Software Controlled Cell Bit-Density to Improve NAND Flash Lifetime

Xavier Jimenez, David Novo and Paolo Ienne
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH–1015 Lausanne, Switzerland
xavier.jimenez@epfl.ch, david.novobruna@epfl.ch and paolo.ienne@epfl.ch

ABSTRACT

Hybrid flash architectures combine static partitions in *Single Level Cell (SLC)* mode with partitions in *Multi Level Cell (MLC)* mode. Compared to MLC-only solutions, the former exploits fast and short random writes while the latter brings large capacity. On the whole, one achieves an overall tangible performance improvement for a moderate extra cost. Yet, device lifetime is an important aspect often overlooked. In this paper, we show how a dynamic SLC-MLC scheme provides significant lifetime improvement (up to 10 times) at no cost compared to any classic static SLC-MLC partitioning based on any state of the art Flash Translation Layer policy.

Categories and Subject Descriptors

B.3.1 [**Memory Structures**]: Semiconductor Memories

General Terms

Design, Experimentation, Measurement, Performance, Reliability

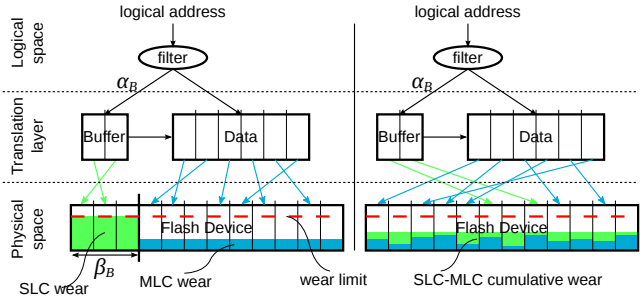
Keywords

NAND Flash Memory, Flash Endurance, FTL, SLC, MLC

1. INTRODUCTION

NAND flash memory is the leading data storage technology for mobile devices, such as MP3 players, smartphones, tablets or netbooks. It features low power consumption, high responsiveness and mobility. However, flash technology also comes with several disadvantages. The device has a very specific physical organization, which results in a coarse granularity of data accesses. A flash memory is a type of EEPROM and, as such, memory cells need to be erased before being written again. Moreover, a flash memory cell can only be written a limited number of times before wearing out. The severity of those limitations is somehow mitigated by a software abstraction layer, called *Flash Translation Layer (FTL)*, which interfaces between common file systems and the flash device.

DAC 2012, June 3-7, 2012, San Francisco, California, USA.



(a) Hard Partitions (b) Soft Partitions

Figure 1: Hard versus soft partitioning. The FTL redirects random writes to a buffer and sequential writes directly to the data partition. The buffer is mapped to SLC to benefit from speed and low energy, while the data uses MLC for density. When writes are unbalanced across buffer and data, a hard partition will wear faster than the other, while soft partitioning (the contribution of the present paper) allows to balance the wear on the global device.

The basic functionality of an FTL is the translation of logical addresses to physical addresses. Such translation can be done following different policies which directly affect read/write performances, device cost and lifetime. The address translation table is typically stored in expensive SRAM cells, which makes the table size a critical design parameter of any FTL. A small translation table is cheap but implies coarse grained data manipulation (block-level) that results in poor performance and lifetime. Instead, a large table enables fine grained data manipulation (page-level), and thus higher performance and lifetime, but at a significant increase in device cost.

Hybrid-FTLs [4, 5, 6, 8, 9, 10] combine the two mappings by dividing the flash memory in two regions: a large data partition, which is addressed at the block-level, and a small log buffer partition, which is addressed at the page-level. The purpose is to direct random writes to the log buffer so that they can be written back to the data partition in-order as big chunks. Such an FLT requires a filter, illustrated in Figure 1a, to decide whether a particular data should be stored in the buffer partition or in the data partition.

