Improving NAND flash performance with read heat separation

Roman Pletka, Nikolaos Papandreou, Radu Stoica,
Haris Pozidis, Nikolas Ioannou
IBM Research, Zurich
{*rap, npo, rst, hap, nio*}*@zurich.ibm.com*

Tim Fisher, Aaron Fry,
Kip Ingram, Andrew Walls
IBM Systems, USA
{*fisher, aaronfry, ingram, awalls*}*@us.ibm.com*

***Abstract*—The continuous growth in 3D-NAND flash storage density has primarily been enabled by 3D stacking and by increasing the number of bits stored per memory cell. Unfortunately, these desirable flash device design choices are adversely affecting reliability and latency characteristics. In particular, increasing the number of bits stored per cell results in having to apply additional voltage thresholds during each read operation, therefore increasing the read latency characteristics. While most NAND flash challenges can be mitigated through appropriate background processing, the flash read latency characteristics cannot be hidden and remains the biggest challenge, especially for the newest flash generations that store four bits per cell.**

In this paper, we introduce read heat separation (RHS), a new heat-aware data-placement technique that exploits the skew present in real-world workloads to place frequently read user data on low-latency flash pages. Although conceptually simple, such a technique is difficult to integrate in a flash controller, as it introduces a significant amount of complexity, requires more metadata, and is further constrained by other flash-specific peculiarities. To overcome these challenges, we propose a novel flash controller architecture supporting read heat-aware data placement. We first discuss the trade-offs that such a new design entails and analyze the key aspects that influence the efficiency of RHS. Through both, extensive simulations and an implementation we realized in a commercial enterprise-grade solid-state drive controller, we show that our architecture can indeed significantly reduce the average read latency. For certain workloads, it can reverse the system-level read latency trends when using recent multi-bit flash generations and hence outperform SSDs using previous faster flash generations.

I. Introduction

Today, 3D-NAND flash is the prevalent memory technology in consumer and enterprise solid-state drives (SSDs) as well as mobile devices. The transition from 2D to 3D NAND flash enabled drastic storage density increases and cost reductions but also helped to address endurance limitations [1], [2]. Storage density continues to increase by the addition of more layers and stacking groups of layers, enhancing the number of bits stored per memory cell.

All these techniques affect the latency characteristics in multiple ways [3]. First, storing more bits per cell exponentially increases the number of threshold voltage distributions. For instance, when moving from triple-level cells (TLC) to quad-level cells (QLC), the number of threshold voltage levels grows from 8 to 16. Figure 1 gives an overview of the typical average read latency ranges from different 3D-NAND devices on the market. The read latencies of multi-level cells (MLC), TLC, and QLC, are normalized to the average latency of the single-level cell (SLC) technology. As can be seen, latency rises with increasing numbers of bits stored per cell.

Second, the denser the threshold voltage distributions, the higher is the likelihood that these distributions overlap, hence, more elaborate read algorithms are needed [4]. Flash media characterization has shown that QLC NAND flash is significantly more susceptible to errors than previous generations [5], [6]. The impact of these drawbacks on read latency is one of the main reasons why SSDs using QLC NAND flash are not widespread in the enterprise storage market today.

The latency degradation is not equal across all flash page types [7]. As we will discuss, some flash pages may have a lower latency and error rate than other pages. By placing frequently read data on fast flash pages, we can achieve a reduction in the average latency, reduce the number of high-latency reads which results in a tighter latency distribution, and reduce the probability of read retries. This idea is promising conceptually, but challenging to implement in practice. Implementing RHS in a flash controller is constrained by computational, space, and bandwidth limitations as the technique must be implemented in hardware and is part of the critical data path. More specifically, the challenges are:

- The size of the read heat metadata is strictly limited by the DRAM capacity available in an SSD. Alternatives such as paging read heat metadata from flash, introduce additional latency and would defeat the purpose.
- Flash controllers have an upper bound on the computational complexity as they cannot afford to introduce additional latency in the critical path. Generally, the chip real estate and power envelope is limited.
- It is not evident how a controller should use read heat information with minimally interfering the data path and other background operations.
- Initially there is no read heat information available, which is undesirable for real applications, as the read latency can be initially very high, but improve over time.

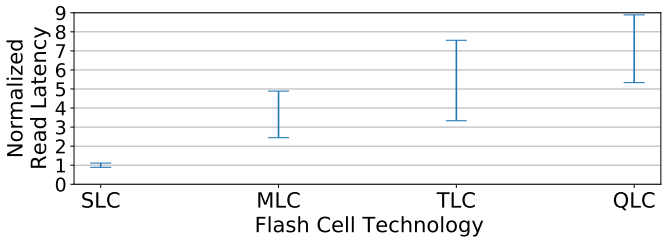


Fig. 1: Average read latencies of different 3D-NAND flash cell technologies normalized to a typical average SLC latency.

The contributions and findings of this paper can be summarized as follows:

- We present a new controller architecture that accommodates RHS in a non-disruptive fashion.
- We present different bit encoding schemes as well as scalable and lightweight read heat tracking strategies and explain the various tradeoffs involved.
- We evaluate the efficiency of RHS for various heat tracking schemes. This is the first study that analyzes the effects from read and write skew overlap and the ability to adapt to workload changes.
- We implemented our proposal in a simulation environment, analyze the potential average read latency reduction that can be achieved, and show that a QLC controller can achieve equal or better performance characteristics when compared with a traditional TLC controller architecture for certain workloads.
- We demonstrate the end-to-end efficiency of our technique by implementing it in a commercial enterprise-grade SSD.

The remainder of this paper is structured as follows: Section II presents relevant NAND flash properties and controller architectures. Our new read-heat aware controller architecture and read-heat tracking strategies are introduced in Section III. Then, we describe our simulation environment and evaluate our controller architecture by means of extensive simulations and on our real SSD controller (Section IV). Section V gives a survey of related work before concluding our paper in Section VI.

