Enabling Intra-Plane Parallel Block Erase in NAND Flash to Alleviate the Impact of Garbage Collection

Tyler Garrett
University of Pittsburgh
Pittsburgh, Pennsylvania
tmg61@pitt.edu

Jun Yang
University of Pittsburgh
Pittsburgh, Pennsylvania
juy9@pitt.edu

Youtao Zhang
University of Pittsburgh
Pittsburgh, Pennsylvania
zhangyt@cs.pitt.edu

ABSTRACT

Garbage collection (GC) in NAND flash can significantly decrease I/O performance in SSDs by copying valid data to other locations, thus blocking incoming I/O requests. To help improve performance, NAND flash utilizes various advanced commands to increase internal parallelism. Currently, these commands only parallelize operations across channels, chips, dies, and planes, neglecting the block level due to risk of disturbances that can compromise valid data by inducing errors. However, due to the triple-well structure of the NAND flash plane architecture, it is possible to erase multiple blocks within a plane, in parallel, without diminishing the integrity of the valid data. The number of page movements due to multiple block erases can be restrained so as to bound the overhead per GC. Moreover, more capacity can be reclaimed per GC which delays future GCs and effectively reduces their frequency. Such an Intra-Plane Parallel Block Erase (IPPBE) in turn diminishes the impact of GC on incoming requests, improving their response times. Experimental results show that IPPBE can reduce the time spent performing GC by up to 50.7% and 33.6% on average, read/write response time by up to 47.0%/45.4% and 16.5%/14.8% on average respectively, page movements by up to 52.2% and 26.6% on average, and blocks erased by up to 14.2% and 3.6% on average. An energy analysis conducted indicates that by reducing the number of page copies and the number of block erases, the energy cost of garbage collection can be reduced up to 44.1% and 19.3% on average.

CCS CONCEPTS

• **Information systems → Flash memory**; • **Hardware → Memory and dense storage**;

KEYWORDS

NAND Flash, Garbage Collection, Storage

1 INTRODUCTION

SSDs have become a staple choice for storage in servers and data centers over the last decade [7]. As the industry and marketplace have shifted towards a heavier reliance on services such as the adoption of cloud computing, it is important to ensure the performance of SSDs keep up with the demand of the system. One of the biggest issues that exists in SSDs is the need to perform garbage collection (GC) in order to reclaim pages that no longer contain useful data. This operation is related to the organization of the internal components of the SSD, as depicted in Figure 1.

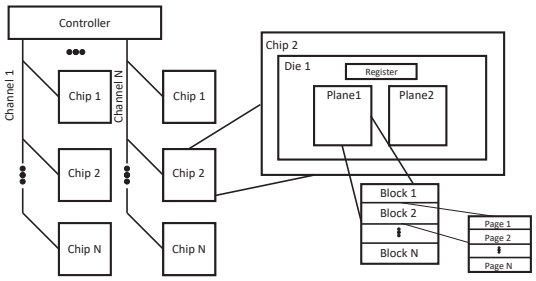

Figure 1: SSD Internal Architecture

Inside the SSD is an array of NAND flash chips. Several chips are strung together sharing a channel over which read, write, and erase commands, along with their corresponding data, can be sent. The array typically consists of multiple channels that can operate in parallel with each other. Inside each chip are one or more dies. Within each die are two planes (sometimes four) which are divided into blocks, while the blocks are further divided into pages. Read and write operations occur at the page granularity, while erase operations occur at the block granularity. The difference in the scope of these operations is because erasing a single page, given the structure of NAND flash, would cause a disturbance to all other pages in within the block, thus leaving the data unreliable[2]. When data in a page needs to be updated, a direct overwrite of the page cannot be performed as NAND flash requires that the page must first be erased. However, as an erase occurs at the block granularity, all other pages in the block are also erased causing data loss. For this reason, SSDs invoke an out-of-place write policy[8]. When a page's data is updated, the Flash Translation Layer (FTL) remaps the logical page of the old data to a new physical page while the original page is marked as invalid leaving two copies: 1) a valid page that holds the new data and 2) an invalid page that holds the out dated data.

