Refresh frequency reduction of data stored in SSDs based on A-timer and timestamps

Marcelino Seif, Emna Farjallah, Franck Badets, Emna Chabchoub, Christophe Layer, Jean-Marc Armani, Francis Joffre, Costin Anghel, Luigi Dilillo, Valentin Gherman

Refresh frequency reduction of data stored in SSDs based on A-timer and timestamps

Marcelino Seif[1], Emna Farjallah[1], Franck Badets[1], Emna Chabchoub[1], Christophe Layer[1], Jean-Marc Armani[1], Francis Joffre[1], Costin Anghel[2], Luigi Dilillo[3], Valentin Gherman[1]

[1]CEA, LIST, Laboratoire Fiabilité et Intégration Capteurs
PC 172, 91191 Gif sur Yvette, France

[2]ISEP
28 rue Notre Dame des Champs
75006 Paris, France

[3]LIRMM
UMR 5506, 161 rue Ada
34095 Montpellier Cedex 5, France

***Abstract*—Increasing the number of bits per cell and technology scaling are ways to reduce the cost per gigabyte of flash memories and solid-state drives (SSDs). Unfortunately, this trend has a negative impact on data retention capability and cycling endurance. Periodic data refresh allows dealing with a reduced retention time and, indirectly, may be used to improve cycling endurance. A worst case data refresh frequency is not optimal in the presence of important temperature variations as it may become unnecessarily pessimistic and alter the SSD response latency and energy consumption. Here, a flexible data refresh methodology is proposed based on approximations of the Arrhenius-curves employed to describe the temperature impact on the retention capability of flash memories. These approximations may be implemented with the help of a small module called A-timer. For an asymmetric temperature distribution between 30°C and 70°C, it is estimated that the refresh frequency can be reduced by more than 63× and almost 3× for respectively charge detrapping and SILC failure mechanisms.**

Keywords—Flash memory; SSD; reliability; endurance; data retention; timestamps; Arrhenius law

I. INTRODUCTION

Solid-state drives (SSDs) based on NAND flash memories offer power and performance advantages together with improved shock and vibration resistance as compared to traditional hard-disc drives (HDDs) [15]. Since these benefits come with a higher cost per gigabyte, technology scaling and development of multi-level cells (MLCs) are used to fill the cost gap between SSDs and HDDs. Unfortunately, both trends affect cycling endurance, i.e., the cumulative number of program/erase (P/E) cycles that can be sustained by each memory cell. For example, cycling endurance is decreased by one decade for each additional bit stored in a MLC [5][15][26].

Besides cycling endurance, the reliability of non-volatile memories is quantified in terms of their data retention time, i.e., the longest period of time during which information can be reliably stored. In flash memories, data retention capability is particularly sensitive to the operating temperature and the number of endured P/E cycles. For example, the retention time of 24nm MLC flash memories may suffer a drop of 3 decades between 25°C and 85°C for a temperature dependence following the Arrhenius law with an apparent activation energy (E_{aa}) of 1.1eV [26]. Furthermore, the data retention time of these memories is reduced by a decade after only 3k P/E cycles [26].

Due to this mutual dependence, solutions aimed to deal with an insufficient data retention time may also be used to improve cycling endurance. As a matter of fact, the allowed number of cumulated P/E cycles can be augmented as long as a mechanism is provided to cope with the resulting retention time reduction. Besides the use of stronger error correcting codes [1], an efficient approach to deal with an insufficient retention time is to periodically check and refresh the stored data [2][15][18][19]. A refresh operation can be executed in-place by injecting only the missing amount of charge in the floating gates of the target flash memory cells or by reprogramming the concerned data at a different physical location [2][3]. In-place programming has smaller energy, response latency and P/E cycle overheads [2]. It was shown that the number of P/E cycles safely sustained by an MLC flash may be doubled if the guaranteed data retention time is reduced from three months to three days and data refresh operations are used as a compensation [19]. An increase of the number of P/E cycles larger than one decade is reported in [2] for a similar retention time reduction.

Endurance improvement based on data refresh operations is especially suitable for flash-based systems that may have only short downtimes. For example, this is the case of SSDs in data centers with a system downtime of maximum few hours per year [25]. Flash-based storage class memories [6] and HDD caches [10] are also potential beneficiaries since their contents may be saved to the underlying HDD storage before the power supply is switched off.