Considering that a significant amount of memory accesses go to the small buffer partition, previous work [3, 6, 7, 11]

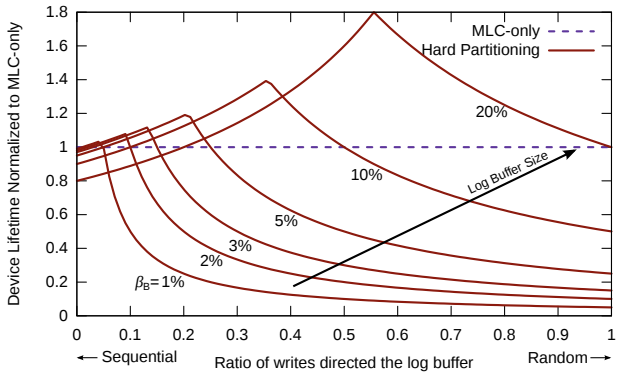


Figure 2: Hard partitions lifetime model for different buffer sizes as a function of the write ratio to the buffer. The model is normalized to a 2-bit MLC-only flash device lifetime. For large sequential accesses, where a FTL will more likely bypass the buffer, the device lifetime is bounded to the data partition. Small and frequently updated random accesses will wear out the buffer first, reducing the device lifetime to the buffer partition one. For reasonable amounts of random writes in practically affordable SLC buffers, lifetime is easily reduced by one order of magnitude.

proposed to build the buffer partition on *Single Level Cell (SLC)* flash, which provides high performance and low energy consumption but poor density, and the larger data partition on *Multiple Level Cells (MLC)* of lower performance but higher density. As a result, the flash device exhibits performances close to SLC (particularly for random data accesses) while keeping the area efficiency of MLC to a great extent. However, this previous work largely disregarded the effect of such SLC-MLC partitioning on the device lifetime.

Managing the SLC and MLC partitions as distinct physical parts of the flash device, as suggested by all the previous pieces of work, can lead to a serious reduction in lifetime. Figure 1a suggests how the extensive use of the buffer partition, due to a particular application write pattern, results in an unbalanced wear causing the device to fail well before most of its cells deteriorate above their maximum wear level (the large data partition is still healthy).

A more quantitative view on the impact of SLC-MLC partitioning to the device lifetime is illustrated in Figure 2, which will be further analyzed later. The figure shows the device lifetime, normalized to an MLC-only device for different buffer sizes as a function of the ratio of writes directed to the log buffer. Localized stress applied to small buffer (on the right of the graph) will easily reduce lifetime by an order of magnitude.

Accordingly, this paper proposes a soft SLC-MLC partitioning which changes the physical allocation of the buffer depending on the wear of the device. Such technique relies on the fact that an MLC can be managed as an SLC while largely keeping the performance benefits of a physical SLC. Figure 1b illustrates the soft partitioning technique, where each cell has a cumulated wear from MLC- and SLC-mode that can be globally balanced. The proposed soft partitioning can be coupled to the existing hybrid FTLs to significantly increase device lifetime while keeping the benefits in performance, energy and area shown on a hard partitioning.

The rest of the paper is organized as follows. In Section 2, we introduce combined SLC-MLC architectures and analyze their lifetime limitations. The soft partitioning is covered in Section 3. Lifetime measurements for soft and hard partitions are presented in Section 4. The related work is presented in Section 5 and Section 6 concludes our paper.

2. FLASH LOG MANAGEMENT

We principally identify two main NAND flash memory technologies: SLC and MLC. The latter stores multiple bits per memory cell providing a larger bit density and hence a smaller cost per bit. Several bits can be stored in a single cell by using multiple voltage levels; e.g., four voltage levels can store two bits. Manipulating MLCs is trickier than SLCs: a higher precision is now required to differentiate the multiple voltage levels making programming and reading significantly more complex, and therefore resulting in lower performances and higher energy consumption. Furthermore, MLC is more sensitive than SLC to the charge loss and neighboring cell interferences that typically affect flash reliability, which translates into a shorter lifetime. Therefore, MLC offers a higher bit density than SLC at the expense of a lower performance, higher energy consumption and reduced lifetime.

Hybrid FTLs combine an SLC log buffer partition with an MLC data partition to improve the performance of MLC: the more hot data (frequently accessed data) directed to the log buffer, the closer to the SLC performance. However, log buffers need to be carefully dimensioned: The smaller the buffer partition, the higher the bit density of the flash device. Yet, the impact of such partitioning on the device lifetime needs to be carefully considered.