II. Background

We first describe some of the relevant NAND flash memory properties and then give an overview of existing flash controller architectures.

A. Properties of NAND flash memory devices

NAND flash cells store data as electrical charge placed in the floating gate or charge trap layer of the transistor forming the cell. Typically, several thousand cells are connected through a single word-line (WL), while several hundreds or more than a thousand WLs form a block. Depending on the amount of charge stored, the cell belongs to one of several states or levels. Figure 2 depicts the probability density function (PDF) of the threshold voltage V_{th} distribution in a TLC NAND flash device and the associated states. The number of states l is given by the number of bits n stored in a cell as $l = 2^n$. When no charge is stored, the cell is in the erase state E. All other states, $L1$ to $L7$, are reached after programming when some charge is placed into the cell.

A WL is organized into n physical pages, one for each bit stored in the cells forming that WL. Figure 2 illustrates an exemplary mapping of the most-significant bit (MSB), the

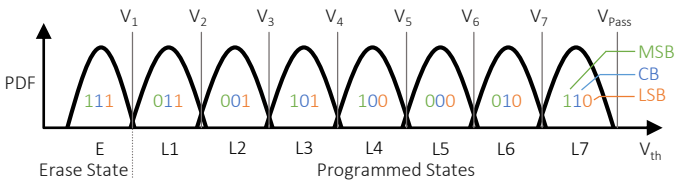


Fig. 2: Threshold voltages V_{th} in a TLC flash device.

Page	E	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	# Thr
P_a (LSB)	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
P_b	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	2
P_c	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	4
P_d (MSB)	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	8

(a) ORBC: Exponentially increasing number of read thresholds.

Page	E	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	# Thr
P_a (LSB)	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
P_b	1	1	1	0	0	0	0	1	1	0	0	0	0	1	1	1	4
P_c	1	1	0	0	1	1	0	0	0	0	0	1	1	1	1	0	5
P_d (MSB)	1	0	0	0	0	1	1	1	1	1	0	0	1	1	0	0	5

(b) PBBC: Single read threshold for lower page and balanced number of read thresholds for other page types.

Page	E	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	# Thr
P_a (LSB)	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0	0	3
P_b	1	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	4
P_c	1	1	1	0	0	0	0	0	0	1	1	1	1	0	0	1	4
P_d (MSB)	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	1	4

(c) MBBC: Maximal balanced binary coding scheme.

TABLE I: Exemplary Gray coding schemes.

center bit (CB), and the least-significant bit (LSB) to the erase and program states of a TLC flash cell. A physical page is the granularity at which data is programmed. As charge cannot be removed from cells, programming can only increase the amount of charge in a cell and an extra erase operation is needed to remove the charge of all cells in a block. When reading a physical page, only a single bit of information is extracted from each cell of the selected WL. As only a single read voltage level can be tested at a time, the read out is done in an iterative process where each one of the tested threshold voltages from the set of read voltages V_1,...,V_7 adds up to the read latency.

B. Gray coding

Gray coding schemes, also known as reflected binary coding, order the binary values in a code such that two successive values differ in only a single bit [8]. In flash, charge variations in cells may result in the crossing of a single read voltage level such that a neighboring state is detected upon a read operation instead of the originally programmed one. Such crossings increase the raw bit error rate (RBER). Preferably, one would like to minimize the number of affected bits from a single crossing event. The properties of Gray codes guarantee that only one physical page sees an increase in the error count from a single crossing.

There are many variants of Gray codes. Here, we focus on three particular Gray codes to pinpoint their effectiveness in RHS: The original reflected binary coding (ORBC) [8], a partially-balanced binary code (PBBC), as well as a maximally balanced binary code (MBBC) [9]. Typically, flash devices only implement a single coding scheme.

An example of the ORBC is given in Figure 2, where the read threshold voltage V_4 is applied to read the LSB, thresholds V_2 and V_6 the CB, and V_1, V_3, V_5, and V_7 the MSB. Table I gives an overview of the three selected Gray coding schemes for a QLC NAND flash device. The four physical pages in a WL are sorted in the programming order. In ORBC, reading a P_d page implies testing eight read thresholds resulting in a significantly higher read latency than the P_a page. PBBC retains the single threshold for the P_a page, but balances the number of read thresholds for all other page types while the MBBC balances all read thresholds.

C. Controller architectures

Early multi-bit NAND flash devices only supported a single mode meaning that the provided command set only allowed for programming all bits in the cells to operate the device reliably. Conventional SSD controllers use flash devices of the same type and operate those devices in the advertised mode only. The use of a single tier is still common in enterprise storage as it provides consistent performance and has low complexity [2].

To improve write bursts while keeping the cost advantage of multi-bit NAND Flash, *hybrid flash controllers* were introduced [10], [11]. From a consumer perspective, hybrid flash controllers have the potential to approach the read performance of SSDs at a much lower cost, using SLC for skewed workloads. A small number of SLC devices were combined with many MLC devices where the SLC portion was used as a fixed-size cache [10]. This architecture writes data first into the SLC cache and destages valid data later to multi-bit flash when the SLC cache is full. While these controllers improve write latency and throughput for bursty write workloads with significant idle times, the sustained write performance could fall below that of a single tier SSD. This is because the controller must perform two writes internally for every host write, in the worst case. From a cost perspective, it is beneficial to build SSDs from a single flash type and leverage the ability of recent flash generations to operate in either a high-density multi-bit mode or in a high-performance single-bit mode [11].

Later, the second generation of hybrid controllers introduced *adaptive SLC caches* where the amount of SLC blocks is adjusted based on the current capacity utilization [12], [13]. Such controllers show excellent performance when the used capacity is low, but when the used capacity increases, the SSD will hit the same sustained write performance issue. A more recent approach tries to adjust the tier sizes and data placement strategy as a function of workload properties to optimize write performance and endurance [14].

