A Flash Compression Layer for SmartMedia Card Systems

Keun Soo Yim, Hyokyung Bahn, and Kern Koh

Abstract — *Flash memory based SmartMedia Card is becoming increasingly popular as data storage for mobile consumer electronics. Since flash memory is an order of magnitude more expensive than magnetic disks, data compression can be effectively used in managing flash memory based storage systems. However, compressed data management in flash memory is challenging because it only supports page-based I/Os. For example, when the size of compressed data is smaller than the page size, internal fragmentation occurs and this degrades the effectiveness of compression seriously. In this paper, we developed a flash compression layer (FCL) for the SmartMedia Card systems. FCL stores several small compressed pages into one physical page by using a write buffer. Based on prototype implementation and simulation studies, we show that the proposed system offers the storage of flash memory more than 140% of its original size and expands the write bandwidth significantly.* [1]

Index Terms — Flash memory, SmartMedia Card, NAND-type flash memory, data compression, storage expansion.

I. INTRODUCTION

As the computing paradigm is shifting from the desktop to the ubiquitous environment, mobile computing devices such as PDAs, smart cell phones, and digital cameras are becoming increasingly popular [1]. These mobile devices usually employ flash memory as data storage because of its small size, lightweight, shock resistance, and low-power consumption. There are two major types of flash memory products, namely, NAND-type and NOR-type flash memories. NOR-type flash memory is generally deployed as code storage because it offers byte I/O and fast read operations [7]. However, NOR-type flash memory is more expensive than NAND-type flash memory in terms of the cost per byte ratio, and hence NAND-type flash memory is more and more widely used as large data storage such as SmartMedia Card systems.

Since flash memory is a version of EEPROM, there are two performance challenges in flash memory based storage systems [2], [3]. One is the write bandwidth problem. Because a write operation is slow and an erase operation should be preceded before the write operation, the write bandwidth often becomes the performance bottleneck. The other is the storage capacity problem. Since flash memory is an order of

magnitude more expensive than magnetic disks, the limited storage capacity should be managed efficiently.

One of the effective ways to improve these two challenges is adopting data compression techniques [4], [5]. By transferring and storing data in a compressed form, we can expand both the write bandwidth and the storage capacity. However, unlike existing compression techniques that are used for NOR-type flash memory [4], [5], data compression in NAND-type flash memory is difficult to utilize because it only supports *page I/O* [6]. For example, when the size of compressed data is smaller than the page size, internal fragmentation occurs as shown in Fig. 1, and this degrades the effectiveness of compression seriously. The fragmentation area is almost impossible to fill with any small amount of data in practical terms because an erase operation should be preceded for a group of adjacent flash pages and this incurs too much overhead. As a result, when the I/O unit size and the flash page size are identical, there is no benefit of compression in NAND-type flash memory. Note that this phenomenon does not occur in other compression systems such as RAM and NOR-type flash memory because they support *byte I/O*.

In this paper, we present an efficient flash compression layer for NAND-type flash based SmartMedia Card that alleviates the internal fragmentation problem significantly. Based on prototype implementation and simulation studies, we show that the proposed system offers the storage of flash memory more than 140% of its original size, and expands the write bandwidth significantly. Nevertheless, it does not result in any additional read latency in I/O operations.

The remainder of this paper is organized as follows. The next section presents a detailed description of the proposed flash compression layer. Next, the performance of the proposed system is evaluated by prototype implementation and simulations. Finally, we summarize and conclude in the last section.
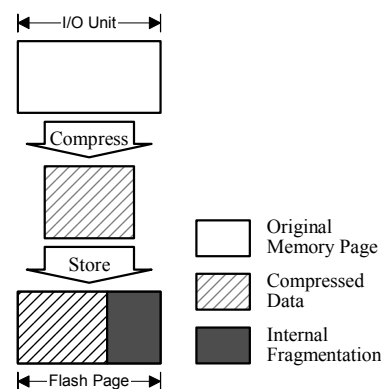

Fig. 1. Internal fragmentation problem.

II. THE FLASH COMPRESSION LAYER

In this section, we present the *flash compression layer* (FCL) for SmartMedia Card systems. We propose two design schemes, namely the *large-unit compression scheme* (LCS) and the *internal packing scheme* (IPS) for better performance.

