SpartanSSD: a Reliable SSD under Capacitance Constraints

Hyeon Gyu Lee*, Juwon Lee†, Minwook Kim*, Donghwa Shin‡, Sungjin Lee◇,
Bryan S. Kim•, Eunji Lee‡, and Sang Lyul Min*
*Seoul National University, †FADU, ‡Soongsil University, ◇DGIST, •Syracuse University
*{flewin, ace, symin}@snu.ac.kr, †jwlee@fadutec.com, ‡{donghwashin, ejlee}@ssu.ac.kr,
◇sungjin.lee@dgist.ac.kr, •bkim01@syr.edu

2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) | 978-1-6654-3922-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISLPED52811.2021.9502476

***Abstract*—In this paper, we present an SSD design that is resilient to sudden power-off failures. Modern SSDs use a large number of capacitors that act as energy reserves to persist both host data and SSD metadata in the unforeseen event of a power outage. However, these capacitors take up a large footprint that limits the SSD's density. We present a series of design choices that significantly reduce the SSD's dependence on capacitors, all the while meeting the durability, consistency, and power-on time constraints. We demonstrate that at a modest performance overhead of 11%, the amount of required capacitance is reduced by 97.87%.**

***Index Terms*—energy efficiency, flash memory, reliability**

I. Introduction

SSDs are a central component in today's data-intensive systems during this *Age of Big Data*. This is thanks to not only the low latency and collectively massive throughput of flash memory, but also the sophistication of the SSD architecture that hides the quirks of the underlying medium. To respond to the high demand for storage capacity in data-intensive applications, the SSD density has increased rapidly in recent years, expanding by 100× over the past ten years [1], [15].

Conventionally, SSDs use a small amount of volatile DRAM as both a cache and a buffer in front of the relatively slow flash memory. This allows the frequently accessed data such as the translation pages to be resident in memory and achieves a faster response time for data writes. However, the use of volatile memory may lead to a data loss or a device failure in an expected power outage, seriously threatening the storage reliability [19]. To address this undesired situation, the enterprise-class SSDs are equipped with an array of capacitors to act as an energy reserve. Such electrical circuitry is known as power-loss protection (PLP) in SSDs [8], [13], [16], and it ensures that the SSD has enough power to write data in the event of a power-loss.

Unfortunately, the reliance on capacitors is not a sustainable solution because the improvement in capacitance fails to keep up with the rapid growth of SSDs. Al(aluminum) and Ta(tantalum)-electrolytic capacitors used in SSDs have increased in density through miniaturization by tenfold from 1960 to 2005 [4]. This rate, however, is still much slower than the SSD's increase in capacity and the amount of internal-DRAM it houses. In 2011, a typical 2.5-inch SSD had 256GB capacity with 256MB of DRAM; by 2018, a high-capacity 2.5-inch SSD boasted a 30TB with 40GB of DRAM [1], [15]. This density gap between battery and memory technologies requires a redesign of the existing battery-based protection mechanisms for sustainability.

SSD Model	Manufacturer	Class	PLP	Capacitor
950Pro, 850Pro	Samsung	Client	None	-
M500	Micron	Client	Partial	Ceramic
M500DC	Micron	Enterprise	Full	Tantalum
PM863, SM863	Samsung	Enterprise	Full	Tantalum
DC1000B	Kingston	Enterprise	Full	Tantalum
DC S3700, S3500	Intel	Enterprise	Full	Aluminum

TABLE I: **Power Loss Protection in SSDs [8], [13], [16]**.

This paper presents an SSD design called *SpartanSSD* that protects against a power fault while minimizing the number of capacitors. It achieves both (1) durability by elastically managing the small size of PLP-protected memory for journaling and (2) consistency by selectively aborting operations that can be safely recovered upon power-on. These techniques are combined to minimize the required capacitance while meeting the power-on recovery time constraint.

We implement SpartanSSD on a Xilinx Zynq Ultrascale+ ZCU102 evaluation board and demonstrate the effectiveness of our approach under a wide variety of workloads. The performance evaluation with micro- and real-world benchmarks shows that SpartanSSD decreases the amount of required capacitance by 97.87% at the cost of 11% of performance overhead.