III. Data-placement Architecture

In this section, we first propose our data-placement architecture, describe lightweight alternatives for read heat tracking, and discuss the implications for a real controller.

A. Description of the architecture

Figure 3 illustrates our controller architecture with a data placement unit supporting RHS for QLC flash. The architecture is generic such that it is applicable to all types of flash controllers from Section II-C. All new components are marked in yellow. In flash controllers, a data placement unit picks erased blocks for writing data (1). A fully programmed block is moved to the occupied block pool (2). When the number of erased blocks is low, garbage collection (GC) selects a victim block, with the (close to) least amount of valid data from the occupied block pool (3) [15]. All data still valid is read and written to a new location (4), and cleaned blocks are erased and made available for data placement. Besides GC, other internal processes such as wear leveling (WLL) may select

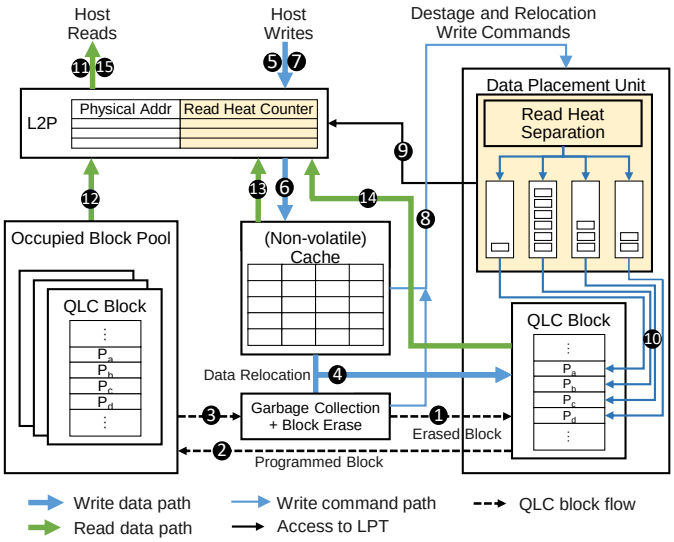


Fig. 3: Generic controller architecture with a data placement unit supporting RHS.

blocks from the occupied block pool. Finally, the logical-to-physical (L2P) table is used to map logical pages, also known as logical block addresses (LBAs), to physical locations in flash.

Host writes (5) are first placed into a cache (6). There are several types of caches typically used in SSDs. Simple destage buffers tend to be very small and are used to hide the flash write latency. Using a non-volatile destage buffer permits writes to be acknowledged to the host (7) before they are written to flash. They usually consist of storage class memory (SCM) such as MRAM, battery-backed DRAM or SLC flash. The cache could also be resized dynamically using a variable percentage of the flash blocks in SLC mode. In this case, blocks are initially configured in SLC mode as long as the used capacity is low which enables tracking of read heat information before data is destaged to QLC. As such a cache can hold up to a quarter of the total capacity, the L2P can be leveraged to point to either a location in the occupied block pool or the cache. This has the advantage that it already allows for the adjustment of the read heat while the host write is placed into the cache as the L2P has to be accessed.

When the non-volatile cache gets full, data is evicted from the cache using a cache replacement policy such as least-recently used (LRU) or first-in first-out (FIFO). The evicted data is then prepared for destaging to QLC and a relocation write command is queued to the data placement unit (8). The data placement unit maintains separate queues for each read latency class. The read latency classes discriminate between the different latency characteristics of physical pages. For example, using ORBC, each page type corresponds to one read latency class where the P_a page maps to the lowest and the P_d page to the highest read latency class. With PBBC, a single read latency class could cover write requests for the P_c and P_d pages. In our design, the queues can hold write commands to fill up to two QLC blocks. This is sufficient to achieve good separation as we will show in Section IV. Next, the read heat counter corresponding to the LBA being relocated is read from the L2P (9). The RHS unit maps frequently read LBAs to the lowest read latency class and rarely read LBAs to

the highest read latency class and so on. When accessing the read heat counter, the data placement unit may update its value depending on the used read heat scheme described below.

In flash, the physical pages in a block must be programmed in sequential order to minimize the propagation of programming errors. Hence, upon programming a physical page, the data placement unit dequeues a write command from the queue corresponding to the read latency class that best matches the current page type (10). In the case no such command is available, another one is taken from a neighboring queue. As soon as a page is programmed, the space in the cache can be freed.

Host reads (11) fetch data from either the location at which it is stored in the occupied block pool (12), the cache (13), or the QLC block in the data placement unit (14) that is currently being written. The location is determined by an L2P lookup. After the completion of the read operation, data is delivered to the host (15). All host read operations increment the corresponding read heat counter using one of the algorithms described below irrespective of where the page read is located.

B. Tracking read heat information

The key parameters that influence the efficiency of read heat tracking, are, the granularity at which read heat is tracked, the resolution of the read heat counters, the procedure to update the heat counters, and the mapping from the read heat counter value to a read heat stream.

1) Read heat counters: Preferably, we would like to collect read heat information for every LBA at a reasonable resolution. As the L2P maps LBAs, typically 4 KiB or 16 KiB today, to their physical location, it therefore sounds reasonable to integrate the read heat counters into the L2P to minimize memory accesses. However, per-LBA heat information at high resolution increases the size of the L2P by roughly 50% which is a non-negligible cost factor when the L2P is kept in DRAM for performance reasons as is the case for enterprise-level SSDs [1]. Therefore, we extend the L2P with a space efficient k-bit saturating counter to track the read heat. Using $k=2$, such a 2-bit counter increases the L2P by only a few percent and allows for tracking 4 heat levels which matches the maximum number of QLC page types. This permits a simple mapping from read heat to latency classes and hence QLC page types.