A. The Large-unit Compression Scheme

A large I/O unit is effective in reducing internal fragmentation. Fig. 2 shows the case when the size of an I/O unit is twice larger than the flash page size. This saves one flash page compared to the case when the I/O unit size and the flash page size are identical. We call this simple scheme the *large-unit compression scheme* (LCS). However, a larger I/O unit induces longer read and write latencies and also incurs additional read and write operations of the adjacent pages for random accessing. Moreover, a larger compression unit needs a more complicated hardware compressor and its associative memory devices.

To address these technical hurdles, we present a novel compressed page management scheme which stores several small compressed pages into one physical page by using a write buffer. We call this scheme the *internal packing scheme* (IPS).

B. The Internal Packing Scheme

In this subsection, we describe the internal packing scheme for the flash compression layer. This scheme compresses each memory page individually, and temporarily maintains a group of the compressed pages in a write buffer (Fig. 3). A write buffer should be non-volatile in order to tolerate failures and it should also support byte I/O. We use a battery backed SRAM for the write buffer because it satisfies the aforementioned conditions.

The grouping of compressed pages aims to minimize internal fragmentation. In this paper, the following three policies are used in the grouping of compressed pages [12].
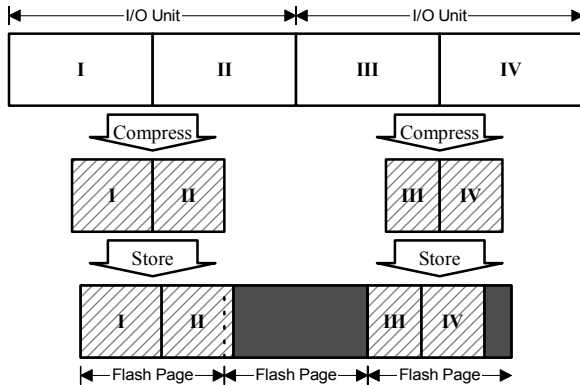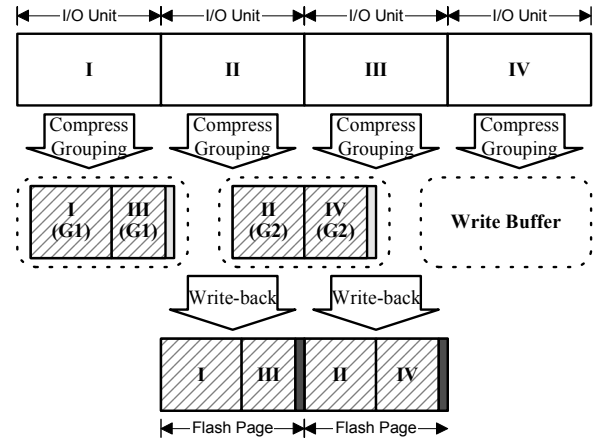

Fig. 3. The Internal Packing Scheme. (I-IV: memory page)

First, the best-fit policy greedily puts the compressed page into a group which induces minimum internal fragmentation after performing the grouping. Second, the worst-fit policy places the compressed page into a group that induces maximum internal fragmentation after performing the grouping. Third, the first-fit policy stores the compressed page into the first group which has enough space. Fig. 4 shows an example of the three policies. With the given write sequence, the figure shows the result of each policy after all writes are performed.

When it is not available to store the incoming compressed page into any of the write buffer groups due to the capacity limit, the internal packing scheme performs a write operation for one of the groups which has the minimum internal fragmentation. The scheme also writes the metadata of the compressed pages simultaneously into a *spare area*, which is attached to each physical page in NAND-type flash memory. The metadata consist of the reverse translation information from the physical page number to the logical page number, the compressed page size, and the error check code (ECC). Since the size of the spare area is either 16 bytes or 64 bytes depending on the product, there is a limitation in the maximum number, namely N, of compressed pages that can be stored in one flash page. Equation (1) shows this limitation.
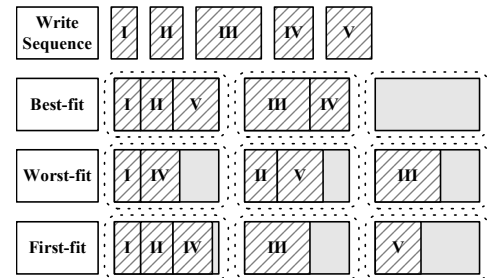

