RAID configurations are plagued by performance and scalability bottlenecks due to hardware. These can be avoided by relying on software RAID spanning multiple SSDs attached to simple controllers and exposed directly (without any hardware functionality). In the following subsection we describe the organization and characteristics of our testbed.

2.1 Experimental Testbed

The used hardware testbed comprises eight Intel X25-E SSDs attached to two LSI 9211-4i STAT2 controllers (Figure 1). The SSDs are exposed as simple SATA2 devices without any additional hardware enhancements through the controllers. We neither use the hardware RAID features of the controllers nor do we use any hardware cache on the controller. Based on previous work showing that a software RAID on top of single devices is a way to overcome the performance bottlenecks of a hardware RAID controller [Petrov et al. 2010] we configured all devices in a software RAID0 volume using the Linux MD package. In doing so we configured Linux to use the NOOP IO-Scheduler and turned of the read-ahead on the the devices. The hardware testbed (Figure 1) comprises a total of 48GB RAM, two 4-Core Intel Xeon 5630 CPUs (two hardware threads per core) and is based on a QPI technology. The system operates under 64 bit Gentoo Linux (2.6.34-gentoo-r12).

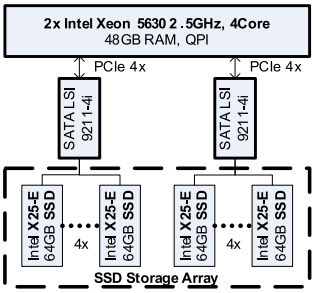

Fig. 1. Experimental testbed architecture

	Seq. Throughput [MB/s] (256KB)	Seq. Throughput [MB/s] (512KB)	Rand. Throughput [IOPS] (4KB)	Rand. Throughput [IOPS] (8KB)	Rand. Throughput [IOPS] (16KB)
Read	2018	2027	279 776	184 469	106 186
Write	1477	1487	41 882	30 458	16 384

Fig. 2. Performance figures overview

We measured the performance of the IO-system using various benchmarks such as IOMeter, fIO or XDD under different operating systems and validated the results with Oracle Orion [Oracle 2010]. Table 2 and Figures 3 and 4 reports the sequential and random throughput as well as sequential and random latency for different block sizes. Table 2 clearly shows the performance dependence on the blocksize. The random throughput for 4KB blocksize is approximately 2.5x better than the random throughput for 16KB blocksizes. Using this architecture we achieve 6x better random read performance, respectively 2x better random write performance over the testbed described in [Petrov et al. 2010].

The logical question now is whether the database can benefit from the hardware improvements. If so what aspects contribute to the best improvement? Last but not least, do established assumptions

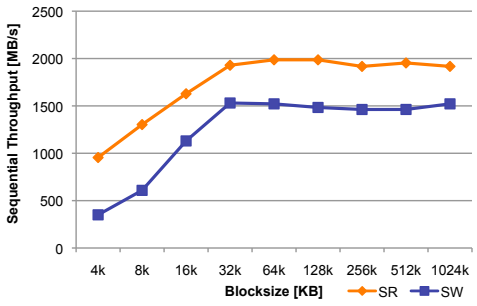

Fig. 3. Measured sequential throughput

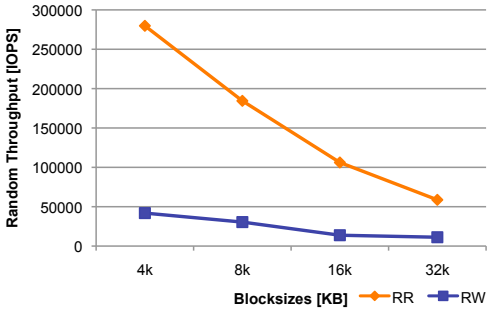

Fig. 4. Measured random throughput

regarding the influence of hardware components such as CPU change in presence of SSD storage with the above characteristics?

We investigated our hypotheses by performing TPC-C experiments on a testbed comprising a MySQL database and SSD storage described above. The used benchmark - DBT2 [DBT2 2010] is an open source TPC-C implementation [TPC-C 2010] on top of MySQL version 5.5.9 with innoDB 1.1.5. DBT2 is instrumented according to the TPC-C specification (Section 4.2.2 of the TPC-C specification [TPC-C 2010]), i.e. to use 20 database connections and 10 terminals per warehouse. The standard MySQL codebase uses a static page size of 16KB, whereas the best random performance of our testbed is achieved with 4KB blocksize. Therefore we reconfigured and recompiled MySQL for the two sizes, as well as for 8KB. For all experiments the block size of the xfs file system was set to the default value of 4KB. The TPC-C tests were performed on all page size variations 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 1800 warehouses amounting to approx. 150-180GB, strongly depending on the database page size. As defined in Section 4.2 of the TPC-C specification [TPC-C 2010] 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.

