IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

Analysis of Write Amplification with Time Locality and/or Multi-Write Access

Saher Odeh and Yuval Cassuto

CCIT Report #881
March 2015

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

CCIT Report #881 March 2015

Analysis of Write Amplification with Time Locality and/or Multi-Write Access

Saher Odeh
Technion EE Department
sahero@tx.technion.ac.il

Yuval Cassuto
Technion EE Department
ycassuto@ee.technion.ac.il

ABSTRACT

Write amplification is a central performance figure of solid-state storage devices. As a result, significant effort has been devoted in the storage systems community to analyzing write amplification in useful scenarios. Analysis here means that a parametrized model is chosen for the device and its incident workload, such that the expected average write amplification can be obtained mathematically. While reaching this goal necessitates a detailed mathematical argumentation, such analysis offers the great benefit of predicting performance without need to deploy or simulate the device. In this paper we develop an analysis framework to calculate the write amplification for two novel and practically important scenarios: workloads with time locality, and devices with multi-write capabilities. The former captures a central feature of real-world workloads, while the latter addresses a promising feature likely to be added to next-generation solid-state storage devices.

1. INTRODUCTION

Non-volatile storage devices have become central components in almost every information system. It is likely that the adoption of such devices will continue to grow fast, thanks to the superior access performance they offer and the impressive density scaling they sustain. In particular, NAND-flash based storage devices are currently the champions of non-volatile storage in terms of performance and density for unit cost. All their advantages notwithstanding, there are two major issues that complicate the adoption of NAND-flash storage devices in real systems. One is their non-deterministic access performance due to address indirection done at the flash translation layer (FTL). The other issue is their limited endurance, which degrades their reliability as they age in the system. Both of these issues are significant, and thus must be addressed by the device vendors during design, and monitored by the customers during operation. Nevertheless, a major challenge is that both performance variability and endurance issues are driven by complex hardware and software mechanisms in the device. Consequently, the storage community has sought deterministic and clear characterizations of these issues, which will allow evaluating and comparing different storage devices. The principal such characterization adopted by the storage industry is the *write amplification* (WA), which is defined as the average number of physical writes performed per user write. The write amplification emerged as a central performance measure, because it captures both the access-performance and endurance issues: writing more internally in general implies worse read/write rates for the user, as well as higher cell wear.

While not a perfect characterization of non-volatile storage performance, the write amplification has fostered great innovation as a tool to design better storage devices and to understand their behaviors. The most immediate use of the write amplification is to benchmark existing or newly proposed device architectures against relevant workload traces. In this usage, we run a workload trace through the architecture in discrete-event simulation, count the total number of physical writes performed by the device, and normalize by the number of user writes in the trace to get the empirical write amplification. To get a more comprehensive quantitative evaluation of the device, we complement the workload traces with synthetic workloads generated by some probabilistic model, most commonly the *uniform workload*. The advantage of synthetic workloads is that they are easier to obtain than recorded traces, and thus can test devices for longer, more realistic durations.

To make the device evaluation even more efficient, we are interested in methods to derive the write amplification analytically, obviating the need to run long and resource-consuming simulations. Indeed, several works in the literature successfully provide analytical tools to estimate the write amplification exhibited by a storage device given its design, e.g. the garbage collection policy it employs, and the workload that it serves. In the sequel we survey some of these works. In the mean time we note that in general the write-amplification analysis task becomes very difficult very fast as we attempt to capture more realistic device architectures and workload models.

1.1 Our Contributions

The study reported in this paper contributes several novel write-amplification analysis tools for important workload models and device architectures. Specifically, we list the following items as our main contributions

1. Defining a new model for workloads with *time locality*.

2. Deriving the write amplification for *workloads with time locality*. The analysis yields the write amplification value given the workload's time-locality level, which is specified as the parameter of the time-locality model in item 1 above.

3. Deriving the write amplification of a *multi-write* architecture, which allows 2 writes to a physical page before its erasure. The analysis yields the 2-write write amplification value given the workload's time-locality level.

4. Deriving the write amplification of the *advanced* multi-write architectures we previously reported in [Odeh 2014]. The importance of this contribution is that these advanced architectures were shown to give superior write-amplification performance over the simple architecture with multi-write across the entire storage space.

5. The analysis solutions in items 2-4 above are shown to give very accurate prediction of the write amplification when compared to simulations of the same setups.

1.2 Problem Setup

The setup description starts with the standard definitions of a storage device with T physical blocks, each comprising N_p physical pages. To the device are assigned in total $U \cdot N_p$ logical pages, which are written and read by the user. The parameter ρ is called the *over-provisioning* factor, and is defined as $(T-U)/U$. The standard way to manage the device is by placing user-written pages sequentially on a block called *the frontier*. When the frontier block becomes full, a new clean block becomes the frontier. Blocks are cleaned in a process called *garbage collection* (GC), and we assume throughout the paper that the block chosen for cleaning in GC is the one with the minimal number of valid pages on it. This selection policy is called here *min-valid*, and is also known in the literature as the *greedy* cleaning policy. A more detailed background on this general setup can be obtained from a number of previous works listed in the next sub-section.

In the paper we specialize this setup in two ways. In the first part consisting of Sections 3 and 4, the device follows this standard operation, but while serving workloads with *time locality*. Time locality is a common feature of real-world workloads, by which a logical page written by a user at a given time has a higher likelihood to be written again in the near future. A simple model of time locality is given in Section 3 using two parameters: p specifies the probability to select a page-write from a pool of recently written logical page addresses, and h is the size of this recent-page pool. The device write amplification under this time-locality workload model is analyzed in Section 4. To the best of our knowledge, this is the first time write-amplification analysis is developed for time-locality workloads. The key novelty in analysis with time locality is to analyze the state of blocks in different time epochs throughout their life cycle. In the second part consisting of Sections 5 and 6, we depart from the standard device model and analyze a device with multi-write capabilities, that is, a device whose physical pages can be written *two times* before the block is cleaned. The importance of addressing the multi-write feature stems from recent work [Odeh 2014, Yadgar 2015] showing evidence of the great promise that multi-writes can bring to storage devices. The key novelty in analysis with multi-write capabilities is moving from tracking page states as valid/invalid, to also distinguish between valid pages that were already written twice and valid pages that can still accept an in-place update. In Section 5 the device supports multi-write on all pages, while in Section 6 the multi-write capabilities are offered on only part of the physical pages. This latter approach was shown to offer superior write-amplification performance, hence the importance of its analysis.