2) Incrementing read heat: When using low resolution counters, the read heat should be incremented probabilistically because the counters saturate quickly otherwise. For a k-bit saturating counter we denote p_i as the probability of incrementing the counter from value i to $i+1$ where $0 \leq i < 2^k - 1$ and introduce

$$\hat{p} = [p_0, ..., p_{2^k-2}] \quad (1)$$

as the probability vector defining a read heat increase scheme. For write heat tracking, it has been shown that such a heat increase scheme is optimal and increasing the number of heat levels quickly leads to diminishing returns [16].

Note that SSD internal or higher-level scrub reads are not indicative of the read heat of data and should not increase the read heat counters. Without going into further details, this can be achieved, for example, using existing protocol features.

[1]We assume a 16 bit read heat counter and an SSD with a storage capacity of 10 TB using a logical page granularity of 16 KiB. This requires a mapping entry of at least 4 Bytes to address the full LBA space to which 2 Bytes would be added for the read heat counter.

	Host Write	GC/WLL Write	Host read
RRHD	–	–	Probabilistic
RHGW	Reset	Reset	–
RHWO	Reset	–	–
DGWO	–	Decrement	–

TABLE II: Summary of read heat decrease schemes.

3) Decrementing read heat: We consider four different read heat decrease schemes as summarized in Table II:

- *Random read heat decrease (RRHD)*: Whenever a read operation results in an increase of the read heat counter, another used LBA is randomly selected and its read heat is decremented.
- *Reset on host or GC/WLL writes (RHGW)*: When an LBA is overwritten or relocated by GC or WLL, the read heat counter is reset to zero.
- *Reset on host writes only (RHWO)*: The reset of the read heat counter is only done upon a host write to the LBA. Internal relocations do not change the read heat information.
- *Decrement on GC/WLL writes only (DGWO)*: Any internal relocation of an LBA will decrement its read heat counter by one. Host writes preserve the value of the read heat counter.

All four schemes are simple to implement in hardware. RRHD uses more resources than the others as it requires two accesses to the L2P and the controller has to select a counter for decreasing heat of an LBA that actually holds data.

C. Discussion on the architecture

The accuracy of RHS is influenced by the workload properties (e.g., the read and write skew, the used capacity, the overlap between the read and write LBA ranges), the size of the cache, the capacity of relocation write command queues, and the read heat tracking strategy. With a fixed-size cache, the gathering of read heat information while data is residing in the cache is very limited. When the cache is a destage buffer, a good separation can only be achieved after data is relocated from QLC-to-QLC.

From this perspective, a hybrid controller architecture with an adaptive SLC cache size has huge advantages and works in synergy with RHS. First, blocks are initially configured to operate in SLC mode and read heat information is collected over a long time frame until the device reaches a certain used capacity point at which destaging to QLC starts. Once blocks are converted to QLC, accurate read heat information is then available.

Second, a dynamically resizable SLC cache can be enhanced with an additional destage buffer. This has the following advantages: Updating read heat counters comes at no extra cost because the L2P has to be accessed at the same time.

Third, the SLC GC can be a simple circular buffer. Hence, SLC blocks remain about the same time (measured in incoming writes) in the pool during which read heat information is collected. Thus, RHS is significantly less sensitive to the GC policy of the QLC pool where the choice of an adequate GC algorithm is essential due to its limited endurance. For these reasons, our SSD controller evaluated in Section IV-F implements this architecture.

Note that our architecture can be combined with so-called superblocks that group flash blocks from different chips and planes together as well as write heat separation [2], [17]. We consider these approaches out of scope of this paper because they do not influence RHS.

IV. Evaluation

In the first part of the evaluation, we focus on the average read latency at queue depth one (i.e., when a single read I/O is submitted at a time). As real SSD controllers vary significantly in their implementation, we want to exclude all effects not related to RHS such as the handling of internal parallelism and command prioritization. This allows us to study the fundamental aspects of RHS. Later on, we present results from a commercial enterprise-grade controller where we implemented RHS.

Most of our tests use skewed synthetic workloads. It has been shown that real-life workloads typically exhibit a skewed access pattern [18], [19], [20]. The skewed synthetic workloads we use in our evaluations follow a Zipfian distribution [21] to closely mimic real-life workload types. They have been widely used to study device characteristics [2], [22]. We denote these workloads as *Zipf x/y* where we adjust the skew factor of the Zipfian distribution such that x percent of generated operations access y percent of the LBA space for the given device size.

A. Simulation environment

Our simulation environment is inspired from a real state-of-the-art flash controller developed in-house. The full flash translation layer (FTL) is simulated with a usable capacity of 1TB consisting of more than ten thousand flash blocks. We found that scaling up the capacity does not affect the actual results obtained. The simulator uses a cyclic buffer GC policy which has the advantage of inherently relocating data and hence emulating retention or read disturb limits. This is reasonable as we are not interested in internal write amplification here.

All flash operations are emulated and the actual read latency values obtained from characterization are used to determine the achievable latency reductions. The simulator uses a minimally-sized cache large enough to hold data for two full QLC blocks on which the data placement unit performs data separation. Upon a host write, a relocation write command is immediately queued to the data placement unit. Host writes and GC queue relocation write commands until the queues are full. Then, the data placement operation is performed. The presented results are normalized to the average read latency of an SLC device from Figure 1.

B. The RHS potential of different bit encoding schemes

We analyze the achievable latency reductions for the Gray coding schemes presented above for various read workloads. The L2P mappings are initially randomized such that each LBA points to a different physical location, a typical precondition workload for characterizing SSDs [23]. Next, we issue a large enough Zipfian write workload (i.e., two full device writes) during which we perform RHS. Using the apriori knowledge of the read workload that we will exercise afterwards, we can easily assign each LBA to the correct read latency class to get the optimal data placement. Finally, we issue 2TB of reads following the desired read workload and measure the number of reads for each page type to evaluate the average read latency for each scheme.