As the SSD accumulates more data it eventually needs to reclaim the space being taken up by the invalid pages. GC is triggered when the number of free pages within a plane drops below a given threshold. When a block is selected for GC it may still contain some valid pages. In order to ensure these pages are not lost, the valid data is first copied to another block in a different part of the SSD. After all copies are created, the previously valid pages are

ISLPED '18, July 23–25, 2018, Seattle, WA, USA

ACM ISBN 978-1-4503-5704-3/18/07…$15.00
https://doi.org/10.1145/3218603.3218627

marked invalid, along with the rest of the block, allowing the erase operation to take place.

The internal movement of data and erasing latency have a negative impact on the response times to service read and write requests. In order to improve performance SSDs rely on the parallelization of operations. Given the internal structure of the SSD, parallelism can be achieved at the channel, chip, die, and plane levels. Commands can be dispatched over all channels and then on all chips simultaneously. To further parallelize within the chip itself, advanced commands are utilized. The Multi-Die Interleave command allows for the same operation to be performed on all dies of a chip at the same time. Take a step deeper and the Multi-Plane advanced command executes the same command on both planes of a die [3]. However, within the plane itself there are currently no commands to parallelize operations.

To try and mitigate the impact GC has on the system this paper proposes a new advanced command called Intra-Plane Parallel Block Erase (IPPBE) that leverages the structure of the NAND flash plane to parallelize the erase operations. This command brings parallelism a step deeper to the block-level. By reclaiming more space in the given erase latency, it is possible to reduce the number of times GC is triggered, thus reducing the response times of I/O requests and the number of valid pages copied. Experimentation shows that read/write response times can be reduce by up to 47.0%/45.4% and 16.5%/14.8% on average respectively and the number of page copies by up to 52.2% and 26.6% on average. An analysis of energy consumption was also investigated and shows that energy used by GC operations can save up to 44.1% and 19.3% respectively.

2 MOTIVATION AND RELATED WORK

The copying of valid pages and the erase operation during GC impose a performance and endurance burden on the SSD. As the demand for SSDs to have more space continues, more bits are being stored per cell. However, by doing so there is a decrease in the lifetime of each page causing the number of tolerable writes prior to wearing out to decrease significantly. The extra writes that occur during GC can accelerate this degradation process and decrease the overall lifetime of the SSD.

When GC takes place in a plane, it stalls incoming I/O requests to itself and other planes in the die [11]. This stalling happens because the planes of the die share a common register. **(1)** As the contents of the valid pages are being copied out, the data must first be read from the page to the die register. **(2)** Then the data is transferred out via the channel to the channel controller where ECC is used to correct any errors[13]. During this transfer time the channel is also blocked. **(3)** Next, the data is sent to the die register of its new physical address and **(4)** then written to a clean page. The blocking of the channels and dies delays other requests from being serviced along these paths. The erase operations is also an issue as it is the longest single operation on the order of several milliseconds. Figure 2 illustrates this process.

Prior works have attempted to address these issues in different ways and can be generally summarized into the following three categories: 1) Decrease the latency of GC [2], 2) Decrease the frequency of/delay GC [4, 11], and 3) Try to circumvent GC when it is

