A Hotness-aware Write Buffer Management Scheme for the Lifetime Extension of Flash-based Solid State Drives

Pingguo Li

***Abstract*—Flash-based solid state disks(SSDs) provide high I/O performance, low energy, and shock resistance. However, random writes limit its application in the update intensive applications such as Web services. Existing buffer management schemes usually employ a hot page detection mechanism to reduce random writes. Since the mechanisms usually take reuse distance or frequency as a parameter to identify frequently updated page/block, which become inefficient in handling fast changing workloads. In this paper, an efficient buffer management scheme, named HWB, is proposed, which reduces random writes by identifying frequently updated page/block. To improve both buffer hit rate and data sequentiality, HWB divides write buffer space into a hot region, a warm region, and a cold region. Frequently updated date is stored in the hot and warm region at page granularity, meanwhile infrequently updated date is kept in the cold region at block granularity. A replacement and flush policy, which combined with a temperature predictor, is designed to improve buffer hit ratio and reduce random write. HWB is evaluated using extensive trace-driven simulations. Its evaluated results show that HWB significantly reduces random write and extends SSD lifetime compared to existing buffer management schemes.**

***Index Terms*—Lifetime extension, Buffer partitioning model, Temperature predictor, Buffer management scheme**

I. INTRODUCTION

NAND flash-based solid state drives(SSDs) have been widely deployed, ranging from portal devices to enterprise storage systems. SSDs offer high I/O performance, low energy consumption, and shock resistance[1], however, they suffer from the overhead of small random writes due to the physical nature of NAND flash memory such as the erase-before-write, and limited endurance. Random write could incur more erase operations, while its erase count is limited for a given block. For example, the erase limit of multi-level cell (MLC) is about 10^3, and that of triple-level cell (TLC) is only about 10^2 [2]. NAND flash memory should be worn out quickly if it is left untreated.

To overcome this issue, build-in RAM such as DRAM is widely employed in SSDs to absorb repeated write requests and change access patterns for NAND flash memory. If the size of a DRAM is configured as 0.1% to 0.3% of its back-end magnetic disk storage, the disk can fully leverage buffer and greatly reduce random writes[3], however, it is unfeasible to increase the capacity of a single-chip DRAM duo to energy consumption and cost. A manufacture normally integrates a relatively small-size DRAM to an SSD. Some classic buffer management algorithms such as LRU, LFU and ARC[4] are proposed to overcome the disadvantageous influence caused by limited buffer space. Since the natural characteristic of NAND flash memory is rarely considered, they become ineffective once the working set size of a workload is more significant than the capacity of the buffer space.

Recently, several buffer management approaches are proposed to prolong the lifetime of an SSD.CFLRU[5] divides buffer into a working region and a clean-first region, and always selects clean pages in the clean-first region as victims over dirty pages unless there are not any clean pages. Based on CFLRU, CFDC[6] divides the clean-first region into a clean page region and a dirty-page region to reduce the search cost. The proposed mechanisms try to reduce random write by discarding clean pages first. However, they become invalid when write requests prevail over reading requests in a workload, because they cannot efficiently identify whether a dirty page is cold or not, and class hot pages may be mistaken as class cold pages and be evicted prematurely, which increase erase operations to NAND flash memory. Note that class cold pages are infrequently updated pages, and class hot pages are frequently updated pages. Random write usually includes class cold and hot pages. LRU-WSR [7] identifies if a dirty page is cold or not by a reuse distance threshold, and only class hot pages can be stored in buffer. Note that a re-use distance represents the number of distinct pages between two consecutive accesses to the same page in a request sequence. Like LRU-WSR, BPAC[8] also identifies whether a dirty page is cold or not based on a re-use distance. The difference is that BPAC sets up two re-use distance thresholds: PIRD and BIRD. To avoid the disadvantageous influence caused by random writes, the write buffer is divided into a page region, and a block region. When the page region is full, the dirty page with over PIRD is moved to the block region. If the block region is also full, the block with over BIRD is evicted and written to NAND flash memory. The proposed mechanisms identify whether a dirty page/block is cold or not by a re-use threshold, and only pages/blocks within the threshold can be stored in buffer to improve buffer hit ratio and reduce random writes, however, they become inefficient in handling fast-changing workloads because the re-use distance of a page/block usually fluctuates frequently over time, which is observed in Section II.

