Maximizing IO Performance Via Conflict Reduction for Flash Memory Storage Systems

Qiao Li*, Liang Shi*§, Congming Gao*, Kaijie Wu*, Chun Jason Xue†, Qingfeng Zhuge *,and Edwin H.-M. Sha*
*College of Computer Science, Chongqing University, Chongqing, China
† Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong
Email: {qiaoli045, shi.liang.hk, albertgaocm, kaijie}@gmail.com, jasonxue@cityu.edu.hk, {qfzhuge, edwinsha}@gmail.com

***Abstract*—Flash memory has been widely deployed during the recent years with the improvement of bit density and technology scaling. However, a significant performance degradation is also introduced with the development trend. The latency of IO requests on flash memory storage systems is composed of access conflict latency, data transfer latency, flash chip access latency and ECC encoding/decoding latency. Studies show that the access conflict latency, which is mainly induced by the slow transfer latency and access latency, has become the dominate part of the IO latency, especially for IO intensive applications. This paper proposes to reduce the flash access conflict latency through the reduction of the transfer and flash access latencies. A latency model is built to construct the relationship among the transfer latency and access latency based on the reliability characteristics of flash memory. Simulation experiments show that the proposed approach achieves significant performance improvement.**

I. Introduction

During the recent years, the technology of flash memory has been advanced in many ways, including the bit density improvement from 1 bit per cell to the latest 6 bits per cell [3] [4], technology scaling from 65nm to the latest 10nm technology [7]. However, with these development of flash memory, significant performance degradation has been found in the process of IO requests in flash memory storage systems [1]. The degraded performance is the result of many reasons. The increased access latency (read/write/erase operations) and the latency induced by complex ECC schemes, such as Low-Density Parity-Check Code (LDPC) [1] [3] [14], are the two major components. In addition to these two issues, access conflict induced IO latency is enlarged with the slowdown on the processing of IO requests in the flash memory storage systems [5] [2].

Access latency of IO requests on flash memory storage systems includes at least three major parts: **the access conflict latency**, which depends on the queueing time when IO requests are pending in the IO queue, waiting for the process of the preceding IO requests; **the data transfer latency**, which is the time required to transfer data between flash controller and flash memory chip; **the access latency** (sensing for read and programming for write), which is determined by the type of flash chips and operations. In addition to the three parts, ECC encoding during write operations and ECC decoding during read operations also contribute to the overall latency of their corresponding operations. Several works have been proposed to optimize separate parts of the IO latency [1] [14] [12] [13] [11]. Gao et al. [2] proposed to exploit the parallelism of solid state drives to reduce the access conflict latency. Data transfer latency is reduced with the introduction of high speed interface between flash chips and flash controller [3]. Access latency is improved with enhanced ECC schemes or relaxed reliability requirement of flash memory [9] [6] [13]. However, none of these works takes the relationship among these three parts into consideration.

In this work, we propose a comprehensive approach to improve the performance of flash memory storage systems by taking the relationship among all the three main parts into consideration. Based on the study of the characteristics of the three types of latencies, we found that, while access conflict latency is always the dominant factor for most IO intensive applications, it is in fact the consequence by the other two factors: slow data transfer, and/or slow flash accesses. Hence instead of directly reducing the access conflict latency, the main idea of this work is to reduce the data transfer latency and flash access latency. The access conflict latency is then reduced naturally. The data transfer and flash access latency reduction is realized by exploring the reliability characteristics of flash memory. In this paper, we take LDPC [1], the highly recommended ECC scheme for the state-of-the-art flash memory due to the increased raw bit error rates [14] [3], as the default ECC scheme. As a side effect of offering stronger error correction capability, however, LDPC also introduces large data sensing and transfer latencies of read requests. When LDPC is applied on flash memory storage system as ECC, an interesting trade-off between the read and write latencies can be seen. On one hand, reading data that have a lower raw bit error rate can be done with less numbers of sensing levels for LDPC decoding, hence incurring less sensing and data transfer latencies for read requests. On the other hand, writing data faster (i.e., programming with a larger step size ΔV_{pp} [9] [6]) can reduce the programming latency of write requests but at the cost of leaving the written data a higher raw bit error rate [10]. To make the situation even more complex, it can be seen that, though a faster write leads to a slower read of the same data in the future, it does help in reducing the access conflict latency (waiting time) of the requests scheduled immediately after this write!

The rest of the paper is organized as follows. Section II discusses the proposed conflict reduction approach for read and write requests. Experiments and result analysis are presented in Section III. Finally, Section IV concludes the paper.