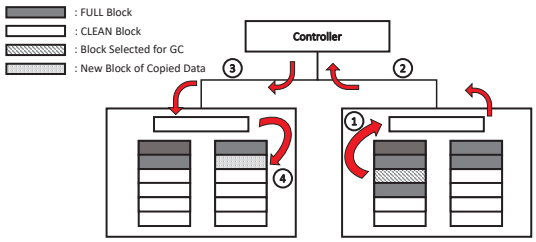

Figure 2: Garbage Collection Process

happening [13]. In [13] another advanced command is used called Copyback where instead of pages being moved out over the channel and dispatched to a new physical page, they are read to the register and written to a page of a different block within the same plane. This frees up the channel, however, by not sending the data over the channel to the controller the data does not run through ECC at the controller. In addition, [13] utilizes RAIN to build in redundancy so that in the event a request is blocked by a plane performing GC the parity can be used to generate the blocked data. Unfortunately, the need for redundancy requires additional capacity. [2] uses a different approach to try and decrease the latency of GC by erasing portions of a block called sub-blocks. These sub-blocks require some pages (or wordlines) to serve as buffers to prevent the erasing of sub-blocks from disturbing data in the other sub-blocks of the block. Pages serving as buffers cannot hold any data and therefore reduce the capacity of the SSD. Dividing the block up in this way means it is easier to find a section to reclaim with few valid pages to move. However, it was found that the erase latency of erasing a single sub-block is essentially equal to that of the latency to erase the entire block. This is due to the fact that the erasing process does not change by having less pages. Therefore, less space may be cleared given the same amount of time. As previously stated, when a plane goes through the GC procedure it occupies the register when copying pages, thus blocking the other plane on the die from servicing incoming requests. [11] decides to make use of this idle time that the other plane experiences by running GC on both by using the Multi-Plane advanced command to move pages and erase in parallel. The idea is that performing GC early in the other plane will delay the need for GC in the future. Although, this method has its limitations. Due to the nature of the Multi-Plane command these operations must share the same block address and page index. For the read and write commands the pages being read or written to must share an index, otherwise they will be moved one at a time, serially. Additionally, the plane that needs to reclaim space will select the candidate block with the least number of valid pages in order to copy less pages. However, the Multi-Plane command requires that the block in the other plane be the same block index and this block may have many valid pages to move, worsening the latency of GC. Endurance may also be harmed as the adjacent plane may not need to reclaim space yet and therefore the page movements and erasing are unnecessarily speeding wearing.

The GC problem is not one that exists equally at all points in the lifetime of the SSD. It is a by-product of SSD aging and more data being stored/updated overtime. The issue worsens as blocks and pages wear out and are replaced with a finite amount of built-in spares. In particular, once GC begins to happen it will occur much

more frequently. Since GC is based on a threshold of how many free pages remain in a plane, the GC process fights to get back over the acceptable threshold. However, if reclaiming one block is not enough then it must continue to serially erase blocks until enough space as been reclaimed.

Since SSDs have become a prominent storage media, it is advantageous to find solutions that do not impose drastic changes to the internal architecture that may require changes to the infrastructure of the system. With these previous designs in mind, this paper seeks to address GC by leveraging the NAND flash structure as it is without making sacrifices to capacity and maximize the efficiency of each GC. Intra-Plane Parallel Block Erase is able to meet these requirements as well as be combined with existing designs to further enhance them.

3 INTRA-PLANE PARALLEL BLOCK ERASE

3.1 Overview

Intra-Plane Parallel Block Erase, or IPPBE as it will be referred to throughout the rest of this paper, is, to the best of the author's knowledge, the first advanced command to parallelize at the block-level within the same plane. Its design is based on exploiting the triple-well structure shared by all blocks residing in a plane[8, 12]. The well plays an important role during the erase operation. As the well is biased with a high voltage, the wordlines of the block being erased are grounded to allow the electrons stored in each cell to be flushed out. The blocks that are not being erased float their wordlines. Blocks are selected via a block decoder in the plane. By redesigning the decoder to select multiple blocks at a time, multiple blocks can be grounded during the erase operation and therefore be erased simultaneously. Erasing more than one block at a time increases the space reclaimed during an erase operation without increasing the erase latency. In addition, future GCs will be delayed due to more space being reclaimed per GC. IPPBE can also accelerate the process of getting a plane's free page count back over the GC barrier so that it can resume servicing reads and writes sooner.

3.2 Erase Operation

To understand IPPBE's design it is important to understand the erase operation's protocol. [2] found that erasing portions of a block did not reduce the latency of the erase operation. This makes sense as the steps taken during the erase operation do not change regardless of the number of pages in a block. [8] describes this process in detail and Figure 3 provides a visual for the steps taken to erase a block.

