Page Allocation Scheme for Anti-Fragmentation on Smart Devices

Jaewon Kim[1,2], Changwoo Min[1,2], Jeehong Kim[1], Dong Hyun Kang[1], Inhyeok Kim[1], Young IK Eom[1]
Sungkyunkwan University, Republic of Korea[1] Samsung Electronics, Republic of Korea[2]
{delicious[1], multics69[1], jjilong[1], kkangsu[1], kkojiband[1], yieom[1]}@skku.edu, {jaewon31.kim[2], changwoo.min[2]}@samsung.com

***Abstract*—In embedded smart devices, efficient memory management is critical because they have relatively small main memory to reduce cost and power consumption. Though Input Output Memory Management Unit (IOMMU), which is recently adopted on embedded smart devices, gives more free spaces in system memory, it increases memory allocation time when memory space is highly fragmented. In this paper, we propose a new page allocation scheme, called A-GPBM, to reduce fragmentation of anonymous pages and secure more physically contiguous pages. Experimental results show that our algorithm decreases unusable free space index for order 4 from 0.93 to 0.32.**

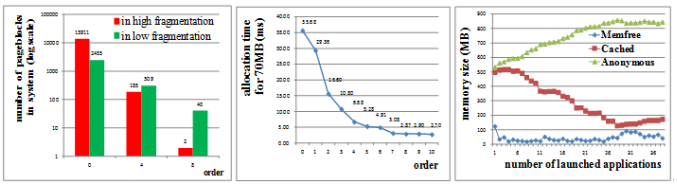
***Keywords*—fragmentation; buddy allocator; hot-cold cache; input output memory management unit**

I. INTRODUCTION

In embedded smart devices, fast response is required when applications run. However, embedded smart devices have limitations on computing power, battery capacity, and memory capacity. Especially, since delay in memory allocation causes slow response, memory management scheme that efficiently utilize small memory space is necessary[1].

Peripheral processors such as Graphics Processing Unit (GPU) and Image Signal Processor (ISP) access memory directly and need chunks of physically contiguous memory. However, when contiguous memory is insufficient due to fragmentation, allocating big chunks of memory is difficult. Therefore, Operating System (OS) used to solve this problem by reserving chunks of contiguous memory at booting time. But, such reserved memory is one reason of memory shortage since the only pre-determined processes are authorized to use the region. Input Output Memory Management Unit (IOMMU) is the HW device that translates address for Direct Memory Access (DMA) capable processors. Although IOMMU was designed to support virtualization, adopting IOMMU on smart devices concomitantly resolves the reserved memory problem since OS does not need reserved memory any longer. Therefore, it increases free spaces in system memory[4].

However, IOMMU-based applications should get large chunks of memory from buddy allocator since the memory was not reserved, where this situation incurs memory allocation overhead. Especially, if it is allocated with small memory chunks, it takes more allocation time because allocation process should be repeated more. In order to discover the mentioned overhead on real devices, we performed motivation experiments. Fig. 1(a) shows that, in high fragmentation, the allocation is performed with lower order. Fig. 1(b) shows that allocation with small size takes more time. Moreover, Fig. 1(c) shows memory status during



(a) Allocation in camera startup (b) Allocation time with order (c) Memory in application startup
Fig. 1. Motivation experiments

launching several applications, and explains that there is plenty of anonymous pages overall. In this paper, to mitigate the fragmentation more effectively, we propose a new page allocation scheme with the consideration on anonymous page.

Previous researches to mitigate fragmentation are categorized to defragmentation and anti-fragmentation. Defragmentation such as Compaction[2] creates contiguous pages when contiguous pages are unavailable. Anti-fragmentation such as Grouping Pages By Mobility (GPBM)[3] allocates and frees in a way keeping fragmentation low. However, defragmentation has scanning and migration overhead causing delay at allocation time. Meanwhile, anti-fragmentation has additional procedure at every allocation and freeing.

II. A-GPBM ALGORITHM

GPBM manages and allocates free pages by classifying free pages into three types. (1) Unmovable: cannot be migrated, most of kernel core, (2) Reclaimable: can be reclaimed easily, inode and directory caches, (3) Movable: can be migrated, anonymous page and page cache[3].

We focus on different deallocation time between anonymous page and page cache although both of them are categorized into Movable. To speed up future access, pages for page cache are not deallocated although a process using them terminates. Then they are reclaimed when free spaces in system memory become scarce. Meanwhile, all pages used for anonymous pages are deallocated just after a process using them exists. We propose Anonymous page aware GPBM (A-GPBM) algorithm to mitigate fragmentation. A-GPBM allocates physically contiguous pages for anonymous pages mapped into a process so that the number of contiguous free pages increases when the process terminates.