II. Read and Write Conflict Reduction for Performance Improvement

In this section, the latency relationship along the IO path is exploited for performance improvement. A latency model is first proposed to build the relationship among data sensing latency of read requests (SL), data transfer latency of read requests (TL),

§Liang Shi is corresponding author: shi.liang.hk@gmail.com.

and data programming latency of write requests (PL) along the IO path of requests. Note that the data transfer latency of write operations is constant. Based on the model, SL, TL and PL can be well traded off with each other to improve overall performance. Then, the reduction of SL, TL and PL is exploited to reduce the access conflict latency (CL).

A. Latency Model for Data Sensing, Transferring and Programming

In this section, we discuss the latency model for read requests and write requests. Note that the latency of requests in flash memory is highly correlated with the ECC scheme in the flash memory controller [14] [9] [6]. In this work, we assume LDPC as the default ECC scheme. In the following part, we first construct the relationship between SL/TL and error correct capability of LDPC, and the relationship between PL and LDPC. Finally, latency model is proposed.

1) SL and TL with LDPC: Recently, several works have showed that soft-decision scheme achieves stronger error correction capability at the cost of longer data sensing and transfer latency because of more reference voltages, i.e. more sensing levels [3] [1] [14]. Assume that the correctable raw bit error rates by LDPC with N reference voltages are $CBER_{LDPC}(N)$ and the raw bit error rates of flash cells are RBER. The relationship between LDPC and SL/TL can be represented as follows:

$$\begin{aligned} & SL(N) \propto N \\ & TL(N) \propto \lceil \log(N+1) \rceil \\ where \quad & RBER < CBER_{LDPC}(N); \end{aligned} \tag{1}$$

This formula implies that SL is proportional to N and TL is a logarithmic function of $N+1$ with the precondition that RBER can be corrected by LDPC.

2) PL with LDPC: Write requests are processed by ISPP in current flash memory [10]. PL is determined by the programming step size ΔV_{pp} of ISPP. Previous works have shown that PL is inversely proportional to ΔV_{pp} [10] [9] [6]. However, different ΔV_{pp} have different reliability characteristics on flash memory [9] [6]. A larger ΔV_{pp} always results in a higher RBER [9]. The RBER is $RBER(\Delta V_{pp})$ for ISPP with ΔV_{pp}. Note that the data transfer latency of write requests is different from that of read requests. Data transfer latency of write requests is a constant value, and is determined by the interface bandwidth of flash chips. Based on the above descriptions, the relationship between PL and LDPC is constructed as follows.

$$\begin{aligned} & PL(\Delta V_{pp}) \propto \frac{1}{\Delta V_{pp}} \\ where \quad & RBER(\Delta V_{pp}) < CBER_{LDPC}(N); \end{aligned} \tag{2}$$

This formula means that PL is inversely proportional to ΔV_{pp}. In order to support the applied ΔV_{pp}, the number of reference voltages in LDPC should make sure that $CBER_{LDPC}(N)$ is larger than $RBER(\Delta V_{pp})$.

3) Latency Model: Based on the above two relationships, the latency model among SL, TL and PL is summarized as follows.

$$\begin{aligned} & SL(N) \propto N \\ & TL(N) \propto \lceil \log(N+1) \rceil \\ & PL(\Delta V_{pp}) \propto \frac{1}{\Delta V_{pp}} \\ where \quad & RBER(\Delta V_{pp}) < CBER_{LDPC}(N); \end{aligned} \tag{3}$$

Once the number of reference voltages N is determined, $CBER_{LDPC}(N)$ can be easily computed. RBER is determined by many factors of flash memory, including retention time, P/E cycles, temperature, ΔV_{pp}, and cell-to-cell interference [9] [6]. Several flash device models has been proposed for the relationship among these factors. In this work, the flash device model in [9] is applied to calculate RBER of flash memory cell.

The latency model shows that PL and SL/TL can be traded off with each other to achieve read and write reduction.

B. Conflict Reduction for Read and Write Requests

When several IO requests access flash memory storage systems in a short time interval, many IO requests will wait in the IO queue for the process of the outstanding IO requests. These IO requests are called conflict requests. In the following part, read and write conflict reduction approach is proposed for read and write requests, respectively.

1) Read Conflict Reduction: Read requests may introduce access conflict especially when LDPC is implemented as the ECC scheme. In order to reduce the access conflict introduced by read requests, a simple approach is to reduce SL and TL. However, SL and TL are hard to reduce. The reason is that SL and TL are determined by the number of reference voltages, which has to provide required accuracy of sensed probability information to correctly decode data. As presented in Section II-A, SL and TL can be determined by PL for the programmed data. SL and TL of reading a page of data become more significant when the data were programmed with faster write speed. In order to reduce SL and TL, straightforward approach is to program the data with slower speed. However, simply programming all data with slow speed introduces bad write performance, which would introduce more conflicts.