After all the data has been copied out, the erasing can being. First, all the wordlines of the selected block are programmed to get each cell into the same state. Next, the wordlines of the selected block are grounded and the remaining blocks are left floating. From this point the process enters a loop of electrical pulses and verification. The iP-well, common to all blocks in the plane, receives an electrical pulse of V_Erase. After which, all worldlines are read to verify that the cells have been erased and read as 1. If not, V_Erase is increased by V_step and the electrical pulse is once again applied to the iP-well. This process continues until it can be verified that all cells have been successfully erased. It is normal for erase pulses to be applied several times before an erase is successful. It should

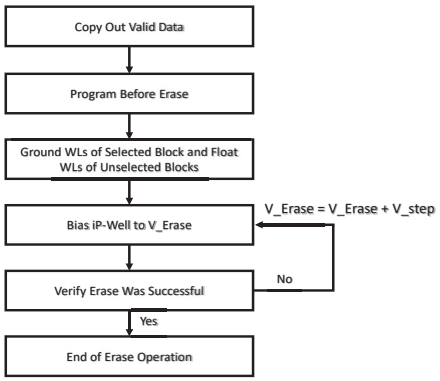

Figure 3: Erase Operation Procedure

be noted that there is a limit to the number of times this loop can run before the erase is deemed a failure.

This procedure is the same regardless if the number of pages per block is 64 or 256. Rather than reducing the size of the erase unit to a portion of a block, IPPBE does the opposite by adding more blocks. By doing so, multiple blocks can be reclaimed in the same amount of time it takes to erase a single block, maximizing the erase operation.

3.3 NAND Flash Plane Triple-Well

In this design the triple-well structure is not altered, however, it is the key component of enabling IPPBE. In NAND flash a triple-well structure is used to construct the plane. There are three sections that the well is divided into: 1) iP-well, 2) N-well, and 3) P-type substrate [6, 8, 12]. As mentioned prior, the iP-well receives the high voltage electrical pulse during the erase operation. When a block being erased has its wordlines grounded during this pulse, the electric potential between the wordlines and the iP-Well induces Fowler-Nordheim tunneling, enabling the trapped electrons in each cell to be flushed out[8]. It is not desirable to have all blocks experience this electron tunneling. Since this iP-well is shared by all blocks in each plane, the other blocks need to inhibit their wordlines to prevent unexpected erasure and loss of data. To do so, blocks not selected for erasure leave their wordlines floating. This maneuver raises the capacitive coupling so that the electric potential between the wordlines and the well is not sufficient to induce electron tunneling. IPPBE takes advantage of this phenomena. Rather than instructing only one block to ground its wordlines, multiple can be instructed to ground their wordlines during the electric pulse. Since the biasing of the well has a duration that lasts a fixed interval of time, then erasing more than one block does not result in additional time spent performing the operation. An energy analysis was conducted using the FlashPower simulator and shows that for every block added during this erase operation the energy expended grows linearly[9]. However, by using IPPBE, GC is called less frequently and therefore overall energy consumption is decreased. This is discussed further in the Section 5.

3.4 Decoder and Command Addressing

When the SSD decides which block will be erased a block decoder is used to set the wordlines. However, given that in IPPBE's design

additional blocks are necessary, a slight modification needs to be made to the decoder to allow multiple blocks to be addressed at the same time. [3] notes that the addressing for operations is simply channel, package, chip, die, plane, block, page, and sub-page. Since the erase operation is not concerned with pages or sub-pages these bits could be repurposed to add the address of other blocks being addressed simultaneously. IPPBE addressing could potentially be modified to combine with and enhance previous GC schemes. For instance, IPPBE could be altered to address multiple sub-blocks across different blocks within a plane. In addition, if combined with the Multi-Plane command multiple block erasure could occur simultaneously within a plane as well as in other planes belonging to the same die.

3.5 Block Selection

Prior to IPPBE, determining which block is best to reclaim was fairly simple. The most traditional approach is termed Greedy GC, which means the block selected was the one with the largest number of invalid pages [8]. The idea being that by selecting the block with the most invalid pages there would be less valid pages that need to be copied out. This way the most space possible is reclaimed when erased. However, with IPPBE careful attention should be made to selecting which blocks to reclaim. For simplicity, consider selecting two blocks to erase. The plane free page count is reduced to the point of triggering the need for GC. Within the plane, two blocks have all invalid pages except for two valid pages each. IPPBE can erase both blocks after all four valid pages are moved out prior. In this case a few extra pages are moved initially to reclaim twice as much space. This will delay the need for GC longer than erasing only one. However, there is also a scenario where GC is triggered and there are many valid pages in the plane. In this case a block that is entirely invalid may be selected, however, the block with the second least number of valid pages only has 60 percent of its pages invalid. This means that, assuming no free pages in the block, 40 percent of the pages must be copied out. In this case it is better to only erase one block. If both were selected to be reclaimed, the time spent moving all the valid pages would counter-act the benefit of simultaneously erasing blocks.