II. Power-Loss Protection in SSDs

The use of volatile memory in storage extends far before the advent of SSDs: HDDs use a small DRAM (of tens of MB in size) to buffer data before writing to disk. Based on this, the host system expects the use of volatile buffers in the storage device, and thus the storage interface has explicit synchronization commands such as `FLUSH`. These commands force all buffered data to be persisted to the non-volatile media, and will only return once completed. They essentially act as a barrier that guarantees all data before the command are made durable.

However, immediately flushing all buffered data to the flash memory upon a synchronization request not only lowers the throughput by decreasing the parallelism of the SSD, but also increases the user-perceived latency with bulky synchronous writes. For this reason, enterprise SSDs typically use capacitors for power-loss protection (PLP). The capacitors reserve energy and provide emergency power to SSD so that all volatile data can be persisted in flash memory in a sudden power outage.

Symbols	Description	Value
T_{prog}	Time to program a page to flash memory	1300μs
T_{erase}	Time to erase a block in flash memory	3800μs
P_{prog}	Power during perform program operation in one chip	82.5mW
P_{erase}	Power during perform erase operation in one chip	82.5mW
S_{page}	The size of one physical flash page	8KB
S_{total}	The total size of protected data	depends
N_{chip}	The total number of chips in the board	64

TABLE II: **Flash memory parameters [12], [20].**

Table I summarizes the policies for power loss protection (PLP) across several SSDs. While client SSDs such as the Samsung 950Pro and Micron M500[1] forgo PLP, enterprise SSDs use capacitors that supply enough energy during power-loss to persist all data in the volatile memory to a temporary location in flash [8], [9], [13], [16], [17]. Such data include not only the user data, but also the SSD's metadata such as the mapping table and validity bitmap. When the power is restored, SSD reloads the data dumped during power-loss and resumes normal operation from that state [10].

Furthermore, PLP also ensures the *consistency* of SSD by allowing the in-progress operations to safely complete in the event of a crash. If the power fault occurs unexpectedly in the middle of an operation that changes the SSD-internal status, SSD can lose consistency and lead to a complete device failure. For example, a free page is allocated for a new write, but the power outages before writing the actual data into it, it will lead to a space leak for SSD. To avoid this, Intel SSDs with PLP technology adopts the energy-storing capacitors that provide enough energy (power) to complete any commands in progress and to make sure that any data in the temporary buffers are committed to the non-volatile NAND Flash media [8], [19]. The case of Samsung SSDs immediately halts the in-progress operations in the face of a crash and checks the data integrity during recovery for consistency [10].

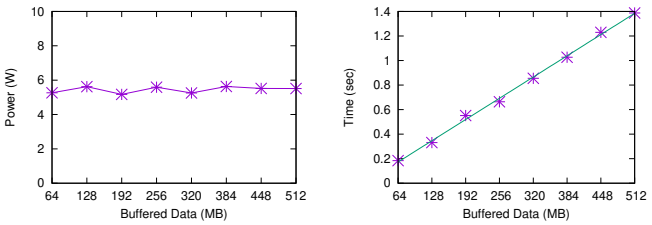
Our goal is to achieve the same level of data protection as enterprise SSDs, ensuring the durability of acknowledged writes and the consistency of SSD, while reducing the amount of SSD-internal capacitance.

III. Design of SpartanSSD

The reliance on capacitors makes the emergency shutdown and recovery simple: all the contents in volatile memory are dumped to a temporary location in flash during a shutdown, and the state of the memory is restored upon recovery. However, this approach limits the storage density of the SSD and increases the overall cost. Instead, we present an alternative design called *SpartanSSD* that meets the durability and consistency requirement and the recovery time constraints without using large capacitors.

We investigate how much energy is required for durability enforcement (to persist all volatile buffer data to flash) and consistency guarantee (to complete all in-progress operations) in SSD, respectively. To this end, we measured the power consumed and time elapsed on our SSD prototype board during

[1]M500 finishes only the in-progress program operations without protecting volatile memory.



(a) Power consumption (b) Elapsed Time

Fig. 1: **Power model for durability enforcement.** This result is measured on our SSD prototype board (XilinxZynq Ultrascale+ ZCU102 board with 8 NAND Flash chips) using MSO54 Tektronix Mixed Signal Oscilloscope and DC current probe.

