2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)

Understanding the design trade-offs of hybrid flash controllers

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

***Abstract*—Over the last few years, NAND flash manufacturers have steadily increased the number of bits stored per cell to achieve significant cost reductions. However, the increased density does not come without drawbacks. All key flash performance metrics, including latency and endurance, significantly degrade as bit density increases. Particularly, sustained write throughput is the worst affected as writes are roughly one order of magnitude slower than reads and further require precursory block erases in the background. As a result, many recent flash controllers operate flash blocks both in single-bit (high endurance and performance) and in multi-bit (high density) mode. In theory, such hybrid controllers are a great way of hiding flash technology limitations. A controller can use a small percentage of the flash blocks in single-bit mode as a cache which allows orders of magnitude higher write bandwidth and endurance in environments where the access patterns of the workload are skewed and bursty. In practice, however, many devices fall short of expectations when write performance varies significantly and utilization increases.**

We argue that a principled approach is required to understand the design trade-offs of hybrid NAND flash controllers. To this end, we develop a modeling framework for estimating the performance and endurance of hybrid controllers. The modeling framework computes the internal data movement generated by a hybrid controller by relying on advanced analytical models that offer both accurate and fast predictions. The data flow is then translated into higher-level metrics that quantify upper bounds for the overall performance of an SSD such as write throughput, latency, and device endurance. Using our modeling framework, we compare different controller architectures, identify their strong and weak points, and show that there is room to improve the efficiency of the hybrid controllers used today.

***Index Terms*—hybrid flash controller, modeling, QLC, flash memory**

I. Introduction

A. Background

The main advantages of the NAND flash storage technology, namely low cost, low latency, high throughput, high density, and shock-resistance have made it the storage media of choice for many enterprise and consumer applications. However, flash technology comes with its own drawbacks. Flash cells must be erased in bulk before being written (programmed) and the erase process is performed at a coarser granularity (blocks) than read and program operations (pages). In addition, pages have to be programmed consecutively within a block which essentially prohibits out-of-place updates. Lastly, and more importantly, flash cells have limited endurance: they experience high stress during the program and erase processes, which degrades them gradually and eventually renders them unreliable.

TABLE I
Typical characteristics of 3D NAND Flash devices.

Operation	Flash cell technology			
	SLC	*MLC*	*TLC*	*QLC*
Page read	20 – 25 µs	55 – 110 µs	75 – 170 µs	120 – 200 µs
Page program	50 – 100 µs	0.4 – 1.5 ms	0.8 – 2 ms	2 – 3 ms
Block erase	2 – 5 ms	5 – 10 ms	10 – 15 ms	15 – 20 ms
Endurance	100,000	15,000	3,000 – 5,000	800 – 1,500

Flash-based Solid-State Drives (SSDs) therefore require sophisticated controllers that implement a wide variety of techniques to overcome flash idiosyncrasies. A flash controller introduces an indirection layer, called the Flash Translation Layer (FTL), whose main objective is to translate logical to physical addresses and to provide a simple interface to applications, while hiding the physical media constraints. The FTL greatly improves write performance as it allows out-of-place writes and minimizes internal data relocations thanks to efficient block cleaning algorithms [1]. It further implements wear-leveling through techniques like health binning [2] which moves the endurance limit from the worst blocks to the average of all blocks. Modern flash controllers also employ lower-level techniques to reduce media errors. Examples include, strong error correction codes that help to detect and repair read errors and dynamic threshold voltage shifting that reduces the raw bit error rate [3], [4].

The continuous pressure on cost reduction led to the introduction of multi-bit flash cells. The first generation, Multi-Level Cells (MLC) are capable of storing two bits per cell. Triple-Level Cells (TLC) followed, which store three bits per cell and, more recently, Quad-Level Cells (QLC) have been introduced reaching four bits. Unfortunately, the increase in bit density comes with a reduction in performance and a severe hit in reliability, as the margins between the voltage levels are tighter. In Table I we summarize typical endurance and performance metrics of 3D NAND devices where only the bit density varies. The values presented are summarized from publicly available sources such as press releases (e.g., [5]), presentations (e.g., [6], [7]) or computed from product specifications (e.g., [8]–[10]). While there is roughly one order of magnitude difference between page reads and programs as well as page programs and block erases for the same cell technology, latencies have dramatically increased with the move from SLC to QLC: The program latency has increased

2375-0227/19/$31.00 ©2019 IEEE
DOI 10.1109/MASCOTS.2019.00025

the most, by 30×, while the read latency has increased by 7× and block erase latency by 5×. At the same time, endurance has dropped by a factor of 100×.

The significant degradation in latency and endurance has lead to current QLC-based SSDs having a lower write performance and fewer guaranteed full device writes per day (DWPD) compared to previous SSD generations. Even though program and erase latencies can theoretically be hidden through background destages and erases, they ultimately affect the steady state write throughput of SSDs. Traditional controllers that operate with a single type of flash cell technology are therefore not suitable for QLC flash as they will suffer from poor sustained write performance and low endurance.

The drop in endurance is the most significant challenge for enabling QLC in SSDs as it directly affects the device lifetime. From an SSD controller design perspective, it is therefore essential to address endurance first and I/O performance only afterwards . Hence, in this paper we first focus on endurance upper bounds, and then determine the associated write performance characteristics. To further improve read performance, we refer to multi-layer caching hierarchies that have been extensively studied in the past.

B. Hybrid Flash Controllers

A promising venue to address the limitations of multi-bit flash technologies is to leverage the ability of the most recent flash chip generations to operate in either a high-density multi-bit mode or in a high-performance single-bit mode. *Hybrid controllers with a fixed-size SLC cache* were first introduced in [11], [12] where a small, fixed amount of SLC flash is used as a cache for the main MLC data storage tier. Later, controller improvements led to *adaptive SLC caches* where the amount of SLC blocks is based on the current device utilization (i.e., amount of stored data) [13], [14]. QLC-based SSDs available today use this type of hybrid controller architecture to implement a dynamically-sized SLC cache [8]–[10].

In Fig. 1, we show the specified write throughput numbers of several recent SSDs that utilize hybrid controllers with adaptive SLC caching. The *Bursty* write bandwidth represents the performance of the SSD when writing to the SLC tier, while the *Sustained* bandwidth represents the long-term write performance when the SLC cache becomes full and the internal SLC to QLC data destages and garbage collection start to interfere with user I/O. The specified device endurance of these QLC SSDs is only between 0.1 and 0.3 DWPD, a drop proportional to the PEC endurance presented in Table I. However, real device-level performance and endurance characteristics depend heavily on the workload properties, including workload skew and device utilization [15], as well as the error mitigation activities performed by the controller. It is therefore difficult for users to predict the actual performance and endurance of SSDs in the real world.

C. Summary & Contributions

From a controller developer's perspective, designing a hybrid controller is challenging as the right architectural decisions are not obvious. In this paper, we propose a principled approach to understand the behavior of hybrid SSD controllers. We focus primarily on explaining the write performance and endurance of hybrid SLC/QLC-based controllers, given that the most limiting characteristics of the latest flash generation are the high page program and block erase latencies and the low endurance. We develop a modeling framework that can predict the write performance and endurance of a hybrid SSD as a function of all relevant parameters and design choices: controller architecture, SSD hardware resources, NAND flash properties, type of write workload and capacity utilization. The framework leverages internally advanced analytical models to estimate the dataflow inside a hybrid controller and then applies optimization techniques to automatically tune the parameters of the controller to identify the best operating regime.

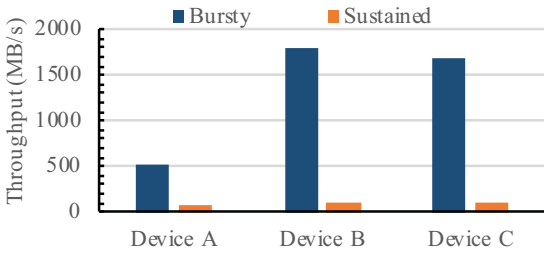


Fig. 1. Write performance for existing hybrid SLC-QLC SSDs.

We believe that the modeling approach presented in this paper can help developers to better understand the trade-offs in hybrid controller design options and strike a balance between maximizing performance and endurance while keeping the implementation complexity reasonable. However, we must bear in mind that our modeling approach can only deliver sound upper bounds for an ideal hybrid SSD controller based on technology characteristics and workload properties from which we cannot deduce a functional controller implementation. We see the modeling work as an educational tool to study the intrinsic trade-offs involved when designing hybrid SSD controllers. A real hardware controller environment imposes strict limitations on the available resources in terms of CPU, memory, power consumption, ASIC or FPGA real-estate, which cannot be fully addressed by modeling and are out of the scope of this paper. Therefore, this modeling framework is not intended to reflect performance aspects that depend on implementation-specific controller details. Also, the modeling algorithms presented are not meant to be directly implemented inside an FTL.