To solve this problem, thresholds are introduced to limit the number of pages being copied. The SSD first finds the best block to erase similar to Greedy. Then, it finds the second best block. Next, an analysis is done to determine if the number of pages that would need to be copied are below the threshold. If yes, then both can be reclaimed. If not, then the second block is not selected to be reclaimed.

3.6 Garbage Collection Barrier

One of the issues with GC is that once it starts to occur it typically will continue to happen. The trigger is usually the passing of a threshold that brings the number of free pages below a given percentage of the total number of pages in the plane. Once this happens, space must be reclaimed continuously until the number of free pages has crossed back over the threshold, or Garbage Collection Barrier (GCB). However, once enough space is reclaimed the plane can return to servicing read and write requests. Because not enough time may be provided to claim more space, there may

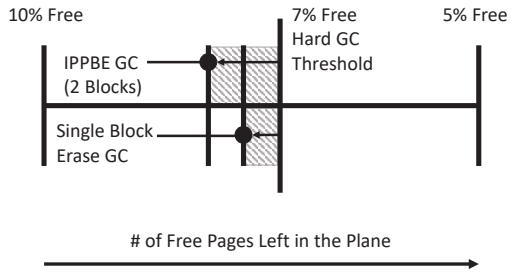

Figure 4: Comparison of IPPBE and Single Block Erase with Garbage Collection Triggered at 7 Percent Free Page Count

be thrashing between the plane reclaiming space and writing to the plane. IPPBE helps to prolong this need to return to GC by reclaiming more space and further distancing the number of free pages from the GCB. Figure 4 illustrates this concept. On the other hand, there may be a large write that pushes the free page count far past the barrier. IPPBE can accelerate the process of returning to the other side of the GCB by reducing the number of erase operations it takes to get there. Figure 5 visualizes IPPBE's implementation.

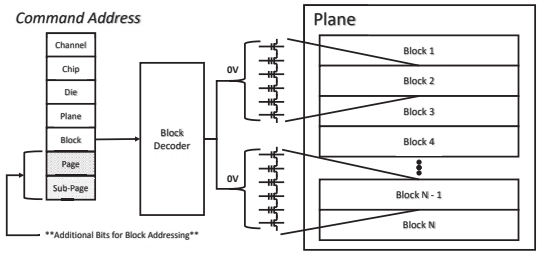

Figure 5: IPPBE Design

4 EXPERIMENTAL SETUP

IPPBE performance was evaluated using a combination of two different simulators *SSDSim* [3] and *FlashPower* [9]. SSDSim is an event based simulator that uses traces[10] of I/O requests to track read, write, and erase operations through a desired SSD configuration. The configurations for the evaluation of this paper can be seen in Table 1. SSDSim's GC algorithm was modified to implement IPPBE's design by introducing additional mechanisms to find multiple candidate blocks to reclaim. Timing schemes also received alterations to ensure that erasing multiple blocks only occupies the channel and chip the same way as erasing a single block. For the purpose of this work the maximum number of blocks erased at a time was limited to two. A static allocation scheme was implemented, prioritizing in the following order: channel, chip (also referred to in some works as way), die, and plane with channel being highest and plane being lowest. Since not every trace is the same size the SSD needs to be aged to force GC to occur. SSDSim does this by injecting invalid pages into the SSD. However, this aging process has been found to not be consistent with how SSDs truly age [5]. To solve this issue, the configuration used is relatively small. This allows GC to occur naturally rather than being artificially imposed. GC was set to trigger when the number of free pages in a plane dropped below 7%. FlashPower was used to evaluate the the energy consumption of the design. Given the same

configuration as in Table 1 the simulation can output the energy consumption of the read, write, and erase operations. Both of these simulators have been validated against the real device.