The results are shown in Figure 4. When the reads follow a uniform random distribution on the entire LBA space, the data placement has almost no influence and the read latency matches the average read latency of all page types irrespective of the Gray coding scheme used. Significant latency reductions can be attained with skewed workloads: For all Gray coding schemes latency is decreasing with increasing workload skew.

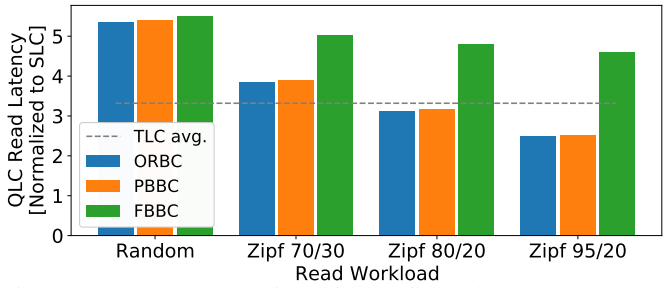


Fig. 4: QLC read latencies with optimal data placement as a function of the read workload. The latency numbers are normalized to the typical average SLC read latency. The dashed line corresponds to a typical TLC device with no RHS.

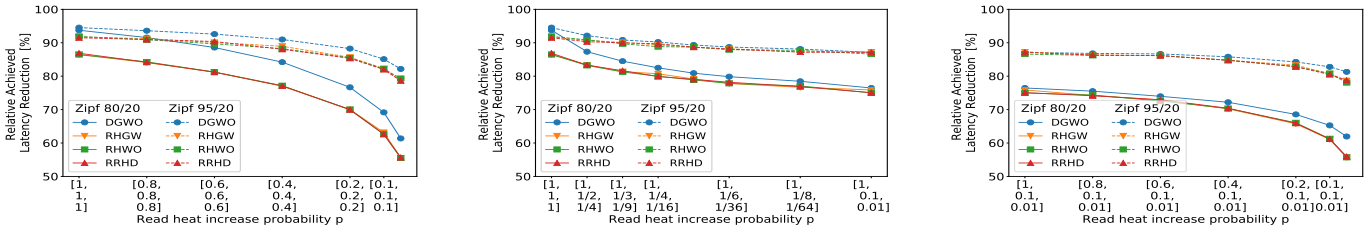
Optimal data placement with ORBC and PBBC outperform a TLC device without RHS when the skew is higher than a Zipfian 80/20 and achieve more than 2.1× reduction in average read latency for Zipf 95/20. When the skew is higher than a Zipfian 80/20, they outperform a TLC device without RHS. Further, ORBC is only marginally better than PBBC, indicating that the capability to identify the read hot data is key for performing RHS. For FBBC, despite the thresholds being maximally balanced among all page types, a slight read latency reduction can still be observed owing to the P_a page that requires sensing one threshold less than the other page types.

C. Comparing read heat tracking strategies

We now study the sensitivity of the different read heat tracking schemes. In contrast to the previous test, RHS is performed based on the read heat information gathered during the test without prior knowledge. The comparison focuses on the ORBC, which exhibits the highest read latency reduction above. To determine how close the different tracking strategies approach the theoretically achievable read latency reduction, we evaluate a set of workloads using a read-write ratio of 80/20 and 98/2 where both, reads and writes, follow Zipfian distributions with the same skew factor but with an offset that minimizes their overlap (i.e., most reads go to an LBA region that is rarely written to and vice versa).

Figure 5 illustrates the results from a large set of read heat increase and decrease strategies. The first strategy uses the same heat increase probability for all p_i (Figures 5a and 5d). In the other two strategies, the increase probability drops exponentially with higher read heat values. In Figures 5b and 5e the first increase probability p_0 is fixed at $p_0 = 1$ to quickly detect whether a page has been read. The results in Figures 5c and 5f vary p_0 between 1 and 0.05, but use the same exponential decrease for p_1 and p_2. These increase schemes are combined with all four read heat decrease strategies from Section III-B3.

Overall, there is no single read heat tracking strategy that performs best for all scenarios and the performance varies significantly. First, a high read heat increase probability only performs well when the read-write ratio is reasonably low (Figure 5a). As soon as the read-write ratio increases and the workload is less skewed, read heat counters saturate quickly. In some cases,



(a) Equal read heat increase probabilities, 80% reads, 20% writes.

(b) Exponential read heat increase probabilities with $p_0 = 1$, 80% reads, 20% writes.

(c) Combination of read heat increase probabilities, 80% reads, 20% writes.

(d) Equal read heat increase probabilities, 98% reads, 2% writes.

(e) Exponential read heat increase probabilities with $p_0 = 1$, 98% reads, 2% writes.

(f) Combination of read heat increase probabilities, 98% reads, 2% writes.

Fig. 5: Sensitivity of RHS to combinations of read heat increase and decrease strategies. The results show the relative achieved latency reductions compared to optimal placement as a function of various read heat probability increase vectors.

almost no benefit from RHS can be measured (Figure 5d). Second, equal read heat increase probabilities p_i are very sensitive to workload properties. They may work very well for highly skewed workloads, but perform poorly with less skew (Figures 5b and 5d). Therefore, these strategies should generally be avoided. Third, a reasonable high probability (i.e., $p_0 \gtrsim 0.4$) is important – especially when the workloads are less skewed (Figures 5c and 5f). Highly skewed workloads see significantly less accesses in the cold region and can hence tolerate a lower p_0.