Enforcement	SSD-PLP	SpartanSSD-JNL
Consistency	0.020J (0.6%)	0.020J (0.6%)
Durability	3.576J (99.4%)	0.0618J (1.7%)
Total	3.596J (100%)	0.0818J (2.3%)

TABLE III: **Energy consumption: SSD-PLP vs. SpartanSSD-JNL.** SSD-PLP fully protects the entire buffered data with sufficient capacitors. SpartanSSD-JNL represents the intermediate version of the proposed architecture that uses elastic journaling.

PLP's power-off sequence, while varying the amount of data within volatile buffer from 64MB to 512GB. Figure 1 plots the power and time consumption as a function of the data buffered in DRAM. We observe that the power consumed with respect to the amount of data buffered in DRAM is near-constant, and the time to completion linearly scales with the amount of data. This shows that the power consumption during PLP's power-off is dominated by the intensive read from DRAM and write to flash. Based on this power model and the physical operating parameters of our SSD (Table II), the energy consumption of SSD with PLP on a power outage can be expressed as follows:

$$E_{full} = P_{prog} \cdot \left\lceil \frac{S_{total}}{S_{page}} \right\rceil \cdot T_{prog} + P_{erase} \cdot N_{chip} \cdot T_{erase} \quad (1)$$

where the first and second term represents the energy consumption for durability (persisting all data in buffer) and consistency enforcement (completing all on-going operation, in the worst-case), respectively.

Table III summarizes the energy consumed by an SSD using PLP (full set of capacitors) and our design (reduced capacitors). We further break down the energy consumed for enforcing consistency or durability. For SSD-PLP, 99.4% of the energy is used for persisting the volatile data, while only 0.6% is used for completing the in-progress operations. Motivated by this, we first propose a buffer management technique that significantly reduces the energy consumed for guaranteeing durability. Our key strategy is as follows.

There are four types of data cached: user data in the volatile memory, mapping data, a mapping table directory, and other

Type	Size	Ratio
User Data Buffer	128KB	0.02%
Mapping Table	512MB	97.87%
Mapping Table Directory	1MB	0.19%
Metadata for Allocation	1MB	0.19%
Metadata for GC	9MB	1.72%
Total Buffer Memory	523.125MB	-
SSD Capacity	512GB	-

TABLE IV: **Components of the in-storage buffer.**

miscellaneous metadata (such as a free block list). These are all protected by the PLP circuitry, but some of these data are recoverable without explicitly writing them to flash. We take advantage of this to reduce the amount of capacitance needed.

The user data requires full protection to ensure data durability because there is no way to recover them if lost. Fortunately, this requirement is not difficult to achieve because the buffer space for user data is relatively smaller than that for the mapping table. Table IV shows the size and ratio of each component in the volatile buffer for a typical 512GB SSD. User data are first buffered in volatile DRAM to maximize the high degree of parallelism in SSDs. If an SSD has 4 channels and 2 ways per channel with 8KB page size, about 128KB of memory (twice the size of all pages that can be written in parallel) are used for data buffering. This, however, only accounts for 0.02% of the volatile DRAM.

In contrast, all metadata in volatile memory can be restored after a power failure. They essentially represent the status of the SSD, which changes according to the user request or the internal activities. Even if they are lost, they can be restored by scanning the entire device for integrity. This approach, however, fails to recover fast after a sudden power-off, and modern SSDs, especially the enterprise-class, use a large number of on-board capacitors to protect the metadata. At the same time, the reliance on capacitors has reached its limit: in particular, the size of the mapping table linearly scales with the storage capacity where several GBs of DRAM is required for a TB range SSD. While prior work with memory-efficient mapping structure such as D-FTL [6] reduce the amount of required DRAM, they are rarely used in practice due to its limited performance. With this observation, SpartanSSD does not protect the mapping table, but achieves the same goal through the sophisticated update mechanism.

SSDs intrinsically have a mechanism that tradeoffs the performance during normal operations and the performance of sudden power-off recovery (SPOR). At one end, the SSD can choose to re-construct the entire mapping table by scanning the out-of-bound area of each page that records the logical address of that physical data. This leads to a prohibitively long recovery time. At the other end, the SSD can persist the translation page that maps the logical to physical translation for that data and all subsequent metadata updates as soon as the user data is written to flash. This is called *checkpointing* in SSDs. Frequent checkpointing reduces the number of pages to scan on recovery, but increases the overhead during normal operations.