SSD Configuration	
Channels: 2	Read Latency: 75us
Chips Per Channel: 2	Write Latency: 1.5ms
Dies Per Chip: 1	Erase Latency: 3.8ms
Planes Per Die: 2	Channel Transfer: 25ns
Blocks Per Plane: 2048	Overprovision: 25%
Pages Per Block: 64	GC Threshold: 7%
Page Size: 2KB	

Table 1: Simulation Configurations

5 RESULTS

Evaluation of IPPBE's design is divided into endurance, performance, and energy. For comparative purposes Greedy GC is used as a baseline for which IPPBE is normalized against. Each scheme was evaluated using a suite of traces from [10].

5.1 Endurance

The SSDs lifetime is a growing concern especially given that SSDs using MLC or TLC designs to hold multiple bits per cell have a significant reduction in the lifetime of pages due to accelerated wear-out[1]. GC worsens this problem by requiring pages be moved thus inducing extra writes. IPPBE tries to limit this by reducing the number of pages movements and block erases. Figure 6 and Figure 7 compare the total number of pages moved and blocks erased during GC. Experimentation shows that the number of page movements are reduced by up to 52.2% and 26.6% on average and the reduction in the number of blocks erased in some traces can be up to 14.2% and 3.6% on average. When IPPBE is applied, both blocks move their pages together then perform the erase. Since no operations occur in between moving pages of the different blocks when using IPPBE, the pages are likely written to the same active block. Data in pages that are being moved are most likely read intensive data since they are among the last few pages yet to invalidate in the block. These frequently read pages are less likely to be updated and may need moved in a later GC. The more fragmented the valid pages are across the plane, the likelihood of needing to copy data during the next GC increases. IPPBE naturally clusters this type of data into the same block. As a result the next GC on the plane moves less pages because there is a higher chance of being able to select a block with less valid pages. The traces able to reduce page copying, due to clustering of valid pages, saw a decrease in blocks erase because IPPBE was able to reclaim more blocks with a larger number of invalid pages, thus increasing the amount of space reclaimed per erase. This increase in erase efficiency can accumulate overtime leading to less blocks needing to be erased because enough space has already been reclaimed. This is more apparent in larger and longer traces such as *prxy0* and *prn0*. This reduction in page copies and block erasures helps to alleviate the endurance burden imposed by GC.

5.2 Performance

GC is detrimental to performance of SSDs because when required it takes priority over all other requests to the die. Therefore, it is

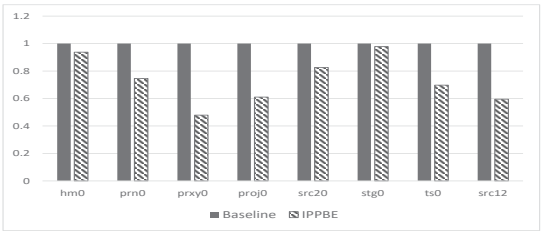

Figure 6: Valid Pages Moved During Garbage Collection

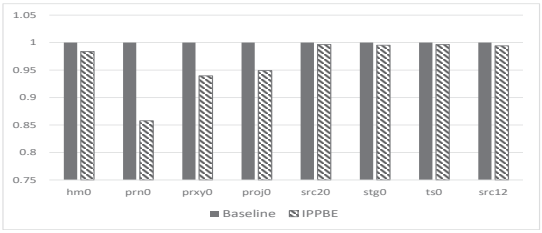

Figure 7: Blocks Erased

advantageous to limit the time spent reclaiming space. Figure 8 demonstrates that IPPBE can reduce the total time spent performing GC by up to 50.7% and 33.6% on average. The reduction is very closely related to the number of times the GC mechanism is invoked. Figure 9 helps show the relationship between the number of times GC is triggered and the time spent on GC. GC frequency is reduced up to 56.4% and 46.7% on average. The reason being is IPPBE can more quickly get back to the other side of the GCB by reclaiming more space per unit of time. However, reducing the frequency of GC operations does not always result in a linear reduction in time spent on GC. The reduction in time spent can be attributed to its heavy reliance on the ability to reduce the number of valid page copies. For instance, *stg0's* reduction in page copies is the smallest and in turn sees the smallest reduction in time spent on GC. This is because the cumulative latency of moving pages is often worse than that of the erase operation. Nevertheless, being able to efficiently erase and move pages leads to a long term reduction. While it is possible the initial use of IPPBE could see a longer GC latency due to page movements, this eventually leads to high efficiency of GC thus limiting its need and time spent performing the operations associated. Figures 10 and 11 demonstrate that reducing the time spent on GC has a ripple affect on the overall average read/write response times of the trace. The average read response time is reduced up to 47.0% and 16.5% on average. The average write response time also decreases by up to 45.4% and 14.8% on average. The results vary based on how large of a portion the baseline's time is spent on GC.

