Optimizing Power Consumption of Memory Deduplication Scheme

Jinwoo Ahn and Dongkun Shin
College of Information and Communication Engineering
Sungkyunkwan University
Suwon, Korea
creator.ahn@gmail.com, dongkun@skku.edu

Abstract — **Recent mobile consumer devices are suffering from limited memory and power consumption. The deduplication technique will be helpful to reduce memory footprint by identifying the same content memory pages. Linux is adopting the Kernel Samepage Merging (KSM) scheme for memory page deduplication. However, current KSM can invoke significant power consumption due to its inefficient scanning. In consumer devices, if a page is not merged with other pages during the last several scanning rounds, it is less likely to be merged at the following scanning rounds. Considering such a feature, a novel KSM scanning technique is proposed, called N-chance partial scanning. Experimental results show that the proposed technique can reduce the power consumption by up to 67.6%.**

Keywords—memory; deduplication; power consumption; KSM;

I. INTRODUCTION

Recent mobile devices such as smartphone and tablet computer require a large capacity of main memory to support more complex and versatile applications. For example, recent smartphones are equipped with 2-GB DRAM. Although recent mobile smartphones are equipped with a larger capacity of main memory, they are still memory-hungry since each application requires more working memory size. It is not easy for device manufactures to increase the size of DRAM further, since the mobile devices are sensitive to power consumption, cost, and device size.

The page sharing scheme can be used for a memory over-commitment technique. If multiple memory pages have the same content, the memory footprint can be reduced by sharing one physical page. The Kernel Samepage Merging (KSM) kernel module in Linux uses a scanning based mechanism to detect the duplicated pages [1]. The KSM kernel-thread daemon, called *ksmd*, periodically scans the heap memory region looking for identical pages. Currently, it can handle only anonymous memory pages.

KSM maintains two red-black-trees (RB-trees) as its main data structures. The stable tree stores shared pages that have already been merged, while the unstable tree records pages that are not yet merged but sharing candidates because they do not change frequently. For every scanned page, KSM searches the same content page in the stable tree first. If the same content page is found, the scanned page is merged into the stable tree. Page sharing is implemented by replacing the page-table-entries of the duplicate pages with a shared KSM page. The shared page is marked copy-on-write (COW), and thus any modifications to the shared page will generate a trap and breaks the sharing.

If the same content page is not found from the stable tree, the unstable tree is examined. Before searching in the unstable tree, the checksum value of the scanned page is calculated. If the calculated checksum value differs from the one recorded in the previous scan round, the checksum value record is updated and the scan continues with the next page. This is to prevent KSM from consuming CPU time at frequently-changed pages, which are called volatile pages in KSM. If the checksum has no change, the same content page is searched in the unstable tree, the sharing pages are merged and inserted into the stable tree. If there is no same content page in the unstable tree, the page is inserted into the unstable tree to be a candidate for page sharing with other pages. The pages in the unstable tree are not protected from being modified.

The scanning-based comparison process of *ksmd* goes on repeatedly, and *ksmd* scans pages incrementally. The scanning can be computing intensive and it is important to tune KSM to the current workload. Users can configure how many pages must be scanned after a certain period of sleep time, which represent in fact the maximum merging rate. Considering that the battery-backed mobile devices are generally power-hungry as well as memory-hungry, too frequent memory scanning should be avoided to prevent too large power consumption overhead of KSM. However, infrequent scanning will fail to find short-lived identical pages, and thus it will not be effective in reducing memory footprint.

In order to minimize the KSM overhead without reducing memory deduplication gain, it is important to focus on the memory pages that have large chances of page sharing. We propose a low-overhead KSM scanning technique, called *N-chance partial scanning*, which skips the scanning of the pages that have not been merged during the past *N* number of KSM scan rounds.

II. N-CHANCE PARTIAL SCANNING

Since KSM excludes frequently-updated (volatile) pages from the page sharing candidates, most of the sharing candidates are not updated during the repeated scanning rounds. Therefore, if a page is not merged with other pages during the last several scanning rounds, it is less likely to be merged at the following scanning rounds. The N-chance partial scanning counts the number of failed scan rounds of each page, and skips the searching on the same content page for the pages that exhausted N number of chances. If the page content is

modified or the page is merged with other sharing pages, the count value is initialized.

Fig. 1 shows the distribution on the number of scan rounds where a page is merged with other pages by KSM. More than 98% of the deduplicated pages are merged within three scan rounds by KSM. From this result, we can know that three chances are sufficient to find identical pages.

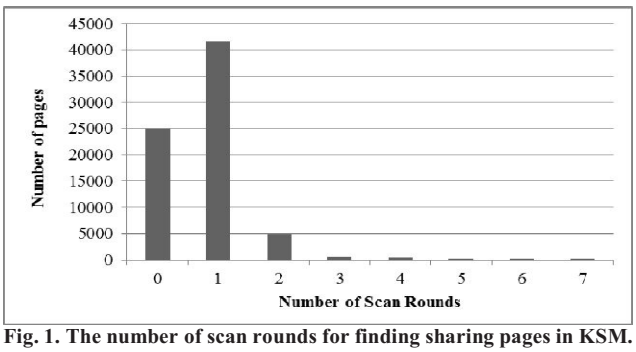

Fig. 1. The number of scan rounds for finding sharing pages in KSM.