SpartanSSD offers a reasonable compromise between two extremes by making use of *journaling*. By recording the changes of the mapping table in a specific area (journal), SpartanSSD restores the mapping table fast, consulting the journal only, instead of scanning the entire device. While some previous studies considered journaling to ensure data durability in SSD [16], it is naively designed for SSDs without PLP and the effectiveness is not investigated. Other metadata are negligible in size compared to the mapping table, so the low cost of protecting them justifies the use of PLP on them.

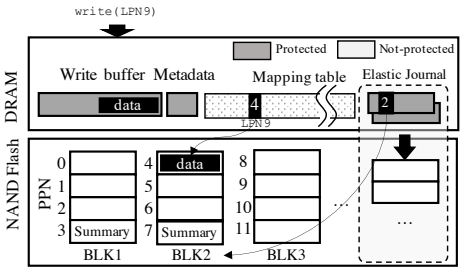


Fig. 2: **Elastic journaling of SpartanSSD.**

A. Durability Enforcement with Elastic Journaling

In this paper, we carefully design the journaling technique in consideration of the SSD characteristics, aiming to effectively overcome the fundamental limitation of the capacitor-based PLP technique.

We start by discussing a simple journaling model that can be applied to SSD. This model logs *the mapping entry* that translates the logical page number (LPN) to the physical page number (PPN) in the journal area. The journal area is maintained in a battery-backed volatile memory. When the actual data is written to flash, the mapping entry is not only updated in the mapping table (volatile memory) but also recorded in the journal area. Once the journal becomes full, the SSD performs a checkpoint where all the mapping table pages with updates are forced to flash and the journal space is reclaimed for recycling. Because updates to the mapping table pages are written in out-of-place, the mapping directory that keeps track of the location of mapping table pages must also change. The size of the mapping directory is relatively small compared to the mapping table, and it is protected by the PLP.

The principle of journaling in SpartanSSD is borrowed from the metadata journaling and write-ahead logging used in file systems and databases. However, there is a major difference in the context of SSDs. In SSD, capacitor-backed DRAM is a critical resource that cannot be added and expanded after deployment. In file systems and databases, a separate secondary storage is typically assigned for journaling and write-ahead logging.

SpartanSSD thus presents *elastic journaling* that is designed to overcome the above limitation through efficient and elastic management of the journal area. Typically in journaling, once the journal area becomes full, the system goes through a checkpoint wherein all of the logged updates are applied to the actual data location. Checkpoints are expensive and detrimental to the overall performance. If the journal size is not large enough to accommodate the write footprint, we can increase the frequency for checkpoints. In SpartanSSD, the frequency of checkpointing and the utilization of the journal area is decoupled by elastically extending the storage area. Once the journal area is full and the SSD is under heavy I/O traffic, the logged contents are persisted into storage as-is without performing a checkpoint. This extended journal area is later read during an idle time for the checkpoint operation.

In addition, we further optimize journaling by recording only the block number of where the new data is written instead of logging the block number and page number for all updates.

Enforcement	SpartanSSD-JNL	SpartanSSD-JNL/EA
Consistency	0.0200J (24.5%)	0.0068J (8.4%)
Durability	0.0618J (75.5%)	0.0618J (75.5%)
Total	0.0818J (100%)	0.0697J (83.9%)

TABLE V: **Energy consumption: SpartanSSD-JNL vs. SpartanSSD-JNL/EA.** SpartanSSD-JNL/EA executes the erase abortion in conjunction with elastic journaling.

This coarse-grained journaling is possible only in SSDs because SSDs maintain the logical page number (LPN) in the out-of-band area within each page. On the recovery, SpartanSSD visits the blocks recorded in the journal area and updates the mapping table by referencing LPN in the out-of-band area. Figure 2 shows the SpartanSSD architecture with block-level mapping entry journaling. Compared to the page-level mapping entry journaling, this consumes less space in the journal area, and thus can mitigate the overhead of checkpointing.