Depending on the application write pattern, an unbalanced wear can occur between the buffer and data partitions. Each partition lifetime is proportional to its technology endurance and size, and inversely proportional to the ratio of writes directed to it. For example, let us take a budget of 100 cells, reserve 5% for an SLC buffer and the rest to the MLC data partition. Consider that the endurance of an SLC is about 10 times larger than of an MLC and, for this particular example, that each partition receives 50% of the writes. The MLC partition will then last $0.95/0.5 = 3.8$ times longer than an MLC-only device taking 100% of the writes. On the other hand, the SLC partition includes only 5% of the cells, which can store half of the bits of an MLC, and is 2.5% of the capacity an MLC-only device; accordingly, the lifetime of the SLC partition corresponds to $0.025 \times 10/0.5 = 0.5$ times the lifetime of an MLC-only device. This indicates that a device with such a hybrid configuration will last half of the time of an MLC-only device, which is already significantly shorter than the lifetime of an SLC-only device.

In order to compute analytically the lifetime of a hybrid flash device, we define α_B and α_D as the write ratios directed to the buffer and data partitions, respectively. Let β_B and β_D respectively be the ratio of the device's cells allocated to the buffer and data, and L_B and L_D be their respective lifetime normalized to MLC-only. Considering an n-bit MLC technology and an SLC endurance γ times larger than MLC, the lifetime of a hybrid device can be expressed as follows:

$$L_B = \frac{\gamma\beta_B}{n\alpha_B} \quad \text{and} \quad L_D = \frac{\beta_D}{\alpha_D}. \qquad \text{(1a, b)}$$

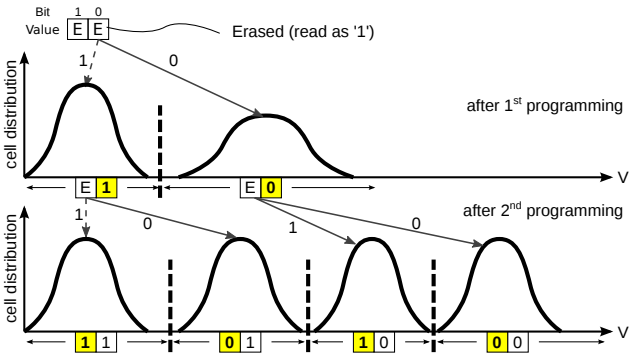


Figure 3: Programming of a 2-bit MLC. Each bit of a cell is programmed separately. Programming the first bit, or LSB, requires to target a single level (staying at the erased level is free) and does not need to be very precise. Programming the MSB, requires to read the current state of the cell and targets potentially 3 different levels, which requires more precision and time. Using cells in SLC-mode consists of programing only the first bit of each cell.

The hard partitions device lifetime corresponds to

$$L_H = min(L_B, L_D). \quad (2)$$

Assuming $n = 2$ and $\gamma = 10$, Figure 2 plots Equation 2 and represents the device lifetime, normalized to an MLC-only device, for different buffer sizes, β_B, and different ratio of writes directed to the log buffer, α_B. We observe that for reasonable buffer sizes (e.g., 5% and less), the lifetime of hybrid devices drops significantly when more than 25% of writes are directed to the buffer. Around one fifth of the cells should be allocated to the buffer to ensure a lifetime close to the MLC-only's.

The inability to balance the wear between its partitions is the main problem of hard partitioning. This, as shown in Figure 2, can seriously compromise the viability of hybrid-FTLs. In the following section we propose a different partitioning that overcomes this issue extending device lifetime.

3. SOFT PARTITIONING

Soft partitioning breaks the rigidity of hard partitioning by changing the physical placement of the log buffer depending on the device wear. This is enabled by the fact that MLC can be manipulated to obtain SLC write performances and that the FTL can keep track of the cumulative wear (SLC and MLC) to decide on the actual physical allocation.

3.1 Faster MLC: Managing MLC as SLC

MLC can also be used to store a single bit and recover the performance, energy consumption and lifetime benefits of SLC [6]. Figure 3 illustrates the programming of a 2-bit MLC. Each cell represents two bits, namely the LSB and the MSB, which are part of different pages and are programmed separately. Before starting the actual programming, the cell needs to be first erased. Then, the LSB is programmed targeting a single voltage level, which does not need to be very precise. In a final step, the MSB is programmed, which requires to first read the current state (i.e., the LSB value) to then push the cell voltage to either of the 3 different levels (see solid arrows in the figure). This second programming requires higher precision and it is typically three to four times longer than the LSB programming.