In this paper, we propose a novel write buffer management

Pingguo Li is a lecturer of the College of Biomedical Engineering, Hubei University of Science and Technology, Xianning 437000, P. R. China (e-mail: lpg@hbust.edu.cn)

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

scheme, called HWB, to extend the lifetime of an SSD by identifying frequently updated pages/blocks. We first analyze the update behavior of dirty pages in several real-world workloads, and then provide our observations. Based on the observations, HWB is proposed to reduce erase operations due to random writes. Our experiments on simulation show that proposed HWB significantly reduces random writes, and prolongs the lifetime of an SSD over the state of the art schemes under enterprise workloads.

We made the following contributions in the paper.

We analyze the update behavior of dirty pages in real world workloads from[9][10],and find that the lifetime of an SSD are considerably related with class FS pages and class FL pages. Keeping them in buffer can extend SSD lifetime significantly.

We propose a novel buffer management scheme, called HWB, which includes a temperature prediction model, a buffer partitioning policy and a replacement and flush policy. The temperature prediction model is used to compute the temperature value of a dirty page/block. The buffer partitioning policy adjusts dynamically region size according to workload access pattern. The replacement and flush policy decides where a page/block should be inserted into in buffer and which dirty page/block should be evicted first when buffer space is not free.

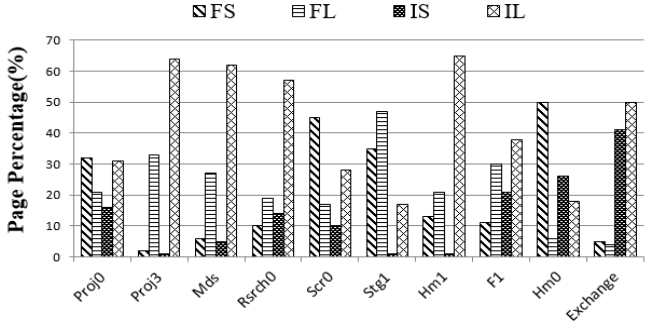
We evaluate HWB using extensive trace-driven simulations. Our evaluation results show that HWB significantly reduces random writes, and prolongs SSD lifetime compared to existing buffer management schemes.

II. OBSERVATION AND MOTIVATION

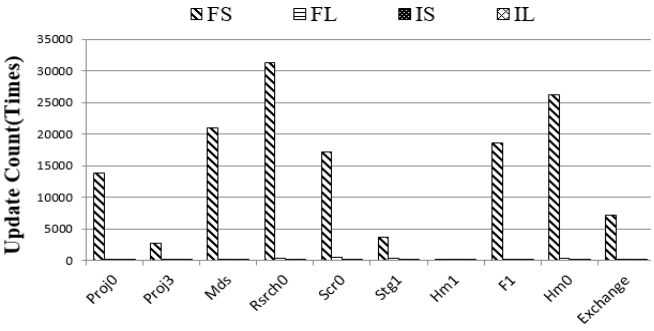
A. Observation

In this section, we study a wide range of real-world workloads from [9][10] to identify the update behavior of dirty pages. According to [11], dirty pages are also divided into class FS, FL, IS, and IL pages. Class FS and FL pages will be more likely to be updated again in the future than class IS and IL pages. Buffer pollution is mainly caused by class IS and IL pages.

We plot the percentage and the update frequency of class FS, FL, IS, and IL pages in ten workloads in Fig.1.As shown in Fig.1(a),they show different distribution characteristics in every workload. For Exchange, class FS, FL, IS and IL pages are made up of 5%,4%,50% and 41% respectively, and the total number of class IS and IL pages is ten times more than that of class FS and FL pages, while for stg1, class FS, FL, IS and IL pages are made up of 35%, 47%, 1% and 17% respectively, and the total number of class FS and FL pages is 4.6 times more than that of class IS and IL pages.