5.3 Energy Analysis

An energy analysis was conducted to investigate the energy impact of using the IPPBE command. The FlashPower simulator was used to calculate the energy consumption of the read, write, and erase operations given the SSD configuration provided in Table 1. The results indicated the following energy consumptions: *Read* = 2.1uJ; *Write* = 12.3uJ; and *Erase* = 22.5uJ. Calculation of energy consumption for GC was as follows: Each page copy requires a

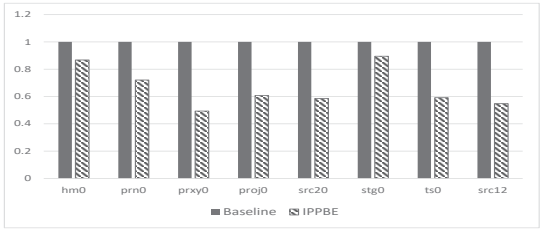

Figure 8: Total Time Spent on Garbage Collection

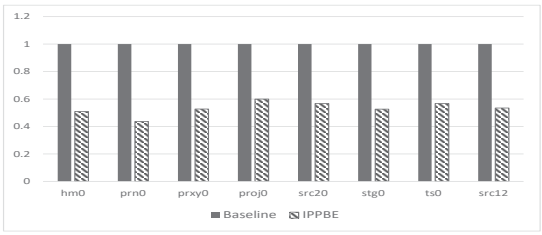

Figure 9: Garbage Collection Calls

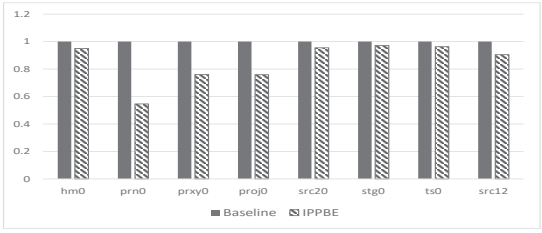

Figure 10: Average Trace Write Response Times

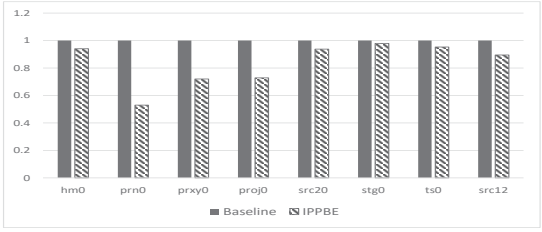

Figure 11: Average Trace Read Response Times

read and write command so the energy to move pages during GC is the sum of the read and write energy multiplied by the number of pages moved. Then, this value is summed with the product of the erase energy and number of blocks erased. Using FlashPower and [9] it can be concluded that the energy consumption of adding an additional block during the erase pulse causes the erase energy to increase linearly as the majority of the energy is consumed by the bitlines. Figure 12 shows that the savings in energy expended by GC can be as high as 44.1% and 19.3% on average. The results are a by-product of the reduction of pages moved and blocks erased in each trace. The GC energy savings most closely resembles the results of the pages movements because there tends to be significantly more pages moved than blocks erase, therefore the write energy becomes the dominate contributor. It should be noted that these values change based on the size of the plane.