In all the different setups our objective is the same: to derive the write amplification of a storage device operated in that setup. In every case the answer is given as a single value of the variable WA calculated by a system of equations as a function of constants describing the device (T, U, N_p, ρ), and the workload (p, h). (It turns out that only the p parameter of the time-locality model affects the resulting write amplification.) In the second part an additional constant r is given as input, describing the expansion factor required to support the multi-write capabilities. Because in most of the setups exact analysis is impossible or very difficult, our derivations include steps that are approximations. In all such cases we properly justify the approximations. In addition, at the end of each section we compare the analysis results (including the approximations) to simulation plots in order to prove their good accuracy. Solving for WA is done by defining additional variables internal to the device operation, and deriving sufficiently many equations to be able to solve for all these variables. Then the desired WA is calculated from these internal variables. Per our previous remark, it is understood that in some of these equations the equality is only an approximation. The complexity of solving each setup is primarily expressed by the number of equations required to solve it. In that respect, the setups are ordered in increasing complexity, where Section 4 uses 1 equation, Section 5 uses 3 equations, and the most complex Section 6 uses 5 equations. To avoid notation clutter, we use the same letters for the variables and auxiliary variables across sections and setups. So it is important to note that in each section the same variables appear, but with different relations between them, as induced by the particular setup. In contrast, all the notations using the letter p with subscripts represent static probability values that do *not* depend on the setup, only on the external parameters.

1.3 Prior Work

The objective of this paper is to extend write-amplification analysis to fundamental and practical setups including time locality and multi-write capabilities. In that it is definitely only one link in a fruitful line of work analyzing write amplification in a variety of real-life setups.

Relevant work started much earlier than the advent of solid-state flash-based storage devices. Much of this earlier work was motivated by the introduction of log-structured file systems [Rosenblum 1992], whose operation has fundamental similarities to the standard FTL operation used in flash-based storage devices. Probably the most important contribution to this line of work is the analysis of uniform workloads under the min-valid (aka greedy) cleaning policy pursued in [Robinson 1996, Desnoyers 2012]. [Desnoyers 2012] also extends the analysis to non-uniform *hot/cold* workloads defined in [Robinson 1996]. In this hot/cold analysis, it is assumed that the hot/cold property of pages is static, and

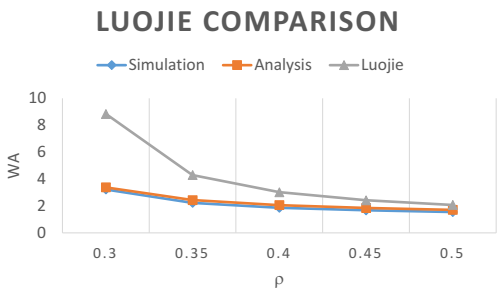

Figure 1: Analysis comparison to [Luojie 2012B]. New analysis improves accuracy considerably.

that the device knows enough on the workload to separate between hot and cold pages. In contrast, in our time-locality analysis all pages behave the same, and we do not assume the device has any knowledge on the workload. The importance of the works of [Robinson 1996, Desnoyers 2012] stems from the very powerful techniques developed based on Markov-chain steady-state analysis (a similar model also appears in [Bux 2010]). In fact, all of our results in this paper use these Markov-based techniques, appropriately (and non trivially!) extended to address more complex scenarios. [Hu 2009] also presents an analytical model for computing the write-amplification for the uniformly distributed workload under the windowed-greedy cleaning policy. The mathematical toolbox for write-amplification analysis was significantly enriched following a recent line of work by van-Houdt using the mean-field model [Van 2013A, Van 2014, Van 2013B]. Among the achievements of this work are analyzing write-amplification for sub-optimal cleaning policies (which are often more practical than the greedy policy), and studying more refined hot/cold scenarios. A complementary line of work [Agrwal 2010, Luojie 2012A] contributes a simpler approximated technique for write-amplification analysis, which allows reaching good estimates without invoking a full-fledged Markov analysis. This approximated analysis was also extended to architectures with all multi-write access in [Luojie 2012B]. We compare our results for the all multi-write analysis to this prior work, and conclude that our Markov-based method gives significantly better estimates, see Figure 1. We conjecture that with multi-write access there is limited validity to the approximation assumptions that the authors of [Luojie 2012B] extend from [Agrwal 2010]. It is also noted that this prior approach cannot accommodate time locality into the analysis.

2. THE UNIFORM SINGLE-WRITE CASE

We start the analysis by considering standard no multi-write device under the uniform workload, which was widely studied in previous works. In this basic configuration, the device uses the standard flash mapping architecture, without multi-write coding, on a uniformly distributed workload. To aid the analysis of the more advanced cases later in the paper, we now review the Markov-chain analysis technique from [Desnoyers 2012]. By that technique, we divide the blocks in the device to *states* representing the numbers of valid pages in the block, and then use the Markov state diagram to derive the mean number of pages copied in GC

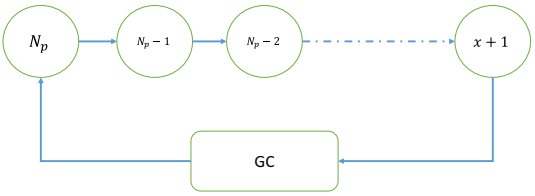

Figure 2: Markov-chain model for uniform workload without multi-write.

under the *min-valid* GC policy. This Markov-chain model is given in Figure 2. Let x denote the mean number of valid pages copied back in GC events. A block at state $i \in [x+1, N_p]$, where $[a,b]$ stands for the set $\{a, a+1, ..., b\}$, has i valid pages. We assume that the state $x+1$ is the state with the minimal number of valid pages that a block may have, and ignore all states with fewer valid pages assuming they occur with negligible probability.

We define a discrete time axis where each time unit is a single-page *user* write. Internal writes due to GC are not counted in the progression of time. New blocks arrive after GC at the state N_p, and exit from the state $x+1$ to a block with x valid pages which is then garbage-collected. The rate at which blocks transition from $x+1$ back to N_p by GC is $\frac{1}{N_p-x}$, that is, every $N_p - x$ user writes the previous frontier is replaced by a new one. We assume that after a sufficient amount of time, the device reaches an equilibrium, where the fraction of pages at a given state is constant. Then from flow conservation the transition rates between every pair of neighboring states is equal.

Let f_i denote the fraction of blocks in state i. The transition rate from state i to state $i-1$ is the probability that a user write is addressed to a page in one of the Tf_i blocks in state i. Overall there are iTf_i such valid pages, and from the uniform workload this probability equals $\frac{i \cdot T \cdot f_i}{UN_p}$. Equating this probability to the GC rate gives

$$\frac{i \cdot T \cdot f_i}{UN_p} = \frac{1}{N_p - x} \; : \; i \in [x+1, N_p] \, .$$

And now substituting the over-provisioning factor $\frac{T}{U} = 1+\rho$ and reordering we get

$$i \cdot f_i = \frac{1}{1+\rho} \cdot \frac{N_p}{N_p - x} \; : \; i \in [x+1, N_p] \, . \qquad (1)$$

Now to obtain an expression for x, we sum-up f_i for all i, and equate to 1, as this sum counts all the block fractions that make up the entirety of the device's blocks

$$1 = \sum_{x+1}^{N_p} f_i = \frac{1}{(1+\rho)} \cdot \frac{N_p}{N_p - x} \cdot \sum_{x+1}^{N_p} \frac{1}{i}$$

$$\backsimeq \frac{1}{(1+\rho)} \cdot \frac{N_p}{N_p - x} \cdot \ln\left(\frac{2N_p+1}{2x+1}\right) . \qquad (2)$$

The last (approximate) equality follows from the approximation

$$\sum_{x+1}^{N_p} \frac{1}{i} \backsimeq \int_{x+\frac{1}{2}}^{N_p+\frac{1}{2}} \frac{1}{i} di = \ln\left(\frac{2N_p+1}{2x+1}\right) .$$

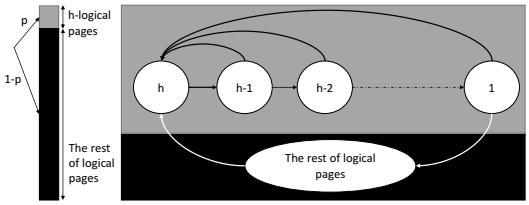

Figure 3: Schematic description of the time-locality model with the locality parameter p.