More specifically, our contributions are as follows:

- We introduce novel analytical models that are able to predict internal data movement inside an SSD with a hybrid controller architecture. We expect these analytical models to be useful in many other settings, including the design of other types of SSDs (see Section V), to predict cache hit rates in other settings, or to understand write amplification overhead in log-structured storage systems.
- Starting from the analytical models, we develop a modeling framework that is able to predict high-level met-

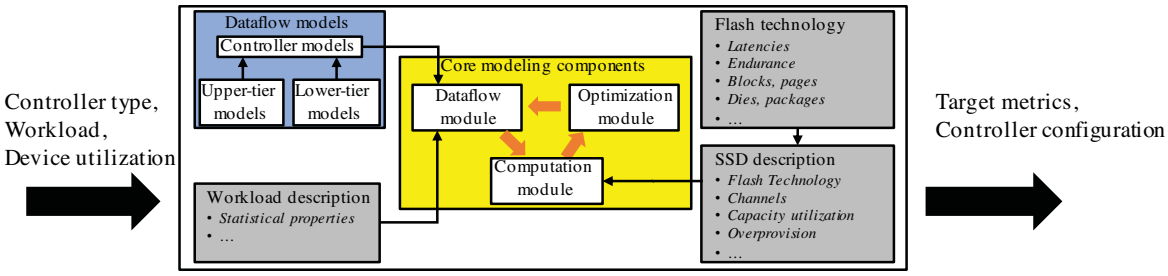


Fig. 2. The high-level architecture of the modeling framework including the core components (yellow), basic dataflow models (blue), and configuration modules (gray).

rics such as write bandwidth, write latency, or device endurance. These predicted upper bounds are helpful when exploring and understanding the strengths and weaknesses of hybrid controllers.

- Using the modeling framework, we show that there is a large room to improve the efficiency of hybrid controllers used today. The key is to be able to optimally size the single and multi-bit tiers dynamically based on workload properties and leverage write heat information to improve data placement and destaging. In particular, we quantify the space for improving a QLC-based SSD, as QLC is the most limiting Flash technology available today.

D. How to read this paper

We recognize that thoroughly understanding the dataflow models is time consuming. Although the dataflow models are a significant contribution of the paper, the manner in which each model is derived is not critical for understanding the modeling approach or the results presented. The output of any modeling algorithm can be reproduced through simulation. As such, the modeling algorithms should be viewed as accelerators rather than enablers. Therefore, we suggest the reader to focus first on understanding the high-level architecture of the framework, how a concrete controller design is modeled, and how the best operating point is identified through parameter optimization.

II. Modeling framework

In this section we provide an overview of our modeling framework. We start with the goals and assumptions and then present the high-level architecture and internal components of the framework. We then describe how the basic models are leveraged to compute the controller-wide dataflow which is ultimately transformed into the target metrics of interest (i.e., upper bounds on performance or endurance). At the end, we discuss further aspects that address the approaches we took to validate our models.

A. Overview of the architecture

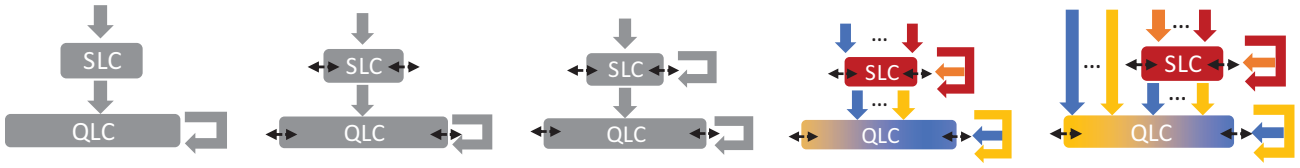
The goal of the modeling framework is to predict certain target metrics, including endurance and write performance, of a controller architecture by taking into account all relevant parameters encountered in an SSD controller. As our modeling framework describes the upper bounds of the target metrics in steady-state, the amount of stored data does not change over time and the statistical properties of the workload remain constant. However, using results from different steady-state configurations can be used to determine optimal configuration parameters in a real controller such that its configuration can be adapted dynamically at run-time. Conceptually, modeling works by answering two interconnected questions: 1) what is the best configuration for a given controller architecture (i.e., by determining the optimal size of the single and multi-bit tiers and the amount of valid data stored therein); and 2) how many internal flash operations (e.g., page reads, programs, and block erases) are performed on average for each user write.

Even though our modeling framework can be used for any hybrid controller configuration where the tiers could operate in any combination of single and multi-bit modes, the remainder of this paper focuses solely on SLC combined with QLC NAND flash, because the typical characteristics of QLC are the most limiting ones.

Fig. 2 outlines the high-level architecture of the modeling framework. Its inputs are the type of the controller architecture, a workload description, and the current device utilization. The outputs being generated include the actual value for a chosen target metric and the optimal values for the internal controller configuration parameters. Internally, the framework is composed of three core modeling components (yellow), a library describing the data flow models in a tier and between them (blue), and a set of configuration modules (grey).

B. Core modeling components

1) Dataflow module: The internal data movement between the tiers is computed by the dataflow module based on the given controller model, the local upper and lower tier models, and the workload description. For a chosen controller architecture, the appropriate controller model is selected. Internally, the controller model leverages the appropriate algorithms that model the upper and lower tiers. A wide range of controller models are illustrated in Fig. 3 and discussed in more detail below. Here, the upper tier operates in SLC mode and the lower tier operates in QLC mode. The dataflow module can be thought of as logically connecting the dataflow models by adjusting their inputs to describe how data moves across the tiers. For example, most hybrid controller architectures write new data first to SLC and later destage it to QLC. Therefore,



(a) Fixed-sized SLC cache.

(b) Adaptive SLC caching and background destaging.

(c) Adaptive SLC caching with background destaging and SLC-to-SLC relocation.

(d) Adaptive SLC caching with heat-aware SLC eviction and relocation.

(e) Adaptive SLC caching with heat-aware data placement, eviction, and relocation.

Fig. 3. Overview of hybrid controller architectures.

the output dataflow from the SLC tier (described as a data-eviction frequency, the amount of data evicted, and the write heat distribution of the pages) represents the input dataflow for the QLC tier.

2) Computation module: Using the output of the dataflow module, the computation module converts the internal dataflow into the target metrics of interest. The computation of each target metric is self-contained and can reflect either the SSD write performance (e.g., average flash chip busy time, write throughput, improvement over a QLC-only controller, etc.) or SSD endurance (amount of data written, DWPD, improvement over a QLC-only controller, etc.).

3) Optimization module: The identification of the optimal controller configuration is performed in the optimization module. The optimal controller configuration is obtained from the set of all computed target metrics for the various controller parameters. Initially, the modeling process starts by assigning predefined values to the tunable parameters. For example, a controller could set 10% of the free physical capacity as an SLC cache. The optimization module then repeatedly adjusts the input parameters of the dataflow module until the best operating point is found. Continuing the example, the optimization module would increase or decrease the SLC cache size until the device endurance is maximized.

The modeling framework relies on numeric differentiation using finite difference approximations when searching for the configuration that maximizes the objective function (i.e., maximum write throughput or endurance). For example, when optimizing the size of the SLC tier to achieve the best endurance, the derivative w.r.t. the SLC allocation is computed as follows:

$$\frac{\partial E(s)}{\partial s} = \frac{E(s+\epsilon) - E(s)}{\epsilon}$$

where E is the estimated controller endurance, s is the SLC size expressed as the fraction of flash blocks used in SLC mode, and $\epsilon \ll 1$ is a configurable fixed size increment.

C. Dataflow models

The dataflow models consist of tier and controller models:

1) Tier models: The mathematical models of the tiers can be grouped into two classes, one class for the upper tier and another class for the lower tier of the hybrid controller. The conceptual difference between the two classes is that data can be evicted from an upper tier, while it cannot be evicted from a lower tier. Consequently, a lower tier always has to relocate data internally, while an upper tier may dynamically decide to either destage data or recirculate it internally. Each tier model has a specific set of input parameters and output values. The output values are derived through analytical equations or numerical algorithms from the input parameters.

2) Controller models: The controller models describe the data flow between the tiers as well as how incoming user data is assigned to a tier. Each controller model is responsible for computing the system wide dataflow by calling the appropriate elementary upper and lower tier models. Fig. 3 depicts different hybrid controller architectures we have modeled sorted in an increasing order of implementation complexity.