Another problem is that a worst-case refresh frequency may become excessively pessimistic with a consequent impact on response latency and energy consumption in the presence of important variations of the operating temperature which could be due to variable workload and/or environmental conditions [14].

This paper is focused on the reduction of the number of data refresh operations in flash memories via an integration over time of the temperature impact on their data retention capability. A small module, called A-timer, is proposed to facilitate the tracking of the temperature impact even when the sys-

tem power supply is off. Upon an A-timer warning, the remaining retention time of each valid memory block can be evaluated based on the difference between the current A-timer state and a timestamp provided by the A-timer itself during the last program operation of the block. Data needs to be refreshed if this difference is above a certain threshold that may depend on the number of P/E cycles endured by the memory block. For asymmetric distributions of the operating temperature between 30°C and 70°C, simulation results point out refresh frequency reductions of more than 63× and almost 3× when the dominant failure mechanism is charge detrapping and SILC, respectively.

A timestamp can be seen as a metadata type assigned to a flash memory block besides the number of endured P/E cycles, page validity flags, logical address mapping information, etc. [15]. Timestamps have been proposed to limit the number of refresh operations in DRAM-based L2 and L3 cache memories in order to improve their power consumption and response latency [20]. Timestamps have also been used to evaluate data retention age, i.e., the elapsed time since data was programmed, and enable online estimation of the raw bit error rate in flash memories [7].

The technique proposed here is orthogonal to solutions used to (a) reduce the number of retention errors like the utilization of read reference voltages which are aware of data retention age [4] or (b) cope with the limited cycling endurance of flash memories such as address translation, wear leveling, storage over-provisioning and data compression [15][22][23].

Possible Arrhenius curve approximations are presented in Section II. Estimated refresh frequency reductions for different temperature distributions are reported in Section III. Aspects related to the utilization of timestamps are considered in Section IV. Concluding remarks are drawn in Section V.

II. Time Integration of Temperature Impact on the Data Retention Time of Flash Memories

A flash memory cell consists of a MOS transistor with a floating gate or a charge trap layer, buried in the dielectric between the channel and the control gate. Data is programmed via the injection/erasure of electric charge to/from the floating gate or the charge trap layer. Unfortunately, charge leaks and data losses are unavoidable, especially at high temperatures and if the oxide around the floating gate has endured a large number of P/E cycles. The temperature impact on the data retention time τ_{RET} is usually described with the help of the Arrhenius law:

$$\frac{\tau_{RET}(T_{use})}{\tau_{RET}(T_{reference})} = \exp\left[\frac{E_{aa}}{K}\left(\frac{1}{T_{use}} - \frac{1}{T_{reference}}\right)\right]$$

where T_{use} and $T_{reference}$ are absolute temperatures, K is the Boltzmann constant and E_{aa} is the apparent activation energy that characterizes the dominant failure mechanism [9][11].

The main failure mechanisms responsible for data leaks in flash memory cells are *charge detrapping* and *stress-induced leakage current* (SILC) [16]. The temperature dependence of these mechanisms is illustrated in Figure 1. Charge detrapping has an E_{aa} between 1.1eV and 1.2eV [9]. In the case of SILC, the E_{aa} is between 0eV and 0.3eV, which implies a reduced sensitivity to thermal variations [9]. Either of these mechanisms can become the dominant one depending on the technology used or the range of operating temperatures [11][17]. Without affecting the genericity of the proposed approach, we will consider separately each of the two failure mechanisms over a full operating temperature range.

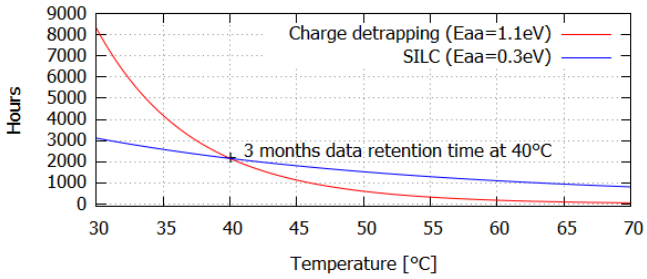

Fig. 1 Flash memory retention time as a function of temperature. Over the entire temerature range, the variations are 3.8× and 136× for *SILC* and *charge detrapping*, respectively. Flash memories in enterprise-class SSDs should ensure a power-off data retention time of 3 months at 40°C [8].