(a) the Percentage of Class FS, FL, IS, and IL Pages

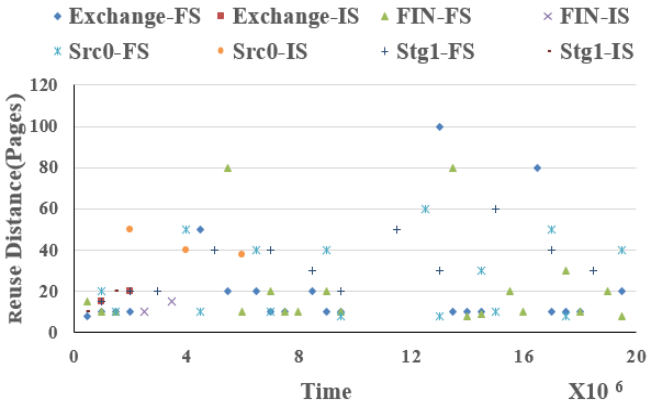


(b) the Update Frequency of Class FS, FL, IS, and IL Pages

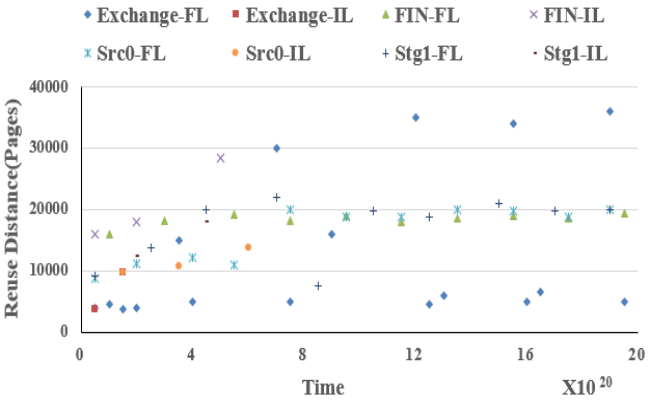
Fig.1 the Percentage of Class FS, FL, IS, and IL Pages (Fig.1(a)), the Update Frequency of Class FS, FL, IS, and IL Pages (Fig.1(b))

As shown in Fig.1(b), we can see that dirty pages have apparent features in update frequency. Specifically, the average update times of class FS pages is 10^4 more than that of class IS or IL pages, and the average update times of class FL pages is 10^2 more than that of class IS or IL pages. It means that SSD lifetime could be extended if class FS and FL pages are kept in buffer as long as possible.

Observation 1, class FS, FL, IS and IL pages show different distribution characteristics in the update frequency in every workload, and the average update times of class FS/FL pages is significantly more than that of class IS/IL pages.



(a) Class FS and IS Pages



(b) Class FL and IL Pages

Fig.2 Trends of Re-use Distance of Class FS and IS Pages(Fig.2(a)), Trends of Re-use Distance of Class FL and IL Pages(Fig.2(b))

To analyze the update behavior of all pages, we select a few pages from each class, and draw their re-use distance over a period of time as shown Fig.2.We can see that class FS, FL, IS and IL pages shows different characteristics in the re-use distance. Specially, the re-use distance of class FS pages fluctuates frequently between greater extremes. For example, the re-use distance of class FS pages in Exchange workload is ten at time 10^5, while one hundred at time 10^{12}. Compared

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

with the re-use distance of class FS pages, the reuse distance of class FL pages fluctuates less, but it is usually longer. We find that the re-use distance and the update frequency of class IS and IF pages also have similar characteristics with these of class FS and FL pages before they are not accessed again. It means that a page/block is hard to be categorized based on frequency or re-use distance, especially early in the course of the access. Similar phenomenon is also observed in other workloads as shown fig.2.

Observation 2, Class FS/FL pages are easily mistaken for class IS/IF pages, or vice versa in the early phases of the access.

B. Motivation

Compared to a hard disk drive, an SSD shows higher I/O performance, however, it could suffer from random writes, especially writes caused by frequently updated pages.

Buffer hit helps to absorb frequent updates, and reduce the amount of random writes to NAND flash memory. However, it is not efficient when the working set size of a workload is bigger than the capacity of buffer space. Since our observations show that there are a lot of infrequently updated pages/blocks, we are motivated to design a buffer management scheme to improve buffer hit ratio and reduce random writes by identify frequently updated pages/blocks.

III. Design of HWB

A. Overview

As shown Fig.3, HWB includes a temperature predictor, a partitioning model and a replacement and flush policy. The temperature predictor is used to evaluate the temperature of a dirty page. Buffer space is divided into three regions by the partitioning model, and the size of each region is adjusted dynamically when a workload pattern changes during runtime. The replacement and flush policy is in charge of managing dirty page/block insertion and eviction. When a write request is coming, the temperature prediction model is called to evaluate the temperature value of the page/block, and then the replacement and flush policy inserts the page/block into corresponding buffer region based on its temperature value.

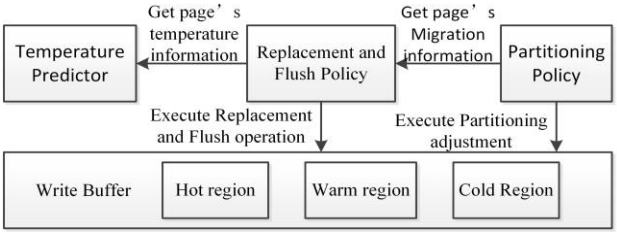


Fig.3 An Overall Architecture of HBM

B. Temperature predictor

In general, with the increasing frequency and shortening reuse distance, a page's temperature is gradually rising, as a result, it has more opportunity to be re-accessed in the future, or vice versa. Based on above reasons, we compute the temperature of a page/block combining frequency and reuse distance. The temperature of a page/block can be derived as Equation 1.