D. Configuration modules

The configuration modules consist of the workload description, the characteristics of the flash technology, and controller specific parameters. Knowing the controller type, the test workload, and the device utilization, we derive the workload description which summarizes the statistical properties of the write-hot and write-cold data sets and their sizes that will be used in the dataflow module. The characteristic parameters of the flash cell technology used and the internal SSD properties (such as the number of flash channels and flash chips, over-provision, etc.), provide the required configuration parameters for the computation module.

We have general-purpose tier models for arbitrarily skewed write workloads as well as specific tier models for uniform random writes. Although a general-purpose model can also model random writes, we chose this approach for a number of reasons. First, models for random writes are easier to understand than others for arbitrary skewed write distributions. Understanding how to model random writes is the necessary first step towards understanding the generalized models. Second, general-purpose models can be used to verify the correctness of the uniform random models. Third, from a practical perspective, the uniform random models are significantly faster to compute as they can be described using closed-form analytical formulas, while the models for skewed writes are numerical algorithms that involve performing operations on large vectors.

E. Further details

We implemented our modeling framework in Matlab. To speed up computation, the modeling framework automatically caches results at two levels as files. The first level of caching is at the individual dataflow models that are being computed. The second level of caching is at the controller design level and

TABLE II
INPUT PARAMETERS FOR THE MODELING FRAMEWORK.

Notation	Description
N	Number of logical flash pages in the modeled dataset taking into account the device utilization
C	The capacity of a tier measured in the number of physical flash pages
r	Ratio between capacities in multi-bit vs. single-bit mode (for an SLC/QLC controller $r = 4$)
α	Over-provisioning at maximum physical capacity in QLC mode and 100% device utilization
α_{slc}, α_{qlc}	Instantaneous over-provisioning computed as the fraction of the physical spare capacity divided by the used capacity in a tier
s_{d_t}, s_{d_a}, s_{d_c}	Total, write-active, and write-cold user data set sizes relative to maximum physical capacity in QLC mode ($s_{d\_t} = s_{d\_a} + s_{d\_c}$)
$\vec{F}_w$	Write heat (the probability distribution of writing each page in the dataset according to the modeled workload)

includes the whole modeling framework such that the output after optimization is cached. Further, using simple scripts, we can accelerate the modeling in the computation module by executing different workloads, capacity utilization points, or controller types in parallel. In a multi-server execution environment, we rely on a shared file-system (e.g., NFS) to consolidate the cache files in the same directory structure. This strategy allows the modeling processes to reuse intermediary results whenever possible without any synchronization concerns.

In order to validate the correctness of the modeling framework, we use several testing strategies that help to identify the source of most implementation and logical errors:

- We validate all closed-form analytical formulas by comparing their output with numerical solutions to the same set of initial data flow equations from which the formulas were derived. This step ensures we catch mathematical derivation errors.
- To validate the tier models for random write workloads, we used a separate software simulation environment that mimics data movement in an SSD. This step identifies dataflow modeling mistakes that can occur in the first stage of developing a tier model when we initially focus on random write workloads.
- To validate the tier models for arbitrarily skewed write workloads, we first compare the output of a generic tier model for a random write workload with the output of the corresponding random write model. We then use the simulation environment to validate that the tier can accurately model arbitrarily generated write distributions. This step ensures we do not introduce errors when generalizing a tier model to support arbitrarily skewed writes.

III. ANALYTICAL MODELS AND TARGET METRICS

This section starts by introducing the modeling parameters used, followed by presenting the analytical algorithms used by the dataflow models. Finally, we describe how to compute the target metrics of interest (write performance or endurance).

A. Modeling parameters

There are several classes of parameters we use for modeling, namely *data sizes*, *over-provisioning ratios*, *frequencies* and *frequency distributions*. All parameters classes represent relative quantities and have values in the $[0, 1]$ interval. Using relative values enables many inconsequential parameters to be abstracted in the modeling process. As stated, our goal is to compute the relative rates of the data movement and only later transform these rates into actual target metrics based on the specifications of an SSD.

Data sizes (denoted by s) are expressed relative to the physical SSD capacity rather than in absolute units. For example, a total user data size of 0.5 means that the user data occupies 50% of the physical SSD capacity when all blocks are set in QLC mode.

Over-provisioning (denoted by α) represents the relative ratio of the spare capacity to the data size ($\alpha = \frac{physical - logical}{logical}$). We distinguish between the maximum theoretical over-provisioning (α) when the logical capacity of the device is filled with user data and all blocks are in QLC mode versus the tier-specific instantaneous over-provisioning (α_{slc}, α_{qlc}) that takes into consideration the current capacity of the tier and the size of the data stored within.

A *frequency* (denoted by f) represents the rate, frequency, or probability of a page movement. It is computed either relative to the global user writes or to the local tier writes. For example, the rate at which pages are evicted from SLC, f_{ev}, represents the average number of evictions per SLC write. If all writes are first stored in SLC, then f_{ev} also represents the global ratio of user writes to SLC evictions.

Distributions (denoted with $\vec{F}$) are large vectors where each element describes the frequency of a page movement occurring (e.g., a particular page being written, evicted or relocated). The sum of all probabilities in a distribution is equal to or lower than 1. Distributions are used only when modeling skewed workloads. All vector formulas presented in this paper use element-wise arithmetic operations. We use the symbols $\odot$ for element-wise multiplication and $\oslash$ for element-wise division.

1) Input configuration parameters: Table II shows the notations of the input parameters given to the core modeling components. The over-provisioning α, which denotes the fraction of additional spare capacity of the total available physical capacity, is key to control the garbage collection overhead [16]. In our modeling framework, we extend this notion and define the instantaneous over-provisioning of a tier α_{slc} and α_{qlc} as the fraction of the spare capacity within a tier normalized to the current utilization. For controller architectures using write heat information, we further define the normalized size of the write-active and write-cold data sets (s_{d_a} and s_{d_t}) as well as the write heat distribution of all pages $\vec{F}_w$. These are directly derived from the workload skew and the utilization.

TABLE III
OUTPUT VALUES OF THE DATAFLOW MODULE.

Notation	Description
s_{slc}, s_{qlc}	Fraction of the physical flash blocks assigned to the SLC or QLC tiers.
s_{d_slc}, s_{d_qlc}	Fraction of the user data stored in the SLC or QLC tiers.
f_{w_slc}, f_{w_qlc}	Frequency of writing a page to the SLC or QLC tiers. Expressed relative to the user write frequency.
f_{gc_slc}, f_{gc_qlc}	Frequency of relocating a page (SLC-to-SLC or QLC-to-QLC) normalized to the frequency of writes to the tier.
f_{inv_slc}, f_{inv_qlc}	Frequency of new writes invalidating (updating) a page in the SLC or QLC tiers (i.e., the hit rate of the tier).
$\vec{F}_{w\_slc}$, $\vec{F}_{w\_qlc}$	Frequency or probability distribution of the page writes to the SLC or QLC tiers..
$\vec{F}_{gc\_slc}$, $\vec{F}_{gc\_qlc}$	Frequency or probability distribution of page relocations inside the SLC or QLC tier. Expressed relative to the writes to the tier.

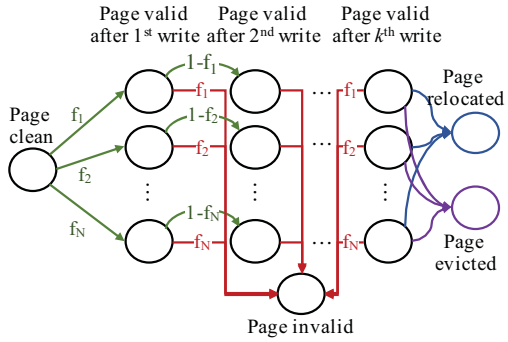


Fig. 4. A Markov chain model for computing the probabilities with which pages are evicted, relocated or invalidated.

In particular, the write heat distribution $\vec{F}_w$ contains the update frequencies for each data page in the dataset sorted in decreasing order. We chose this approach to be able to model write workloads irrespective of how they are expressed. When using a synthetic write distribution (e.g., a Zipfian distribution), we simply use the probability mass function to compute the individual page update probabilities. Otherwise, when using a given I/O trace, we scan the trace and compute the frequencies of the page writes. If a vector becomes too large, the write distribution can be optionally compressed. The compression process is straight-forward as it essentially ensures that the CDF lines for the original and the compressed vectors are overlapping once the axes are normalized to the $[0-1]$ range. By default, our workload distributions have 10^8 elements, which translates to a ∼1.5TB user data set assuming the standard 16kB QLC flash page size.