In the presence of important temperature induced variations of the data retention time, the usage of worst case refresh frequencies can be avoided by (a) regularly checking the operating temperature and (b) calculating the time integral of the temperature impact on data retention time. A warning is triggered and refresh operations may be executed when the time integral reaches a threshold value that indicates a potential data retention hazard. In the case of DRAM memories, which have their data retention capability also affected by the operating temperature, a single temperature measurement per refresh period may be sufficient due to the relatively short refresh interval, e.g., a few tens of milliseconds [12].

Since many SSDs today contain at least one temperature sensor [14], the time integral of the temperature impact can be computed by the SSD controller if it is regularly interrupted to execute the following operations:

- check the output of the temperature sensor;
- select the temperature acceleration factor from a precomputed table;
- add the selected value to the content of a status register;
- trigger a warning if the difference between the current status register value and the value when the last warning was triggered exceeds a preselected threshold.

The performance overhead of this monitoring procedure is small as it has a reduced number of instructions and is executed relatively infrequently. On the other hand, nowadays SSDs contain several channels with many raw flash chips per channel. This may induce a temperature gradient over several flash chips and the need to increase the number of temperature sensors, e.g., a temperature sensor per flash chip. In this case, the performance overhead of the above operations may rise quickly as they have to be executed for each temperature sensor. The power overhead is also important since one has to keep track of the temperature impact when the power supply is off.

In order to reduce these overheads, we propose a small hardware module, called A-timer, able to compute the mentioned time integral and alert against potential data retention hazards. The considered A-timer implementation is shown in Figure 2. The output of a digital temperature sensor is used to select an increment value for a status register. Warnings are issued each time the value stored in the status register is increased by a certain threshold value. The status register should not be reset even after a warning is triggered in order to use its states as timestamps. Consequently, after each warning, the threshold in Figure 2 has to be incremented with a *ΔThreshold* value that represents the difference to the next status register state for which an alert has to be issued. The status register only needs to be updated at a low frequency since the temperature variation rate is relatively low compared to system clock frequency. The low frequency clock generator can be an oscillator circuit or a frequency divider of a system clock signal.

The status register in Figure 2 is updated at a constant rate but with a variable increment value that approximates the temperature impact on the guaranteed data retention time as illustrated by the dashed curves in Figure 3. The dashed curves are maintained below the corresponding solid curves in order to avoid data retention hazards.

In the case of the charge detrapping mechanism, the red dashed curve is obtained by partitioning the operating temperature range into intervals over which the approximated data retention time τ_{APP_RET} is kept constant. τ_{APP_RET} is doubled at each transition from one temperature interval to its neighbor on the left-hand side. τ_{APP_RET} can be expressed with the formula below which involves the increment period τ_{CLK}, the increment value of the status register and *ΔThreshold*:

$$\tau_{APP\_RET}(T) = \tau_{CLK} \frac{\Delta Threshold}{Increment(T)} \quad (1)$$

Since the data retention time is reduced by 136× from 30°C to 70°C, the full operating temperature range is partitioned into 8 subintervals characterized by increment values equal to 2^i ($0 \le i \le 7$) with 2^7 assigned to the temperature interval that starts at 70°C.

A similar approach is followed for the approximation of the SILC-based Arrhenius curve represented by the blue dashed curve in Figure 3. As in the case of the charge detrapping mechanism, the temperature range is divided in 8 intervals over which τ_{APP_RET} is kept constant. At the left-hand side end of each temperature interval a constant value is added to τ_{APP_RET}. This value represents 35% of the guaranteed retention time at 70°C in order to bridge the 280% gap between the guaranteed retention times at 70°C and 30°C in 8 steps. The increment values for the status register can be calculated based on (1).

The solid curves in Figure 3 have been scaled to guarantee a power-off data retention of only 3 days at 40°C. This enables at least the doubling of the allowed number of P/E cycles as compared to the situation depicted in Figure 1 [2][19]. This relatively small data retention time may be useful for data center SSDs for which one should expect much shorter downtimes, e.g., maximum few hours per year [25].