Based on above descriptions, we propose a recording-based re-programming approach for SL and TL reduction. The basic idea is to record the information of data that have experienced access conflicts but were not programmed with slow speed. Then, during idle time, these data are re-programmed with the slow program speed. The approach is designed based on the idea that if the data have caused conflicts, they have a higher probability to cause conflicts in the future. Then, after slowly re-programming of these data, the future access conflict would be reduced when the data are accessed again. Note that since we only re-program the data that have caused access conflicts and were programmed using the normal or fast program speed, the re-programming overhead is small, which will be presented in the experiment section. In addition, many write requests are also programmed with slow speed to further increase the probability of fast read and reduce the re-programming probability, which will be presented in the next section.

Take Figure 1 as example to show the process of read conflict reduction. In this figure, assume that there are 4 pending read

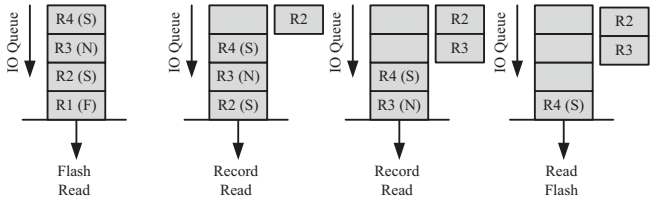
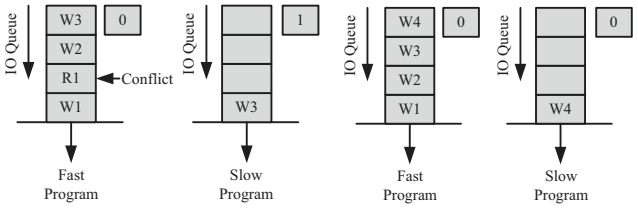


Fig. 1. Read Conflict Reduction Process.



(a) Case 1: Slow Programming (b) Case 2: Normal Programming

Fig. 2. Write Conflict Reduction Process.

requests in the IO queue with different read speeds represented by Slow (S), Normal (N) and Fast (F). R1 is the next scheduled read request. As shown in Figure 1, even though R1 induces access conflict, it is not recorded since R1 is a fast read and can not be accelerated anymore. R2 is a slow read request with access conflict. When R2 is scheduled, the page number of R2 is recorded in a queue maintained in the flash controller. R3 is a normal speed read request and can be speeded up. In this case, it is also recorded. Finally, for R4, it does not introduce conflict to the IO queue. Even though R4 is a slow read, it is not recorded for re-programming to avoid additional overhead to the flash memory.

2) Write Conflict Reduction: Generally speaking, write requests are more time consuming than read requests in flash memory, which will induce serious access conflict. In order to solve this problem, a simple approach is to speed up the process of programming operations. However, fast programming operations would result in slow read operations. From the latency model presented in Section II-A, fast programming increases RBER, which then demands a much larger number of reference voltages to reliably sense the data for LDPC decoding. In this case, SL and TL are increased.

In order to solve the problem, write conflict reduction approach is proposed, which uses fast programming operations only when there are access conflict in the IO queue. If conflicts are eliminated, normal or slow programming operations are used to avoid bad read performance and reduce the re-programming overhead. The speed for write requests during conflict free is determined by the types of the last read request. If the last read request is conflict, then slow programming operation is applied to increase the possibility of fast read in the future. Otherwise, normal speed programming operation is applied.

Take Figure 2 as example to show the process of write conflict reduction. Assume there are 4 IO requests pending in the IO queue and W1 is the next scheduled write request. W1 is processed with fast programming operation. In this case, access conflict is reduced and pending requests in the queue would be processed faster. When the last write request in the queue is processed, two cases could happen. For the first case (Figure 2(a)), the last read request R1 is conflict. The write request, W3, is then processed with slow program speed to reduce the latency of future read requests. A tag, read conflict tag, is added in the IO queue to record types of the last read request. For the second case (Figure 2(b)), the read conflict tag is not set, then the last write request, W4, is processed with normal programming speed.

III. Experiment and Analysis

In this section, we first present the experimental methodology. Then, the experimental results are presented, followed by the analysis of the improved performance.

A. Experiment Setup

In this paper, we use trace driven simulator to verify the proposed approach. The developed simulator is configured with 8 channels with 8 flash chips equipped in each channel. Each chip has two flash planes, and each plane has 2048 blocks. Each block has 64 pages. The page size is 4KB. A page mapping based flash translation layer, greedy garbage collection, and dynamic wear leveling approaches are implemented in the simulator. Eight IO intensive workloads are selected to evaluate the proposed approach [8].