2) Dataflow metrics: Table III introduces the parameters that characterize the dataflow, i.e., how a controller moves data inside an SSD. Depending on the workload and the controller design, not all of these parameters are required at all times. For example, probability distributions are not necessary for random workloads.

B. Model derivation techniques

The derivation of the dataflow models relies on the following key observations.

Markov chain for page states. First, we compute the probabilities that a given page is in either of the valid, invalid, evicted or relocated states. Fig. 4 shows how these probabilities evolve as a function of the number of writes

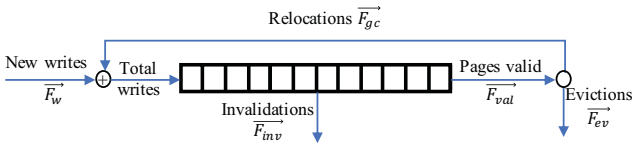


Fig. 5. Dataflow of a storage tier.

experienced between the time a page is written and the time the page is evicted or relocated. The state (the round circle) at row i and column j represents the scenario where the page with id i is valid after j writes. The probability of reaching a state is the sum of probabilities of all possible paths to that state, while the probability of a path is the multiplication of all probabilities along the path. For example, the probability of page i being written and still valid after k writes is $f_{val} = f_i \cdot (1-f_i)^k$. The probability distribution of all pages being valid after k writes is $\vec{F}_{val} = \vec{F}_w \odot (\vec{1} - \vec{F}_w)^k$ (written using vector notations).

In the particular case of random writes, given that all update probabilities are equal, the valid page states have equal probability and can be collapsed vertically. The probability of a page having valid data after k writes then becomes $f_{val} = (1 - \frac{1}{N})^k$, where N is the number of pages in the dataset actively updated. For deriving closed form formulas, we further apply Euler's limit to obtain: $f_{val} \simeq e^{-\frac{k}{N}}$. A similar modeling technique is also used in [17].

Dataflow balance equations. We make the observation that the inflow and outflow from a tier, as shown in Fig. 5, must be equal: the incoming number of page writes must be equal in steady-state to the number of pages being evicted or invalidated:

$$\vec{F}_w + \vec{F}_{gc} = \vec{F}_{ev} + \vec{F}_{val} = \vec{F}_{ev} + \vec{F}_{inv} + \vec{F}_{gc}$$

For random writes, these equations can be further transformed and simplified to reflect the average page flow rates by applying the sum operator $\sum$. Assuming that the incoming write rate is defined to be 1 ($\sum \vec{F}_w = 1$), we have:

$$1 + f_{gc} = f_{inv} + f_{val} \quad (1)$$

$$f_{val} = f_{ev} + f_{gc} \quad (2)$$

The invalidation rate, f_{inv}, is equal to the fraction of the active data present in the tier:

$$f_{inv} = s_{d\_tier} \quad (3)$$

Next, we observe that the physical pages in the tier, C, are consumed (written) by either new user writes or by relocation writes. As relocation writes do not produce invalidations, the number of physical pages in the tier that can store new incoming data is $C(1-f_{gc})$. The number of physical pages in a tier can be further expressed in terms of the relative tier size to data size: $C = N\frac{s_{tier}}{s_{d\_tier}}$. Therefore, the rate at which valid pages exit the tier is:

$$f_{val} = (1-\frac{1}{N})^{N\frac{s_{tier}}{s_{d\_tier}}(1-f_{gc})} \approx e^{-\frac{s_{tier}}{s_{d\_tier}}(1-f_{gc})} \quad (4)$$

Size and distribution of valid user data. We observe that in order for a page to be invalidated (overwritten) by a new user write two independent conditions must hold: 1) the page must be already present in the tier and 2) the page must be referenced by the new write. We can express this relationship in terms of probability distributions in a vector form as:

$$\vec{F}_{inv} = \vec{F}_{d_a} \odot \vec{F}_w \implies \vec{F}_{d_a} = \vec{F}_{inv} \oslash \vec{F}_w$$

where $\vec{F}_{d_a}$ is the vector of the probabilities of each page being present in the tier, $\vec{F}_{inv}$ is the probability distribution (i.e., the rates) with which pages are invalidated, and $\vec{F}_w$ is the write probability distribution of the workload. The total active data size present in the tier is: $s_{d\_tier} = \frac{\sum \vec{F}_{d_a}}{N}$.

C. Detailed analytical dataflow models

In this Section we describe the analytical dataflow models for the upper and lower tiers. For each tier we use separate models for uniform random and skewed workloads.

1) Upper-tier model with LRW eviction for random writes: We start by modeling the simplest controller architecture where data is first written to the upper tier (SLC) and then evicted (de-staged) to the lower tier (QLC), as shown in Figures 3(a) and 3(b). We consider a random write workload and an eviction policy that destages the least recently written (LRW) blocks first. The goal of this upper tier architecture is to absorb spikes in the write I/O workload and ensure low write latency and high instantaneous throughput. During idle periods, the data in the upper tier can be destaged to QLC when needed. From a development perspective, this design is attractive due to the low implementation complexity. Most hybrid QLC controllers on the market appear to use this architecture [8]–[10]. Algorithm 1 represents the analytical modeling function for the upper tier.

Algorithm 1 Upper-tier with LRW eviction - Random Writes

LRW_cache_RW(s_{d_a}, s_{slc}) $\rightarrow$ [s_{d_slc}, f_{inv_slc}, f_{ev}]

1: $rel\_sz = \frac{s_{slc}}{s_{d\_a}}$ (relative upper tier size to data size)
2: $f_{ev} = e^{-rel\_sz}$
3: $f_{inv\_slc} = s_{d\_slc} = 1 - f_{ev}$

The formulas presented in the algorithm are a straightforward application of Equations 1-4. Note that this upper tier model does not have any internal relocations ($f_{gc} = 0$) and that pages are evicted from the cache after exactly $C = \frac{N \cdot s_{slc}}{s_{d\_a}}$ user writes.

2) Upper-tier model with LRW eviction for skewed writes: For arbitrarily skewed writes, the modeling algorithm is conceptually similar and follows the techniques presented in Section III-B. The main difference is that we operate with frequency distributions and we must compute the individual frequency for each page of the dataset (note the usage of vector operations). Algorithm 2 shows the upper-tier modeling function for skewed write workloads.

Algorithm 2 Upper tier with LRW eviction - Skewed Writes

LRW_cache(s_{d_a}, s_{slc}, $\vec{F}_w$) $\rightarrow$ [s_{d_slc}, f_{inv_slc}, f_{ev}, $\vec{F}_{ev}$]

1: $N = |\vec{F}_w|$ (number of logical pages)
2: $C = N \cdot \frac{s_{d\_a}}{s_{slc}}$ (number of physical pages in the upper tier)
3: $\vec{F}_{ev} = \vec{F}_w \odot (\vec{1} - \vec{F}_w)^C$ (page eviction distribution)
4: $\vec{F}_{inv} = \vec{F}_w - \vec{F}_{ev}$ (page invalidation distribution)
5: $f_{ev} = \sum \vec{F}_{ev}$ (average eviction frequency)
6: $f_{inv\_slc} = 1 - f_{ev}$ (average invalidation frequency)
7: $s_{d\_slc} = \frac{\sum \vec{F}_{d_a}}{N} = \frac{\sum(\vec{F}_{inv} \oslash \vec{F}_w)}{N}$ (average active data size)

3) Upper-tier model for occupancy-aware eviction: The previous upper-tier model with LRW destage can be improved by introducing the ability to perform SLC-to-SLC relocations as is illustrated in Fig. 3(c). An occupancy-aware upper tier works by cleaning blocks using a LRW policy and either relocating data to SLC or destaging data to QLC based on how the current cache utilization compares to a tunable threshold. The design improves on the fixed LRW eviction policy especially in scenarios where the hot write set fits in the cache as it avoids unnecessary data destages. The utilization threshold for the data destage is tuned by the optimization module. The dataflow model for the upper tier with partial LRW eviction using random writes is described in Algorithm 3.