With the exception of the digital temperature sensor and the low frequency oscillator, the gate count of the A-timer in Figure 2 amounts to about 1k 2-input NAND-equivalent gates which is at least one order of magnitude smaller than the gate count of a low power processor. For example, the Cortex-M0 processor has 12k gates at its minimum configuration [27]. This difference is amplified by the fact that a processor-based solution also requires program and data memories. A similar difference is expected to characterize the power consumption as processor low power techniques can also be applied to the A-Timer. Such techniques may include entering a deep sleep mode between consecutive temperature measurements or usage of an ultra-low leakage (ULL) logic cell library. Moreover, in an interrupt-driven operating mode, the duty cycle of the A-timer, i.e., the portion of time when it is active, is minimal since only one addition operation has to be performed.

The power consumption of the A-timer in Figure 2 is estimated at a few hundreds of nW for a CMOS 65nm technology. The operation of an A-timer during the rather short data center power outages [25] can be ensured with the help of a small backup battery. A different approach may be to (a) provide just enough energy to save the content of the A-timer status register into a NV-buffer when a power outage is detected and (b) consider the worst-case temperature evolution during the blackout period. Access to real-time clock information is a feature granted by the JEDEC standard 4.51 and can be used to get the outage duration [21].

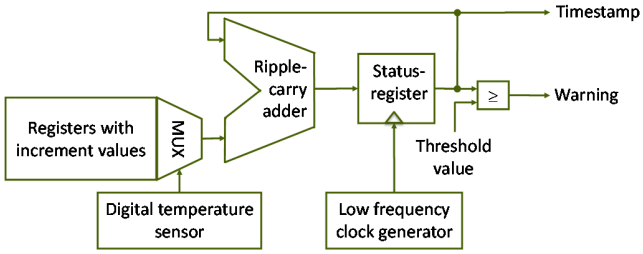

Fig. 2 Block diagram of a possible A-timer implementation.

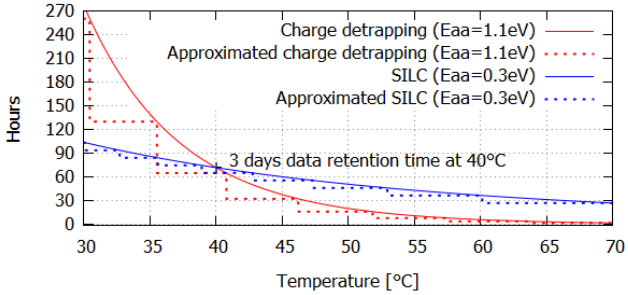

Fig. 3 Arrhenius curve approximations that can be achieved with the circuit in Figure 2.

III. Estimated Refresh Frequency Reductions

In order to estimate the refresh frequency reductions enabled by the proposed Arrhenius curve approximations, we considered different temperature distributions between 30°C and 70°C. Temperatures within this range have been measured in data centers SSDs [14]. Symmetrical and asymmetrical temperature distributions were taken into account via uniform, normal and Gamma distributions. The standard de-

viations of the normal and Gamma distributions were set to 3.3°C and 6.6°C, i.e., 1/12 and 1/6 of the whole temperature range. Figure 4 illustrates the probability density functions of the considered temperature distributions with a standard deviation equal to 3.3°C. The distributions represented by the dashed curves in Figure 4 are mirror images of those represented by solid curves with respect to a vertical middle axis.

Considering the approximation of the guaranteed data retention time given by each dashed curve in Figure 3, the mean of the approximated data retention time can be calculated as follows:

$$\tau_{\text{MEAN\_APP\_RET}} = \sum_{i=0}^{\text{sup}} \frac{CDF(T_{i+1}) - CDF(T_i)}{CDF(70°C) - CDF(30°C)} \cdot \tau_{\text{APP\_RET}}(T_i) \quad (2)$$

where:

- CDF is the cumulative distribution function of the probability density functions in Figure 4;
- *sup*=7 for both approximations in Figure 3;
- $CDF(T_{i+1})$-$CDF(T_i)$ is the probability that the operating temperature is in the interval [T_i and T_{i+1});
- $\tau_{APP_{RET}}(T_i)$ is the approximated data retention time in the operating temperature interval [T_i and T_{i+1});
- The T_i values represent the operating temperatures where the considered dashed curve touches the Arrhenius curve in Figure 3.