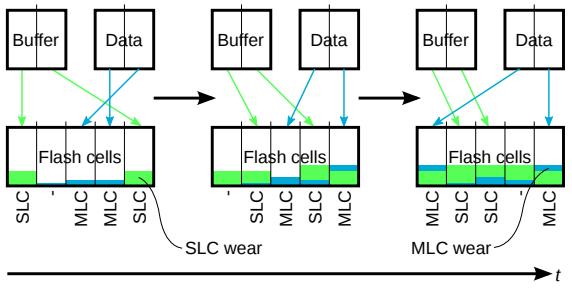


Figure 4: Software-controlled log buffer. A practical scenario of a hybrid-FTL, where cells switch between SLC- and MLC-mode regularly in order to balance the overall wear.

Interestingly, programming only the LSB of MLC shows performances very similar to SLC, which motivated previous researchers to propose the use of MLC as SLC for the statically allocated log buffer partition of Figure 1a. Thereby, performance is obtained at the expense of density in an MLC device. Such way of manipulating an MLC will be referred to as *SLC-mode* in the rest of the paper. Here, we propose to go one step further and opportunistically change the physical allocation of the buffer depending on the device wear.

3.2 Software-Controlled Log Buffer

Whereas hard partitions is typically applied on hybrid SLC-MLC hardware, the soft partitions scheme is applied to a completely homogeneous hardware architecture, made only of one or more MLC chips. The FTL is able to configure selectively regions of the flash chip to SLC-mode or MLC-mode at will, with the intention to evenly distribute the wear throughout the whole device. While small buffers are likely to die first for hard partitions, a soft partitioning can spread the localized stress over the complete device.

Figure 4 illustrates the evolution of a device using hybrid-FTL on soft partitions. Focusing on the leftmost physical block, one can see how this is initially allocated to the buffer, thus managed as SLC, then is freed from both partitions to be later allocated to the data partition, managed as MLC. Such transitions are typically triggered by the FTL wear balancing algorithm. Notice that the wear of the block increases through time and results from the times that the block is programmed as SLC and MLC.

Accordingly, the FTL needs to consider a global wear metric that includes the effects of both, the MLC-mode and SLC-mode, to decide the physical allocation of the log buffer. Such metric is detailed in the following subsection.

3.3 Lifetime of Soft Partitions

Let ω_{MLC} and ω_{SLC} be the wear associated to an MLC and SLC, respectively. The lifetime function of the soft partition scheme with respect to MLC-only is a linear function which is proportional to the write ratio directed to the buffer, α_B. In the one extreme, when the MLC-mode is exclusively used ($\alpha_B = 0$), the device lifetime is equal to the MLC-only. In the other extreme, when the SLC-mode is exclusively used ($\alpha_B = 1$), the device lifetime is limited by the wear of the SLC writes and the sensitivity of the MLC reads, which

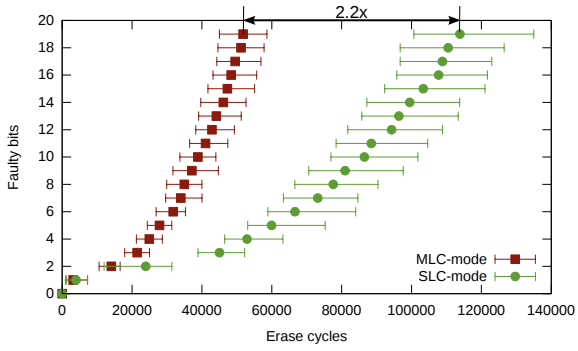


Figure 5: Comparison of SLC-mode and MLC-mode wear speed. Erase cycles are applied on the cells during which, in the case of SLC, only the first bit is programmed. Every hundred cycles, both bits are programmed and are verified for errors. We observed that SLC-mode requires in average 2.2 times more erase cycles than MLC to achieve the same error rate.