Using exponentially decreasing probabilities for p_1 and p_2 with a reasonably high p_0 exhibits the least sensitivity to workload properties, albeit other configurations may occasionally perform better. DGWO reduces the average read heat latency generally better than all other schemes, while RHGW, RHWO, and RRHD perform similarly well.

D. Read and write skew overlap

Real-world workloads are not only likely to be skewed, but the read and write skew also tend not to overlap [24]. Here, we analyze the efficiency of RHS as a function of the overlap of the read and write skew.

The results in Figure 6 have been obtained using a moderate and a highly skewed workload (Zipf 80/20 and Zipf 95/20 for both, reads and writes) and using two different read-write ratios (i.e., 80/20 and 98/2). Our simulations are based on a fully utilized SSD. The read and write workloads therefore operate on the entire LBA space. All curves are plotted as a function of the offset between the read and write skew where the x-axis denotes the offset as a percentage of the total capacity by which the distribution of the writes is shifted relative to the distribution of the reads. We report the relative achieved latency reduction as a percentage of the maximum achievable reduction with optimal placement. Again, the simulations use ORBC. We only show the read heat decrease strategies of DGWO and RHGW to improve readability. We have experimented with all strategies and can confirm that RRHD and RHWO perform similarly to RHGW.

DGWO performs almost always better than RHGW irrespective of read and write skew offsets. With a moderate workload skew Zipf 80/20, the read write ratio has a strong influence on the latency reduction achieved. When the read write ratio is 80/20, the ability to identify the most read cold data by incrementing the read heat with a probability of one upon the first read, significantly improves the achieved latency reduction.

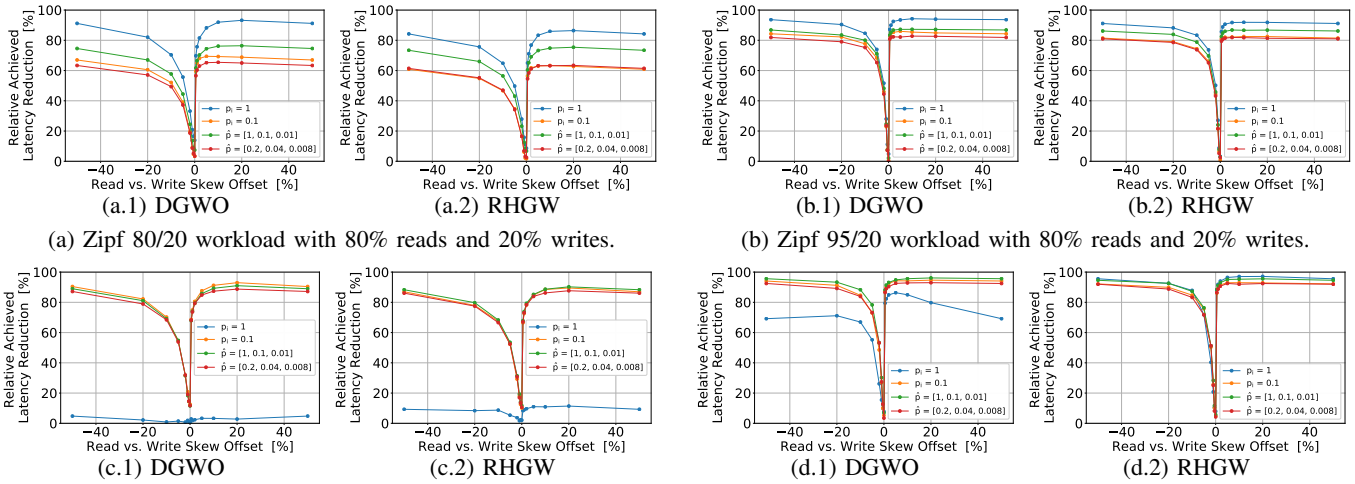
With a high read write ratio of 98/2 and the same moderate workload skew Zipf 80/20, all strategies except the one using $p_i = 1$ for heat increase achieve between 80 and 93 % latency reduction for a wide range of read-write skew offsets. When the workload skew is high, all strategies except DGWO with $p_i = 1$ achieve high average latency reduction over a large range of read-write offsets. Only within a small range of about 10 % of overlap between the read and write hot LBAs, do the algorithms exhibit a significant drop in achieved latency reduction. Almost no latency reduction can be achieved when the read and write workloads completely overlap.

E. Adaptation speed to workload changes

An important metric is the speed at which RHS adapts to workload changes. Here, we select the heat tracking strategy $\hat{p} = [1, 0.1, 0.01]$ and DGWO combined with ORBC, that performs consistently well and mostly outperforms the other schemes as presented above. In this experiment, all L2P mappings are randomized, all read heat counters are reset, and no RHS is performed initially. We then perform a large amount of reads following a skewed Zipfian distribution.

Next, we issue the same Zipfian-type of workload as writes at an offset that minimizes the overlap with reads executed before.

Every 10 % of the LBA space written, the same read workload as before is executed and the read latency is measured. The writes are used to trigger GC and the gathered read heat information so far is used in RHS when data are written to new locations. The sequence is repeated until RHS is no longer able to reduce read latency.



(a) Zipf 80/20 workload with 80% reads and 20% writes.

(b) Zipf 95/20 workload with 80% reads and 20% writes.

(c) Zipf 80/20 workload with 98% reads and 2% writes.

(d) Zipf 95/20 workload with 98% reads and 2% writes.

Fig. 6: Relative achieved latency reduction for DGWO and RHGW as a function of the read and write skew offset.

Figure 7 shows the average latency reduction relative to optimal data placement as a function of the number of writes. After writing an amount of data that corresponds to 20 % of the logical capacity, a significant latency reduction can be achieved when the workload skew is high. In particular, for a Zipf 95/20 workload, 76.5 % of the possible reduction is already achieved at this point. Even a lightly skewed workload such as Zipf 70/30 exhibits benefits from RHS at this point, although it only reaches 21.9 % latency reduction. Zipf 80/20 and 95/20 reach their maximum latency reduction after an amount of data that corresponds to 30 % of the logical capacity has been written and used for RHS while Zipf 70/30 reaches this level at about 40 %.