Algorithm 3 Upper tier with partial LRW eviction - random writes

LRW_cache_RW(s_{slc}, s_{d_a}, u) $\rightarrow$ [s_{d_slc}, f_{ev}, f_{gc_slc}]

▷ input u is the target utilization of the upper-tier ($u = \frac{s_{d\_slc}}{s_{slc}}$)
1: $f_{inv\_slc} = s_{d\_slc} = s_{slc} \cdot u$
2: $f_{ev} = 1 - s_{d\_slc}$
3: $temp = -\frac{1}{s_{slc} \cdot u} e^{-\frac{s_{slc}}{s_{d\_a} \cdot s_{d\_a} \cdot u}}$ (simplifying notation)
4: $f_{gc\_slc} = -\frac{s_{slc} \cdot W(temp)}{s_{d\_a}} - \frac{1 - s_{d\_slc}}{s_{d\_slc}}$

The modeling equations are derived from the dataflow balance equations specialized for random writes. We first note that the size of the user data stored in the tier and the invalidation rate are determined by the target utilization (Algorithm 3, line 1). We then substitute Equations 3 and 4 in Equation 1 to obtain:

$$\underbrace{1 + f_{gc}}_{tier\ writes} = \underbrace{s_{d\_slc}}_{f_{inv\_slc}} + \underbrace{e^{-\frac{s_{tier}}{s_{d\_tier}}(1-f_{gc})}}_{f_{val}}$$

This equation accepts a solution in terms of the Lambert-W function [18] that is shown on lines 3 and 4 in Algorithm 3.

4) Upper-tier model for heat-aware eviction: An ideal controller tracks write heat and leverages this information to prioritize data destage and reduce cleaning overhead through heat segregation. The destage process is optimal when evicting the coldest pages rather than the least recently written pages. Having the hottest data in SLC maximizes the rate of invalidations in SLC and minimizes the rate of evictions to QLC.

Algorithm 4 Upper tier with partial LRW eviction - skewed writes

LRW_cache_SW(s_{slc}, $\vec{F}_w$, u) $\rightarrow$ [s_{d_slc}, f_{ev}, f_{gc_slc}]

1: $s_{d\_slc} = s_{d\_a} \cdot u$
2: $k = |\vec{F}_w| \cdot s_{slc} \cdot u$ (top k^{th} pages are upper-tier resident)
3: $\alpha_{slc} = \frac{1-u}{u}$ (over-provisioning for the upper-tier resident pages)
▷ Use Alg. 7 to model write amplification for the upper-tier resident pages
4: f_{gc_slc} = WA_SW($\vec{F}_w[0:k]$, α_{slc})
5: $\vec{F}_{ev} = \vec{F}_w[k:]$; $f_{ev} = \sum \vec{F}_{ev}$

From a modeling perspective, the pages of the dataset can be separated into two sets based on the write frequency: a hot set that resides in SLC and a cold set that is initially written to SLC and then destaged in the background to QLC (the SLC thus acts as a temporary destage buffer to reduce latency). Modeling the SLC-to-SLC relocations involves computing the cleaning overhead of the write workload corresponding to the pages in the hot set only. We show in Section III-C7 how the cleaning overhead is computed for arbitrary workloads. The size of the hot set denotes the number of pages resident in the SLC tier and is computed based on the target utilization of the SLC tier, which is a tunable controller parameter and is subject to optimization.

5) Lower-tier model for random writes: Previous work has shown that the garbage collection policy for a random write workload has a negligible dependency on the cleaning overhead [17], [19]. We use the closed-form analytical formula from [17] that computes the relocation frequency (f_{gc}) in terms of the Lambert-W [18] function. The only difference here is that we need to compute first the instantaneous over-provisioning, α_{qlc}, as a function of the amount of data stored in the tier and the currently available storage capacity in the tier: $\alpha_{qlc} = \frac{s_{qlc} - s_{d\_qlc}}{s_{d\_qlc}}$. The size of the data stored in the tier is computed based on the modeling output of the upper tier.

Algorithm 5 Lower-tier relocations - random writes

WA_RW(α_{qlc}) $\rightarrow$ [f_{gc}, wa]

1: $f_{gc} = -\frac{W(-(1+\alpha_{qlc})e^{-(1+\alpha_{qlc})})}{1+\alpha_{qlc}}$
2: $wa = \frac{1}{1-f_{gc}}$

6) Lower-tier model for skewed workloads using LRW cleaning: Algorithm 6 describes the exact modeling steps. For a LRW cleaning policy, the storage capacity can be considered a circular log-structure where updates are appended to the tail and cleaning happens at the log head where the oldest written block is located. When low on space, valid pages are relocated from the head to the tail of the log structure, followed by advancing both head and tail pointers.

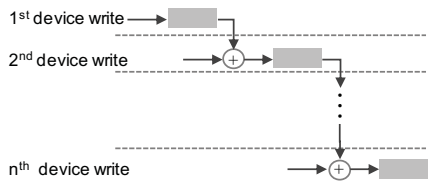


Fig. 6. Unrolling a LRW log-structure into a hierarchy of LRW caches.

Algorithm 6 Lower-tier relocations with LRW cleaning - skewed writes

WA_LRW_SW($\vec{F}_w$, α) $\rightarrow$ [f_{gc}, wa]

1: $N = |\vec{F}_w|$ (Number of logical pages)
2: $C = N(1+\alpha)$ (Number of physical pages)
3: $f_{gc} = 0$; $\vec{F}_{gc} = 0$ (Initialization)
4: **repeat**
5: $f_{gc\_old} = f_{gc}$; $\vec{F}_{gc\_old} = \vec{F}_{gc}$
▷ Pages are either written or relocated with distribution:
6: $\vec{F}'_w = \vec{F}_w(1 - f_{gc\_old}) + \vec{F}_{gc\_old} \cdot f_{gc\_old}$
7: $C' = C(1 - f_{gc})$ (Physical pages written with new data)
▷ Once written, pages survive until GC with probability:
8: $\vec{F}_{gc} = (\vec{1} - \vec{F}_w)^{C'}$
▷ The overall distribution of pages relocated in epoch i^{th}:
9: $\vec{F}_{gc} = \vec{F}'_w \odot \vec{F}_{gc}$
10: $f_{gc} = \sum \vec{F}_{gc}$
11: **until** $|f_{gc} - f_{gc\_old}| < tol$

Note that writes at the log tail are caused either by: a) page relocations or b) by user updates. Assume page i is present in the log. At each new user write, the probability that the page escapes invalidation and remains valid is $1 - \vec{F}_w(i)$. The cleaning process does not produce invalidations as it only relocates internally existing valid pages. After k updates, the probability the page is still valid becomes $(1 - \vec{F}_w(i))^k$.

The main difficulty is determining how many user writes can be accommodated between the time a page enters the circular log structure (by being written at the log tail) and the time the page is GC-ed (when it reaches the log head). Initially, when the log has no data (the device did not see any writes), the number of user writes accommodated is N (Algorithm 6, line 2). However, as pages start to be relocated, less and less writes can be accommodated before a page is relocated. Accommodating a lower number of user writes increases in turn the GC rate. This leads to a circular dependency between GC rate and the actual capacity of the log structure that is reflected in lines 6-7.

We solve the circular dependency problem by logically unrolling the log structure over time by considering an infinite set of LRW caches with the same capacity that are connected. Fig. 6 shows how the unrolling takes place. Our algorithm repeatedly computes the input (line 8) and output (line 9) page distributions for the log-structures until they convergence.

Algorithm 7 Relocations for Heat Segregation - skewed writes

WA_SW($\vec{F}_{inv}$, α_{qlc}) $\rightarrow$ [f_{gc}, wa]

▷ Constants

H (number of write heat streams to model)

$tol = 10^{-3}$ (maximum wa error tolerated)

$N = size(\vec{F}_w)$ (total number of logical pages)

$M = \alpha_{qlc} \cdot N$ (number of physical pages)

▷ Variables

$pp(1:H) = lp(1:H) = 0$ (the number of physical/logical pages assigned to each stream)

$der(1:H)$ (the wa derivative for each stream)

$tprob = \frac{\sum \vec{F}_{inv}}{H}$ (streams have an equal write frequency)

▷ Initialization

$sort(\vec{F}_w)$ (if needed)

for ($i = j = 0$; $i < H$; i++) **do**

  while $prob(i) < tprob$ **do**

    $prob(i)$ += $\vec{F}_w(j++)$

    $log\_pages(i)$++;

  $pp(i)$ = $\frac{M \cdot N}{lp(i)}$

  $der(i)$ = derivative(WA_LRW_SW($\vec{F}_w(i)$, $\alpha_{qlc} = \frac{pp(i)}{lp(i)}$))

wa = compute_WA()

▷ Optimize space allocation

repeat

  i = rank(min(der)) ; j = rank(max(der))

  $step$ = optimize_space_allocation_change()

  $pp(i)$ += $step$; $pp(j)$ -= $step$

  $der(i)$ = derivative(WA_LRW_SW($\vec{F}_w(i)$, $\alpha_{qlc} = \frac{pp(i)}{lp(i)}$))

  $der(j)$ = derivative(WA_LRW_SW($\vec{F}_w(i)$, $\alpha_{qlc} = \frac{pp(j)}{lp(j)}$))

  wa_{old} = wa ; wa = compute_WA()

until $|wa - wa_{old}| < tol$

7) Lower-tier model for heat-aware data segregation: It is well understood that heat segregation is key to reducing write amplification [1], [17]. Heat segregation groups data into streams with similar update frequencies for data placement and has been implemented in some controllers [3]. We leverage the algorithm to compute the write amplification for each stream from [17] and adapt it to the hybrid controller architecture illustrated in Figures 3(d) and 3(e).