corresponds to $\frac{\omega_{MLC}}{n\omega_{SLC}}$, considering that lifetime is inversely proportional to wear. This corresponds to the ratio of the MLC and SLC wear divided by n, as MLC is able to store more bits than SLC. Notice that such lifetime is different with respect to an SLC-only device, where the lifetime is limited by the sensitivity of the SLC read (larger than MLC, as discussed in Section 3.1). When $\frac{\omega_{MLC}}{n\omega_{SLC}} \geq 1$, the soft partitions ensures a lifetime larger or equal to MLC-only. Accordingly, the indicated lifetime function corresponds to

$$L_S = \left(\frac{\omega_{MLC}}{n\omega_{SLC}} - 1\right)\alpha_B + 1. \quad (3)$$

However, $\frac{\omega_{MLC}}{n\omega_{SLC}}$ is a parameter that can not be found in the specifications of a standard MLC flash device, as chip manufacturers do not characterize its usage in SLC-mode. We built an FPGA-based platform to interface ONFI [1] compliant NAND flash chips and extracted experimentally this parameter. The latest NAND flash chips generations not being available for average consumers, we unsoldered some from USB sticks. We wore out cells programming them either in SLC- or MLC-mode, while periodically measuring faulty bits. In order to isolate the wear factor from the read reliability, faulty bits are gathered by programming in MLC and reading back the LSB and MSB. On Figure 5, we report the average number of cycles required to observe a certain amount of faulty bits for the first time on a 2-bit MLC. Error bars represent the positive and negative mean differences. We observe for this particular chip that 2.2 times more cycles are needed in SLC-mode to reach the same error rate than MLC. In our experiments, we will assume $\frac{\omega_{MLC}}{2\omega_{SLC}} = 1.1$.

4. RESULTS

In this section, the proposed soft partitioning technique is coupled with three published FTLs and the results of running three different data traces are compared to the hard partitioning reference.

4.1 Experimental Setup

We developed a trace-driven flash simulator in order to measure the execution time and erase counts of several FTL

	financial2	file copy	alix 8 d.
Total Data	1860MB	3606MB	5566MB
Mem Footprint	383MB	2767MB	1030MB
Data/Footprint Ratio	4.86	1.30	5.40
Weighted Req. Avg.	52.0KB	500KB	230KB
Weighted Req. Std. Dev.	65.8KB	21.3KB	177KB

Table 1: Benchmark characteristics.

executing three realistic traces. The 'financial2', obtained from the UMass Trace Repository [2], was gathered from an OLTP application running at a large financial institution. We generated two other disk traces from a tiny home server running a light Linux distribution. The traced storage is a 16 GB Compact Flash, which contains the systems main partition and a swap partition. The first trace, 'alix 8 days' is a 1-week trace running a web server. The second trace, 'file copy' was obtained from writing several GBytes of MP3 files. Some of the characteristics of the selected benchmarks are included in Table 1. The ratio between total data and memory footprint indicates the level of data reuse, while the ratio between the average and the standard deviation of the weighted request size indicates how different are the sizes of the different requests. Accordingly, we can conclude that 'file copy' includes sequential memory requests of similar size while 'financial2' and 'alix 8 days' include significant data reuse with memory requests of varied sizes.

We implemented three different FTLs, namely FAST [9], ROSE [4] and ComboFTL [7]. FAST is one of the first Hybrid-FTL, while ROSE is the most recent amelioration of FAST known by the authors. Although they originally use an MLC buffer, we allocate the buffer to SLC, which, as motivated in Section 3.1, increases performance and extends device lifetime. The only side effect is the increase in area, which we assume to pay off for reasonable buffer sizes. Lastly, ComboFTL includes an SLC buffer that gives multiple chance to victim data upon eviction. If the victim data is considered as being likely to be updated, it can be fed back into the buffer avoiding an expensive migration to the MLC partition.

The simulated flash characteristics were extracted from our measurement on a 4GBytes NAND flash chip. The flash chip shows a program latency of $400\mu s$ and $1600\mu s$ for the first bit and second bit, respectively. It has a read latency of $120\mu s$ and erasing takes $4ms$. We set the number of cells to 80 billion, corresponding to a 20GBytes MLC-only device.