Fig. 2. The Large-unit Compression Scheme. (I-IV: memory page)

Fig. 4. An example of grouping policies.

$$N = \left\lfloor \frac{S_{spare} - S_{ECC}}{\lg \frac{S_{storage}}{S_{page}} + \lg S_{page}} \right\rfloor = \left\lfloor \frac{S_{spare} - S_{ECC}}{\lg S_{storage}} \right\rfloor \qquad (1)$$

where S_{spare} is the spare area size, S_{ECC} is the ECC length, S_{page} is the flash page size, and $S_{storage}$ is the total storage capacity of the flash memory, respectively, in terms of bits.

Fig. 5 shows the block diagram of the flash compression layer with the internal packing scheme. In this paper, we design the flash compression layer on the flash translation layer (FTL) [2], [3], [8] so that the erase operation is logically hided. This architecture has significant benefits in reducing the latency of I/O requests. Write requests can be performed faster because write operations are generally absorbed by the write buffer. Read requests can also be performed faster in both of the two cases. First, if the requested data are in the write buffer, the read operations are performed faster because the write buffer is faster than NAND-type flash memory. Second, if the requested data are not in the write buffer, the NAND-type flash memory should be accessed. However, we can also expect faster read operations in this case because most write requests and a certain fraction of read requests are executed concurrently on the write buffer.

There are various efficient compression algorithms that can be used for high-speed hardware compressors. In this paper, we use the X-RL algorithm [9], [10] because it has some desirable properties such as the good compression ratio with a small compression unit and simple hardware implementation.

The main performance defect of a compressed memory system is usually the decompression time overhead for read operations. However, unlike the on-chip cache and main memory compression systems [13], a read/write operation does not induce extra time overhead in the proposed flash compression layer because the speed of X-RL compressor/decompressor is faster than I/O bus speed, and hence, compression/decompression does not induce extra time overhead.
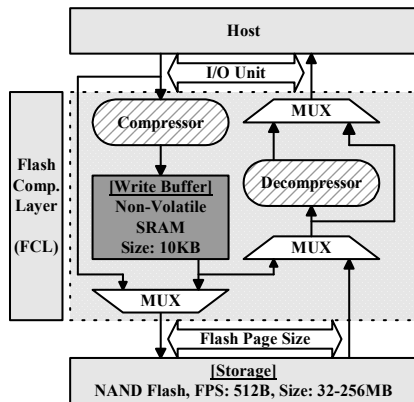

Fig. 6. Prototype of SmartMedia Card with the flash compression layer.

III. PERFORMANCE EVALUATION

To evaluate the performance of the proposed flash compression layer, we implemented a prototype of a SmartMedia Card system that embeds NAND-type flash memory as shown in Fig. 6. We also performed trace-driven simulations to evaluate the performance under various design conditions. We used the *Canterbury Corpus* benchmark, which is a de facto standard for the evaluation of lossless data compression algorithms [11]. Hence, the simulation results can be directly compared with related works that use this benchmark.

As mentioned in Section I, our goal is to expand the storage capacity and write bandwidth by using the flash compression layer. To measure theses two goals, we use the *effective compression rate* as the performance metric, that represents the number of used flash pages over the number of source pages. To compare various design conditions precisely, we define two additional metrics, namely the *compression rate* and the *internal fragmentation rate*. The compression rate is defined as the ratio of the compressed data size to the source data size, thereby, a lower compression rate means the better performance. The internal fragmentation rate is defined as the ratio of the internal fragmentation size to the source data size. Then, the effective compression rate can be obtained by adding the average compression rate and the average internal fragmentation rate.

Fig. 7 shows the compression rate of the benchmark as a function of the compression unit size. As shown in the figure, a larger compression unit generally results in a better compression rate. However, we observed that the compression rate is stabilized when the size of the compression unit is larger than 2K bytes. Hence, to maximize the performance in terms of the compression rate, the size of the compression unit should be at least 2K bytes. However, a larger compression unit induces longer I/O latencies and also requires a more complicated hardware design.