To do so, our new algorithm divides the written pages in several datasets based on heat, partitions the spare storage capacity across the datasets, and finally computes the relocation overhead for each dataset. The space allocation is then iteratively optimized until we determine the minimal cleaning overhead. Algorithm 7 describes the lower-tier model for heat-aware data segregation.

D. Computing target metrics

Write performance and write endurance are the main target metrics being evaluated by the computation module. *Write performance* is quantified by first computing the total expected flash chip busy time to service a user write by taking into account all the associated internal write operations. The maximum write throughput can then be estimated from how many I/Os the SSD can service in parallel. The number of maximum parallel I/Os supported is a function of the number of NAND flash devices, dies, packages, and channels. The expected busy time needed to handle a single flash page is computed in Algorithm 8.

Algorithm 8 Computation of the average write latency

compute_write_lat(f_{w_slc}, f_{w_qlc}, f_{gc_slc}, f_{gc_qlc}) $\rightarrow$ [t]

▷ r, p, e = page read, page program, and block erase latencies

▷ n = pages per flash block in SLC/QLC mode

$wa_{slc} = \frac{1}{1-f_{gc\_slc}}$; $wa_{qlc} = \frac{1}{1-f_{gc\_qlc}}$

$t = f_{w\_slc}(p_{slc} + (wa_{slc} - 1)(r_{slc} + p_{slc}) + \frac{e_{slc} \cdot wa_{slc}}{n_{slc}})$
$+ f_{w\_qlc}(p_{qlc} + (wa_{qlc} - 1)(r_{qlc} + p_{qlc}) + \frac{e_{qlc} \cdot wa_{qlc}}{n_{qlc}})$

Device endurance is determined by first computing the number of useful PECs per block expected to store new data, excluding the number of PECs wasted due to internal data movement. The PECs wasted due to internal data movement can be conceptually viewed as a *wear amplification* that depends on the write amplification of each tier, the PECs of the two modes, and the capacity ratio between the SLC and QLC modes. Although flash blocks are not expected to have a uniform endurance (PEC count), established wear leveling algorithms [2] are able to distribute wear optimally and ensure that all blocks have a similar lifetime. We further assume that a SLC/QLC wear leveling mechanism is implemented that can swap blocks between the two pools if required by wear leveling. Given that each block has a total PEC budget that can be used in both the SLC and QLC modes, we can convert the SLC wear to QLC wear and vice-versa. The conversion can linear (e.g., ten SLC PECs equal 1 QLC PEC) or can follow an arbitrary function determined by the particular NAND flash technology used. The conversion function should ideally be provided by the chip vendor although it can also be determined through characterization. For simplicity, we assume a linear conversion between SLC and QLC wear as described in Algorithm 9.

Algorithm 9 QLC PEC equivalent endurance

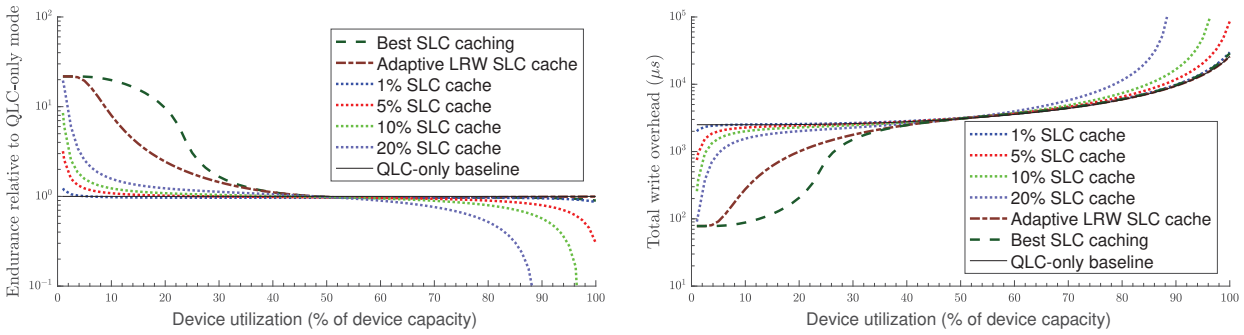
compute_endurance(f_{w_slc}, f_{w_qlc}, f_{gc_slc}, f_{gc_qlc}) $\rightarrow$ [pec]

$wa_{slc} = \frac{f_{w\_slc}}{1-f_{gc\_slc}}$; $wa_{qlc} = \frac{f_{w\_qlc}}{1-f_{gc\_qlc}}$

$pec = \frac{pec_{qlc}}{wa_{qlc} + r \cdot wa_{slc} \frac{pec_{qlc}}{pec_{slc}}}$

IV. Modeling results

In this section, we present some of the applications of our modeling framework. We focus on understanding three critical design options, namely, the importance of adapting the SLC tier and SLC occupancy to the current device utilization, the benefits of using a heat-aware data placement and destage, and the advantages of bypassing the SLC tier when beneficial.



(a) Write endurance.

(b) Average write overhead.

Fig. 7. Relative performance of hybrid controller architectures versus a QLC-only controller for uniform random writes.

Methodology. We face two main challenges when presenting modeling results. First, the absolute endurance and write throughput of a hybrid controller depend on the available SSD hardware resources (i.e., number of flash devices, channels, interface, etc.). We abstract resource-specific aspects by presenting the endurance of hybrid controllers relative to that of a QLC-only controller with the same hardware resources. For write performance, we present the expected chip busy time, i.e., the cumulative latency of all flash operations in both modes associated with a single user write. The chip busy time can then be converted to write throughput based on the actual SSD hardware resources which determine the maximum number of parallel I/O operations supported. Second, the characteristics of a flash technology are generally considered proprietary information. We resort to using the average values of publicly known 3D NAND flash characteristics presented in Table I, and make sure that the target metrics of the modeling results are robust to changes of the flash technology metrics.

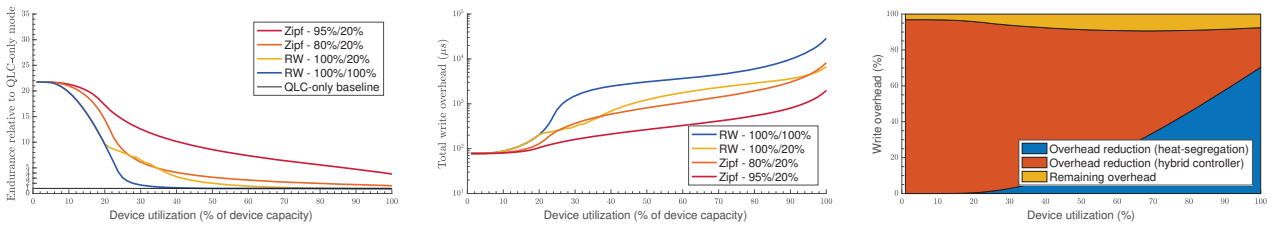
Workloads. It has been shown that real-world workloads tend to be skewed where the majority of writes target a small percentage of the user data [3], [20], [21]. To cover a wide range of workload skews we utilize four types of workloads: a) a workload with no skew composed of uniform random writes to the whole dataset (denoted RW 100/100); b) a lightly skewed workload composed of hot/cold data where 20% of the user data is updated randomly (RW 100/20); c) a medium skewed workload following a Zipfian distribution [22] where 20% of the dataset is updated 80% the time (Zipf 80/20); and d) a highly skewed Zipfian workload where 20% of the dataset is updated 95% of the time (Zipf 95/20).

Analysis dimensions. The device utilization has a significant impact on performance. First, at low utilization, the size of the SLC tier can be large enough to hold all frequently updated user data such that the controller achieves SLC-equivalent endurance and performance. Second, the instantaneous overprovisioning is high which results in a low write amplification that is close to one. Therefore, we present all results as a function of the device utilization to capture the dynamic controller behavior. We expect the device utilization in typical enterprise SSD deployments to be in the range of $15-85\%$, with an average utilization of 60%.

