W-Buddy: Wear-Out-Aware Memory Allocator for SCRAM-based Low Power Mobile Systems

Dongyoung Seo
Sungkyunkwan University
Suwon, Korea
kkaka@skku.edu

Dongkun Shin
Sungkyunkwan University
Suwon, Korea
dongkun@skku.edu

Abstract— Since recently emerging storage class RAM (SCRAM) devices are power-efficient, byte-addressable, and non-volatile, they are expected to replace the power-hungry DRAM in mobile consumer devices. However, SCRAM has a limited number of program and erase cycling, requiring a wear-leveling technique. Since the locality in memory access pattern may intensify the discriminated wearing of memory blocks, it is required to change the operating systems such that memory accesses are distributed evenly over all memory space for wear-leveling. We propose a novel memory allocator, called W-Buddy, which selects a free memory chunk to be allocated considering the wear-outs of memory chunks. Our experimental results show that the proposed W-Buddy achieves fourteen time longer lifetime compared with the conventional buddy memory allocator.

Keyword: SCRAM, Wear-leveling, Memory Allocator

I. INTRODUCTION

For battery-powered mobile consumer devices such as mobile phone and mobile pad, power consumption is an important metric. Since DRAM must be refreshed periodically, it is one of power-hungry devices in the mobile system. Therefore, low-power mobile DRAMs are generally used to minimize the power consumption. Recently, as a next-generation non-volatile memory, the storage class RAM (SCRAM) such as phase-change RAM (PCRAM), spin-transfer-torque magnetic RAM (STT-MRAM), and resistive RAM (RRAM) is emerging to replace the power-hungry DRAM [1]. Since SCRAM is byte-addressable and has a shorter latency than NAND flash memory, it is reasonable to use SCARM as main memory. The non-volatility of SCRAM will reduce overall power consumption of mobile devices as well as simplify the system software, which is now inevitably complex in order to handle the volatile main memory.

However, the memory cells of SCRAM have a limited lifetime because they wear out at every program-and-erase operation. The limited lifetime is critical to use SCRAM for main memory. Especially, the main memory access pattern is highly localized making a significant imbalance on the wear-outs of memory cells. Therefore, a wear-leveling technique is essential to balance the wear-outs of different memory cells. Recently, many wear-leveling techniques for SCRAM have been proposed. However, most of them are device-oriented, which means the wear-leveling module is implemented within memory controller and it monitors the write pattern from host and takes some action for wear-leveling. For example, Dhiman *et al.* [2] suggested a memory controller which maintains the update counts of every 4 KB memory page for wear-leveling purpose. StartGap [3] technique periodically rotates the logical-to-physical address mapping in memory controller to change the memory page allocated for update-intensive data.

For a more active technique, we can consider a host-driven wear-leveling technique which changes the host access pattern to distribute the write requests evenly over all memory space. The host-driven wear-leveling can be used along with the device-driven wear-leveling making a synergetic effect.

Unfortunately, since current operating systems are designed for DRAM-based systems, they have no consideration on the wear-leveling for main memory. For example, the buddy memory allocator of Linux operating system allocates the first free memory chunk suitable for the request size without any information on the wear-outs of memory blocks. Therefore, write requests are likely to be concentrated on only a portion of overall memory blocks. To avoid such imbalance on wear-outs, we should revisit the operating systems considering the wear-leveling issue of SCRAM.

In this paper, we propose a wear-out-aware memory allocator for SCRAM-based mobile systems. The buddy memory allocator is redesigned such that it allocates a less-worn-out memory block. To the best of our knowledge, this is the first study to reform operating systems considering the wear-leveling issue of SCRAM. The experiments show that the wear-out-aware memory allocator can increase the lifetime of SCRAM significantly with a negligible overhead.

II. WEAR-OUT-AWARE MEMORY ALLOCATOR

Linux uses the buddy memory allocator for physical memory management. Every memory chunk in the buddy allocator is managed based on the order of the chunk. The size of memory chunk in the *n-th* order is $2^n \times S$ where S is the smallest chunk size. All memory chunks are managed by linked lists according to their orders. The buddy allocator chooses a memory chunk to be allocated by finding a free chunk from the head of the corresponding order list. If a chunk is deallocated, it is attached to the head of the corresponding linked list. Therefore, if an application repeats memory allocation and deallocation frequently, the recently-allocated memory chunk will be reallocated making a significant imbalance on the wear-outs of memory chunks.

We propose a novel wear-out-aware buddy allocator, called W-Buddy, for SCRAM-based memory. W-Buddy manages the update counts of memory chunks and prefers to allocate a less-worn-out chunk to balance the wear-outs of memory chunks.

978-1-4673-1501-2/12/$31.00 ©2012 IEEE

Fig. 1 shows how W-Buddy manages memory chunks. If we assume the total memory size is 32 KB and the smallest allocatable chunk size is 4 KB, the memory is managed at four different orders from 0 to 3. The chunk in the order *i* can be divided into two buddy sub-chunks in the order *i*-1, and vice versa. For a memory allocation request, W-buddy selects one of free chunks from the chunk list in the proper order.