Since the I/O unit size and the flash page size are equally 512 bytes in the current NAND-type flash memory products, we use 512 bytes as the compression unit size of the internal packing scheme. In the case of the large-unit compression

Fig. 5. Block diagram of the flash compression layer when the internal packing scheme is used.

scheme, we use 1K bytes and 2K bytes as the size of the compression unit for better performance in terms of the compression rate and the internal fragmentation rate. Note that these sizes are twice and four times larger than the flash page size.
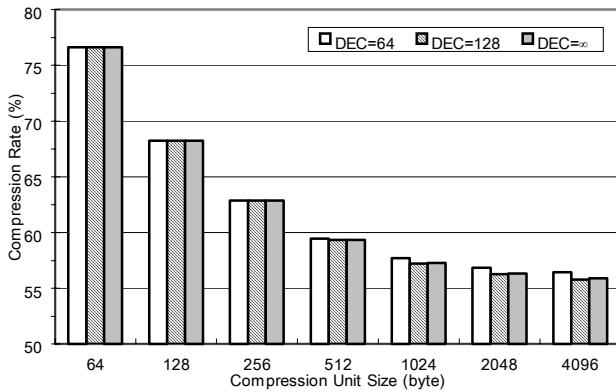

Fig. 7. The compression rate of the flash compression layer as a function of the compression unit size and the dictionary entry count (DEC) of the X-RL compressor.

The figure also provides the compression rate with different dictionary sizes as shown in the legend. The dictionary is used in the X-RL compressor to find the correlation between source data and compressed data. The experiment shows that the dictionary size of 128 entries is enough to achieve the best compression rate. Since one dictionary entry needs only 4 bytes, the total dictionary size is 512 bytes and this does not induce the crucial cost problem even though the dictionary is maintained in SRAM.

To manage the compressed pages efficiently in NAND-type flash memory, internal fragmentation should also be minimized because I/O operations are performed in a page basis. Fig. 8 shows the measured internal fragmentation rate as a function of the I/O unit size and the flash page size when the large-unit compression scheme is used. Since the flash page size (FPS) is 512 bytes in the current products, the internal fragmentation rates are 26.2% and 20.9%, respectively, when the I/O unit sizes are twice and four times larger than the flash page size.
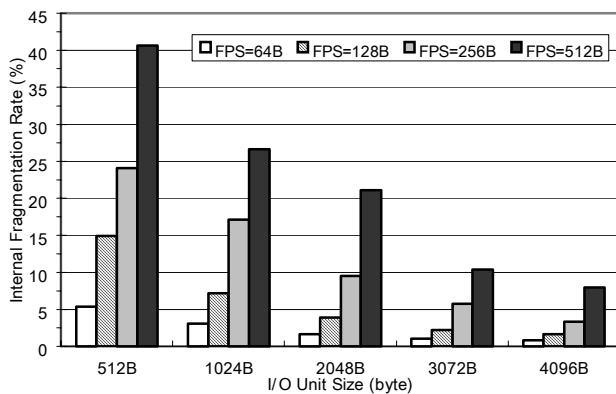

Fig. 8. The internal fragmentation rate of the large-unit compression scheme as a function of the I/O unit size and the flash page size (FPS).

The figure also shows that a larger I/O unit and a smaller flash page result in a better internal fragmentation rate. However, just a simple use of a large I/O unit or a small flash page causes another critical problem. A large I/O unit incurs long latency and additional I/O operations for adjacent pages, and a small flash page requires not only a more complicated CMOS layout but also a larger address translation table for FTL.

The internal packing scheme is an alternative approach of reducing internal fragmentation without changing the underlying sizes of the I/O unit and the flash page. Fig. 9 shows the internal fragmentation rate of the internal packing scheme with different grouping policies as a function of the write buffer size. In the figure, the bold lines represent the average internal fragmentation rates of the large-unit compression scheme whose I/O unit sizes are 1K bytes and 2 K bytes, respectively. As can be seen from the figure, the internal packing scheme always induces smaller amount of internal fragmentation than the large-unit compression scheme though it uses the smaller I/O unit size of 512 bytes.