A. Importance of adapting the SLC cache size and utilization

Fig. 7 shows the impact of adapting the cache size to the device utilization. We use a random write workload to compare three types of controller architectures as we vary the device utilization. The first controller type uses a fixed-size SLC cache that is representative of the first generation of hybrid controllers. We use four different SLC cache sizes (the dotted lines) where 1%, 5%, 10%, and 20% of the total flash blocks are statically set to the SLC mode. The second controller type represents the second generation of hybrid controllers with adaptive SLC caches that adjust the SLC size to the device utilization and use a simple LRW destage policy (the brown line). The third controller type adapts optimally both the SLC capacity and the SLC utilization (the green line). It represents the best achievable performance by a hybrid controller that uses an SLC buffer to store new user data.

We note several trends. First, surprisingly, controllers with a fixed SLC to QLC ratio provide little benefit. Endurance and throughput are noticeably improved in only a narrow device utilization range $(0-5\%)$, elsewhere both metrics degrade when compared to a QLC-only controller. When device utilization is high, the fixed SLC caches turn out to be significantly worse at steady state compared to QLC-only and an optimally-sized hybrid controller. The capacity invested in the SLC tier would be better put to use as extra overprovisioning for the QLC tier that would significantly reduce write amplification and therefore improve endurance and write performance. The SLC tier is still helpful to reduce write latency as long as data can be destaged fast enough in the background to QLC, however, either endurance and throughput are severely impacted or the device capacity is artificially reduced. Second, a controller that adapts the SLC size performs better at both low and high device utilization. The range where endurance and throughput improve extends to $\sim 30\%$, while at high device utilization endurance and throughput almost match the behavior of the QLC-only baseline. Third, a controller that introduces SLC-to-SLC relocations and adapts the SLC occupancy (Fig. 3(c)) further extends the achieved gains. The additional improvement is significant, however, it is confined to the $10-25\%$ utilization range.



(a) Write endurance. (b) Average write overhead. (c) Breakdown of the average write overhead.

Fig. 8. Relative performance of hybrid controller architectures versus a QLC-only controller when increasing write skews.

Overall, we observe that the benefits of hybrid controllers can be significant – both endurance and write throughput can be improved by more than an order of magnitude by the judicious use of an SLC tier. However, the benefits are limited to a narrow utilization range where the actively written data mostly fits in the SLC tier. We conclude that accurately sizing the tiers and the target SLC utilization is key to both maximizing endurance and performance and also reducing the drawbacks of a hybrid controller at high device utilization.

B. Importance of heat-aware data placement

As real-world I/O workloads tend to be skewed, modeling a workload using uniform random writes to the whole address space is not indicative of the actual controller performance. It is therefore of great interest to study the expected endurance and write throughput improvements as a function of write workload skew. We achieve this by modeling the performance metrics of a heat-aware adaptive controller (Fig. 3(d)) for various skewed workloads as a function of the device utilization.

Fig. 8(a) shows the relative endurance improvement of this hybrid controller over a QLC-only baseline. The QLC-only controller performs heat-segregation in the same way and is using the same number of heat streams as the hybrid controller, therefore all endurance gains are a consequence of the hybrid controller architecture. As expected, the endurance and throughput benefits of the hybrid controller increase as the workload becomes more skewed. However, the endurance gains are unevenly distributed. For the low utilization range $(0-20\%)$, endurance is improved by $10-20\times$ as the working set fits into the SLC tier. With increasing utilization, more data is being destaged to QLC, and we observe a sharp drop for all workloads. The endurance for the RW 100/20 workload does not improve significantly if the device utilization is higher than 50%. However, for the medium-skewed Zipf 80/20 workload, we see significant improvements of at least $2\times$ across the practical utilization range and 50% higher endurance at maximum device utilization. For the highly-skewed Zipf 95/20 workload, the endurance improvements grow to over $4\times$ over the practical utilization range.

Fig. 8(b) shows how the write overhead increases (and therefore how write throughput drops) as a function of the device utilization. Surprisingly, the write throughput gains are higher than the endurance gains. We see an improvement of $32-3\times$ for the Zipf 80/20 and $32-12\times$ for the Zipf 95/20 workload when comparing them to a controller not using heat information. The reduction in write overhead can be attributed to two factors: a) the overall write amplification reduction achieved by heat segregation in both the SLC and QLC tiers and b) the decrease in QLC writes due to the hybrid controller architecture that stores write hot data in SLC. We break down the write overhead reduction and attribute it to the two sources. Fig. 8(c) shows that the majority of the write throughput improvement can be attributed to the hybrid controller architecture.

These results show that both, write throughput and endurance can be significantly improved by leveraging write heat segregation. It is hence paramount to accurately track heat information to improve data placement and QLC destages.

C. Impact of flash media evolution

The properties of flash media in the two SLC/QLC modes (Table I) have a significant impact on the absolute endurance and write performance metrics. However, it is not immediately obvious that the relative ratio between the SLC and QLC metrics changes the performance of the presented controller architectures. As QLC flash technology is evolving, it is not unreasonable to expect improvements, for example, due to the introduction of newer materials, better manufacturing processes or more accurate read voltage shifting.

As an example, we show the relative endurance gain that can be achieved when introducing the ability to selectively write to one of the tiers depending on the workload, write heat, and device utilization, rather than always writing data first to the SLC tier (architectures (d) vs. (e) in Fig. 3). The ability to selectively write data to either tier introduces a significant amount of complexity and is worth considering only if it provides measurable benefits as it complicates the internal data flow. We use the lightly skewed RW 100/20 workload as it showed little endurance improvement at high device utilization (see Fig. 9). Each line represents a different ratio between the supported SLC and QLC program erase cycles. The experiments presented so far correspond to the brown line ($100,000$ SLC PEC / $1,150$ QLC PEC ≈ 87). At this relative SLC/QLC endurance ratio there is little benefit from selective data placement – it only improves endurance by less than 5%. However, as the relative SLC/QLC PEC ratio decreases (QLC endurance improves relatively to SLC), the selective data destage leads to higher endurance gains. For example, if the SLC/QLC PEC ratio is reduced to $10\times$, then selective

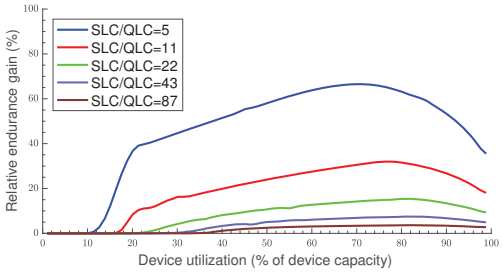


Fig. 9. Relative endurance gain by introducing selective tier data placement.

data placement provides up to 25% additional endurance for an otherwise unfriendly workload.

V. Related work

Hybrid flash controllers. The two types of existing hybrid flash controller architectures that have been introduced in Section I-B are hybrid controllers with a fixed-size SLC cache [11], [12] and controllers with adaptive SLC caches [8]–[10], [13], [14]. A more recent trend is to combine flash and 3DXP memory in the same device [23]. Such a design suffers from the same problems as fixed-size SLC caches as they cannot adapt to the workload or device utilization. The cache will either be too small (and therefore ineffective) or too big (and therefore needlessly increase cost and reduce storage density). Nonetheless, our modeling framework can be used to predict the performance or endurance of such SSD controller architectures. The only change required to the modeling framework would be to include a model of a 3DXP cache, i.e., an upper-tier model that supports in-place updates. Although not discussed for brevity, we note that the LRW upper-tier (III-C1) and heat-aware upper-tier model (III-C4) are easy to adapt to account for the lack of write amplification when using phase-change memory.

Other hybrid controller designs study the benefits of decreasing the bit density in worn-out blocks with the goal of extending endurance [24], [25]. Similarly, two worn-out pages can be combined into a single page [26]. As the assignment of distinct physical parts of flash devices to a pool can lead to unbalanced wear, a soft-partitioning scheme to balance the wear of the pools is proposed in [27]. However, such approaches result in a reduction of the device capacity, a consequence undesirable in commercial SSDs.

Analytical modeling. Our modeling framework builds upon previous efforts aimed at understanding and quantifying write amplification through mathematical models [16]. Analytical models for computing write amplification for random workloads were first derived in [17], [19]. The model we use for computing the relocation rate for arbitrarily skewed workloads was inspired from [17], [28]. We extend existing work by introducing new models for the cache tier designs and propose an improved mathematical model to compute write amplification for arbitrary workloads.