Now, by rearranging equation (2) we get the final equation

$$\frac{(1+\rho)\cdot(N_p - x)}{N_p} = \ln\left(\frac{2N_p+1}{2x+1}\right). \quad (3)$$

Using equation (3), we numerically calculate x. The write amplification then equals

$$WA = \frac{N_p}{N_p - x}. \quad (4)$$

In the sequel we extend this Markov analysis technique above to workloads with time locality and with multi-write. Toward that we next formalize the notion of time locality.

3. TIME-LOCALITY MODELING

We now specify a model for workloads with time locality. The same model that we use here for analysis was used in our previous work [Odeh 2014] for experimental evaluation of multi-write architectures with time-locality workloads. We model time-locality workloads with two parameters describing the "locality level" of the workload. The first parameter among the two is the *probability* for a write to address a page taken from a pool of recently written pages, which is called the *recent-pool*. We denote this probability by p, and we refer to this type of write as a *local* write. The page written in a local write is chosen uniformly from the recent-pool. A non-local write, happening with probability $1-p$, is chosen uniformly from the entirety of the logical pages. See Figure 3 for a pictorial description. Hence the probability for a certain page to be written as a non-local write is

$$p_c = \frac{1-p}{UN_p}. \quad (5)$$

We note that the probability in equation (5) is identical to the uniformly distributed workload when setting $p = 0$. The recent-pool *size* is the second parameter of the workload, denoted by an integer h. The recent-pool acts as a queue, i.e, written pages enter the tail of the queue, and leave at the head of the queue when h more recent pages have entered the queue. Hence the probability for a certain write to be a particular page in the recent-pool is

$$p_h = \frac{p}{h}. \quad (6)$$

The probability that a write causes a page to exit the recent-pool is equal to the probability that a new page enters it. This probability is equal to the probability that the write is non-local, and to a page not in the recent-pool. Since the latter event's probability is close to 1 (assuming $h \ll UN_p$), we approximate this probability to

$$p_{exit} \backsimeq 1 - p. \quad (7)$$

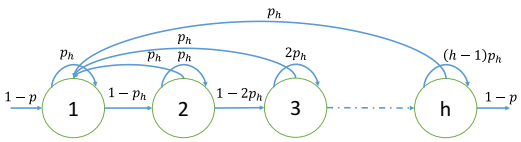

Figure 4: The recent-pool queue.

An important component in the analysis is the probability that a recently user-written page exits the recent-pool queue without being written again by the user. This probability is denoted by p_e, which turns out to equal p_{exit} as shown in the subsequent argument. To see that, we examine a newly user-written page. As a result of the write, it enters the tail of the recent-pool queue. Those pages that are not re-written while in the queue advance through the states of the queue, until they finally exit at the head of the queue as a result of a non-local write. To calculate the probability that this happens, we use the state diagram in Figure 4 . The states in Figure 4, numbered 1 through h, represent the pages in the queue, where state 1 corresponds to the tail of the queue, i.e, where newly user-written pages enter, and the state h corresponds to the head of the queue, i.e, the last state that a page reaches before it exits the queue. The edges in the figure represent the possible transitions of pages between the states, each with the probability for such transition in a given user-write. The transitions can be categorized to the following scenarios given a fixed state i:

1. With probability $p_{exit} \backsimeq 1-p$ a page not in the recent-pool is written, in which case the new page enters at the tail of the queue, i.e, at state 1. All of the pages in the pool shift one state to the right and the head of the queue, i.e, the page at state h, exits the pool.
2. With probability p_h the page at state i is written, in which case the page moves back to the tail of the queue, i.e, to state 1, and the pages to its left move one state to the right.
3. With probability $(i-1)\,p_h$ a write is issued to a state lower than i, leaving the page at its current state i.
4. With probability $(h-i)\,p_h$ a write is issued to a state higher than i, transferring the written page to the tail of the queue, and shifting the pages at states i and lower one state to the right.

To calculate an expression for p_e, we construct a system of equations describing the above behavior of the queue for each state. Let $\Pr(i)$ denote the probability for a page to exit the head of the queue without experiencing any writes given that it has reached the state i. For example, the probability to exit the head of the queue given that the page already reached the state h, i.e, $\Pr(h)$, equals p_{exit} plus the probability to remain in the same state times $\Pr(h)$, see equation (8) part (3). Now the probability for a page in the initial state 1 to exit the head of the queue with no additional writes, i.e, $\Pr(1)$, equals the probability for the page to move one state to the right, times $\Pr(2)$, see equation (8) part (1). And the general case for $\Pr(i)$ is equal to the sum of the probabilities of two scenarios:

1. The probability that the page has stayed in state i times $\Pr(i)$, which is $(i-1)\,p_h \cdot Pr(i)$ from transition type 3 above.

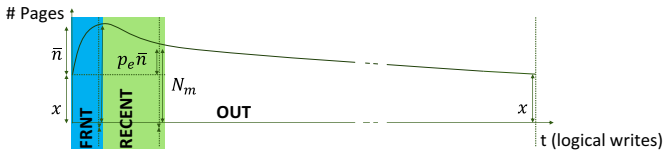

Figure 5: A block's life cycle between two consecutive garbage collections of the same block.

2. The probability that the page has moved one state to the right times $\Pr(i+1)$, which is $(1-i\cdot p_h)\cdot Pr(i+1)$ from transitions 1 and 4 above.

See equation (8) part (2) for the full equation of $\Pr(i)$.

$$\begin{array}{ll}(1) & p_e = Pr(1) = (1-p_h)\cdot Pr(2)\\ (2) & Pr(i) = (i-1)\,p_h Pr(i) + (1-ip_h)\,Pr(i+1)\\ (3) & Pr(h) = (h-1)\,p_h\cdot Pr(h) + (1-p)\end{array} \quad (8)$$

After solving equation (8) we get:

$$p_e = (1-p)\,. \quad (9)$$

An interesting implication of equation (9) is that the probability that a page gets invalidated while in the recent-pool does not depend on the recent-pool size h, only on p. As a result, the parameter p will be the key determinant of the write-amplification performance.