This is important for hybrid controllers with dynamically resizable tiers: As initially all blocks are in SLC, they have time to track sufficient read accesses before blocks are converted to QLC. When the capacity used increases and the SLC cache shrinks, data is destaged to QLC blocks and written to flash pages in accordance with their read heat. Therefore, such a design can offer both, high storage capacity and good read performance. For non-hybrid controllers or controllers with a small cache, the amount of writes to get good RHS is significant such that actively identifying and relocating read hot data currently stored in slow pages is advisable [25].

F. Measurements on a real SSD

We implemented RHS in a commercial enterprise-grade SSD controller with QLC flash using ORBC and measured its efficiency. We preconditioned the drive with sequential and random writes such that 80 % of the logical capacity was used. Using two concurrent read workloads, the first one issuing 95 % of the reads to 5 %, and the second one 5 % of the reads to 95 % to the written LBA space, we measured the read latency distribution first without RHS. We then added a concurrent uniform random write workload. At this point, all overwrites were placed using the read heat information gathered. We then measured the read latency again.

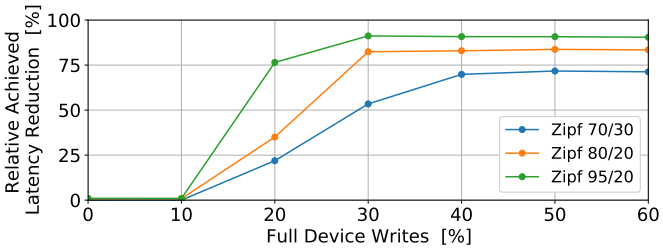


Fig. 7: Speed of adaptation to a new workload.

Figure 8 illustrates the cumulative density functions (CDF) of the two measured read latency distributions. The x-axis is normalized to the average QLC latency without RHS. The four steps visible in the plot are caused by the latency characteristics of the different page types. We clearly see that the levels change with RHS. While the number of the fastest P_a page reads has increased from 24.2 % to 44.0 %, the number of the slowest P_d page reads has been reduced by 4× to only 5.5 % of all reads. Overall, RHS reduced the average latency by 17.1 %.

V. Related Work

Gray coding schemes are widely used [8], [9]. In data storage systems, they can be used to reduce the raw bit error rate [26]. Choi et al. [7] introduced invalid data-aware (IDA) coding, where the voltage levels are reprogrammed when the lower bit is invalidated to accelerate future reads. The reprogramming of already programmed cells causes additional cell-to-cell interference which increases the RBER of neighboring pages whereas our approach does not affect the reliability. IDA coding could easily be combined with our heat-aware data placement.

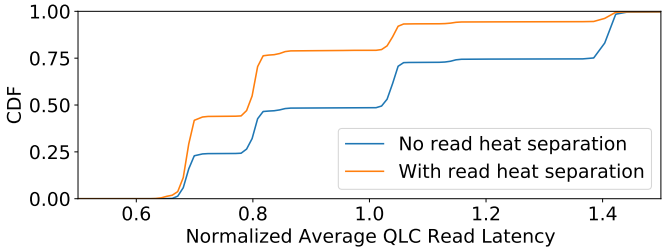


Fig. 8: CDF for the read latency measured in a real SSD normalized to the average QLC latency.

Improving read performance of multi-bit flash based on latency variation of page types was first introduced in Fastread [25]. Data migration is only performed during idle time and only a subset of all pages are being tracked.

Read heat information can be leveraged for other purposes such as mitigating read disturb effects [27] or creating additional data copies to reduce the tail latency for frequently read data [28], [29]. Such techniques are synergistic to our proposal. Given that heat tracking adds a non-negligible amount of complexity, it is beneficial to leverage the read heat information for multiple purposes.

Some existing flash controllers perform *write* heat separation to reduce the internal write amplification of an SSD [2], [30], [31], [32]. There is a tension between performing write and read heat separation simultaneously. We propose to keep the write heat separation logic unchanged and only perform RHS for data that would be normally written inside a block.

VI. Conclusions

We show that in multi-bit NAND flash the selection of an appropriate bit encoding scheme enables read latency variations among different physical page types which can be efficiently exploited. This paper evaluated manifold key aspects and trade-offs in the controller architecture that influence RHS. Our RHS exploits the fact that the access patterns of real-world workloads are typically skewed. Our experiments show that close to the optimal read latency can be achieved, without impacting write performance and with limited controller changes. In contrast to other heat tracking schemes, we demonstrate that read heat tracking at a very fine granularity can be implemented in real controllers supporting large capacities of several tens of Terabytes. For reasonably skewed workloads, our design can even outperform previous faster NAND flash generations at the system level.

RHS efficiency improves with increasing workload skew. We show that more than 80 % of the potential reduction in average read latency can be achieved when the read and write skews are not overlapping. Measurements on a real SSD with RHS show an average latency reduction of more than 17 % on the system level. One key aspect is the efficiency at which the heat tracking scheme can detect never or rarely accessed data locations.

Further, read heat tracking can only start once data has been written. This is a clear drawback for non-hybrid controllers or controllers with a small cache as the data has to be relocated in the background at some point later. Until then, no benefit from RHS is achieved. We therefore strongly suggest the use of a hybrid controller architecture with a dynamically resizeable SLC cache. This also allows for simplifying the SLC GC, decoupling the performance of RHS from the QLC GC policy, and hence offer both, high storage capacity and good read performance.

References

[1] K.-T. Park *et al.*, "Three-dimensional 128 Gb MLC vertical NAND flash memory with 24-WL stacked layers and 50 MB/s high-speed programming," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, 2015.