The parameters for SL, TL and PL are produced as follows: First, we use different number of reference voltages in LDPC to compute the correctable error rates. Then, based on the model proposed in Section II, SL and TL are computed. Finally, based on the flash device model in [9], the supported program speed for the corresponding LDPC is produced. Table I shows the applied parameters used in the work.

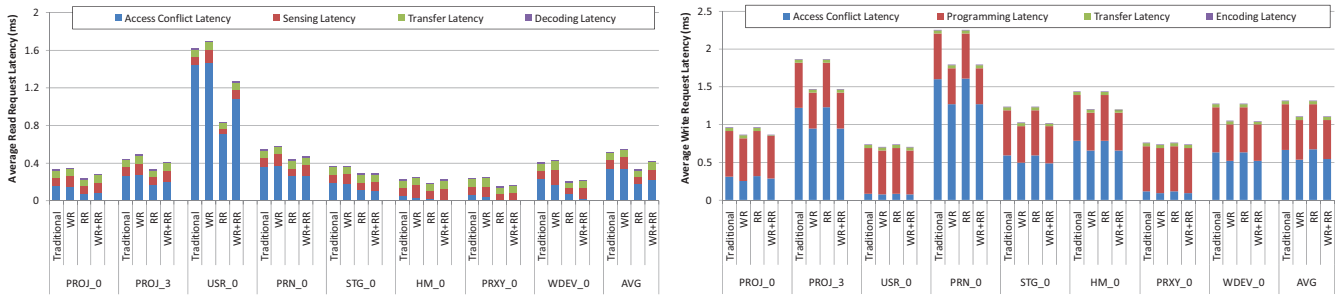
TABLE I. Parameters in This Work.

Latency (μs)	Read		
	Fast	Normal	Slow
Sensing	**30**	**90**	**210**
Data Transfer for Read	**40**	**80**	**100**
Latency (μs)	Write		
	Slow	Normal	Fast
Programming	**800**	**600**	**450**
Data Transfer for Write	40	40	40
Erase	3000	3000	3000
Encoding/Decoding	8	8	8

B. Experimental Results

In this section, the experiment results are presented and analyzed. Four schemes are implemented to show the effectiveness of the proposed approach: Traditional, Read Conflict Reduction (RR), Write Conflict Reduction (WR), and the combination of RR and WR (RR+WR). RR and WR represent that only read or write conflicts are relaxed. RR+WR represents that both RR and WR are implemented.

Figure 3 shows the IO latency for read and write requests, respectively. Compared with Traditional, RR and WR achieve significant read and write performance improvement, respectively. For read requests, RR achieves 36.3% IO latency reduction for read requests as shown in Figure 3(a). The latency reduction for RR scheme comes from two aspects at least: reduction of SL and TL, and reduction of CL. With RR scheme, SL and TL are reduced by 13%. CL is reduced by 47.8%. For write requests, WR achieves 16.2% IO latency reduction for write requests. The latency reduction also comes from two aspects: PL reduction and CL reduction. As shown in Figure 3(b), PL and CL are reduced by 13.2% and 19.5%, respectively. Based on these results, we can find that read and write conflicts can be significantly reduced via read and write conflict reduction. RR+WR combines RR and



(a) IO Latency for Read Requests

(b) IO Latency for Write Requests

Fig. 3. IO Latency Improvement for Read and Write Requests.

WR to improve both read and write performance. As shown in Figure 3, RR+WR averagely reduces IO latency by 20.8% and 16.2% for read and write requests compared with Traditional, respectively. Compared with RR, RR+WR has 15% IO latency increases. The reason comes from that WR relaxes many write operations, which would induce read access latency increases, as presented in Section II-A. With the combination of these two approaches, we can find that the proposed approach achieves balanced performance improvement. From this experiment, we can find that the proposed approach is very efficient in the reduction of CL.

RR requires re-programming operations when there are conflict read requests with slow or normal speed. In order to reduce the overhead of re-programming operations, only slow and normal speed read operations with conflict are re-programmed. In addition, re-programming operations can be further reduced when write requests are processed with slow speed if there are read conflicts. Figure 4 shows the overhead of re-programming operations compared with all IO requests. The re-programming overhead is consistent with CL of IO requests. For example, USR_0 has the largest overhead since it has the largest CL. On average, the percentage of re-programming operation is smaller than 5%. The small re-programming overhead can be negligible through the exploration of the idle time.

