Sub-grouped superblock management for high-performance flash storages

JungWook Park[1a)], **Gi-Ho Park**[2], **Charles Weems**[3], **and ShinDug Kim**[1]

[1] *Supercomputing Lab., Dep. of Computer Science, Yonsei Univ., Seoul, Korea*

[2] *Department of Computer Engineering, SeJong Univ., Seoul, Korea*

[3] *Department of Computer Science, University of Massachusetts, Amherst, MA 01003–4610, U.S.A.*

a) *pjppp@cs.yonsei.ac.kr*

Abstract: In this paper we describe a new superblock management scheme to overcome the problem of increased erase operations, that results from increasing the degree of interleaving of memory banks in flash memory based storage devices. To improve performance, superblock management is used to increase the degree of linear interleaving of flash memory banks. However, increased interleaving may significantly increase the number of erase operations, thus decreasing device lifetime. The proposed management scheme efficiently separates hot and cold data into two different sub-groups, dramatically increasing the efficiency of superblock merging. According to our simulation results, the number of erase operations decreases by around 27.3 percent, which is enough to significantly lengthen overall device lifetime. Read performance is only slightly degraded by our approach.

Keywords: superblock management, NAND flash memory, interleaving, flash translation layer

Classification: Storage technology

References

[1] Samsung NAND Flash datasheet K9XXG08XXM, [Online] http://www.samsung.com

[2] J.-S. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A SPACE-EFFICIENT FLASH TRANSLATION LAYER FOR COMPACTFLASH SYSTEMS," *IEEE Trans. Consum. Electron.*, vol. 48, no. 2, May 2002.

[3] J.-U. Kang, H. Jo, J.-S. Kim, and J. Lee, "A Superblock-based Flash Translation Layer for NAND Flash Memory," *Proc. EMSOFT'06*, pp. 161–170, Oct. 22–25, 2006.

[4] J.-U. Kang, J.-S. Kim, C.-I. Park, H. Park, and J. Lee, "A multi-channel architecture for high-performance NAND flash-based storage system," *J. Syst. Architect.*, vol. 53, pp. 644–658, 2007.

[5] [Online] http://www.futuremark.com/products/pcmark05/

© IEICE 2009
DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009

[6] S.-H. Park, J.-W. Park, J.-M. Jeong, J.-H. Kim, and S.-D. Kim, "A Mixed Flash Translation Layer Structure for SLC-MLC Combined Flash Memory System," *Proc. Workshop on Storage and I/O Virtualization, Performance, Energy, Evaluation and Dependability*, 2008.

1 Introduction

NAND flash memory based storage devices are widely used in various kinds of computer systems. Mobile laptop computers, for example, achieve greater energy efficiency and physical durability. Moreover, as memory process technology advances, fast flash memory access time will be able to overcome the disk I/O bottleneck in desktop computer systems. A NAND flash memory chip is composed of memory cell array, internal control logic, and a page register. The memory array consists of a fixed number of memory blocks, and each block is made up of several pages. In current NAND flash memory with multi-level cells (MLC), each block consists of 128 pages, and each page is composed of 4096 bytes of active data area plus 128 bytes of spare area [1]. A read/write operation on the flash memory is performed at the page-level, where each read operation fetches an entire page into the page register even when just one byte of data is requested. Similarly, a complete page should be written to the page register before any write operation is performed on the cell array.

Access timing is as follows: 50 us is required for fetching a page to the register and an entire page can be read out in 140 us. Filling to a page register for writing also takes 140 us, but a write operation incurs additional latency to reflect the modification of the cell array: 800 us. Moreover, when NAND flash storage devices are assembled into a solid state disk (SSD), consisting of multiple memory chips with an aggregate capacity of tens to hundreds of gigabytes, the individual chips may be has multiple banks [4].

Hence interleaving is a reasonable approach to increasing the overall performance of an SSD with these access characteristics. Theoretically, memory access performance is linearly scalable with the level of interleaving, but in practice it is limited by the given access patterns of real applications. To enhance storage performance, a superblock management scheme is employed to support flash memory bank interleaving. However, increasing the degree of interleaving can also significantly increase the number of erase operations. The appropriate degree of interleaving depends on the I/O parallelism and overhead of the applications using the SSD.