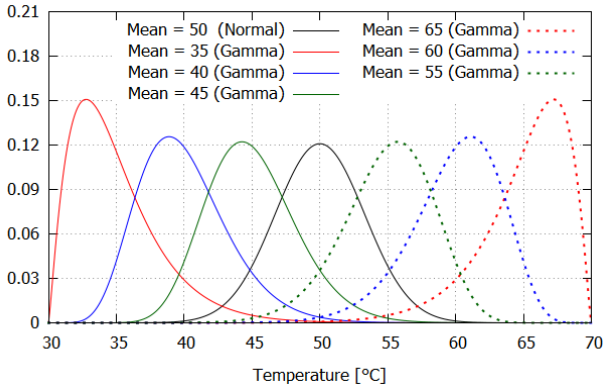

Fig. 4 Probability density functions (pdf) of the considered symmetrical (normal) and asymmetrical (Gamma) distributions of the operating temperature with a standard deviation of 3.3°C.

For each distribution, the estimated refresh frequency reductions are reported in Table I. The improvement with respect to the worst case refresh frequency is calculated as the ratio between $\tau_{MEAN\_APP\_RET}$ and the guaranteed data retention time at 70°C. The comparison with the ideal refresh frequency is given by the ratio between $\tau_{MEAN\_APP\_RET}$ and the expected data retention time for ideal Arrhenius curve approximation.

The refresh frequency reduction with respect to a worst case refresh frequency is sensitive to the E_{aa} value and the shape of the operating temperature distribution. The most important reductions correspond to the largest E_{aa} value and distributions biased towards low temperatures. The smallest frequency reductions were obtained for the Gamma distributions

TABLE I. ESTIMATED REFRESH FREQUENCY REDUCTIONS BETWEEN 30°C AND 70°C.

Distribution	Mean value [°C]	Standard deviation [°C]	Refresh frequency reduction w.r.t.			
			worst case refresh frequency (at 70°C)		ideal refresh frequency	
			1.1eV	0.3eV	1.1eV	0.3eV
Gamma	35	6.6	62.71	3.01	0.72	0.92
Gamma	35	3.3	52.53	2.92	0.70	0.91
Gamma	40	6.6	33.49	2.47	0.71	0.90
Gamma	40	3.3	27.81	2.42	0.72	0.90
Gamma	45	6.6	18.10	2.06	0.72	0.90
Gamma	45	3.3	14.62	2.01	0.72	0.90
Uniform	50	11.5	18.40	1.83	0.73	0.89
Normal	50	6.6	10.13	1.71	0.72	0.88
Normal	50	3.3	7.88	1.68	0.72	0.89
Gamma	55	6.6	5.85	1.44	0.73	0.88
Gamma	55	3.3	4.34	1.43	0.72	0.89
Gamma	60	6.6	3.37	1.21	0.73	0.87
Gamma	60	3.3	2.42	1.15	0.73	0.84
Gamma	65	6.6	2.07	1.08	0.74	0.91
Gamma	65	3.3	1.37	1.02	0.72	0.87

with the mean value at 65°C. The largest frequency reductions were achieved for Gamma distributions with the mean value at 35°C, i.e., 62.7× for E_{aa}=1.1eV and 3× for E_{aa}=0.3eV. The A-timer gain is expected to increase for wider temperature ranges.

For a given E_{aa} and different distributions with the same mean values, the refresh frequency reduction is improved when the standard deviation is larger since the time spent at lower temperatures and, implicitly, $\tau_{MEAN\_APP\_RET}$ are increased. This is confirmed by the uniform temperature distribution for which the refresh frequency reduction is more important than with the two normal distributions.

Comparisons with the ideal refresh frequencies show reductions around 0.71× for E_{aa}=1.1eV and 0.87× for E_{aa}= 0.3eV, which means that the approximated refresh frequencies are respectively 1.41× and 1.15× larger than the ideal frequencies. The better matching in the case of E_{aa}=0.3eV may be explained by the relatively lower variation rate of the Arrhenius curve and the resulting smaller approximation errors.