In terms of grouping policies, the figure shows that the best-fit policy performs the best among the three policies. We also observed that the internal fragmentation rate of the internal packing scheme converges when the write buffer size is larger than 160K bytes. Hence, we use a write buffer of 160K bytes in the implementation.
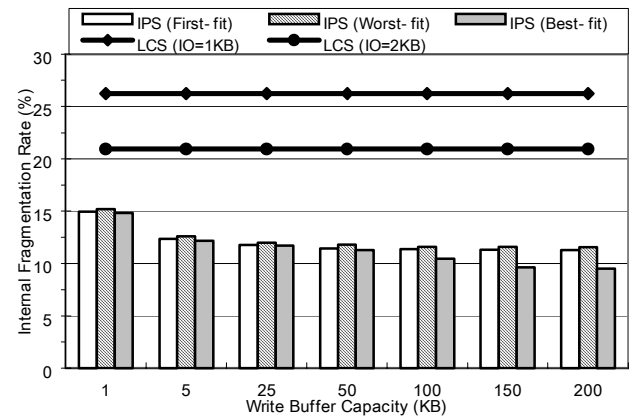

Fig. 9. The internal fragmentation rate of the internal packing scheme with different grouping policies as a function of the write buffer size. LCS = large-unit compression scheme; IPS = internal packing scheme; IO = I/O unit size.

Fig. 10 shows the internal fragmentation rate of the flash compression layer with various applications of the benchmark. As shown in the figure, the internal packing scheme outperforms the large-unit compression scheme for most cases. Specifically, the internal packing scheme shows better performance than the large-unit compression scheme when the application shows good compression rates. For example, the internal fragmentation of the *fax* benchmark is almost eliminated with the internal packing scheme because of its superb compression rate of 16.0%.
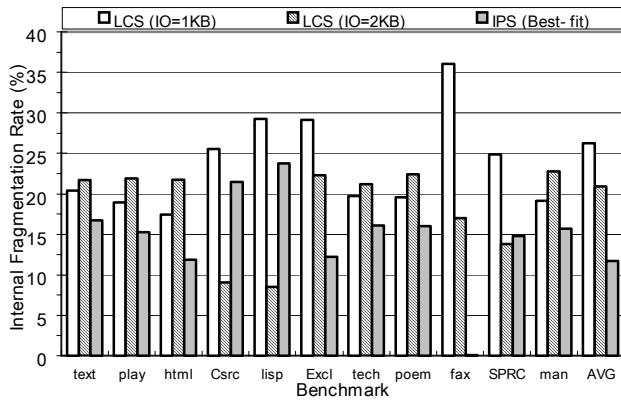
Fig. 10. The internal fragmentation rate of various applications and their average (AVG). LCS = large-unit compression scheme; IPS = internal packing scheme; IO = I/O unit size.

Table I summarizes the performance of the flash compression layer in terms of various performance metrics. Since the internal packing scheme uses the I/O unit of 512 bytes, it shows a slightly worse compression rate than the large-unit compression scheme that uses larger I/O units of 1K bytes and 2K bytes. However, the internal packing scheme reduces internal fragmentation significantly compared to the large-unit compression scheme. As a result, the internal packing scheme shows better effective compression rate than the large-unit compression scheme. We calculate the expansion rate of the storage capacity and the write bandwidth. The expansion rate of the internal packing scheme is over 40% and those of the large-unit compression schemes are in the range of 20% and 30%, respectively.

TABLE I
PERFORMANCE SUMMARY OF THE FLASH COMPRESSION LAYER.

Scheme	Compression rate	Internal fragmentation rate	Effective compression rate	Storage and bandwidth expansion rate
LCS (IO=1KB)	57.3%	26.2%	83.5%	19.8%
LCS (IO=2KB)	56.3%	20.9%	77.2%	29.5%
IPS (Best-fit)	59.3%	11.7%	71.0%	40.8%

IV. CONCLUSION

Internal fragmentation degrades the effectiveness of data compression significantly in NAND-type flash memory because it supports only page I/O. In this paper, we have proposed an efficient flash compression layer for the NAND-type flash based SmartMedia Card system that aims to minimize the internal fragmentation and maximize the effectiveness of compression technology. We have presented two design techniques: the large-unit compression scheme that reduces the internal fragmentation by enlarging the compression unit and the internal packing scheme that stores several small compressed pages into one flash page by using a write buffer. Prototype implementation and simulation

experiments have shown that the internal packing scheme offers the storage of flash memory more than 140% of its original size and expands the write bandwidth significantly.