4. SINGLE-WRITE CASE WITH LOCALITY

In this section we derive an analytical formula for the write amplification for workloads with time locality. We will adapt the Markov-chain analysis of Section 2 to capture the fact that recently written pages have higher likelihood to be invalidated while in the recent-pool. We examine the user-written pages added to the frontier block. Each of these pages will experience one of two scenarios

1. The page does not get written again; in this case the page remains valid and exits the recent-pool queue. The probability for this scenario is denoted by p_v (valid), and is equal to p_e, the probability for a page to exit the recent-pool without being written again, defined in equation (9).

2. The page gets re-written at least once, leaving the physical page invalid. The probability for this scenario is denoted p_{inv} (invalid), and $p_{inv} = 1 - p_e$.

A block's life-cycle from frontier to GC consists of the following four epochs (see Figure 5 for a pictorial description)

1. The frontier epoch: After garbage-collecting a block, its valid pages are written-back to the block along with new user pages until there is no place left for new writes.

2. The recent-page epoch: After the frontier epoch is complete, some of the block's pages are recent user-writes, and are still in the recent-pool.

3. The non-local epoch: This epoch makes up the majority of the block's life-cycle, and writes issued to the block's pages are all non-local writes.

4. GC epoch: The block is selected for garbage-collection under the minimum-valid (min-valid) policy.

Let $\bar{n}$ denote the mean number of free pages available in the beginning of the frontier epoch. Then we have

$$\bar{n} = N_p - x. \quad (10)$$

Let N_m denote the mean number of *valid* pages in a block at the beginning of the non-local epoch, which includes the pages copied during GC and user-written pages that left the recent-pool without re-write. Thus

$$N_m = x + p_v\cdot\bar{n}. \quad (11)$$

In this case also we model the blocks using the same Markov chain as in Figure 2 with one modification: we assume that blocks enter the Markov chain with N_m valid pages instead of N_p in the all-uniform case. Then the Markov chain describes the state of the blocks in the *non-local epoch*. In the non-local epoch, a block transitions from i valid pages to $i-1$ valid pages with the probability

$$p_c\cdot i.$$

As in Section 2, we assume that an equilibrium is reached and that the flow is conserved throughout the chain, hence the flow of block GC at a rate of $\frac{1}{N_p-x}$ is equal to the flow of blocks out of any state. The flow of blocks out of state i equals the rate of *non-local writes* times the probability for such write to reside on a page belonging to such block, which is equal to $p_c\cdot i\cdot T\cdot f_i$. There are only two changes needed to the analysis from Section 2 to adapt to this case. First, the probability for a non-local write is $1-p$; this means that not every user-write will be issued to a block in the non-local epoch. The second difference mentioned above, is that the non-local blocks start at the state N_m instead of N_p, thereby saving the need to deal with local writes within the Markov analysis (the local writes during the recent-page epoch are taken into account through the value of N_m). Hence, the flow equations are

$$\frac{1-p}{UN_p}\cdot T\cdot i\cdot f_i = \frac{1}{N_p-x}\ :\ i\in[x+1,N_m]\,. \quad (12)$$

Equality follows from substitution of p_c from equation (5). Repeating equation (1) with the necessary modifications we get

$$i\cdot f_i = \frac{1}{(1+\rho)\cdot(1-p)}\cdot\frac{N_p}{N_p-x}\ :\ i\in[x+1,N_m]\,. \quad (13)$$

Similarly to section §2, we sum-up f_i for all $i\in[x+1,N_m]$ to obtain an expression with x, and approximate the result to get the following equation for x

$$\frac{(1+\rho)\cdot(1-p)\cdot(N_p-x)}{N_p} = \ln\left(\frac{2\cdot N_m+1}{2\cdot x+1}\right). \quad (14)$$

Using equation (14) we numerically calculate x and then WA by equation (4). We now compare the analysis WA results to the actual write-amplification values acquired from the simulations. In Figure 6 we compare the results for two over-provisioning factors $\rho\in\{0.1,0.3\}$ using a synthetic workload with locality parameter in the range $p\in[0,0.5]$. First we observe that the analysis delivers write-amplification values that are very close to the true experimental results. In particular, the analysis succeeds in predicting the device behavior of degrading performance with

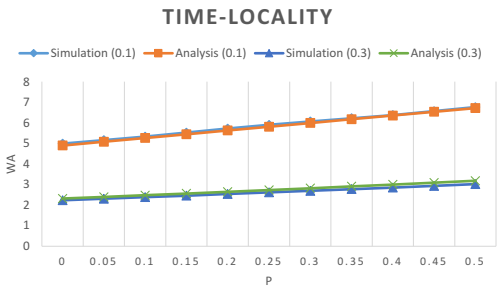

Figure 6: Analysis vs. simulation for the time-locality single-write setup for $\rho \in \{0.1, 0.3\}$.

the increase in time locality. The degradation is more severe when the over-provisioning is smaller, while performance is more flat with larger over-provisioning. Note that it is not clear a-priori that increased time locality should degrade write-amplification performance. On the one hand, increased time locality lowers the invalidation rate of blocks in the non-local epoch, which indeed contributes to increased write amplification. But on the other hand, time locality also has a favorable effect in that blocks enter the non-local epoch with fewer valid pages in the first place, due to invalidations from local writes. So the question that analysis answers is which of the two conflicting effects prevails, which turns out to be the former.

5. MULTI-WRITE CASE WITH LOCALITY

We now move to analyze the write amplification of a storage device with *multi-write* capabilities, i.e., where *2 writes* are allowed in every physical page of the device before erasure. We restrict ourselves to the case of $t = 2$ writes because we expect higher order multi-writes to exhibit diminishing returns. It is possible to extend the analysis to larger t, but some of the steps will require more complex arguments.

Supporting 2 writes through multi-write coding requires storing logical pages in physical pages that are larger by a multiplicative expansion factor $r > 1$. We assume that the value of the expansion factor r is given to us as an outcome of the lower-level controller design. With the 2-write capability, writes may be performed in-place without adding a new page to the frontier, which lowers the WA. On the other hand, the redundancy overhead of the code supporting 2 writes shrinks the over-provisioned storage available to mitigate WA in the first place. Our objective now is to analyze the WA for the 2-write scenario in the presence of time locality as modeled in Section 3. Since the analysis following in the remainder of the section is somewhat more involved than in previous sections, we offer the option to jump directly to the final solution in Section 5.4. This allows to use the resulting analysis tool as a design aid, without requiring to understand the details of how the equations were obtained. That said, we believe that following the analysis at full will provide valuable insight on the behavior of storage devices with multi-write capabilities.