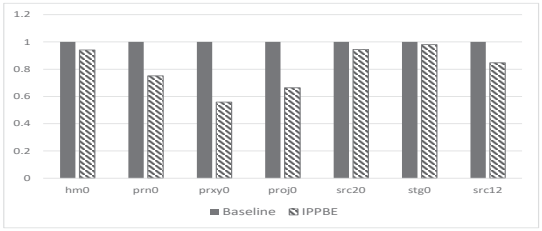

Figure 12: Garbage Collection Energy Analysis

6 CONCLUSIONS

Intra-Plane Parallel Block Erase (IPPBE) is the first advanced command, to the best of the authors knowledge, to bring parallelism to the NAND flash block-level. By taking advantage of the well structure of the plane and modifying the block decoder, IPPBE can improve endurance, performance, and energy consumption by reducing the time spent performing GC. IPPBE alleviates GC's impact by reclaiming more space per erase operation and clustering valid pages in the same block to reduce future page copies. IPPBE can also be modified to operate in conjunction with previous GC design schemes for further enhancement.

REFERENCES

[1] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu. 2017. Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives. *Proc. IEEE* 105, 9 (Sept 2017), 1666–1704. https://doi.org/10.1109/JPROC.2017.2713127

[2] T. Y. Chen, Y. H. Chang, C. C. Ho, and S. H. Chen. 2016. Enabling sub-blocks erase management to boost the performance of 3D NAND flash memory. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/2897937.2898018

[3] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Shuping Zhang. 2011. Performance Impact and Interplay of SSD Parallelism Through Advanced Commands, Allocation Strategy and Data Granularity. In *Proceedings of the International Conference on Supercomputing (ICS '11)*. ACM, New York, NY, USA, 96–107. https://doi.org/10.1145/1995896.1995912

[4] Myoungsoo Jung, Ramya Prabhakar, and Mahmut Taylan Kandemir. 2012. Taking Garbage Collection Overheads off the Critical Path in SSDs. In *Proceedings of the 13th International Middleware Conference (Middleware '12)*. Springer-Verlag New York, Inc., New York, NY, USA, 164–186. http://dl.acm.org/citation.cfm?id=2442626.2442638

[5] J Kim, M Park, and I Shin. 2017. Improving SSD Simulator for High Reliability of Performance Evaluation. *International Journal of Applied Engineering Research* 12, 15 (2017), 4836–4839.

[6] Jin-Ki Kim. 2010. Partial block erase architecture for flash memory. (Sept. 28 2010). US Patent 7,804,718.

[7] Rino Micheloni. 2016. *3D flash memories*. Springer, Dordrecht, [Netherlands].

[8] Rino Micheloni, Luca Crippa, A. Marelli, and Inc Books24x7. 2010;2014;. *Inside NAND Flash memories* (1st ed.). Springer, Heidelberg;New York;.

[9] V. Mohan, T. Bunker, L. Grupp, S. Gurumurthi, M. R. Stan, and S. Swanson. 2013. Modeling Power Consumption of NAND Flash Memories Using FlashPower. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 7 (July 2013), 1031–1044. https://doi.org/10.1109/TCAD.2013.2249557

[10] Microsoft production server traces. 2008. http://iotta.snia.org/traces/list/BlockIO

[11] N. Shahidi, M. T. Kandemir, M. Arjomand, C. R. Das, M. Jung, and A. Sivasubramaniam. 2016. Exploring the Potentials of Parallel Garbage Collection in SSDs for Enterprise Storage Systems. In *SC16: International Conference for High Performance Computing, Networking, Storage and Analysis*. 561–572. https://doi.org/10.1109/SC.2016.47

[12] Kang-Deog Suh, Byung-Hoon Suh, Young-Ho Lim, Jin-Ki Kim, Young-Joon Choi, Yong-Nam Koh, Sung-Soo Lee, Suk-Chon Kwon, Byung-Soon Choi, Jin-Sun Yum, Jung-Hyuk Choi, Jang-Rae Kim, and Hyung-Kyu Lim. 1995. A 3.3 V 32 Mb NAND flash memory with incremental step pulse programming scheme. *IEEE Journal of Solid-State Circuits* 30, 11 (Nov 1995), 1149–1156. https://doi.org/10.1109/4.475701

[13] Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A. Chien, and Haryadi S. Gunawi. 2017. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs. In *15th USENIX Conference on File and Storage Technologies (FAST 17)*. USENIX Association, Santa Clara, CA, 15–28. https://www.usenix.org/conference/fast17/technical-sessions/presentation/yan