4.2 Soft vs. Hard Partitioned Hybrid FTLs

The traces are executed twice by each FTL for several buffer sizes. The first run serves as a warm up and we collect the result with the second run. We assume a fully allocated logical space. We visit a large spectrum of parameters specific to each FTL and only keep the most effective combination for each trace. For FAST and ROSE, reducing execution time will systematically maximize lifetime, whereas for ComboFTL, originally build on hard partitions, a parameter provides a trade-off between endurance and performance. We show two parameter sets for ComboFTL: Combo$_L$ maximizes lifetime while Combo$_P$ maximizes performance.

The time spent in wear leveling is assumed to be negligible compared to the actual data migration process. Such assumption is supported by Chiao et al. [4], where the performance overhead of aggressive wear balancing applied on a Hybrid-FTL is shown to be between 1-2%. Thus, the execu-

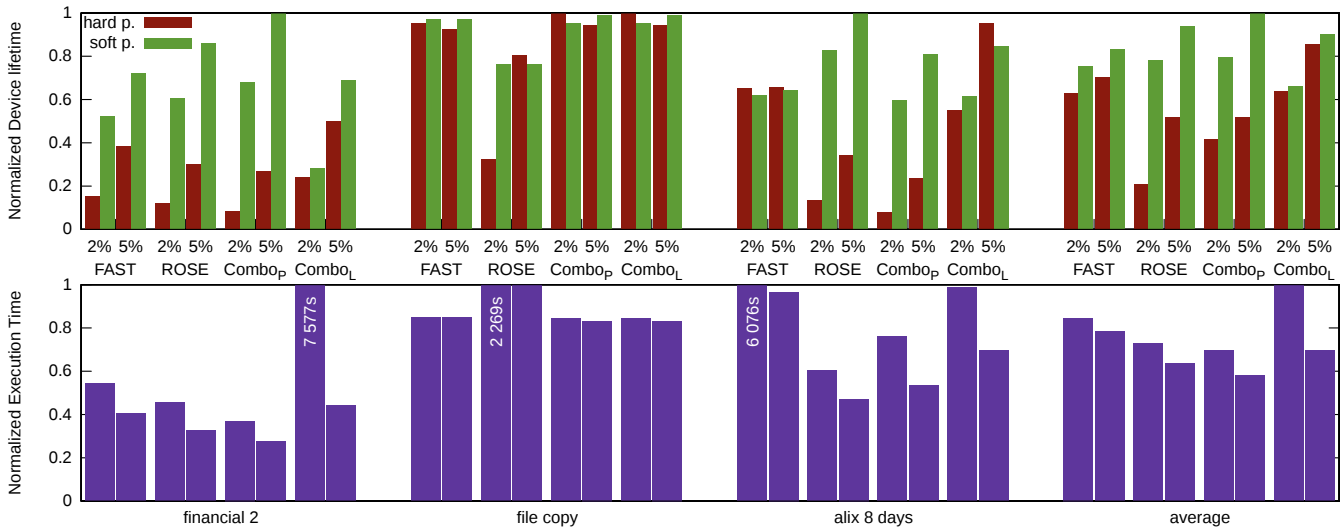


Figure 6: Lifetime and Performance. The results contrast our technique versus hard partitioning for various FTLs implemented with 2% and 5% buffer sizes and normalized for each trace to the largest value. Among a large spectrum of parameters specific to each FTL, only the best results are shown. Combo$_P$ and Combo$_L$ maximize performance and lifetime, respectively. In the case of performance, we assume the difference between hard and soft partitioning to be negligible. Overall, our soft partitioning significantly increases lifetime for practically every considered FTLs and benchmarks.

tion time of both, the soft and the hard partitioning scheme, is assumed to be the same in our experiments.

Figure 6 shows normalized life time (top) and normalized execution time (bottom) of the selected FTLs executing the benchmarks of Table 1 for buffer sizes of 2% and 5% of the total cell budget managed as hard partitions and as the proposed soft partitions. The results are normalized to the largest value for each trace.

The 'financial 2' trace has a large amount of small accesses. Increasing the buffer size will naturally reduce the amount of data evicted from the buffer and results in better performance and lifetime. We observe that the proposed soft partitioning is able to considerably increase the device lifetime with respect to hard partitioning for most of the configurations. We can see that when a lot of stress is put on the buffer, Combo$_L$ is able to extend hard partitions lifetime, sacrificing significantly performance. Interestingly, maximizing lifetime for hard partitions does not improve soft partitions lifetime. Indeed, soft partitions benefits from the fact that SLC-erase cycles wear less the cells while improving performance. Where hard partitioning requires to trade-off lifetime for performance, soft partitioning is able to obtain the best of both.