5.1 The Frontier

We start the analysis by considering the block's life cycle as in Section 4, but with significant adaptations necessary to accommodate in-place re-writes. The first epoch of a block is the frontier epoch, which is similar to Section 4; in this case however, the block holds fewer pages, as each page requires more space to accommodate the expansion of the 2-write code. The remaining epochs are similarly defined as in Section 4, while adapting to in-place re-writes.

We now show the adaptations applied to the analysis of Section 4 to handle this case. The mean number of free pages in a block at the beginning of the frontier epoch, i.e, $\bar{n}$, is now smaller due to the expansion factor

$$\bar{n} = \frac{N_p}{r} - x. \tag{15}$$

In Section 4, each newly written user-write to the frontier had one of two possibilities: either remaining valid until it leaves the recent-pool, or being re-written resulting in the physical page's invalidation. Now a newly written page has one of *three* possibilities before leaving the recent-pool:

1. It does not get written again; hence the physical page still has one available in-place write before invalidation. We refer to this case by v_2. The probability of this case satisfies $p_{v_2} = p_e$ in a similar way to p_v in Section 4.

2. It gets re-written *once*; in this case the physical page has a valid content with no extra writes permitted before erase. We refer to this case by v_1. The probability of this case is equal to the probability that the page does get another local-write before leaving the pool, i.e, probability $(1 - p_e)$; and then exiting the recent-pool without additional writes, i.e, with probability p_e. Assuming independence between the two events, we get $p_{v_1} = (1 - p_e) \cdot p_e$.

3. It gets re-written *twice*; in this case the physical page becomes invalid. We refer to this case by *inv*, short for invalid. The probability for this case is equal to the probability for the page to twice get written while traversing the pool from tail to head. Hence, the probability for this case equals: $p_{inv} = (1 - p_e)^2$.

The probabilities of the 3 possible events above sum to 1. Now the mean number of valid pages in a block at the beginning of the non-local epoch equals

$$N_m = x + (p_{v_1} + p_{v_2}) \cdot \bar{n}. \tag{16}$$

To evaluate the benefit of the 2-write setting, we next need to find an expression for the probability that a user-write results in an in-place update, which is a write addressed to a physical page at state v_2. The higher this probability is, the better the savings are in the consumption of spare storage for incoming writes.

There are two types of in-place writes each happening with a different probability: a) A *local* write to a page at the *frontier* or *recent-page* epochs; we denote this probability by $\Pr(local_{inplace})$; b) A *non-local* write to a page at the *non-local* epoch; we denote this probability by $\Pr(nonlocal_{inplace})$.

To calculate $\Pr(local_{inplace})$, we denote by T_x the expected time, measured in units of user writes, that a block spends in the frontier epoch before moving to the recent-page epoch. So every T_x user writes, $\bar{n}$ user-written pages enter the recent-page queue, and a $(1 - p_{v_2})$ fraction of them will endure an in-place write. Therefore, the rate of in-place

writes hitting pages in the frontier and recent-page epochs is given by:

$$\Pr\left(local_{inplace}\right) = \frac{\bar{n}}{T_x}\left(1 - p_{v_2}\right).$$

Now moving to calculate $\Pr\left(nonlocal_{inplace}\right)$, we denote by DV the mean total number of v_2 pages in blocks at the non-local epoch. To have an in-place update at the non-local epoch, we first need to have a non-local write, whose probability is equal to $1-p$. Then we also need the write to be issued to a page that can accommodate an in-place write, whose probability is equal to $\frac{DV}{U \cdot N_p}$, and the denominator neglects the relatively few pages in recent-page epoch blocks. Altogether, the probability $\Pr\left(nonlocal_{inplace}\right)$ is equal to

$$\Pr\left(nonlocal_{inplace}\right) = (1-p) \cdot \frac{DV}{U \cdot N_p}.$$

If we sum together the in-place probabilities $\Pr\left(local_{inplace}\right)$ and $\Pr\left(nonlocal_{inplace}\right)$ and take the complement, we get the fraction of user writes that result in adding a page to the frontier. We denote this value by a, where

$$a = 1 - \Pr\left(local_{inplace}\right) - \Pr\left(nonlocal_{inplace}\right).$$

We view $a \leq 1$ as the slope of the function tracking the number of free pages in the frontier block at discrete times indexed by user writes. In other words, starting at $\bar{n}$ free pages immediately after GC, each user write on average decreases the number of free pages by a. In the analysis of Section 4 where no in-place writes are possible, the slope a equals 1. Substituting $\Pr\left(local_{inplace}\right)$ and $\Pr\left(nonlocal_{inplace}\right)$, we get

$$a = 1 - \frac{\bar{n}}{T_x}\left(1 - p_{v_2}\right) - (1-p) \cdot \frac{DV}{U \cdot N_p}.$$

Now we can write a simple but important relation between the slope a, the number of free pages in a new frontier block $\bar{n}$, and the number of user writes during the frontier epoch T_x

$$aT_x = \bar{n}.$$

Using the last equation, we substitute $a = \frac{\bar{n}}{T_x}$ in the previous slope equation and get

$$\frac{\frac{N_p}{r} - x}{T_x} = \frac{1 - (1-p)\frac{DV}{UN_p}}{2 - p_{v_2}}. \tag{17}$$

Each T_x logical user writes "cost" $T_x + x$ physical writes: x (defined in Section 2) for GC writes, and T_x in frontier and in-place writes. Therefore, an expression for the WA as a function of T_x and x is given by

$$WA = \frac{T_x + x}{T_x}. \tag{18}$$

Next we derive another equation connecting the unknowns T_x, DV and x.

5.2 Double-Valid Estimation

Let N_{dv} denote the mean number of v_2 pages in a block when it enters its non-local epoch. Note that there are two different origins for these v_2 pages: some were added as GC write-back pages, and some were added by user writes. Since GC written pages do not enter the recent-page queue, the probability to re-write them during the short epochs prior to the non-local epoch is very small. So we can assume without significant loss of precision that all of the x GC written pages are at state v_2. For the user-written v_2 pages we use the notations of Section 5.1, by which $\bar{n}$ user-written pages are added to the frontier, and a p_{v_2} fraction of them remain at state v_2 throughout the recent-page epoch. So altogether we reach the following expression for N_{dv}

$$N_{dv} = x + p_{v_2} \cdot \bar{n}. \tag{19}$$

To calculate DV (the mean total number of non-local epoch v_2 pages), we define the function $n_{dv}(t)$ for the number of v_2 pages in a block, t user writes from the time the block entered the non-local epoch. We have the relation $n_{dv}(0) = N_{dv}$. Then DV is calculated as the sum over all T blocks

$$DV = \sum_{i=0}^{T-1} n_{dv}\left(iT_x\right),$$

where we used the fact that every T_x user writes a new block enters the non-local epoch. To obtain an analytic expression, we define $dt = T_x$, and use a continuous-time approximation

$$DV = \frac{1}{T_x}\sum_{i=0}^{T-1} n_{dv}\left(i \cdot dt\right) dt \backsimeq \frac{1}{T_x}\int_0^{T \cdot T_x} n_{dv}\left(t\right) dt, \tag{20}$$

where the right-hand side follows from substituting $t = i \cdot dt$. Now at the continuous-time domain, we write the differential equation governing $N_{dv}(t)$ and its solution as

$$\frac{dn_{dv}(t)}{dt} = -p_c \cdot n_{dv}(t) \Rightarrow n_{dv}(t) = N_{dv} \cdot e^{-p_c \cdot t}, \tag{21}$$

where p_c, defined in (5), is the probability to hit each non-local epoch v_2 page. Now substituting the right-hand side of (21) in the integral of (20), we get

$$DV \backsimeq \frac{N_{dv}}{p_c \cdot T_x} \cdot \left[1 - e^{-p_c \cdot T \cdot T_x}\right]. \tag{22}$$