In the literature, it has been shown that important improvements of the flash memory lifetime can be achieved based on refresh operations [2]. Refresh operations may also be used to improve the average response latency of a flash memory as they allow to compensate for the retention time degradation resulting from faster but less precise write operations [19].

In the case of read-intensive SSD applications, i.e., *web search*, *MSR-Cambridge* [2], the frequency of functional data updates may become sensibly smaller than the refresh frequency and this may greatly limit the memory lifetime improvement when the refresh operations are based on data relocations. The solution proposed here can be used to sensibly reduce the P/E cycle overhead of data refresh based on relocation operations and further extend SSD lifetime. In the case of read-intensive SSD applications, the magnitude of the lifetime improvement can reach the same amplitude as the refresh frequency reductions reported in Table I. In the case of refresh procedures based on in-place data reprogramming [3], our solution allows to reduce the number of data checks and refresh operations.

IV. Monitoring based on Timestamps and A-Timer

In an SSD, one A-timer can be assigned to handle all flash memories covered by a temperature sensor. This is possible only if a timestamp is associated to each flash memory block. Following an erase operation, a timestamp needs to be updated when the first page of its corresponding block is programmed.

The states of the A-timer status register can be used as timestamps. When a warning is issued by the A-timer, the timestamp of each memory block is compared to the state of the A-timer status register. If the difference is larger than a certain threshold the valid pages are refreshed. If the memory blocks contain pages with the similar *write-hotness*, i.e., functional update frequency [13][24], the block pages will have the tendency to be updated and invalidated simultaneously. This will not only improve the *garbage collection efficiency* [15] but also reduce the number of blocks that may require refresh operations.

Since warnings from one single A-timer are used to verify the timestamps of memory blocks with heterogeneous data, the time period between warnings τ_{WARN} should be smaller than the approximated data retention time τ_{APP_RET} of all memory blocks. Moreover, it is necessary that τ_{WARN} is smaller than $\tau_{APP\_RET}/2$ in order to avoid that all memory blocks are refreshed each time a warning is triggered. The case when τ_{WARN} is equal to $\tau_{APP\_RET}/2$ is illustrated in Figure 5 which indicates the moments when refresh operations have to be executed for two arbitrary memory blocks, B_1 and B_2. If these blocks are programmed between the A-timer warnings at t(i) and t(i+1) the valid data in both blocks should be refreshed after the A-timer warning at t(i+2).

In Figure 5, one can see that B_1 is refreshed shortly before its data retention time is exhausted while B_2 just after its data retention time is halved. This means that the data retention time may be affected by an underestimation between 0 and $\tau_{WARN}=\tau_{APP\_RET}/2$ besides the Arrhenius curve approximation errors. This is the price to pay for managing a large number of memory blocks with one single timer, not necessarily an A-timer. Fortunately, this underestimation affects only the first refresh operation after a memory block is programmed. The underestimation of the data retention time induced by the first refresh operation can be reduced by selecting τ_{WARN} equal to τ_{APP_RET}/M with M>2. This implies an underestimation between 0 and $\tau_{WARN}=\tau_{APP\_RET}/M$. Finding the M value that offers the best performance, power consumption and endurance trade-off for a given SSD application is beyond the scope of this paper.

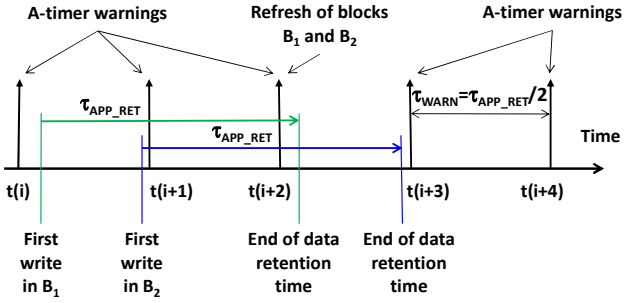

Fig. 5 Refresh operations as a function of A-timer warning period, τ_{WARN}, and the data retention time, $\tau_{APP\_RET}=2\cdot\tau_{WARN}$, scaled at a constant operating temperature, for 2 arbitrary data blocks.