We have analyzed the access patterns of various applications and propose a new superblock management scheme that avoids the increase in erase operations due to excessive interleaving. The blocks in a superblock are divided into two sub-groups according to their spatial locality, thereby avoiding unnecessary erase operations. According to our simulation results, the overall erase count is reduced by around 27.3 percent while preserving almost the same access latency. Thus, the proposed method achieves scalable access

© IEICE 2009
DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009

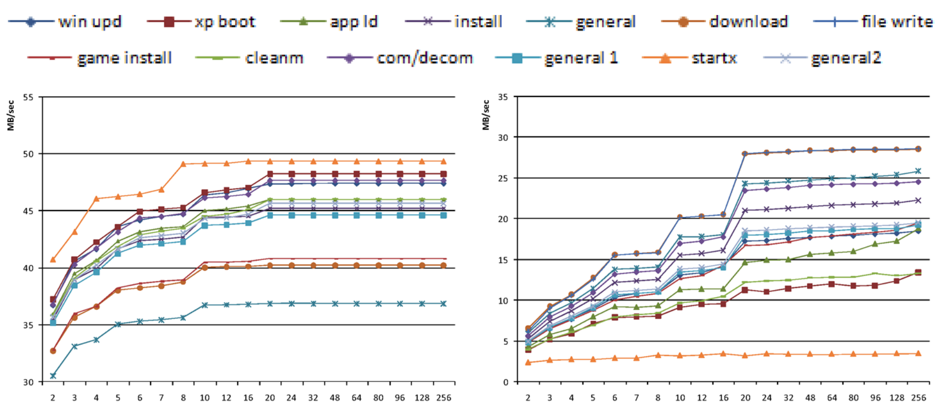
performance, while minimizing the erase count and extending device lifetime.

2 Multi-bank interleaving technique

Thirteen different application traces were used to analyze the degree of I/O parallelism in storage access patterns. Because there can be up to 256 interleaved flash memory banks, At most 256 pages (1 MB) can be interleaved as a single request. Details of the experimental environment will be presented in a later section. The effect of bank interleaving on read/write performance is shown in Fig. 1. Fig. 1 (a) shows the average performance for read requests in mega-bytes per second, where the x-axis denotes the number of banks, and Fig. 1 (b) presents the same result for write requests. Both read and write requests show performance gains as the number of interleaved banks increases. However, performance improvement plateaus when the number of banks exceeds the average request size.

NAND flash memory does not support in-place-update, so a requested logical write address must be re-mapped to another physical address. The Flash Translation Layer (FTL) performs this address translation and remapping [2]. When a page in the flash memory's page register is written, an available physical page is allocated. To modify existing page contents, the data must be written into an extra blank page in another block. The latter is called the update-block while the block containing the page to be updated is called the data block. When all of the pages in the update-block have been used to record updates, both blocks should be merged into a new data block and erased. These characteristics of flash memory present some difficulties in designing an interleaved structure and its management mechanism.

Fig. 2 shows some block mapping examples for an SSD architecture with eight banks of flash memory, where each block is assumed to have eight pages; LBlock means a logical block, PBlock means a physical block, and the numbers in the inner box indicate logical page addresses. Fig. 2 (a) shows a basic mapping scheme that allocates a physical block to each logical block, with the result that the SSD can take little advantage of I/O parallelism [3]. Fig. 2 (b)



(a)Average read performance (b) Average write performance

Fig. 1. Impact of bank interleaving in various environments.