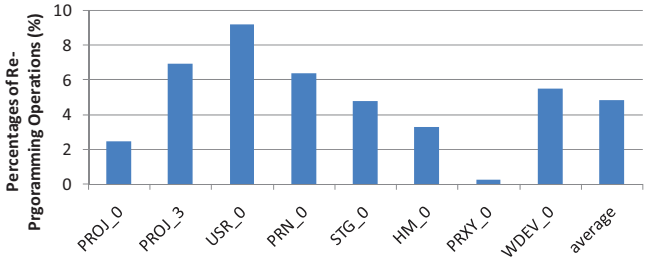


Fig. 4. Percentages of Re-Programming Operations.

IV. Conclusions

In this paper, we proposed a conflict reduction approach for read and write requests to improve the performance of flash memory storage systems. Different from previous works, the reduction of read and write conflicts is proposed to reduce the conflict latency of IO requests, which has been verified as the dominate part in the IO latency. A latency model is first proposed to construct the latency relationship among data sensing, transferring during read operations and data programming. Then, based on the proposed latency model, the conflict reduction approach is proposed to improve the performance of read and write requests with the awareness of access conflict in the IO queue. An efficient implementation with negligible cost is then presented. Simulation results show that the proposed approach works effectively, with around 20.8% and 16.2% IO latency reduction for read and write requests.

V. Acknowledgment

This work is partially supported by the Fundamental Research Funds for the Central Universities (CDJZR185502), National Natural Science Foundation of China (61402059), National 863 Program 2013AA013202, Chongqing High-Tech Research Program csct2012ggC40005, NSFC 61472052, and NSFC 61173014.

References

[1] G. Dong, N. Xie, and T. Zhang. Enabling nand flash memory use soft-decision error correction codes at minimal read latency overhead. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(9):2412–2421, Sept 2013.

[2] C. Gao, L. Shi, M. Zhao, C. Xue, K. Wu, and E.-M. Sha. Exploiting parallelism in i/o scheduling for access conflict minimization in flash-based solid state drives. MSST'14, pages 1–11, June 2014.

[3] K.-C. Ho, P.-C. Fang, H.-P. Li, C.-Y. Wang, and H.-C. Chang. A 45nm 6b/cell charge-trapping flash memory using LDPC-based ECC and drift-immune soft-sensing engine. ISSCC'13, pages 222–223, 2013.

[4] X. Jimenez, D. Novo, and P. Ienne. Phœnix: Reviving mlc blocks as slc to extend nand flash devices lifetime. DATE '13, pages 226–229, 2013.

[5] M. Jung, E. Wilson, and M. Kandemir. Physically addressed queueing (paq): Improving parallelism in solid state disks. ISCA'12, pages 404–415, 2012.

[6] R.-S. Liu, C.-L. Yang, and W. Wu. Optimizing nand flash-based ssds via retention relaxation. FAST'12, pages 11–11, 2012.

[7] S. E. C. Ltd. High-Performance 128-gigabit 3-bit Multi-level-cell NAND Flash Memory. https://memorylink.samsung.com/.../detail.do?newsId=12761, 2013.

[8] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating server storage to ssds: analysis of tradeoffs. ACM Eurosys'09, pages 145–158, 2009.

[9] Y. Pan, G. Dong, and T. Zhang. Exploiting memory device wear-out dynamics to improve nand flash memory system performance. FAST'11, pages 18–18, 2011.

[10] K.-D. Suh, B.-H. Suh, Y.-H. Lim, J.-K. Kim, Y.-J. Choi, Y.-N. Koh, S.-S. Lee, S.-C. Kwon, B.-S. Choi, J.-S. Yum, et al. A 3.3 v 32 mb nand flash memory with incremental step pulse programming scheme. *IEEE Journal of Solid-State Circuits*, 30(11):1149–1156, 1995.

[11] J. Wang, K. Vakilinia, T.-Y. Chen, T. Courtade, G. Dong, T. Zhang, H. Shankar, and R. Wesel. Enhanced precision through multiple reads for ldpc decoding in flash memories. *IEEE Journal on Selected Areas in Communications*, 32(5):880–891, 2014.

[12] W. Wang, Y. Lu, and J. Shu. p-oftl: An object-based semantic-aware parallel flash translation layer. DATE '14, pages 157:1–157:6, 2014.

[13] G. Wu and X. He. Reducing ssd read latency via nand flash program and erase suspension. FAST'12, pages 10–10, 2012.

[14] K. Zhao, W. Zhao, H. Sun, T. Zhang, X. Zhang, and N. Zheng. Ldpc-in-ssd: Making advanced error correction codes work effectively in solid state drives. FAST'13, pages 244–256, 2013.