Although the standard SSD specification requires a fairly long boot-up time constraint of tens of seconds [14] after a power failure, this is loose to be practical in larger systems where an array of SSDs are used [16]. Thus, to accelerate the boot-up time, SpartanSSD uses the block-level mapping journaling and limits the scanning range during recovery. However, even scanning all pages in this range to reconstruct the latest and consistent mapping table still incurs a non-negligible overhead of hundreds of milliseconds.

To further reduce the recovery time, we adopt the *summary page* [3] that records all LBAs for the physical pages for each block at a designated area (e.g. the last page of each block). The recovery process can scan this summary page instead of the spare area of entire pages on each block to reduce the number of reads needed for mapping information. SpartanSSD records a summary page once a block is about to be fully utilized. These summary pages are used during recovery, starting from the first block in the journal area. For blocks without a summary page, SpartanSSD scans the LBAs from the spare area; this open happens for the last block. When checkpointing is complete for the mapping information of all pages in the journal blocks, the journal area is reclaimed and summary pages are invalidated.

Our design uses additional storage space for (1) persisting the journal and (2) maintaining a summary page for each block. The journal is only a few KBs in size and is negligible in terms of storage overhead. The summary page is only one page out of all the pages in a block. For reference, our experimental platform uses flash memory chips with 256 pages per block - this is less than 0.5% storage overhead. The recent trend shows that the number of pages per block will continue to increase, and we expect the storage overhead to become even smaller in the future.

With elastic journaling, SpartanSSD consequently reduces the required capacitance for protecting the mapping table, which accounts for 97.87% of the total energy during the PLP's power-off (see Table III). Only a few KBs of memory needs to be protected with elastic journaling, and thus consumes less than 0.01% of the total energy compared to full mapping table protection.

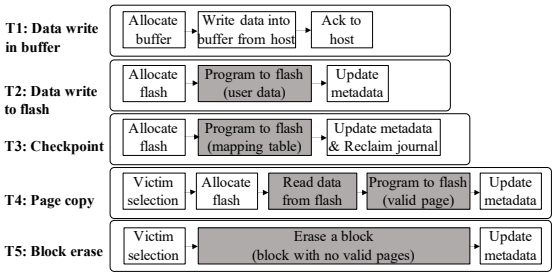


Fig. 3: **Transactions in SpartanSSD.**

B. Consistency Enforcement

Aside from durability guarantees, PLP also enforces that the state of the SSD remains consistent after a power loss. SSD pages that experience power loss during a program or erase operation may contain incorrect data, and the state of the SSD may become inconsistent if multiple operations are not atomically persisted. As shown previously in Table III, the relative amount of energy used for finishing these operations is only less than 1% when protecting the entire DRAM. The journaling technique discussed in the previous section reduces the necessary capacitance by protecting only the user data and journaled metadata, but the relative amount of energy for ensuring all in-progress operations complete increases to 24.5%, as shown in Table V. This subsection discusses how SpartanSSD reduces the required capacitance by judiciously aborting in-progress operations while remaining consistent.

We can classify SSD's internal state change into two: one caused by user writes and the other caused by SSD's internal management such as garbage collection (GC). We also define five transactions (as shown in Figure 3) that make up the user write and GC. In Figure 3, the gray box presents operations accessing the NAND Flash memory, and the white box manipulates data only in DRAM. The user write is divided into three separate transactions and processed in order. The SSD first transfers the data from the host into its internal buffer (T1), persists the data by programming to flash (T2), and finally reflects the changes to the mapping by persisting the updated mapping information through a checkpoint (T3). For GC, the SSD copies each valid pages from the victim to a new location (T4), and then finally erases a block that contains no valid pages (T5). Selecting the victim block incurs no persistent state change, and thus is not classified as a transaction.

For the SSD state to remain consistent even during a sudden power outage, the individual transactions should be committed atomically. Because PLP satisfies this requirement by precluding the untimely abortion of in-progress operations and persisting DRAM with capacitance, the longest transaction dictates the amount of energy needed in the capacitor to enforce consistency in case of a sudden power outage. However, we observe that we can safely abort an ongoing erase operation without losing consistency: if an erase operation fails due to a power loss, we simply need to identify the block that experienced a power loss during recovery time and retry the erase. For this purpose, SpartanSSD logs the block that is about to be erased in the journal. Although aborting an ongoing erase and retrying afterward effectively increases the amount of wear

on that block, a prior work showed that it has a negligible effect on the overall reliability [19], increasing the error rate at most only 0.9%. This is a stark contrast to the increase in error rate for a power failure during a program operation: up to 50%.