© IEICE 2009
DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009

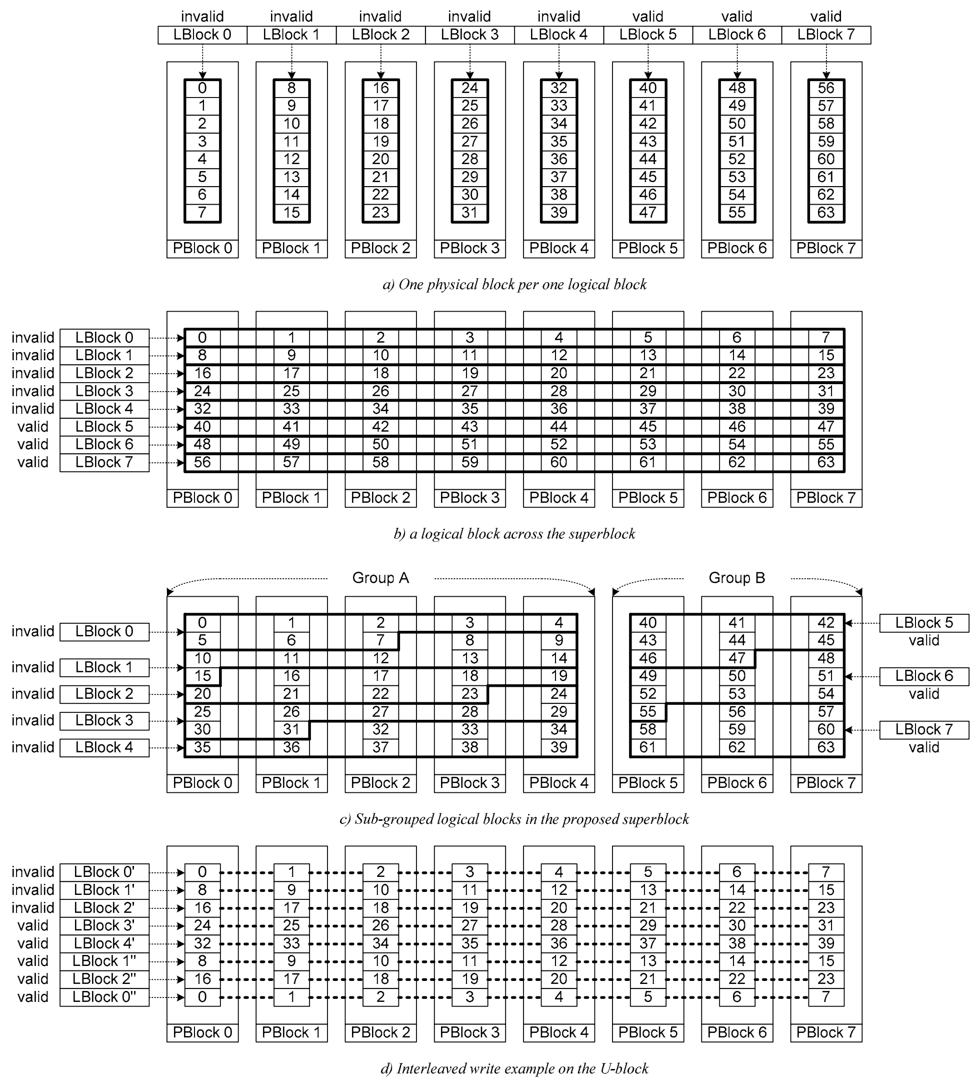


Fig. 2. Block management schemes for multi-bank flash memory.

shows a conventional superblock mapping scheme that maps a logical block across multiple physical blocks. Sequential read and write requests across multiple pages within a superblock can be interleaved. But this superblock management also has a weakness with respect to merge efficiency. Fig. 2 (d) shows an example of a block write sequence on an update-block. After the series of modifications, all of the data superblocks shown in Fig. 2 (b) should be erased, even though LBlocks 5, 6, and 7 were not modified. PBlocks 5, 6, and 7 would not be erased if the blocks were not tied together as a super-block, as was the case in Fig. 2 (a).

Simulation results also show that the total number of erase operations due to merge operations increases linearly: compared with the 2-bank case, the 4-bank super-block has 1.8 times more erase operations; there are 3.3 times more for the 8-bank case, 6.1 times for the 16-bank case, and 11.4 times for the 32-bank case. In general, SSD requires an interleaved superblock scheme to achieve high performance, but flash memory chips are more quickly worn out by the increased number of erase operations that are incurred by the

DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009

superblock mechanism.

3 Separating hot/cold blocks in a superblock

This paper proposes an internal superblock sub-grouping scheme to achieve the performance improvement of interleaving without sacrificing merge efficiency. Specifically, the proposed scheme simply separates the blocks into two sub-groups when a merge operation occurs: those that are likely to be modified again (Group A: hot data) and others (Group B: cold data), as shown in Fig. 2 (c).

Initial block allocations for data-blocks: The initial allocation of data-blocks in the proposed scheme is the same as for the conventional superblock scheme because at allocation time we do not yet have information to indicate any tendency for future access patterns.

Write operations on update blocks: Modification of existing data can occur only via the update-blocks. This operation in the proposed scheme is also the same as the conventional superblock scheme that supports interleaving.

Merging data into a new data-block: When an update-superblock is full, the FTL checks all of the blocks in the data-superblock and the update-superblock to perform the merge operation. Then the modified LBlocks that we expect to be further modified are gathered together as group A. While unmodified LBlocks, which we do not expect to be modified, are collected into group B. These two sub-groups become a new data-superblock after the merge operation.