The 'file copy' trace being mostly made of very large sequential accesses, it bypasses completely the buffer and directs most of the accesses to the data partition, except for ROSE. Having the majority of writes directed to the MLC partition, annihilate most of the benefit of an SLC-MLC combined architecture and it is not surprising to observe similar lifetime between hard and soft partitioning. It also explains why a bigger buffer does not increase performance.

The 'alix 8 days' benchmark is composed of a good balance of both random and large sequential accesses. For FAST, we observed a large amount of sequentiality mispredictions, resulting in poor performances but slightly larger lifetime for hard partitions. Similarly to 'file copy', neither its lifetime nor execution time will be affected much by the buffer size. For ROSE and ComboFTL, increasing the buffer size has similar effects as 'financial 2': it increases performance and lifetime and we observe a large lifetime improvement with the soft partitioning. Also, we observe a case where Combo$_L$ can maximize lifetime further than soft partitions, however this comes at a performance cost with respect to Combo$_P$.

4.3 Generalization of Experimental Results

Figure 7 plots the lifetime results for the different configurations discussed in the previous subsection in the space introduced in Figure 2. New configurations, corresponding to additional log buffer sizes are also added to the figure.

For applications with random accesses, we observe that increasing the buffer size reduces the pressure on the data partition and results in higher ratios of writes to the buffer. In that spectrum of the plot, hard partitioning is only able to outperform soft partitioning for very large buffer sizes. Such region is shaded in the figure and annotated as *high cost*. However, hard partitioning fails to maximize the device lifetime for sensible buffer sizes. Such region is shaded in the figure and annotated as *typical*. The latter corresponds to scenarios that are likely to target a hybrid-FTL to increase the flash performance and that importantly will largely extend device lifetime when adopting the proposed soft partitioning.

For sequential access patterns, all the buffer sizes present a very low ratio of buffer writes, which results in marginal differences between the different cases, except for large buffer sizes. Such region, shaded in the figure and annotated as *sequential*, does not benefit from the hybrid-FTL schemes targeted in this paper.

5. RELATED WORK

The idea of an SLC-MLC combined architecture has been investigated in previous work. Grupp et al. [6] experimen-

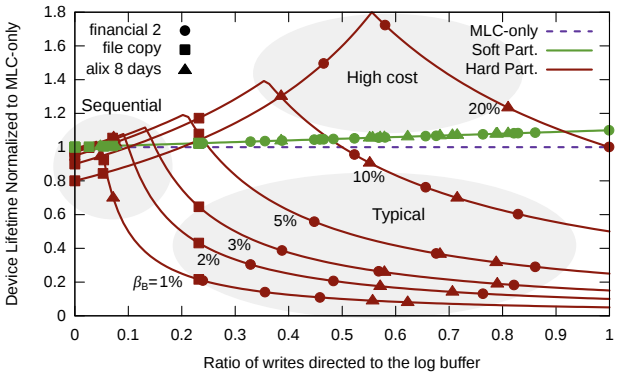


Figure 7: Lifetime models populated by benchmarks results. This graph shows the result of every best combinations of FTLs, traces and buffer sizes. In typical applications including random access patterns, soft partitions do systematically better. Only configurations characterized by a considerably higher cost can reach higher lifetimes by some 50% at most.

tally characterize MLC flash chips reporting performance and energy figures, even to use in SLC-mode. Based on those observations, they propose Mango, an FTL that makes use of SLC-mode to improve response time and reduce the total energy consumed. However, their FTL achieves a lifetime reduction of 35% with respect to MLC. They compensate such reduction by using an error correction code, the benefit of which can also be coupled to our approach.

Chang [3] proposes a Hybrid SSD combining SLC and MLC chips, which is a clear example of hard partitions referred in the paper. In order to extend the lifetime, they adapt the ratio of writes directed to the log buffer to balance the wear on each partition. Thereby, the performance is reduced for most of the cases. Instead, the proposed scheme respects the ratio of writes to the log buffer, which should have been optimized for performance by the FTL, and changes the physical allocation of the log buffer to balance the device wear, obtaining high performance without compromising device lifetime.