$$T_{page} = \partial * F_{self} / F_{avg} + (1 - \partial) * D_{avg} / D_{predict} \quad (1)$$

Where F_{self} is the updated frequency of current page in the last time period, F_{avg} is the average updated frequency of all dirty pages in the last time period, D_{avg} is the average re-use distance of current page, $D_{predict}$ is the reuse distance of current page when it is re-accessed in the future. The predicted distance is attained through a distance predictor discussed later. The value of F_{self} / F_{avg} and $D_{avg} / D_{predict}$ represent the long-term popular and the short-term temporal correlations of current page, respectively. σ is an adjustment factor to adjust the influence caused by F_{self} / F_{avg} and $D_{avg} / D_{predict}$.σ is set to 0.5 when a system boots, and then dynamically is adjusted according to access pattern.

In order to compute page's temperature, a ghost buffer is used to maintaining the metadata of access pages. The metadata of each page includes ID, F_{self} , $D_{last-one}$, D_{last},and D_{avg}. ID is a page address, F_{self} is the frequency of a page, $D_{last-one}$and D_{last} are the page's last but one re-use distance value and last re-use distance value respectively, andD_{avg} is the average value of previous re-use distances.

Based on Section II, we know from the observations that the effective distance of a page depends on their previous experiences. When the previous distance is gradually increased, the predictive distance usually is increased accordingly, or vice versa. In conclusion, the effective distance of a page can be derived as Equation 2.

$$D_{predict} = 2\lambda * D_{last} + \beta \quad (2)$$

Where λ is a scaling factor to estimate how much a page's re-use distance value will change. λ depends on its access pattern, specifically, if the value of last but one re-use distance is smaller than that of last re-use distance, it means that page's temperature is getting cold, and then it may have to wait a little longer for next re-access, or else it may need only to have shorter wait time. Thus, for the former, λ is set to a value of $D_{last} / D_{last-one}$,while, for the latter, λ is set to a value of 0. β is the difference between actual observed value and estimate value.

C. Partitioning model

The partitioning model is in charge of buffer partitioning. In general, all pages may be divided into class hot and cold pages. Class hot pages should be kept in buffer as long as possible to improve buffer hit ratio and reduce random writes. Based on Section Ⅱ,we know from the observations that class hot pages are easily mistaken for class cold pages, or vice versa. As a result, an observation period should be needed to reduce the chances of false positives.

Based on above reasons, the buffer space is divided into a hot region, a warm region and a cold region, as shown in Fig.4. Class hot pages and class cold pages are kept in the hot region and the cold region respectively, while other pages are stored in the warm region. HWB manages the hot region and the warm region at page granularity to improve the buffer hit ratio, and the cold region at block granularity to convert random write into sequential write and reduce erase operations to NAND flash memory.

The size of three regions may be dynamically adjusted according to access pattern, Specifically, adjustment method is as follows: at the beginning of time, the size of each region is equal. At run time, when a dirty page in a region is hit, the replacement and flash policy is called to decide whether the page should be moved to a higher temperature region. If yes, a threshold value(H) is increased by 1. The partitioning model checks H perilously. If H is more than a threshold value(A), and then the size of current region is reduced by 1, and the size

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

of the corresponding region is increased by 1. On the contrary, when a region is full, the page with lowest temperature in the region is moved to a lower temperature region, and L is increased by 1. If L is more than A, the size of the region is decreased by 1, and the size of the corresponding region is increased by 1 accordingly, as shown in Fig.4.

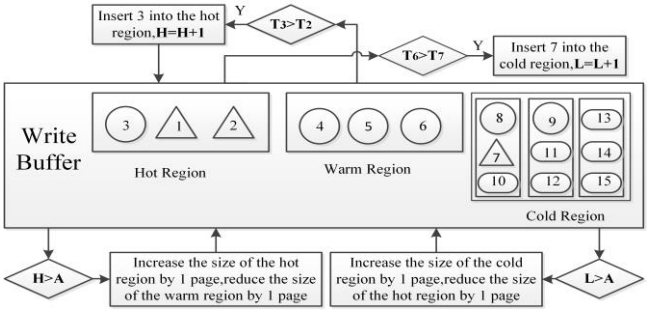


Fig.4 Adjustment Partitioning

D. Replacement and flush policy

A SSD could benefit from buffer hit, however, buffer pollution diminishes the potential benefits. Buffer management policy is an efficient method to reduce the harm caused by buffer pollution. Since buffer pollution is caused mainly by class cold pages. Therefore, class cold pages should be evicted first when buffer is not free.