[2] R. Pletka *et al.*, "Management of next-generation NAND flash to achieve enterprise-level endurance and latency targets," *ACM Trans. on Storage*, vol. 14, no. 4, 2018.

[3] M. Hao, G. Soundararajan, D. Kenchammana-Hosekote, A. A. Chien, and H. S. Gunawi, "The tail at store: A revelation from millions of hours of disk and SSD deployments," in *Proc. USENIX Conf. on File and Storage Technologies*, Feb. 2016.

[4] Y. Cai, E. F. Haratsch, O. Mutlu, and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling," in *Proc. Design, Automation Test in Europe*, 2013.

[5] N. Papandreou *et al.*, "Reliability of 3D NAND flash memory with a focus on read voltage calibration from a system aspect," in *Proc. Non-Volatile Memory Technology Symposium*, 2019.

[6] N. Papandreou *et al.*, "Open block characterization and read voltage calibration of 3D QLC NAND flash," in *Proc. IEEE Int. Reliability Physics Symposium*, 2020.

[7] W. Choi, M. Jung, and M. Kandemir, "Invalid data-aware coding to enhance the read performance of high-density flash memories," in *Proc. IEEE/ACM Int. Symposium on Microarchitecture*, 2018.

[8] F. Gray, "Pulse code communication," US Patent 2,632,058, Mar. 1953.

[9] J. P. Robinson and M. Cohn, "Counting sequences," *IEEE Trans. Comput.*, vol. C-30, no. 1, 1981.

[10] L.-P. Chang, "A hybrid approach to NAND-flash-based solid-state disks," *IEEE Trans. on Computers*, vol. 59, no. 10, 2010.

[11] S. Im and D. Shin, "ComboFTL: Improving performance and lifespan of MLC flash memory using SLC flash buffer," *J. Syst. Archit.*, vol. 56, no. 12, 2010.

[12] D. Glenn, "Optimized client computing with dynamic write acceleration," 2014. [Online]. Available: https://www.micron.com/~/media/client/global/documents/products/technical-marketing-brief/brief_ssd_dynamic_write_accel.pdf

[13] M.-C. Yang, Y.-H. Chang, C.-W. Tsao, and C.-Y. Liu, "Utilization-aware self-tuning design for TLC flash storage devices," *IEEE Trans. VLSI Syst.*, vol. 24, no. 10, 2016.

[14] R. Stoica *et al.*, "Understanding the design trade-offs of hybrid flash controllers," in *Proc. IEEE Int. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2019.

[15] L.-P. Chang, T.-W. Kuo, and S.-W. Lo, "Real-time garbage collection for flash-memory storage systems of real-time embedded systems," *ACM Trans. on Embedded Computing Systems*, vol. 3, no. 4, 2004.

[16] R. Stoica and A. Ailamaki, "Improving flash write performance by using update frequency," *Proc. of the VLDB Endowment*, vol. 6, no. 9, 2013.

[17] B. Van Houdt, "A mean field model for a class of garbage collection algorithms in flash-based solid state drives," *SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, 2013.

[18] L. Cherkasova and M. Gupta, "Analysis of enterprise media server workloads: Access patterns, locality, content evolution, and rates of change," *IEEE/ACM Trans. Netw.*, vol. 12, no. 5, 2004.

[19] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim, "A case for flash memory SSD in enterprise database applications," in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2008.

[20] S. Liu, X. Huang, H. Fu, and G. Yang, "Understanding data characteristics and access patterns in a cloud storage system," in *Proc. IEEE/ACM Int. Symposium on Cluster, Cloud, and Grid Computing*, 2013.

[21] B. C. Arnold, *Pareto distribution*. Wiley Online Library, 1985.

[22] Y. Yang and J. Zhu, "Write skew and zipf distribution: Evidence and implications," *ACM Trans. on Storage*, vol. 12, no. 4, 2016.

[23] L. Bouganim, B. P. Jonsson, and P. Bonnet, "uFLIP: Understanding flash io patterns," in *Conf. Innovative Data Systems Research CIDR*, 2009.

[24] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 1, 2012.

[25] D.-W. Chang, W.-C. Lin, and H.-H. Chen, "Fastread: Improving read performance for multilevel-cell flash memory," *IEEE Trans. VLSI Syst.*, vol. 24, no. 9, 2016.

[26] S. Liu and X. Zou, "QLC NAND study and enhanced Gray coding methods for sixteen-level-based program algorithms," *Microelectronics Journal*, vol. 66, 2017.

[27] Y. Cai, Y. Luo, S. Ghose, and O. Mutlu, "Read disturb errors in MLC NAND flash memory: Characterization, mitigation, and recovery," in *Proc. IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, 2015.

[28] P. A. Misra *et al.*, "Managing tail latency in datacenter-scale file systems under production constraints," in *Proc. EuroSys Conf.*, 2019.

[29] H. M. Bashir *et al.*, "Reducing tail latency using duplication: A multi-layered approach," in *Proc. Int. Conf. on Emerging Networking Experiments and Technologies*, Dec. 2019.

[30] X.-Y. Hu, R. Haas, and E. Eleftheriou, "Container marking: Combining data placement, garbage collection and wear levelling for flash," in *Proc. IEEE Int. Symp. on Modeling, Analysis, and Simulation of Comp. and Telecommunication Systems*, 2011.

[31] P. Desnoyers, "Analytic modeling of SSD write performance," in *Proc. Int. Systems and Storage Conference*, 2012.

[32] M. Shafaei, P. Desnoyers, and J. Fitzpatrick, "Write amplification reduction in flash-based SSDs through extent-based temperature identification," in *Proc. USENIX Conf. Hot Topics in Storage and File Sys.*, Jul. 2016.