Successive refresh operations can be required for static and cold data characterized by a small functional update frequency, e.g., user files, executable files and operating system files. Moreover, such data is likely to be stored via static or dynamic wear leveling approaches in those memory blocks with the largest number of P/E cycles and shortest data retention time and, implicitly, may require refresh operations [15].

In case of SSDs with relaxed wear leveling policies, blocks with different numbers of endured P/E cycles may have different guaranteed data retention times and, implicitly different τ_{APP_RET} values. In this case, τ_{APP_RET} can be approximated to the closest smallest multiple of τ_{WARN}, i.e., $\tau_{WARN} \cdot floor(\tau_{APP\_RET}/\tau_{WARN})$. Consequently, one may have some memory blocks which are refreshed every second A-timer warning, as illustrated in Figure 5, other memory blocks refreshed every third warning and so on. This corresponds to an adaptive refresh technique in which the refresh frequency is modulated by the number of P/E cycles of each memory block [2][3].

Blocks with different data retention capabilities may be found in hybrid SSDs made of different flash memory types, e.g., with 1 bit per cell (SLC) and 2 bits per cell (MLC) [5] [10]. The cycling endurance of SLC and MLC memories differ by at least one decade and it may happen that the MLC cells reach sooner their maximal number of P/E cycles and, implicitly, their minimal guaranteed data retention time. Such exhausted cells can still be used to store static or cold data provided that they are refreshed based on in-place programming.

In order to evaluate the timestamp storage overhead, consider an SSD with 32 raw flash chips partitioned into 4 channels. A flash chip contains 2^{13} blocks with 2^7 pages of 8kB. This implies a total of 2^{18} blocks and the same number of timestamps for the whole SSD. Consider that the SSD has a single temperature sensor and an A-timer is used to measure the operating temperature every second for three years. If one considers the Arrhenius curve approximation shown by the red

dashed curve in Figure 3 and 2^7 the maximum increment value of the A-timer status register, the status register state needs to have at most 34 bits. Although the timestamps correspond to A-timer status register states, they may be shorter as some of the least significant bits of the status register can be neglected. For example, ignoring the least significant 6 bits in the status register induces a measurement error of 63 seconds. A 28-bit timestamp represents a negligible storage overhead compared to a block size of 1MB. For the whole SSD, the required timestamp information amounts to 28×2^{18} bits and can be stored in a single memory block. In comparison, the flash translation layer (FTL) used for the logical to physical translation of page addresses requires a table with 25×2^{25} bits, i.e., a total of 100MB.

As the FTL and other management information, the timestamps can be uploaded into the working RAM of the SSD controller and saved to the non-volatile media when the power is switched-off. The performance overhead of checking 1MB of timestamps is negligible. For example, according to the refresh scheme in Figure 5, the timestamps need to be checked once every 36 hours by the SSD controller if τ_{APP_RET} is equal to 3 days.

V. Conclusions

Periodic refresh of stored data may be used to deal with an insufficient data retention capability and also improve the cycling endurance of flash memories. Unfortunately, in the presence of important operating temperature variations, a worst-case refresh frequency may become excessively large and induce unnecessary penalties. A specially designed module, called A-timer, is proposed to reduce this overhead if one assumes an Arrhenius law for the temperature dependence of the guaranteed data retention time. Despite its simple design, the A-timer is able to efficiently approximate the impact of a changing operating temperature. For asymmetric temperature distributions over a range of 40°C measured within data center SSDs, the estimated refresh frequency reductions may reach 63× and 3× for respectively charge detrapping and SILC failure mechanisms. A methodology based on timestamps was presented to enable the utilization of one A-timer for a whole SSD. The timestamps can be provided by the A-timer itself.

Refrences

[1] D. Bertozzi et al., "Performance and reliability analysis of cross-layer optimizations of NAND flash controllers," ACM Transactions on Embedded Computing Systems, vol. 14, no. 1, Article 7, 2015.

[2] Y. Cai et al., "Flash correct-and-refresh: retention-aware error management for increased flash memory lifetime", IEEE International Conference on Computer Design, pp. 94–101, 2012.