3. 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 [Schindler et al. 2003; 2003; Graefe 2007] there is a compromise between IO efficiency and buffer size. However a larger page size stands in stark contrast to the characteristics of SSDs.

Figures 5, 6, 7 represent the MySQL results of the TPC-C experiments for different page sizes, buffer sizes and CPUs. We observe a max. 20% performance improvement in transaction throughput (NoTPM) due to the page size performance influence (4KB over 16 KB). The measured transaction throughput improvement can be 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.

Let us now consider Figure 8, which contains the NoTPM maxima from Figures 5, 6, 7 and the relation between buffer size, page size and transaction throughput. Clearly for the same number of

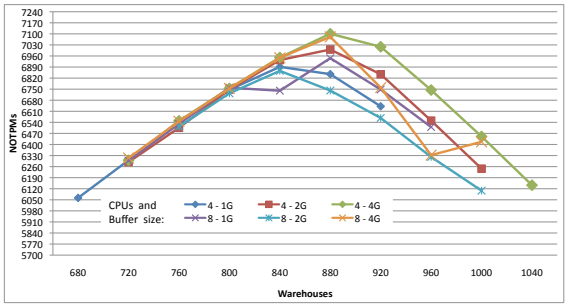

Fig. 5. Transactional throughput [NoTPM] 4KB

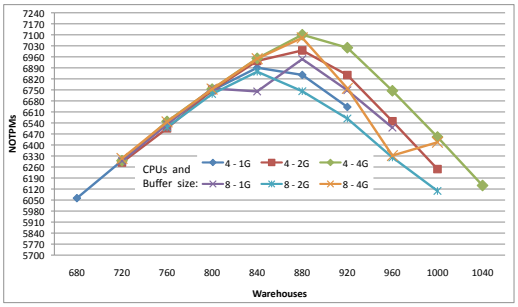

Fig. 6. Transactional throughput [NoTPM] 8KB

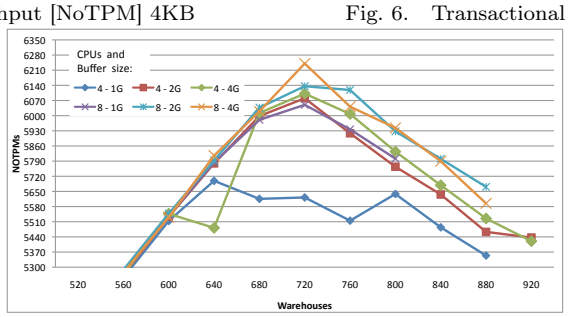

Fig. 7. Transactional throughput [NoTPM] 16KB

	4CPUs			8CPUs		
DB Buffer	P.Size 4KB	P.Size 8KB	P.Size 16KB	P.Size 4KB	P.Size 8KB	P.Size 16KB
1G	6891.14	6419.14	5698.99	6946	6678.41	6050.05
2G	7000.85	6530.1	6080.26	6865.74	6631.93	6136.07
4G	7100.61	6661	6101.82	7080.96	6744.27	6240.77

Fig. 8. Maximum Values for NoTPM

CPUs and the same buffer size there is between 15% and 20% performance advantage for the 4KB page size over 16KB. Consider the data for four CPUs (Figure 8): the NoTPM throughput for 4KB page size and 1GB buffer size is comparable (11.5% better) to the throughput with 16KB page size and 4GB buffer size (page size and buffer size are four times larger). The same result is visible in the eight CPU data. This observation is even more interesting in light of the TPC-C access skew [Hsu et al. 2001] 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. 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 Figure 8 improve with the higher number of CPUs. We depict the transactional throughput over the different database buffer sizes (Figure 9) and number of CPUs (Figure 10). We observe an increase in NoTPM with more CPUs and larger buffer sizes. Although the slope of the curves in Figure 10 decreases this is still a very interesting observation in an IO-bound environment. 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.

4. 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

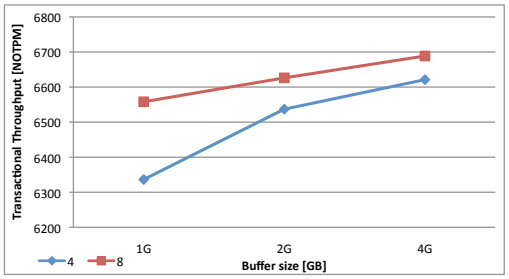

Fig. 9. Increase in transactional throughput (NoTPM) for different buffer sizes (number of CPUs - constant, page size 4KB)

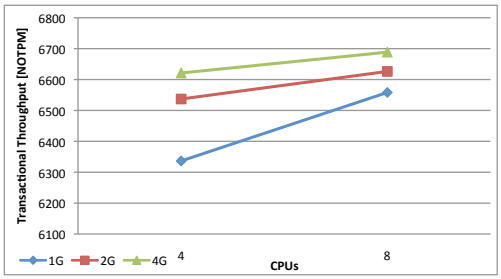

Fig. 10. Increase in transactional throughput (NoTPM) for different number of CPUs (buffer size - constant, page size 4KB)

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, 8KB and 16KB page sizes on SSD storage. We observe a 20% performance and response time improvement for the smaller block sizes. 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 (Figure 2). It is therefore important to support multiple concurrent block sizes to accommodate the requirements of different database objects types and access patterns. Larger block sizes are better suited for tables processed by sequential operations (e.g. full table scan) while smaller block sizes are better for operations yielding random accesses (e.g. index scan).