SSD simulation. Amber [29] and MQSim [30] are SSD simulation frameworks that enable a much faster prototyping of new controller designs in a more friendly development environment that abstracts the constraints imposed when developing a new FPGA- or ASIC-based flash controller. We view simulation frameworks as complementary to our modeling work. A modeling approach is significantly faster than SSD simulation, both when implementing a new controller design or when evaluating its workload-depended behavior. For example, we estimate the number of data points modeled for this paper at 1×10^6 which took $24\,\mathrm{h}$ CPU time to compute, implying an average modeling latency of less than $50\,\mathrm{ms}$ for a dataset $> 1\,\mathrm{TB}$ (the modeling time depends on how many controller parameters need to be optimized and the workload type). A fast simulation framework, even when no user data is stored and many of the SSD details are abstracted, will take at least a few minutes to complete a single simulation run. This yields at least three orders of magnitude increase in runtime or about four CPU years to obtain equivalent results. However, once an efficient controller design is identified through modeling, it is advisable to implement a proof-of-concept in a simulation environment to resolve implementation-specific details and validate its dynamic performance.

VI. Conclusions

This paper aims to predict and explain the behavior of hybrid controller architectures for NAND flash through a modeling framework that estimates the upper bound performance or endurance. Our modeling framework computes the internal data movement in an SSD by relying on novel analytical models that offer both accurate and yet fast predictions. The data flow is then translated into higher-level metrics that quantify upper bounds for the overall performance of an SSD, such as write throughput, latency, or device endurance.

As an example of the capabilities of the framework, we compare multiple controller architectures and show that there is a large room to improve the efficiency of the hybrid controllers used today. We investigate three dimensions for hybrid controllers. First, we demonstrate the importance of accurately adjusting the SLC cache size and the SLC occupancy (amount of data cached in SLC) to the workload properties and to the device utilization. Second, we show the importance of accurately tracking and exploiting write heat information. For real-world workloads, which typically exhibit a large amount of skew, both write endurance and throughput can be increased significantly, even when the hot working set does not fully fit in the SLC tier. Third, we show the importance of re-considering the controller architecture as the storage technology improves over time and the relative values of the SLC and QLC flash metrics shift.

Looking towards the future, the modeling approach presented is more broadly applicable to other hybrid SSDs or mixed technology NVM SSDs. For example, the modeling approach can be used for both prediction of write throughput for conventional single-mode controllers or prediction of the behavior of SSDs that combine multiple types of persistent memory (MRAM, 3DXP, Flash).

REFERENCES

[1] X.-Y. Hu, R. Haas, and E. Eleftheriou, "Container marking: Combining data placement, garbage collection and wear levelling for flash," in *Proceedings of the 19th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '11, Jul. 2011, pp. 237–247. [Online]. Available: http://dx.doi.org/10.1109/MASCOTS.2011.50

[2] R. A. Pletka and S. Tomić, "Health-binning: Maximizing the performance and the endurance of consumer-level nand flash," in *Proceedings of the 9th ACM International on Systems and Storage Conference*, ser. SYSTOR '16. New York, NY, USA: ACM, 2016, pp. 4:1–4:10. [Online]. Available: http://doi.acm.org/10.1145/2928275.2928279

[3] R. Pletka, I. Koltsidas, N. Ioannou, S. Tomić, N. Papandreou, T. Parnell, H. Pozidis, A. Fry, and T. Fisher, "Management of next-generation NAND flash to achieve enterprise-level endurance and latency targets," *ACM Trans. Storage*, vol. 14, no. 4, pp. 33:1–33:25, Dec. 2018. [Online]. Available: http://doi.acm.org/10.1145/3241060

[4] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proc. of the IEEE*, vol. 109, pp. 1666–1704, Sep. 2017.

[5] (2018) Toshiba's 768Gb 3D QLC NAND flash. [Online]. Available: https://www.anandtech.com/show/11590/toshiba-768-gb-3d-qlc-nand-flash-memory-1000-p-e-cycles

[6] (2018) Scaling Flash Technology to Meet Application Demands. [Online]. Available: https://www.anandtech.com/show/13181/flash-memory-summit-toshiba-keynote-live-blog

[7] Jeff Yang. (2018) Raising QLC Reliability in All-Flash Arrays. [Online]. Available: https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2018/20180807_ENST-102-1_Yang.pdf

[8] (2018) Intel 660p SSD. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/660p-series-brief.pdf

[9] (2019) Samsung 860 QVO SSD. [Online]. Available: https://www.samsung.com/semiconductor/minisite/ssd/product/consumer/860qvo/

[10] (2019) Crucial P1 SSD. [Online]. Available: https://www.crucial.com/usa/en/storage-ssd-p1

[11] L.-P. Chang, "A hybrid approach to NAND-flash-based solid-state disks," *IEEE Trans. on Computers*, vol. 59, no. 10, pp. 1337–1349, Oct. 2010. [Online]. Available: http://dx.doi.org/10.1109/TC.2010.14

[12] 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, pp. 641–653, Dec. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2010.09.005

[13] 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

[14] 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. Very Large Scale Integr. Syst.*, vol. 24, no. 10, pp. 3132–3144, Oct. 2016. [Online]. Available: https://doi.org/10.1109/TVLSI.2016.2538182

[15] N. Ioannou, K. Kourtis, and I. Koltsidas, "Elevating commodity storage with the salsa host translation layer," in *IEEE MASCOTS*, Sep. 2018.

[16] X.-Y. Hu, E. Eleftheriou, R. Haas, I. Iliadis, and R. Pletka, "Write amplification analysis in flash-based solid state drives," in *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference*. ACM, 2009, p. 10.

[17] 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.

[18] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the LambertW function," *Advances in Computational Mathematics*, vol. 5, no. 1, pp. 329–359, 1996.

[19] P. Desnoyers, "Analytic modeling of SSD write performance," in *Proceedings of the 5th Annual International Systems and Storage Conference*. ACM, 2012, p. 12.

[20] Y. Etsion and D. G. Feitelson, "Exploiting core working sets to filter the L1 cache with random sampling," *IEEE Trans. Computers*, vol. 61, no. 11, pp. 1535–1550, 2012. [Online]. Available: https://doi.org/10.1109/TC.2011.197

[21] Y. Yang and J. Zhu, "Write skew and zipf distribution: Evidence and implications," *ACM Trans. Storage*, vol. 12, no. 4, pp. 21:1–21:19, Jun. 2016. [Online]. Available: http://doi.acm.org/10.1145/2908557

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

[23] (2019) Intel Optane Memory H10 with Solid State Storage. [Online]. Available: http://www.intel.com/content/www/us/en/products/docs/memory-storage/optane-memory/optane-memory-h10-solid-state-storage-brief.html

[24] X. Jimenez, D. Novo, and P. Ienne, "Phœnix: reviving MLC blocks as SLC to extend NAND flash devices lifetime," in *Proc. of the 2013 Design, Automation and Test in Europe*, ser. DATE '13, Mar. 2013, pp. 226–229. [Online]. Available: https://doi.org/10.7873/DATE.2013.059

[25] E. H. Wilson, M. Jung, and M. T. Kandemir, "ZombieNAND: Resurrecting dead NAND flash for improved SSD longevity," in *Proceedings of the 2014 IEEE 22nd International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '14, Sep. 2014, pp. 229–238. [Online]. Available: https://doi.org/10.1109/MASCOTS.2014.37

[26] H.-Y. Lin and J.-W. Hsieh, "HLC: Software-based half-level-cell flash memory," in *Proceedings of the 2015 Design, Automation and Test in Europe*, ser. DATE '15, 2015, pp. 936–941. [Online]. Available: http://dl.acm.org/citation.cfm?id=2755753.2757031

[27] X. Jimenez, D. Novo, and P. Ienne, "Software controlled cell bit-density to improve nand flash lifetime," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 229–234.

[28] Y. Yang and J. Zhu, "Write amplification with write skew," in *Proceedings of the 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '16, Sep. 2016, pp. 406–411. [Online]. Available: https://doi.org/10.1109/MASCOTS.2016.23

[29] A. Tavakkol, J. Gómez-Luna, M. Sadrosadati, S. Ghose, and O. Mutlu, "MQSim: A framework for enabling realistic studies of modern multi-queue SSD devices," in *Proc. of the 16th USENIX Conference on File and Storage Technologies*, ser. FAST '18, Feb. 2018, pp. 49–66. [Online]. Available: https://www.usenix.org/conference/fast18/presentation/tavakkol

[30] D. Gouk, M. Kwon, J. Zhang, S. Koh, W. Choi, N. S. Kim, M. Kandemir, and M. Jung, "Amber: Enabling precise full-system simulation with detailed modeling of all SSD resources," in *Proceedings of 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '18, 2018, pp. 469–481. [Online]. Available: https://doi.org/10.1109/MICRO.2018.00045