Recently, larger storage devices such as NAND-type flash memory type-II [6] emerge which support the larger I/O unit size of 2K bytes. For future work, we are planning to study and analyze these new products and design an efficient compression scheme that considers new characteristics of these products. We also plan to unify the flash compression layer (FCL) and the flash translation layer (FTL) for better performance.

REFERENCES

[1] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J. A. Tauber, "Storage Alternatives for Mobile Computers," *In Proceedings of the 1st Symposium on Operating System Design and Implementation*, pp. 25-37, 1994.

[2] M. Wu and W. Zwaenepoel, "eNVy: A Non-Volatile, Main Memory Storage System," *In Proceedings of the 6th Symposium on Architectural Support for Programming Languages and Operating Systems*, pp. 86-97, 1994.

[3] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFlash Systems," *IEEE Transactions on Consumer Electronics*, Vol. 48, No. 2, pp.366-375, 2002.

[4] M. Kjelso and S. Jones, "Memory Management in Flash-Memory Disks with Data Compression," *Lecture Notes in Computer Science*, Vol. 986, Springer Verlag, pp. 399-413, 1995.

[5] S. Wells and D. Clay, "Flash Solid-State Drive with 6MB/s Read/Write Channel and Data Compression," *In Proceedings of the 40th International Conference on Solid-State Circuits*, pp. 52-53, 1993.

[6] Samsung Electronics, "256M x 8 Bit / 128M x 16 Bit NAND Flash Memory," http://www.samsungelectronics.com/.

[7] Intel Corporation, "3 Volt Synchronous Intel StrataFlash Memory," http://www.intel.com/.

[8] Intel Corporation, "Understanding the Flash Translation Layer (FTL) Specification," http://developer.intel.com

[9] S. Bunton and G. Borriello, "Practical Dictionary Management for Hardware Data Compression," *Communications of the ACM*, Vol. 35, No. 1, pp. 95-104, 1992.

[10] M. Kjelso, M. Gooch, and S. Jones, "Design and Performance of a Main Memory Hardware Data Compressor," *In Proceedings the 22nd Euromicro Conference*, IEEE Computer Society Press, pp. 422-430, 1996.

[11] R. Arnold and T. Bell, "A Corpus for the Evaluation of Lossless Compression Algorithms," *In Proceedings of the 7th IEEE Data Compression Conference*, pp. 201-210, 1997.

[12] A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating System Concepts*, 6th Ed., pp. 285-287, John Wiley & Sons Inc., 2003.

[13] K. S. Yim, J. S. Lee, J. Kim, S. D. Kim, and K. Koh, "A Novel Cache Organization Technique for Aggressively Compressed Memory Hierarchy," *Technical Report SNU-CSE-OSL-0301*, Seoul National University, 2003.

Keun Soo Yim received the BE degree in electronic engineering and BS degree in computer science from Yonsei University, Korea, as an informatics specialist. He is currently with the school of electrical engineering and computer science at Seoul National University, Korea. His research areas include operating systems, computer architectures, and network systems. He is currently interested in memory compression technologies for the on-chip cache, main memory, and flash memory.

Hyokyung Bahn received the BS, MS, and PhD degrees in computer science from Seoul National University, Korea, in 1997, 1999, and 2002, respectively. He is currently an assistant professor in the department of computer science and engineering, Ewha University, Seoul, Korea. His research interests include operating systems, caching algorithms, Web technologies, reference behavior modeling, genetic algorithms, and distributed systems. Dr. Bahn is a member of the IEEE Computer Society, the IEICE, and the Korea Information Science Society.

Kern Koh received the BE degree in applied physics from Seoul National University, Korea, in 1974, and the MS and PhD degrees in computer science from University of Virginia, United State, in 1979 and 1981, respectively. He was a researcher of Bell Laboratory of AT&T from 1981 to 1983. He is currently a professor in the school of electrical engineering and computer science, Seoul National University, Korea, from 1983. His research interests include operating systems, distributed systems, and computer system performance analysis. Dr. Koh is a member of the IEEE, the ACM, and the Korea Information Science Society.