`Acknowledgement`: This work has been partially supported by the DFG project 'Flashy-DB'.

REFERENCES

Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., and Panigrahy, R. Design tradeoffs for ssd performance. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*. USENIX Association, Berkeley, CA, USA, pp. 57–70, 2008a.

Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., and Panigrahy, R. Design tradeoffs for ssd performance. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*. USENIX Association, Berkeley, CA, USA, pp. 57–70, 2008b.

Chen, F., Koufaty, D. A., and Zhang, X. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. In *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*. SIGMETRICS '09. ACM, New York, NY, USA, pp. 181–192, 2009.

DBT2 . Database test suite. http://osdldbt.sourceforge.net/, 2010.

Do, J. and Patel, J. M. Join processing for flash ssds: remembering past lessons. In *Proceedings of the Fifth International Workshop on Data Management on New Hardware*. DaMoN '09. ACM, New York, NY, USA, pp. 1–8, 2009.

Graefe, G. The five-minute rule twenty years later, and how flash memory changes the rules. In *Proceedings of the 3rd international workshop on Data management on new hardware*. DaMoN '07. ACM, New York, NY, USA, pp. 6:1–6:9, 2007.

Gray, J. and Graefe, G. The five-minute rule ten years later, and other computer storage rules of thumb. *SIGMOD Rec.* vol. 26, pp. 63–68, December, 1997.

He, J., Jagatheesan, A., Gupta, S., Bennett, J., and Snavely, A. Dash: a recipe for a flash-based data intensive supercomputer. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '10. IEEE Computer Society, Washington, DC, USA, pp. 1–11, 2010.

Hsu, W. W., Smith, A. J., and Young, H. C. Characteristics of production database workloads and the tpc benchmarks. *IBM Syst. J.* vol. 40, pp. 781–802, March, 2001.

Hudlet, V. and Schall, D. Ssd != ssd - an empirical study to identify common properties and type-specific behavior. In *Proceedings of the 14. GI-Fachtagung Datenbanksysteme für Business, Techologie und Web*. BTW 2011. pp. 430–441, 2011.

Lee, S.-W. 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*. SIGMOD '07. ACM, New York, NY, USA, pp. 55–66, 2007.

Lee, S.-W., Moon, B., and Park, C. Advances in flash memory ssd technology for enterprise database applications. In *Proceedings of the 35th SIGMOD international conference on Management of data*. SIGMOD '09. ACM, New York, NY, USA, pp. 863–870, 2009.

Lee, S.-W., Moon, B., Park, C., Kim, J.-M., and Kim, S.-W. A case for flash memory ssd in enterprise database applications. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. SIGMOD '08. ACM, New York, NY, USA, pp. 1075–1086, 2008.

Li, Y., He, B., Luo, Q., and Yi, K. Tree indexing on flash disks. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*. IEEE Computer Society, Washington, DC, USA, pp. 1303–1306, 2009.

Nath, S. and Kansal, A. Flashdb: dynamic self-tuning database for nand flash. In *Proceedings of the 6th international conference on Information processing in sensor networks*. IPSN '07. ACM, New York, NY, USA, pp. 410–419, 2007.

Oracle, C. Oracle Corp. ORION (Oracle I/O Calibration Tool) . http://oracle.com/technology/software/tech/orion/index.html, 2010.

Petrov, I., Almeida, G., Buchmann, A., and Ulrich, G. Building large storage based on flash disks. In *Proceedings of the 2010 ACM/IEEE ADMS Workshop in conjunction with VLDB*, 2010.

Schindler, J., Ailamaki, A., and Ganger, G. R. Lachesis: robust database storage management based on device-specific performance characteristics. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*. VLDB '2003. VLDB Endowment, pp. 706–717, 2003.

Shah, M. A., Harizopoulos, S., Wiener, J. L., and Graefe, G. Fast scans and joins using flash drives. In *Proceedings of the 4th international workshop on Data management on new hardware*. DaMoN '08. ACM, New York, NY, USA, pp. 17–24, 2008.

TPC-C. Tpc benchmark c. standard specification. revision 5.11. february 2010. http://www.tpc.org/tpcc/spec/tpcccurrent.pdf, 2010.