Park et al. [11] propose HFTL, an FTL based on an SSD architecture very similar to Chang's. In particular, they propose techniques to exploit multi-banks parallelism and maximize bandwidth. As us, they realize that the device lifetime is limited by the partition with the shortest lifetime, however, they mitigate the problem by sizing the SLC partition to guarantee a lifetime larger than the MLC partition. This oversizing, with 10 to 30% of the cells allocated to the log buffer, significantly increases the cost of the system, not only for the increase in flash cells but mostly for the huge address translation table associated, being only affordable in special niche markets.

Similarly, Im et al. propose ComboFTL [7], which can be tuned to either optimize lifetime at expenses of reducing performance or performance at expenses of reducing lifetime. Figure 6 shows that the combination of ComboFTL optimized for performance with our soft partitions is able to simultaneously achieve the largest lifetime and the higher performance.

Thus, to the best of our knowledge, this is the first work that proposes a soft partitioning of the log buffer present in hybrid FTLs that is continuously reallocated to distribute the device wear extending device lifetime at no cost.

6. CONCLUSIONS

Flash architectures combining SLC and MLC technologies are targeting new cost-sensitive applications displaying frequent random write patterns. The random accesses benefit from SLC performances while devices are primarily in MLC-mode to take advantage of the low cost of flash memory. However, unbalanced pressure on the SLC partition may lead to a premature death of the device. In this paper we have shown an approach that is robust to unbalanced stress and ensures a lifetime at least as long as that of an MLC-only device, showing up to 10 times lifetime improvement compared to the original approach. Interestingly for volume devices, this advantage comes at practically no extra cost.

7. REFERENCES

[1] (2011, Oct.) Open NAND flash interface 2.3. [Online]. Available: http://onfi.org/specifications/

[2] K. Bates and B. McNutt. (2007, Jun.) OLTP application I/O. [Online]. Available: http://traces.cs.umass.edu/index.php/Storage/Storage

[3] L.-P. Chang, "A Hybrid Approach to NAND-Flash-Based Solid-State Disks," *IEEE Trans. Computers*, pp. 1337–1349, Oct. 2010.

[4] M.-L. Chiao and D.-W. Chang, "ROSE: A novel flash translation layer for NAND flash memory based on hybrid address translation," *IEEE Trans. Computers*, pp. 753–766, Jun. 2011.

[5] H. Cho, D. Shin, and Y. I. Eom, "KAST: K-associative sector translation for NAND flash memory in real-time systems," in *Design Automation and Test in Europe*, Nice, France, Apr. 2009, pp. 507–512.

[6] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf, "Characterizing flash memory: anomalies, observations, and applications," in *ACM/IEEE Int. Symp. Microarchitecture*, New York, NY, USA, Dec. 2009, pp. 24–33.

[7] S. Im and D. Shin, "ComboFTL: Improving performance and lifespan of MLC flash memory using SLC flash buffer," *Journal of Systems Architecture*, pp. 641–653, Dec. 2010.

[8] J. Kim, J. M. Kim, S. Noh, S. L. Min, and Y. Cho, "A space-efficient flash translation layer for CompactFlash systems," *IEEE Trans. Consumer Electronics*, pp. 366–375, May 2002.

[9] S.-W. Lee, D.-J. Park, T.-S. Chung, D.-H. Lee, S. Park, and H.-J. Song, "A log buffer-based flash translation layer using fully-associative sector translation," *ACM Trans. Embedded Computing Systems*, Jul. 2007.

[10] S. Lee, D. Shin, Y.-J. Kim, and J. Kim, "LAST: Locality-aware sector translation for NAND flash memory-based storage systems," *ACM SIGOPS Operating Systems Review*, pp. 36–42, Oct. 2008.

[11] J.-W. Park, S.-H. Park, C. C. Weems, and S.-D. Kim, "A hybrid flash translation layer design for SLC-MLC flash memory based multibank solid state disk," *Microprocessors & Microsystems*, pp. 48–59, Feb. 2011.