Program operations, on the other hand, should not be aborted. Hundreds of pages make up a block in flash memory, thus a page program operation occurs much more frequently than a block erase. Aborting these concurrent program operations will complicate the boot-up time as these operations that may have completed out-of-order must be carefully replayed to enforce consistency.

IV. Evaluation

We have built a prototype of our SpartanSSD using a Xilinx Zynq Ultrascale+ ZCU102 evaluation board. The board has an SoC that integrates four ARM Cortex-A53 cores with an FPGA, 16GB DDR4 DRAM, and external connectors for communications. We ran our SpartanSSD on Ubuntu 16.04 with a Linux 4.9.0 kernel. For storage, we used a custom flash card connected to the evaluation board via the FPGA Mezzanine Card interface. The card has 512GB NAND flash chips, including 8 flash channels and 8 flash LUNs per channel. We implemented a customized library to communicate with these flash chips with Connectal, which provides a connection between hardware chips and software FTL bypassing the operating system.

For the performance evaluation, we also implemented four different types of SSDs and compare them against our design.
Baseline: This SSD has no capacitor and immediately synchronizes all of the data to the NAND flash upon a request. Although this configuration is unrealistic in practical systems, we study this to see the lower-bound of the performance.
FullProtection: This version protects entire buffer data and in-progress operations with a sufficient amount of capacitance. This SSD configuration emulates commercial SSDs and it is optimal in terms of performance.
DuraSSD: The original version of DuraSSD [10] incrementally flushes the mapping table while protecting the entire mapping table and user data during a power outage through capacitors. In our evaluation, we further augment DuraSSD to bound the time and energy during power-offs: we implement this by maintaining the number of dirty pages and flushing them when it goes beyond a threshold for the capacitance limit.

We implemented three different versions of SpartanSSD. **SpartanSSD-B** uses a basic journaling model that logs a page-level mapping entry and **SpartanSSD-E** uses the elastic journaling model that journals mapping table updates at the block level with the storage extended journal area. Lastly, **SpartanSSD-ES** recovers the internal state by referencing the summary pages, instead of LBAs stored in the spare area. We maintain two pages (16KB) as a journal area and use them alternatively to receive the incoming requests while checkpointing. For a fair comparison, we set all versions of SSD other than Baseline and FullProtection to use the same size of the capacitor.

We measured the performance of SpartanSSD using synthetic and real-world workloads. For synthetic workloads, we use three workloads with one million requests of the following

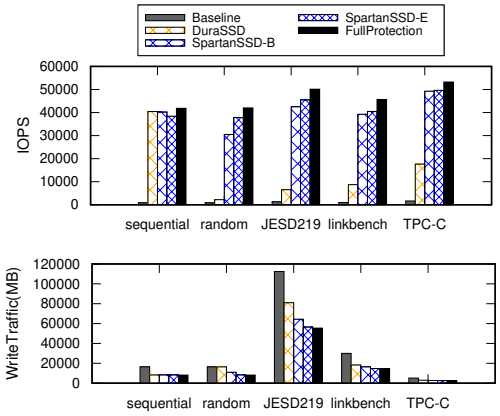


Fig. 4: **IOPS and write traffic.**

Workload	W%	R%	FP(GiB)		DA(GiB)		HOT(%)	
			W	R	W	R	W	R
Sequential	100	0	8.0	0.0	8.0	0.0	20	-
Random	100	0	8.0	0.0	8.0	0.0	20	-
JESD219	67	33	10.0	5.9	54.9	26.8	70	69
Linkbench	86	14	4.8	1.9	14.6	2.3	68	32
TPC-C	52	48	1.1	2.1	2.5	2.3	64	30

TABLE VI: **Workload characteristics.** W and R represents write and reads. Access footprint (FP) is the size of the logical address space range accessed, and Data accessed (DA) is the total amount of read or written data. Hotness (HOT) is the percentage of data read or written in the top 20% of the frequently accessed logical address.