How to replace a cold page: when a write request is coming, there are five possible scenarios in our approaches if buffer is full. These scenarios can cover all the case for a write request. We present each case and corresponding operation as follows:

Case 1) buffer miss and ghost miss: If a write request A is not hit in the buffer and the ghost buffer, the block at the tail of the cold list is wrote back to the NAND flash memory, and then A is inserted into a corresponding block, and the block is moved to the head of the cold list. If the corresponding block is not found, A is inserted into the head of the cold list.

Case 2) buffer miss and ghost hit: If a write request A is missed in the buffer, but hit in the ghost buffer, A's temperature is computed, and is compared with that of the page B at the tail of the warm list. If A's temperature is higher, and then B is evicted, and A is inserted into the head of the warm list. Or else, the page is inserted into the head of the cold list.

Case 3) buffer hit in the cold region: If a write request A is hit in the cold region, HWB computes its temperature and compares it to that of the page B at the tail of the warm list. If A's temperature is higher, and then B is evicted, and A is inserted into the head of the warm list. Or else, the block containing the page is moved to the head of the cold list.

Case 4) buffer hit in the warm region: If a write request A is hit in the warm region, HWB computes its temperature and compares it to that of the page B at the tail of the hot list. If A's temperature is higher, and then B is evicted, and A is inserted into the head of the hot list. Or else, A is moved to the head of the warm list.

Case 5) cache hit in the hot region: If a write request A is hit in the hot region, A is moved to the head of the hot queue.

How to flush a victim page: There are three possible scenarios in our approaches when the flush policy is called. We present each case and corresponding operation as follows:

Case 1) to evict in the cold region: When a dirty block A is evicted from the cold list, there are two kinds of policies: an active and a positive policy.

For the active policy, firstly, HWB computes the effective distance(ED) of the block A at the tail of the cold list based on Equation 3.

$$ED = PT + 2^{\lambda} * D_{last} + \beta \qquad (3)$$

Where Pt is the total number of write requests when a page is accessed at last time.

Secondly, the ED is compared with a current distance(CT). If the ED is less than the CT, A will be evicted immediately. Note that the CT is the total number of write requests after the boot time.

For the positive policy, it is only called when there are not free in the cold region. In this case, the block at the tail of the cold list is evicted directly.

Compared with the passive eviction, the positive eviction may improve the write performance because there are more free in the cold region, however, it may reduce write hit rate due to the blocks discarded early.

Case 2) evict in the warm region: When the warm region is full, HWB moves the page at the tail of the warm list firstly into a corresponding block in the cold region, and then the block is moved to the head of the cold list. If the cold region is full, the block at the tail of cold list is discarded.

Case 3) evict in the hot region: When the hot region is full, HWB estimates the temperature of the page A at the tail of the hot list firstly, and then compare A' temperature with that of the page at the tail of the warm list. A is moved to the head of the warm list if A' temperature is higher, or else, A is inserted into a corresponding block in the cold region, and the block is moved to the head of the cold list. If the cold region is full, the block at the tail of cold list is evicted.

IV. VALUATION

A. Experimental setup

1) System settings

We built an event-based trace-driven SSD simulator by adding a buffer management scheme into the SSDsim [12]. We implemented the following write management schemes in SSDsim simulator: 1) LRU, 2) LRU-WSR, 3) BPAC and 4) HWB.

2) Workloads

We use a set of workloads for experimental evaluation, which are collected from actual enterprise applications and are available in [9] and [10]. They include Financial (FIN), Source control(SCR), Web staging(STG), and Exchange.

B. Experimental results

1)Buffer hit ratio

Fig.5 shows the hit ratio of four buffer management policies for all the workloads. Please note that buffer hit rate is calculated as the ratio between the number of pages hit in the write buffer and total number of writing pages.

As we can see from the Fig.5, HWB outperforms LRU in terms of write buffer hit ratio under all workloads. With 32MB write buffer, the buffer hit ratio of HWB is 48%, 60%, 76% and 31% respectively. By contrast, the buffer hit rate of LRU is 30%, 52%, 57% and 27% respectively. Compared to LRU, our best-case write hit ratio comes from FIN workload, where we improve the write hit rate by 60%, while Our worst-case write hit rate occurs when running Exchange workload, where we only improve the write hit rate by 15%.