If the sub-group organization of a new data-superblock is different from that of the previous data-superblock, all of the blocks within the previous data-superblock should be erased during the merge operation as in the conventional scheme. If the two sub-groups have the same organization and a merge operation is required, only the PBlocks in group A are erased for future use (the PBlocks in group B are not erased). Thus, a new data-superblock is constructed by collecting the LBlocks in group A that are valid in the update-superblock and the empty LBlocks in the group B space. The new logical data-superblock is constructed by simply linking the newly updated group A pages in the new data-superblock to previously unmodified group B pages in the previous data-superblock. A new empty logical superblock is obtained by combining those group A PBlocks erased in the previous data-superblock with the empty PBlocks corresponding to group B in the new data-superblock. Therefore, one complete superblock can be returned to the available superblock pool. The interleaving hardware is assumed to allow grouped bank management to support this logical combination of sub-groups into a superblock.

Possible overhead: The overall erase count decreases with the proposed management scheme, but there are additional issues to consider. First of all, our approach requires more table space for recording block addresses. The conventional interleaved architecture requires only one physical address for a superblock and automatically generates the physical block addresses in

© IEICE 2009
DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009

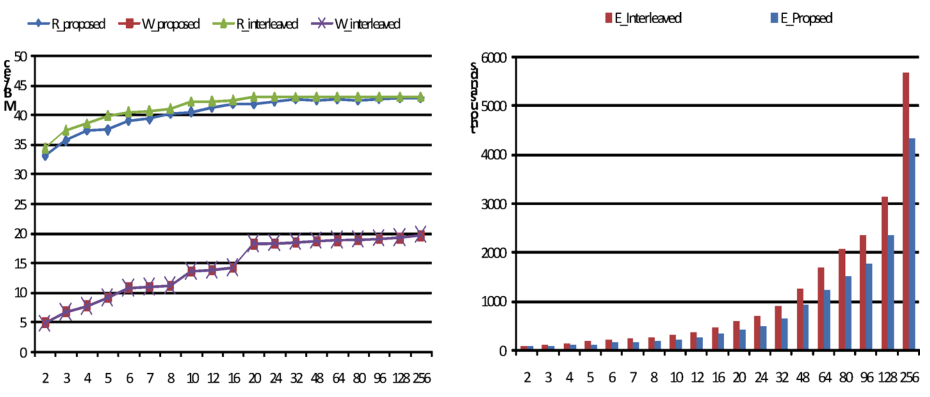
a superblock. In contrast, in the proposed architecture, all physical block addresses must be stored, because fragmented superblocks can be combined with a partial merge. The block table size thus increases from 72 KB to 576 KB when an 8-bank superblock structure in a 128 GB SSD is assumed. That is, the overhead increases by a factor of 8, from 0.00005% to 0.0004%. The increase in block table size may thus be negligible. However, interleaved reading can be limited by sub-grouping. Various applications were evaluated to show that sub-group superblock management can efficiently reduce overall erase count, while still providing spatial locality for good performance.

4 Evaluation and conclusion

A NAND flash memory simulator was developed to determine overall performance and count the erase operations. The simulator is constructed to evaluate a hybrid FTL with various superblock schemes on disk I/O traces [6]. The traces XP_boot, app_ld, general, and file_write were extracted from running the PCmark05 benchmark suite that represents general PC use [5]. Additional traces are gathered from both Windows and Linux environments in our lab: win_upd, install, download, game install, cleanm, com/decom, general1, general2 and startx.

Fig. 3 shows the performance results for the proposed superblock management scheme, compared with the conventional superblock scheme for interleaving. The results shown are for the average of the thirteen traces. The x-axis shows the number of interleaved banks. Fig. 3 (a) shows the performance in MB/sec for read and write operations. Fig. 3 (b) shows the average number of erase operations in thousands. The difference in read performance is expected because dividing a data-superblock into two sub-groups can reduce the effect of interleaving. As a result, read performance decreases by around 2.2% on the average and maximum 5.8% in the 5-bank case. However, the effect on the number of erase operations shows significant improvement, i.e., around 27.3% of erase operations can be avoided on average.

Performance for the write operations is almost the same, because write operations affect only the update-superblocks, which can be fully interleaved in



(a) Average performance results *(b) Overall erase count*

Fig. 3. Performance results of the proposed scheme.

© IEICE 2009
DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009

both schemes. In conclusion, interleaving is useful for achieving high performance I/O in NAND flash based SSD. This paper introduces a sub-grouped superblock management scheme to significantly reduce unnecessary erase operations while preserving most of the benefits of interleaved I/O operations.

Acknowledgments

This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (NO. R01-2007-000-11309-0).

© IEICE 2009
DOI: 10.1587/elex.6.297
Received December 16, 2008
Accepted February 12, 2009
Published March 25, 2009