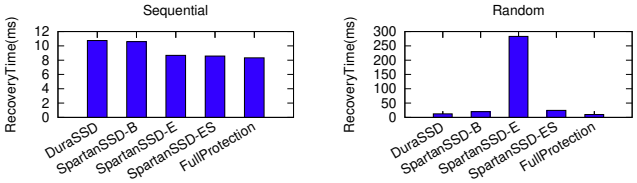


Fig. 5: **Recovery time.**

pattern: 8KB sequential writes, 8KB random writes, and the skewed read-write mixed workload that follows the JESD 219 specification for SSD endurance testing. For real workloads, we use LinkBench [2], a graph processing application based on the Facebook social graph, and TPC-C [18] on MariaDB, an online transaction processing benchmark. Table VI summarizes the characteristics of the workloads in this study.

Figure 4 shows the IOPS and write traffic for different types of SSDs across the workloads. SpartanSSD provides comparable performance to FullProtection, resulting in a performance loss of only less than 17% and 11% on average respectively. This performance degradation is reasonable considering that SpartanSSD-E reduces the adopted capacitance by 97.87% (Table III).

Compared to DuraSSD that uses the same size of capacitor as SpartanSSD, SpartanSSD-E performs better than DuraSSD by 4.31x on average across workloads. This performance gain is obtained because SpartanSSD-E utilizes the small size of battery-backed memory in the more efficient manner through elastic journaling. In terms of write traffic, SpartanSSD-E outperforms DuraSSD by 24.6% because it reduces the checkpoint frequency with coarse-grained journaling.

In particular, SpartanSSD-E provides excellent performance for random write workloads, compared to other alternatives (DuraSSD and SpartanSSD-B). DuraSSD protects the limited

number of mapping table pages that is statically determined by the equipped capacitance. If the number of dirty mapping table pages becomes more than allowance, DuraSSD forces them into the NAND flash to ensure durability. This behavior incurs frequent flushing of the mapping table for the random write workloads, leading to a performance degradation and write traffic increase. SpartanSSD-B also has a similar limitation because it uses the fixed size of journal area. In contrast, SpartanSSD-E can deal with a large working set efficiently using elastic journaling, and thus it provides the best performance for random and JESD219 workloads that has large footprint at a time window.

One counter-intuitive result is SpartanSSD-E shows 4.9% lower IOPS than SpartanSSD-B for sequential workload. Our careful analysis reveals that this is due to the software complexity of elastic journaling higher than other mechanisms.

Figure 5 shows the recovery time of various SSDs at the face of power loss. We deliberately injected a power-off at random points of workloads and measured the recovery time at reboot. For sequential workloads, all types of SSDs provides excellent recovery time less than 11ms, but for random and skewed workloads, SpartanSSD-E increases the recovery time significantly. This is due to the overhead of referencing the out-of-band area for each page to obtain the LPN information. SpartanSSD-ES eliminates this overhead through the summary page. For the sequential workloads, the mapping page updates have a strong locality and there is few blocks to checkpoint in recovery, and thus the overhead of referencing out-of-band area has less impact.

V. Related Work

SpartanSSD focuses on reducing the amount of capacitance while guaranteeing durability and consistency. While a few prior work approach this problem from a device technology perspective (via using PRAM [5] and SLC mode [7]), our work more closely relates to DRWB [11] and DuraSSD [10] that addresses the concern via a software design.

DRWB (Dual-Region Write Buffer) [11] divides the SSD-internal buffer into a small protected region and a large unprotected one, and uses the small protected area for differential logging of data updates in the unprotected area. DRWB, however, only considers the user data and does not address the intricacies relating to SSD's metadata that not only takes up the majority of the space in the volatile DRAM but also is critical for guaranteeing the SSD's consistency.

DuraSSD [10] is an SSD prototype that enhances the write performance in database and NoSQL systems by relying on capacitors. Based on the observation that the frequent flushing of the data in the SSD-internal cache imposes a serious performance degradation, the authors use a battery-backed DRAM to decouple the durability guarantee to the host and the programming of the data to flash. While DuraSSD presented an incremental backup of the mapping table to reduce the flushing overhead in a power crash, it essentially does not reduce the capacitor size because the entire mapping table should be persisted in the worst case.