Compared with LRU-WRS and BPAC, HWB also has higher performance because HWB can efficiently identify the hot page through the temperature predictor and keep them in buffer cache as soon as long, which significantly improve

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

buffer hit rate. Similar to HWB, BPAC also tries to identify hot page by the threshold policy, however, it makes no distinction between different kinds of pages, as a result, hot pages may be mistaken as cold pages, and evicted too early. As for LRU-WRS, which tries to keep hot pages in buffer as long as possible, often mistaken cold pages as hot pages and pollutes the limited buffer space in the end.

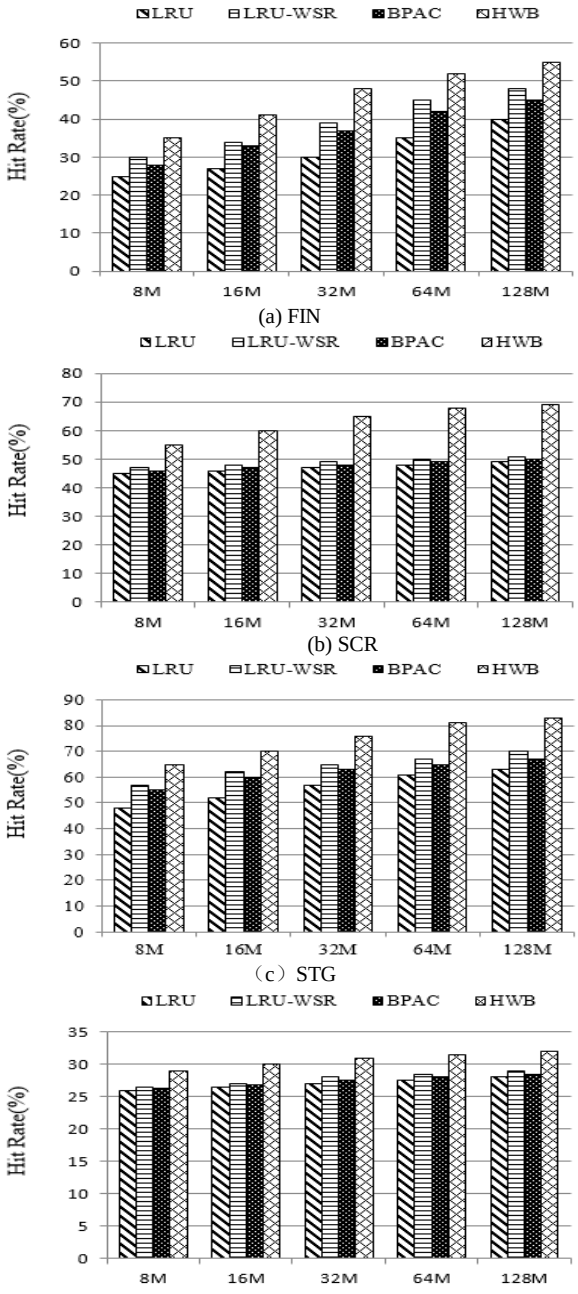


Fig.5 Buffer Hit Ratio of FIN(Fig.5(a)), Buffer Hit Ratio of SCR(Fig.5(b)), Buffer Hit Ratio of STG(Fig.5(c)), and Buffer Hit Ratio of Exchange(Fig.5(d))

We note that the average write hit rate of Exchange workload is lower than that of any other workload, while the average write hit rate of STG workload is higher than that of any other workload. One main reason is that Exchange contains less class hot pages, such as class FL and FS pages, than any other workloads, while STG includes more hot pages than any other workloads.

We also note that LRU-WSR write hit ratio is higher than that of BPAC. One main reason is that LRU-WSR manages buffer by page list, while BPAC manages buffer by block list, which makes hot pages easier to be evicted prematurely.

2)Erase Count

Erase count is number of blocks erased during garbage collection. Note that the erase number of a SSD is limited, so replacement policy should try to reduce erase operation.