[3] Y. Cai et al., "Error analysis and retention-aware error management for flash memory," Intel Technology Journal, Volume 17, Issue 1, pp. 140–164, 2013.

[4] Y. Cai et al., "Data retention in MLC NAND flash memory: characterization, optimization, and recovery," International Symposium on High Performance Computer Architecture, pp. 551–563, 2015.

[5] L.-P. Chang, "Hybrid solid-state disks: combining heterogeneous NAND Flash in large SSDs," IEEE Asia and South Pacific Design Automation Conference, pp. 428–433, 2008.

[6] G. Crump, "What Is Storage Class Memory?" 2011.

[7] S. Di Carlo et al. "FLARES: an aging aware algorithm to Autonomously Adapt the Error Correction Capability in NAND Flash Memories," ACM Transactions on Architecture and Code Optimization, vol. 11, issue 3, article no. 26, Oct. 2014.

[8] JEDEC Standard, "Solid-state drive (SSD) requirements and endurance test method," JESD218A, February 2011.

[9] JEDEC Standard, "Failure mechanisms and models for semiconductor devices," JEP122G, October 2011.

[10] T. Kgil et al., "Improving NAND flash based disk caches," International Symposium on Computer Architecture, pp. 327–328, 2008.

[11] K. Lee et al., "Activation energies (Ea) of failure mechanisms in advanced NAND flash cells for different generations and cycling," IEEE Transactions on Electron Devices, vol. 60, no. 3, pp. 1099–1107, March 2013.

[12] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: retention-aware intelligent DRAM refresh," ACM SIGARCH Computer Architecture News - ISCA '12 Volume 40, Issue 3, pp. 1–12, 2012.

[13] Y. Luo, Y. Cai, S. Ghose, J. Choi and O. Mutlu, "WARM: improving NAND flash memory lifetime with write-hotness aware retention management," Symposium on Mass Storage Systems and Technologies, pp. 1–14, 2015.

[14] J. Meza, Q. Wu, S. Kumar and O. Mutlu, "A large-scale study of flash memory failures in the field," ACM SIGMETRICS International Conference on Measurement and Modeling of Computer System, pp. 177–190, 2015.

[15] R. Micheloni, L. Cripa and A. Marelli, "Inside NAND Flash Memories", Springer-Verlag, Berlin, 2010.

[16] N. Mielke et al., "Bit error rate in NAND flash memories," IEEE Annual International Reliability Physics Symposium,pp. 9–19, 2008.

[17] N. Mielke et al., "Flash EEPROM Threshold Instabilities due to charge trapping during program/erase cycling," IEEE Transactions on Device and Materials Reliability, vol. 4, no. 3, pp. 335–344, September 2004.

[18] Micron Technology, "TN-12-30: NOR flash cycling endurance and data retention introduction," Technical Note, 2013.

[19] Y. Pan, G. Dong, Q. Wu and T. Zhang, "Quasi-nonvolatile SSD: trading flash memory nonvolatility to improve storage system performance for enterprise applications," High Performance Computer Architecture, pp. 1–10, 2012.

[20] W.R. Reohr, "Memories: exploiting them and developing them," IEEE International SOC Conference, pp. 303–310, 2006.

[21] D. Ruggeri, "The challenges of scaling nonvolatile memory in embedded systems - How Micron e•MMC embedded memory simplifies high-capacity storage," July 31, 2013.

[22] Sandisk, "WP001 - Flash management/A detailed overview of flash management techniques," White Paper, November 2013.

[23] Seagate Technology LLC, "SSD over-provisioning and its benefits," 2016.

[24] R. Stoica and A. Ailamaki, "Improving flash write performance by Using Update Frequency," Proceedings of the VLDB Endowment, vol. 6, no. 9, pp. 733-744, 2013.

[25] N. Sundby and D. Taylor, "Beyond capacity: storage architecture choices for the modern datacenter," White Paper, IDC Analyze the Future, Sponsored by: Toshiba, February 2014.

[26] Swissbit, "X-500 / X-55 series SLC vs. EM-MLC," White Paper, April 17, 2014.

[27] J. Yiu, "The definitive guide to the ARM cortex-M0," ISBN: 978-0-12-385477-3, Elsevier, 2011.