To allocate contiguous pages for anonymous pages, A-GPBM uses 3 techniques. First, in order to avoid that some of contiguous pages are allocated for page cache during contiguous allocation for anonymous page, A-GPBM distinguishes free page allocations between for anonymous page and for page cache. Fig. 2. depicts the flow of single page allocation through A-GPBM. Page cache allocation is handled by the Movable List since page cache does not need contiguous

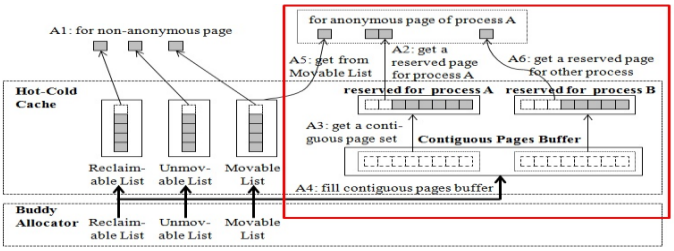


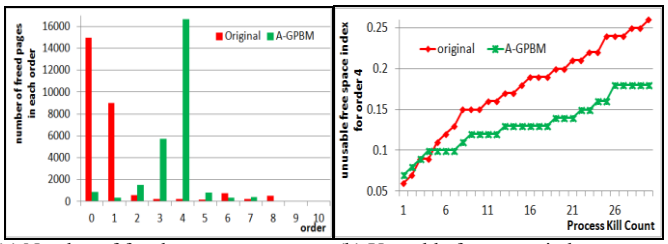
Fig. 2. The flow of single page allocation in A-GPBM

allocation (A1). Allocation for anonymous page is performed by the Contiguous Pages Buffer (CPB) to allocate contiguously. The CPB contains physically contiguous pages and manages them not to be mixed. If the CPB is empty when an allocation request arrives, the CPB is filled with contiguous pages brought from buddy allocator (A4). Second, to allocate contiguous pages for anonymous pages mapped into one process, even when requests for anonymous page from several processes are timely interleaved, A-GPBM classifies allocation requests for anonymous page according to requesting process. A-GPBM reserves contiguous pages for a process, and allocates one of those reserved pages if allocation for anonymous page is requested (A2). If reserved page is unavailable, a new contiguous page set is taken from the CPB and the set is reserved for the requesting process (A3). If the CPB is empty and cannot be filled from buddy allocator, then A-GPBM allocates a free page from either Movable List (A5) or reserved pages for other process (A6). On the contrary to the allocation, if contiguously allocated pages are freed back, those pages are merged as reserved pages and finally moved to buddy allocator through the CPB. Third, when the CPB is filled (A4), A-GPBM takes all pages within a collapsed higher order in buddy allocator to finally mitigate fragmentation in buddy allocator. Traditional hot-cold cache requests order 0 page in 31 times, and buddy allocator splits a higher order if order 0 page does not exist. Because the next contiguous page in the collapsed order is allocated to the subsequent request, some of 31 pages in hot-cold cache can be contiguous. However, the limitation on 31 pages does not guarantee that all pages in the lastly collapsed order move to hot-cold cache at the one filling process. In order to keep more contiguous pages, A-GPBM brings all pages in a collapsed order, up to order 4, into the CPB though the number of pages exceeds 31.

III. Experimental Results

To verify our algorithm, we extended GPBM by modifying algorithm of buddy allocator and hot-cold cache, and performed two experiments on an Android mobile phone with Linux Kernel 3.4. To make certain that fragmentation level improves, we used unusable free space index Gorman and Whitcroft proposed[3]. The index is a ratio of unavailable memory size to total memory size in system for allocation of a requested order of pages. Smaller index means lower fragmentation level and higher availability of contiguous pages allocation. Consequently massive memory can be allocated faster in memory status where the index is low.

We performed the first experiment by implementing two user programs having same algorithm, where requests 4KB memory in every 1ms, 100MB in total. We launched the two programs at the same time and checked contiguity of freed pages after terminating a program. In Fig. 3(a) the x-axis



(a) Number of freed pages in each order in micro benchmark test

(b) Unusable free space index for order 4 in LMK test

Fig. 3. Comparison of fragmentation

means order of contiguous pages and the y-axis means number of merged pages in each order. We observed that difference from order 5 to 10 is negligibly small, meanwhile, A-GPBM caused pages in order 0, 1 to be merged into order 2, 3, and 4. As IOMMU device drive, motivation in this paper, tries memory allocation in the order of order 8, 4, and 0, we determined order 4 as a standard order for the unusable free space index. Then the index on order 4 drastically decreased from 0.93 to 0.32, meaning only 32% of pages under order 4.

In the second experiment, we checked increment of the index on order 4 during consecutive invocations of Android Low Memory Killer (LMK). Because most of freed pages from killed process were for anonymous page, we can verify improvement of A-GPBM by comparing the index after killing a process. In Fig. 3(b) we observed that the index in A-GPBM slowly increased, meaning slowly fragmented. It also means IOMMU-based new applications can be active more quickly because more contiguous pages are generated when a process is killed by LMK in memory shortage situation.

IV. Conclusion

Memory management is important and impacts system performance. In this paper, we proposed a new memory allocation scheme which mitigates memory fragmentation by allocating contiguous pages for anonymous pages. With the proposed algorithm, unusable free space index for order 4 drastically decreased from 0.97 to 0.34 in our micro benchmark test. Since hot-cold cache has been implemented with per-CPU page, we will do additional research for multi core environment.

V. Acknowledgment

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2010-0020730). Young Ik Eom is the corresponding author of this paper.

References

[1] G. Lim, C. Min, Y. I. Eom, "Virtual memory partitioning for enhancing application performance in mobile platforms," Consumer Electronics, IEEE Transactions on, 2013.

[2] M. Gorman and A. Whitcroft, “Supporting the allocation of large contiguous regions of memory,” Ottawa Linux Symposium, 2007.

[3] M. Gorman and P. Healy, “Supporting superpage allocation without additional hardware support," International Symposium on Memory Management, 2008.

[4] R. Mijat and A. Nightingale, "Virtualization is coming to a platform near you," ARM White Paper, 2011.