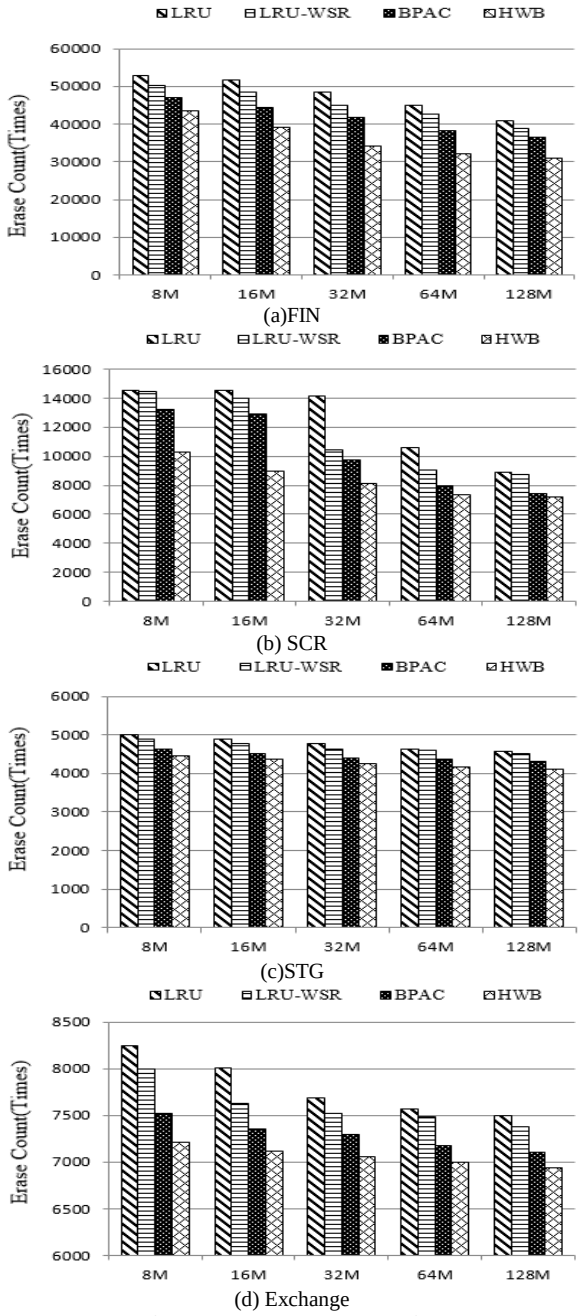


Fig.6 Erase Count of FIN(Fig.6(a)), Erase Count of SCR(Fig.6(b)), Erase Count of STG(Fig.6(c)), and Erase Count of Exchange(Fig.6(d))

Fig.6 plots the erase count of four buffer management policies for all the workloads. As we can see from the Fig.6, HWB outperforms LRU in terms of erase count under all workloads. With 32MB write buffer, the average erase count of HWB is 34050, 7058, 8102 and 4252 respectively. By contrast, the buffer hit rate of LRU is 48570, 7689, 14120 and 4785 respectively.

We note that BPAC outperforms LRU-WRS in terms of erase count under all workloads. The reason is that BPAC can transfer more random write into sequence write, which reduces the number of write operations to NAND flash memory. Similer to BPAC, HWB also use block list to manage cold

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

region, but it could be more effective in identifying hot pages, and reducing buffer pollution, therefore, HWB outperforms BPAC in terms of the number of erase under all workloads.

3)Response Time

Average response time is overall performance of write operation, which is related with not only write hit ratio, but also erase count.