$C_{i,j}$ denotes the *j-th* memory chunk on the order *i*. For each chunk, W-Buddy manages the special information of (N_p, S_{alloc}) so as to search a less-worn-out free chunk efficiently. N_p is the update count of the chunk and S_{alloc} is the allocation status bitmap for the chunk and the sub-chunks at sub-levels. The N_p of a chunk is determined by the N_p of its sub-chunks. If both the sub-chunks of $C_{i,j}$ are free, the N_p of $C_{i,j}$ is set to the sum of update counts of sub-chunks. If only one of sub-chunks is free, the N_p of $C_{i,j}$ is set to two times the N_p of the free sub-chunk. For example, since $C_{2,0}$ has only one free sub-chunk $C_{1,1}$, the N_p of $C_{2,0}$ is 14 (= 2×7). The S_{alloc} of $C_{2,0}$ is (100), which means $C_{2,0}$ is not free but a free sub-chunk exists in the order of 1 and 0.

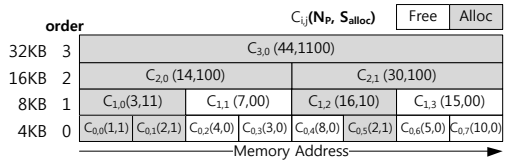

Figure 1. The memory buddy chunk management in W-Buddy

The memory allocation process is as follows: If a 4 KB memory chunk should be allocated assuming the initial memory state in Fig. 1, W-Buddy first examines the S_{alloc} of the highest order chunk $C_{3,0}$. Since the S_{alloc} is (1100), there exists a free chunk in the order 0 where 4 KB memory chunks are managed. Then, by checking the S_{alloc} of sub-chunks, W-Buddy can know that both $C_{2,0}$ and $C_{2,1}$ have a free 4 KB sub-chunk. W-Buddy compares the update counts of $C_{2,0}$ and $C_{2,1}$. Since the update count of $C_{2,0}$ is less than that of $C_{2,1}$, W-Buddy further examines the sub-chunks of $C_{2,0}$. By the searching process, W-buddy will select the chunk of $C_{0,3}$ finally.

If there are memory chunks which have cold or hot data and are not deallocated during a long period, the memory chunks cannot be utilized by the allocation-based wear-leveling. Therefore, if the difference on update counts between the most-worn-out chunk and the least-worn-out chunk exceeds a predefined threshold, W-buddy invokes a compulsory wear-leveling which swaps the cold data and hot data in the chunks.

To maintain the update counts of memory chunks, we use a hardware counter in the memory controller. If the counter monitors all memory write requests and reports them to operating system via interrupt in order to maintain the exact update counts of all memory chunks, there is too much overhead for the hardware counter capacity and interrupt handling. Instead, we use a sampling-based approach, where the memory controller checks the current updated memory chunk and inform OS of the information only at every the sampling period.

III. PERFORMANCE EVALUATION

We used a trace-driven SCRAM simulator and several real-world memory traces to evaluate the wear-leveling performance. We assumed the SCRAM-based memory has the size of 1 MB. The wear-leveling performances of three memory allocators, Ideal, Buddy, and W-Buddy, are evaluated. While the Buddy allocator does not consider the wear-leveling, the Ideal and the W-Buddy allocator perform the wear-leveling. The Ideal allocator counts all memory update requests and allocates the least-worn-out memory chunk. In addition, it supports the compulsory wear-leveling. For W-Buddy, we used the sampling ratio of 100 K update requests.

Fig. 2 shows the distribution of memory chunk update counts under different memory allocators. The Buddy allocator shows significant differences on update counts resulting from the spatial locality of memory update pattern. W-Buddy reduced the standard deviation of update count distribution by 87% and the maximum update count by 93% compared to the Buddy allocator. However, the average update counts are increased by the compulsory wear-leveling. The wear-leveling performance of W-Buddy is lower than the Ideal allocator due to its sampling-based approach. However, W-Buddy has negligible hardware and interrupt overheads while the Ideal allocator makes an interrupt to OS at every memory update.

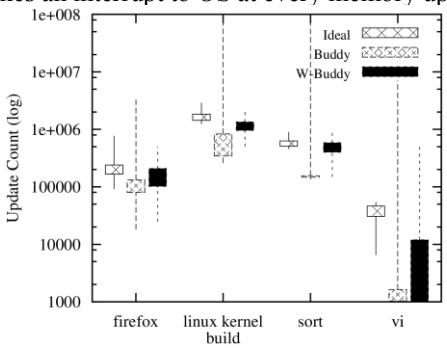

Figure 2. The distribution of memory chunk update counts (The box represents the 25th and 75th percentile, and the upper whisker and lower whisker are the maximum and the minimum update count, respectively)

IV. CONCLUSION

The current operating systems are designed assuming DRAM-based memory devices. Therefore, many subsystems in operating systems should be redesigned for future low-power SCRAM-based memory devices. Especially, the wear-leveling will be an essential feature of memory allocator. We proposed a low-overhead wear-out-aware memory allocator, W-Buddy, as the first study on revisiting operating systems for SCRAM. From experiments, the proposed W-Buddy allocator showed significant improvements on the lifetime of SCRAM.

V. ACKNOWLEDGMENT

This research was partly supported by Mid-career Researcher Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2012-0027613).

REFERENCES

[1] R. Freitas and W. Wilcke, "Storage-class memory: The next storage system technology," IBM Journal of Research and Development, vol. 52, issue 4/5, pp. 439–447, 2008.

[2] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: a hybrid PRAM and DRAM main memory system," In Proc. of the 46th Annual Design Automation Conference, pp. 664–469, 2009.

[3] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based Main Memory with Start-Gap Wear Leveling," In Proc. of the 42nd Annual International Symposium on Microarchitecture, pp. 14–23, 2009