5.3 Minimum-Valid Estimation

The final, and most interesting step in the analysis is the calculation of x, the expected number of valid pages in the min-valid block. The task at hand is to extend the analysis of Section4 to the case of $t = 2$ re-writable pages. The challenge in this extension is that now the blocks' Markov process needs to track the number of pages in each of the valid-page states v_1 and v_2. For that we define a new Markov process where each state is defined by a pair of indices i, j. A block is at state i, j in the Markov process if it has i valid pages at state v_1 (no re-write remaining), and j valid pages at state v_2 (one re-write remaining). The fraction of blocks in state i, j is denoted $f_{i,j}$, and $\sum_{i,j} f_{i,j} = 1$. The total number of valid pages in a block at state i, j is $i + j$. As always with a Markov analysis, we assume that the device has reached a steady state, in particular, the fractions $f_{i,j}$ remain constant in time.

The 2-dimensional Markov diagram for the case of $t = 2$ re-writable pages is given in Figure 7. The diagram describes the dynamics of the blocks during *the non-local epoch*. There are three types of transitions in the Markov diagram of Figure 7. The first type occurs with a non-local write on a v_1 page, which invalidates it and decreases by one the number of v_1 pages in the block, while leaving the number of v_2 pages unchanged. The rate of this transition equals to

$$Rate\left(f_{i,j} \rightarrow f_{i-1,j}\right) = p_c \cdot i. \tag{23}$$

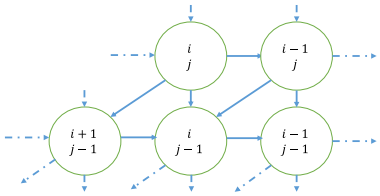

Figure 7: A 2D Markov chain describing $t = 2$ multi-write.

The second type occurs with a non-local write on a v_2 page followed by at least one local write on the same page before its retirement from the recent-page queue. In such case the number of v_2 pages decreases by one, while leaving the number of v_1 pages unchanged. The rate of this transition equals to

$$Rate\left(f_{i,j} \to f_{i,j-1}\right) = p_c \cdot (1 - p_{v_2}) \cdot j. \tag{24}$$

The third type occurs with a non-local write on a v_2 page followed by the retirement of the page from the recent-page queue without experiencing a local write. In such case the number of v_2 pages decreases by one, while the number of v_1 pages increases by one. The rate of this transition equals to

$$Rate\left(f_{i,j} \to f_{i+1,j-1}\right) = p_c \cdot p_{v_2} \cdot j. \tag{25}$$

We now look at diagonal cuts in the Markov diagram, that is, edges between states with $i + j \geq s$ and states with $i + j < s$, for $s \in [x + 1, N_m]$. Recall from Section 4 that the number of valid pages is assumed to be between an upper limit of N_m at the start of the non-local epoch, and a lower limit of x when chosen for GC. Each diagonal cut is crossed by transition edges of the first (23) and second (24) types in the direction of decreasing $i+j$. In the opposite direction we have transitions of total rate $1/T_x$ due to GC, where recall from Section 5.1 that every T_x user writes a new block is cleaned and becomes the frontier. Summing over all the state transitions in diagonal s, we get the following flow conservation equation

$$\sum_{i+j=s} \left[(1 - p_{v_2}) \cdot j + i\right] \cdot p_c \cdot f_{i,j} T = \frac{1}{T_x}. \tag{26}$$

To get the final equation we use the property $\sum_{i,j} f_{i,j} = 1$. By rearranging the sum, substituting (26) and using the logarithm approximation similar to Sections 2,4, we get the following

$$\sum_{i,j} f_{i,j} = \frac{1}{p_c T_x T} \cdot \sum_{s=x+1}^{N_m} \frac{1}{s} + p_{v_2} \cdot \sum_{i,j} \frac{j}{i+j} \cdot f_{i,j}. \tag{27}$$

For the second sum we approximate the average of the individual j to $i + j$ ratios by the global double-valid to total-valid ratio[1].

$$\simeq \frac{1}{p_c T_x T} \cdot \ln\left(\frac{2 \cdot N_m + 1}{2 \cdot x + 1}\right) + p_{v_2} \cdot \frac{DV}{U \cdot N_p}. \tag{28}$$

Finally, by substituting $1 = \sum_{i,j} f_{i,j}$ we get the (approximate) equation

[1]The approximation works well because the ratios are smaller than 1 and bounded away from 0.

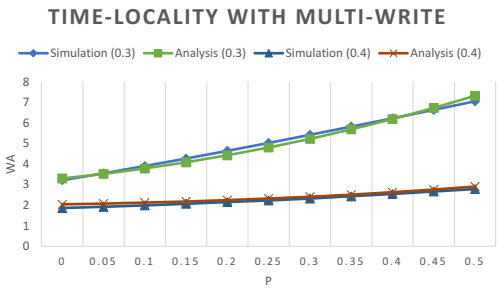

Figure 8: Analysis vs. simulation of the time-locality with multi-write setup for $\rho = \{0.3, 0.4\}$.

$$\frac{1}{p_c T_x T} \cdot \ln\left(\frac{2 \cdot N_m + 1}{2 \cdot x + 1}\right) = 1 - p_{v_2} \cdot \frac{DV}{U \cdot N_p}. \tag{29}$$

5.4 Final Solution for Multi-Write

Altogether, for the analysis of $t = 2$ multi-write architecture we have the 3 unknown variables T_x, DV, x, for which we use the 3 equations (17), (22), and (29). Once we numerically solve this system of equations, we obtain the WA as

$$WA = \frac{T_x + x}{T_x},$$

as prescribed in equation (18).

To test the accuracy of our analysis we compare the WA values from the solution of the 3 equations to the write amplification obtained through simulation of a storage device whose pages support $t = 2$ multi-writes. We use the parameters $\rho = \{0.3, 0.4\}$ and $r = 1.261$ (a capacity-achieving multi-write code for $t = 2$ writes). We plot the resulting values for $p \in [0, 0.5]$, shown in Figure 8. Now with multi-write too we see that the analysis is quite accurate, and that write amplification grows with the time locality. As before, time locality has a mixed effect on page invalidation, and now even more complex due to the effect of double-valid pages on the consumption rate of pages at the frontier. Therefore, it is necessary to solve the derived system of equations to find the true balance between the internal variables, which together determine the actual write amplification value.