VI. Conclusion

This paper presents a novel SSD design for scalable SSD called SpartanSSD. SpartanSSD protects a part of the buffer with a small size of capacitor, but it provides high performance and strong reliability with careful update serialization and judicious data management. Compared to the commercial enterprised-SSD, SpartanSSD reduces a capacitor size by more than 97.87% at the cost of graceful compromise on performance.

Acknowledgment

We thank the anonymous reviewers for their practical and insightful comments. This work was supported by the National Research Foundation(NRF) of Korea (NRF-2020R1A4A4079859 and NRF-2018R1A5A1060031). The Institute of Computer Technology at Seoul National University provided the research facilities for this study. Eunji Lee is the corresponding author.

References

[1] AnandTech, "Samsung 30.72 TB SSDs: Mass production of PM1643 begins," https://www.anandtech.com/show/12448/samsung-begins-mass-production-of-pm1643-sas-ssds-with-3072-tb-capacity, 2018.

[2] T. G. Armstrong, V. Ponnekanti, D. Borthakur, and M. Callaghan, "Linkbench: a database benchmark based on the facebook social graph," in *SIGMOD*, 2013, pp. 1185–1196.

[3] A. Birrell, M. Isard, C. Thacker, and T. Wobber, "A design for high-performance flash disks," *SIGOPS Oper. Syst. Rev.*, vol. 41, no. 2, p. 88–93, 2007.

[4] J. Both, "The modern era of aluminum electrolytic capacitors," *IEEE Electrical Insulation Magazine*, vol. 31, no. 4, pp. 24–34, 2015.

[5] J. Guo, J. Yang, Y. Zhang, and Y. Chen, "Low cost power failure protection for MLC NAND flash storage systems with PRAM/DRAM hybrid buffer," in *DATE*, 2013, pp. 859–864.

[6] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings," in *ASPLOS*, 2009, pp. 229–240.

[7] M. Huang, Y. Wang, L. Qiao, D. Liu, and Z. Shao, "SmartBackup: An efficient and reliable backup strategy for solid state drives with backup capacitors," in *HPCC*, 2015, pp. 746–751.

[8] Intel, "Power loss imminent(PLI) technology," https://www.intel.com/content/www/us/en/solid-state-drives/ssd-power-loss-imminent-technology-brief.html, 2014.

[9] Intel, "Intel® solid-state drive DC P3600 series," 2015.

[10] W. Kang, S. Lee, B. Moon, Y. Kee, and M. Oh, "Durable write cache in flash memory SSD for relational and NoSQL databases," in *SIGMOD*, 2014, pp. 529–540.

[11] D. Kim and S. Kang, "Dual region write buffering: making large-scale nonvolatile buffer using small capacitor in SSD," in *SAC*, 2015, pp. 2039–2046.

[12] Micron, "Micron MT29F MLC NAND flash datasheet," https://www.micron.com/-/media/client/global/documents/products/data-sheet/nand-flash/20-series/2gb_nand_m29b.pdf, 2004.

[13] Micron, "How Micron SSDs handle unexpected power loss," https://www.micron.com/-/media/client/global/documents/products/white-paper/ssd_power_loss_protection_white_paper_lo.pdf, 2014.

[14] Micron, "9200 NVMe SSDs features," Micron SSD Datasheet, 2016, https://www.micron.com/-/media/client/global/documents/products/data-sheet/ssd/9200_u_2_pcie_ssd.pdf.

[15] Samsung, "Samsung SSD 830 series," https://www.samsung.com/us/support/owners/product/128gb-ssd-830-series, 2011.

[16] Samsung, "Power loss protection in ssds - how ssds are protecting data integrity," A Samsung Electronics White Paper, 2016.

[17] Sandisk, "Unexpected power loss protection," http://solidstatedisks.co.uk/Downloads/Sandisk_Unexpected_Power_Loss_Protection.pdf, 2015.

[18] Transaction Processing Performance Council, "TPC benchmark C standard specification," 1990.

[19] H.-W. Tseng, L. Grupp, and S. Swanson, "Understanding the impact of power loss on flash memory," in *DAC*, 2011, pp. 35–40.

[20] G. Wu and X. He, "Reducing SSD read latency via NAND flash program and erase suspension," in *FAST*, 2012, pp. 1–7.