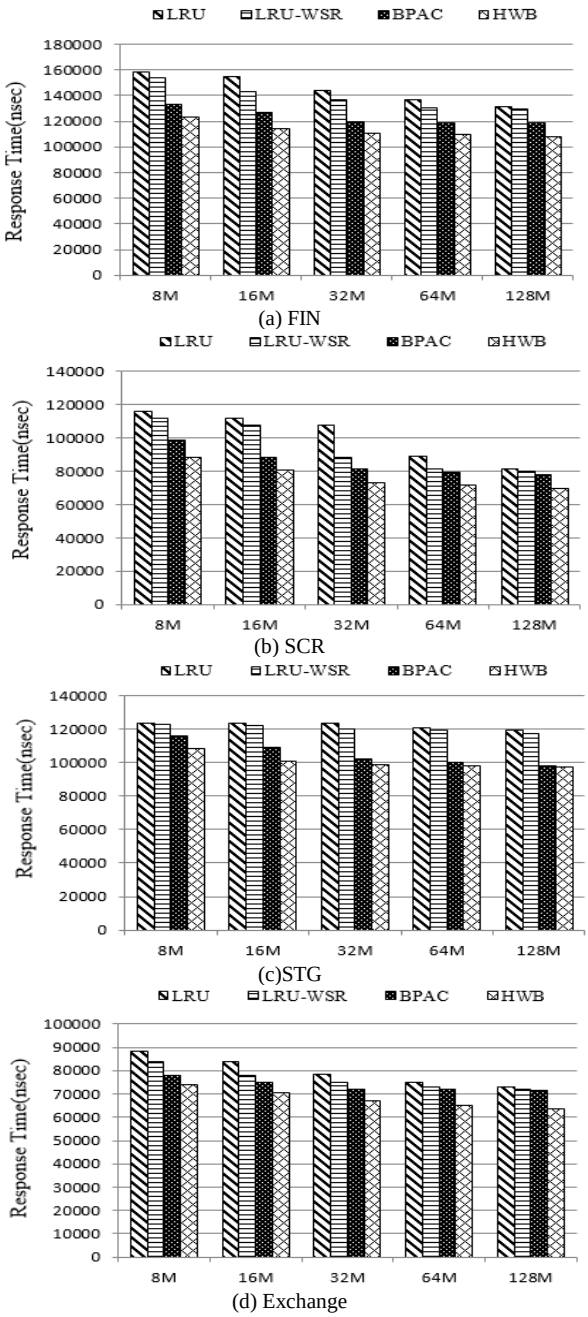


Fig.7 Repsonse Time of FIN(Fig.7(a)), Repsonse Time of SCR(Fig.7(b)), Repsonse Time of STG(Fig.7(c)),and Repsonse Time of Exchange(Fig.7(d))

Fig. 7 plots the response time of four buffer management policies for all the workloads. As we can see from the Fig.7, HWB outperforms any other algorithm in terms of response time under all workloads. For example, with 32MB write buffer, the response time of HWB is 110321ns, 67009ns, 73021ns, and 98501ns respectively. By contrast, the response time of LRU is 143726ns, 78238ns, 107661ns, and 123171 ns respectively. The main reason is that HWB obtains not only higher buffer hit ratio, but also less erase operations compared with LRU.

We note that BPAC outperforms LRU-WRS in terms of response time under all workloads even if its buffer hit rate is lower than that of LRU-WRS. It means that response time has more sensitivity to erase operation, therefore we should try to reduce erase operations to NAND flash memory at the expensive of driving down buffer hit rate.

V. Conclusion

In this paper, we propose HWB, a hotness-aware write buffer management scheme that reduces random writes by identifying infrequently updated page/block. To improve both buffer hit rate and data sequentiality, HWB divides write buffer space into a hot region, a warm region, and a cold region. Frequently updated date is stored in the hot and warm region at page granularity, and infrequently updated date is kept in the cold region at block granularity. We further design a replacement and flush policy combined with a temperature predictor to improve buffer hit ratio and reduce random writes. HWB is evaluated using extensive trace-driven simulations. Its evaluated results show that HWB significantly reduces random write and extends SSD lifetime compared to existing buffer management schemes.

References

[1] N. Agrawal, V. Prabhakaran, T.Wobber, J. D. Davis, M. Manasse, and R. Panigrahy. Design tradeoffs for SSD performance[C]. In Proceedings of the 2008 USENIX Annual Technical Conference, 2008.

[2] H. Yassine, J. Coon, and D. Simmons. Index programming for flash memory. IEEE Transactions on Communications, Vol. 65(5): 1886-1898,2017.

[3] Karedla R, Love J S.Caching strategies to improve disk system performance[J]. Computer, 1994, 27(3):38-46.

[4] Megiddo N.ARC: A self-tuning, low overhead Replacement cache[C]. USENIX File and Storaqe Technologies Conference (FAST'03), San Francisco, CA. 2003.

[5] Seon-yeong Park , Dawoon Jung , Jeong-uk Kang , Jin-soo Kim, Joonwon Lee, CFLRU: a replacement algorithm for flash memory[C], Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems October 2006 .

[6] Ou Y , Theo H?rder, Jin P . CFDC: A Flash-Aware Buffer Management Algorithm for Database Systems[C]. Advances in Databases and Information Systems - 14th East European Conference, ADBIS 2010, Novi Sad, Serbia, September 20-24, 2010. Proceedings. DBLP, 2010.

[7] Jung H , Shim H , Park S , et al. LRU-WSR: integration of LRU and writes sequence reordering for flash memory[J]. IEEE Transactions on Consumer Electronics, 2008, 54(3):1215-1223.

[8] Wu G , Eckart B , He X . BPAC: An adaptive write buffer management scheme for flash-based Solid State Drives[C].2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). IEEE, 2010.

[9] Jstorage performance council.spc trace file format specification.http://traces.cs.umass.edu/index.php/storage/storage.

[10] block traces from SNIA,http://iotta.snia.org/traces/list/BlockIO.

[11] S. Park and C. Park.FRD: A filtering based buffer cache algorithm that considers both frequency and reuse distance[C]. Proc. 33rd IEEE Int. Conf. Massive Storage Syst. Technol. pp. 1-12 2017.

[12] Y. Hu, H. Jiang, D. Feng, L. Tian, H. Luo, and S. Zhang, Performance Impact and Interplay of SSD Parallelism through Advanced Commands, Allocation Strategy and Data Granularity[C], Proceedings of the international conference on Supercomputing (ICS), 2011, pp.96-107.

ISBN: 978-988-14049-1-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)