6. ADVANCED MULTI-WRITE ARCHITECTURES

It has been shown in [Luojie 2012B] that having multi-write access for the entire storage space saves WA only when the over-provisioning is very high. To address that, in a previous work [Odeh 2014] we proposed two mapping architectures that use multi-write access more parsimoniously, and as a result reduce WA significantly for low over-provisioning as well. We thus wish to extend the multi-write time-locality WA analysis to these more practical architectures. As it turns out, for one of the architectures the analysis of Section 5 carries over with almost no change. Alas for the second architecture, which was shown in [Odeh 2014] to offer the best WA performance, we need substantial new ideas for the analysis. These two architectures are discussed in the two sub-sections now following.

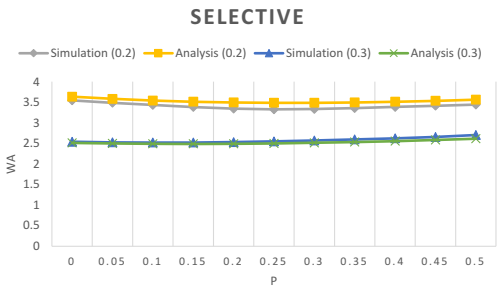

Figure 9: Analysis vs. simulation of the selective architecture for $\rho = \{0.2, 0.3\}$.

6.1 The Selective Architecture

The architecture called *selective* in [Odeh 2014] allocates pages with multi-write capability to all user writes, while GC writes are allocated regular no multi-write pages. Both types of pages reside in the same blocks. Consequently, pages written by the user can be updated in-place once before their erasure, while pages copied back in GC cannot be further updated in place. The rationale behind this architecture is that time locality in the workload renders user writes much more likely for update than "older" copied-back pages.

Comparing to the architecture with all multi-write capability analyzed in Section 5, the selective architecture manifests two minor changes. First, the number of free pages available for user writes in the frontier is larger, because the x copied-back pages do not consume the $r > 1$ expansion factor. Therefore, we now have

$$\bar{n} = \frac{N_p - x}{r}$$

(compare to (15)). Second, the mean number of v_2 pages in the beginning of the non-local epoch does not have a contribution from the x copied-back pages. Hence it changes to

$$N_{dv} = \bar{n} \cdot p_{v_2} \tag{30}$$

(compare to (19)). After applying those two simple changes, the analysis proceeds identically to Section 5 with the same equations. While simple conceptually, these changes induce a vast change on the device write-amplification performance. We present the analysis results versus the simulation results for $\rho = \{0.2, 0.3\}$, $r = 1.261$, and $p \in [0, 0.5]$. See Figure 9, and compare to the much worse results in Figure 8. Another good feature revealed by the analysis is that the performance of the selective architecture is much less sensitive to the time-locality level.

6.2 The Double-Front Architecture

The architecture called *double-front* in [Odeh 2014] applies a similar differentiation between user and GC writes as the selective architecture. The difference is that in double-front we send user-writes with multi-write, and GC writes without multi-write to *separate blocks*. In effect, the double-front architecture manages two write frontiers: one for user writes and one for GC writes – hence its name. The advantage of this architecture over the selective architecture is in reducing the dependence between user-written and GC-written pages

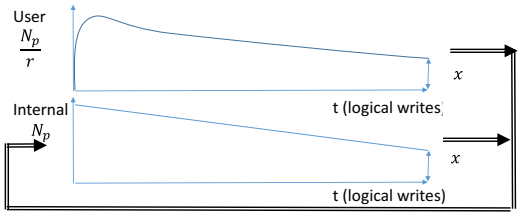

Figure 10: Double-front architecture: either a user or an internal block is selected for GC, and written to the internal-block frontier.

when choosing a block for cleaning, thus achieving lower WA in general. The disadvantages are the more complex management of two frontiers, and harder analysis. As in Section 5, we offer the option to jump directly to the final solution in Section 6.3, but recommend reading through to get valuable insights.

Blocks storing user-writes with multi-write are called *user-blocks* in the sequel, and the ones storing GC writes without multi-write are called *internal-blocks*. Each user-block in the user-frontier has space for N_p/r user writes, while each internal-block in the internal-frontier has space for N_p GC writes. Upon selection for cleaning, a physical block may change its function between being a user-block and an internal-block. For the sake of analysis, we model the collections of user-blocks and internal-blocks by separate Markov chains labeled M_{usr} and M_{int}, respectively. The Markov chain M_{int} is similar to the one used in Section 4 for a no multi-write architecture. The Markov chain M_{usr} is similar to the one used in Section 5 for an all multi-write architecture. So the main challenge here is the joint analysis of these two dependent Markov chains. In the notation of Section 4 we denote the fraction of blocks in state i of M_{int} by f_i^{int}. In the notation of Section 5 we denote the fraction of blocks in state i, j of M_{usr} by $f_{i,j}^{usr}$. Because M_{int} and M_{usr} partition the entirety of blocks, we have

$$\sum_i f_i^{int} + \sum_{i,j} f_{i,j}^{usr} = 1. \tag{31}$$

As we defined T_x in Section 5.1, we now define T_{usr} and T_{int} for M_{usr} and M_{int}, respectively: a new block becomes the frontier user-block every T_{usr} user writes, and a new block becomes the frontier internal-block every T_{int} user writes. Clearly T_{usr} and T_{int} are dependent through the single GC selection policy looking for the min-valid block among both user-blocks and internal-blocks. The relation between the two Markov chains is better understood with the illustration of Figure 10. But for now we can re-write the flow equations from Sections (4),(5) replacing T_x by T_{int} and T_{usr} , respectively

$$i \cdot p_c \cdot f_i^{int} \cdot T = \frac{1}{T_{int}}, \tag{32}$$

$$\sum_{i+j=s} [(1 - p_{v_2}) \cdot j + i] \cdot p_c \cdot f_{i,j}^{usr} T = \frac{1}{T_{usr}}. \tag{33}$$

Note that the values of p_c, p_{v_2} are unchanged from their earlier uses. We define $|M_{usr}| = \sum_{i,j} f_{i,j}^{usr}$ as the fraction of blocks functioning as user-blocks in the device. Then using (32) and the same logarithmic approximation from (2) we

get the following equation

$$\frac{\ln\left(\frac{2\cdot N_p+1}{2\cdot x+1}\right)}{p_c T_{int} T} = 1 - |M_{usr}|. \quad (34)$$

By using the same decomposition from (27) on (33), and skipping some technical details[2], we get

$$\frac{1}{p_c T_{usr} T} \cdot \ln\left(\frac{2\cdot N_m+1}{2\cdot x+1}\right) = |M_{usr}| - p_{v_2} \cdot \frac{DV}{U \cdot N_p}, \quad (35)$$

where N_m is adapted to reflect that in the user-block frontier there are only user writes

$$N_m = (p_{v_1} + p_{v_2}) \cdot \bar{n},\ \bar{n} = \frac{N_p}{r}. \quad (36)$$

Another required adaptation is for DV to reflect that there are only $T|M_{usr}|$ blocks with double-valid pages, and that these blocks are used only for user-writes

$$DV = \frac{N_{dv}}{p_c \cdot T_{usr}} \cdot \left[1 - e^{-p_c \cdot T|M_{usr}| \cdot T_{usr}}\right],\ N_{dv} = p_{v_2} \cdot \bar{n} \quad (37)$$

(compare to (22) and (19)). The next equation is obtained by carrying over (17) from Section 5.1, which gives

$$\frac{\frac{N_p}{r}}{T_{usr}} = \frac{1 - (1-p)\frac{DV}{UN_p}}{2 - p_{v_2}}. \quad (38)$$

Finally, the most interesting equation is the one tying together the two frontiers

$$x\left[\frac{1}{T_{usr}} + \frac{1}{T_{int}}\right] = \frac{N_p}{T_{int}}, \quad (39)$$

where the left-hand side is the rate of internal GC writes cleaning both user and internal blocks, and the right-hand side is the rate of pages becoming available for GC writes at the internal-block frontier.

6.3 Final Solution for Double-Front

Altogether, for the analysis of the double-front architecture we have the 5 unknown variables T_{usr}, T_{int}, $|M_{usr}|$, DV, x, for which we use the 5 equations (34), (35), (37), (38), and (39). Once we numerically solve this system of equations, we obtain the WA as

$$WA = \frac{N_p + T_{int}}{T_{int}},$$

since every T_{int} user writes a full N_p-page internal-block is consumed for non-user GC writes.

To test the accuracy of our analysis we compare the WA values from the solution of the 5 equations to the write amplification obtained through simulation of the double-front architecture. We use the parameters $\rho = \{0.2, 0.3\}$ and $r = 1.261$. We plot the resulting values for $p \in [0, 0.5]$, shown in Figure 11. The first thing to observe is that the analysis still nicely tracks the behavior of the device, but with more visible inaccuracies. This fact is justified by the

[2]The only new assumption we add to get the following is that $|M_{usr}|$ is also the fraction of valid *logical* pages in user-blocks, which is reasonable because the joint GC balances the valid-page counts.

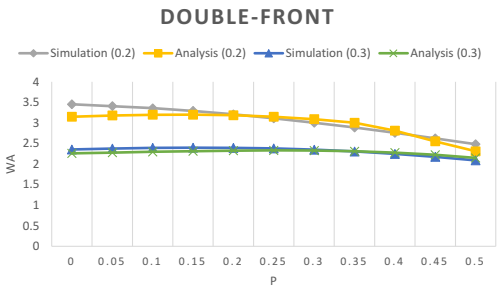

Figure 11: Analysis vs. simulation of the double-front architecture for $\rho = \{0.2, 0.3\}$.

complexity of the analysis in this case, which required some "heavier duty" approximations. The results show that the double-front architecture outperforms all previous architectures. Another advantage is that if we fix all the parameters of the setup, we may often see an improvement in the performance when increasing the time locality. This means that the double-front is an especially attractive architecture for workloads with inherent time-locality features.

7. CONCLUSION

We have developed write-amplification analysis for natural and novel setups with time locality and multi-write access. The main contributions are formally derived systems of equations that can be efficiently solved by storage developers and operators to predict the device performance in real systems. In addition, the analytical derivation has also provided insights on the inner workings of device architectures in the presence of time locality and multi-writes. For future work we identify the following interesting directions

1. Analyzing multi-write with more than two writes.
2. The analysis for sub-optimal garbage collection policies to lower the computational complexity of the garbage-collection algorithm.
3. Combining spatial-locality and hot/cold separation with time locality, to capture the most realistic workloads.

8. REFERENCES

[Agrwal 2010] Agarwal, Rajiv and Marrow, Marcus. A closed-form expression for write amplification in NAND flash, GLOBECOM Workshops (GC Wkshps), 2010 IEEE, pages 1846–1850, 2010. IEEE.

[Bux 2010] Bux, Werner and Iliadis, Ilias. Performance of greedy garbage collection in flash-based solid-state drives. Performance Evaluation, 67(11):1172–1186, 2010.

[Desnoyers 2012] Desnoyers, Peter. Analytic modeling of SSD write performance, Proceedings of the 5th Annual International Systems and Storage Conference, pages 12, 2012.

[Yadgar 2015] Yadgar, Gala, Yaakobi, Eitan and Schuster, Assaf. Write once, get 50\% free: Saving ssd erase costs using wom codes, USENIX FAST, 2015.

[Hu 2009] Hu, Xiao-Yu, Eleftheriou, Evangelos, Haas, Robert, Iliadis, Ilias and Pletka, Roman. Write amplification analysis in flash-based solid state drives, Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference, pages 10, 2009. ACM.

[Luojie 2012A] Luojie, Xiang and Kurkoski, Brian M. An improved analytic expression for write amplification in NAND flash, Computing, Networking and Communications (ICNC), 2012 International Conference on, pages 497–501, 2012. IEEE.

[Luojie 2012B] Luojie, Xiang, Kurkoski, Brian M and Yaakobi, Eitan. WOM codes reduce write amplification in NAND flash memory, Global Communications Conference (GLOBECOM), 2012 IEEE, pages 3225-3230, 2012. IEEE.

[Menon 1995] Menon, Jai. A performance comparison of RAID-5 and log-structured arrays, High Performance Distributed Computing, 1995., Proceedings of the Fourth IEEE International Symposium on, pages 167–178, 1995. IEEE.

[Odeh 2014] Odeh, Saher and Cassuto, Yuval. NAND flash architectures reducing write amplification through multi-write codes, Mass Storage Systems and Technologies (MSST), 2014 30th Symposium on, pages 1–10, 2014. IEEE.

[Robinson 1996] Robinson, John T. Analysis of steady-state segment storage utilizations in a log-structured file system with least-utilized segment cleaning. ACM SIGOPS Operating Systems Review, 30(4):29–32, 1996.

[Rosenblum 1992] Rosenblum, Mendel and Ousterhout, John K. The design and implementation of a log-structured file system. ACM Transactions on Computer Systems (TOCS), 10(1):26–52, 1992.

[Van 2013A] Van Houdt, Benny. A mean field model for a class of garbage collection algorithms in flash-based solid state drives, ACM SIGMETRICS Performance Evaluation Review, pages 191–202, 2013. ACM.

[Van 2013B] Van Houdt, Benny. Performance of garbage collection algorithms for flash-based solid state drives with hot/cold data. Performance Evaluation, 70(10):692–703, 2013.

[Van 2014] Van Houdt, Benny. On the necessity of hot and cold data identification to reduce the write amplification in flash-based SSDs. Performance Evaluation, 82:1–14, 2014.