The N-chance partial scanning can reduce the CPU computing overhead by KSM. Since the *ksmd* thread generally runs as a background thread of low priority, the performance degradation of user applications is insignificant. However, the power consumption of *ksmd* may be harmful to the battery lifetime of mobile devices. The merging rate of *ksmd* thread is determined by two configurable parameters *pages_to_scan* and *sleep_millisecs.* After ksmd scans the *pages_to_scan* number of pages, it sleeps during the *sleep_millisecs*. Each sharing candidate page needs M number of 4KB page comparisons, where M is the number of all pages in the KSM scanning target memory regions. If n, s, and t denote *pages_to_scan, sleep_millisecs,* and the time for one page scanning, respectively, the load of *ksmd* is $nt/(nt+s)$. If k number of pages among the a number of scanning candidates are skipped by the N-chance partial scanning, the load of *ksmd* is reduced to $(n-k)t/((n-k)t+s)$. Therefore, the computing load of *ksmd* is reduced by $100 \cdot ks/(n((n-k)t+s))$%, and the power reduction ratio will be similar.

While the time for one scan round is $M/n \times (nt+s)$ in the original KSM, the time is $M/n \times ((n-k)t+s)$ in the N-chance partial scanning. Therefore, the partial scanning can scan all the KSM target memory regions mush faster than the original KSM. This is profitable to find the short-lived duplicated pages. When n=100 and s=20, the time for one scan round is about 5 minutes. During the full scan time, several short-lived pages may be freed by application itself. The partial scanning can reduce the time for one scan round to 3 minutes. Therefore, it can find more short-lived duplicated pages.

III. EXPERIMENTS

We evaluated the proposed technique with an Android-based smartphone device equipped with 2-GB DRAM. While executing five memory-hungry applications during 20 minutes, the memory gain and power consumption are measured. The memory gain is the size of free memory pages acquired by freeing deduplicated pages. When measuring the power consumption of smartphone device, the airplane mode is used in order to reduce undetermined power fluctuations by network devices.

Three merging rates are used: R1=2, R2=5, and R3=10, as shown in Fig. 2. As a higher merging rate is used, the memory gain increases, but the power consumption is more significant. By adopting the 3-chance partial scanning, the power consumptions are reduced by 54%, 65.7%, and 67.6% at the merging rates of R1, R2, and R3, respectively. In addition, the memory gains are also improved since the N-chance KSM can find more short-lived duplicated pages.

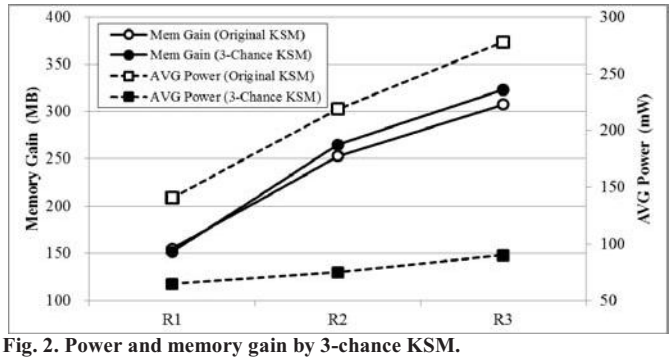

Fig. 2. Power and memory gain by 3-chance KSM.

Fig. 3 shows the size of memory pages that are scanned by 3-chance KSM. More than 70% of target memory pages are skipped since they have exhausted three chances. Therefore, we can know that the N-chance KSM is more efficient than the original KSM. At the second and third rounds, the deduplicated pages are 30% and 12%, respectively. These deduplication ratios are quite large since most of the deduplicated pages are allocated by the applications launched at the time.

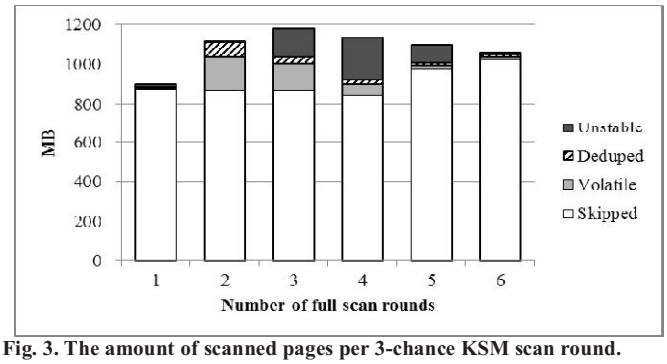

Fig. 3. The amount of scanned pages per 3-chance KSM scan round.

ACKNOWLEDGEMENTS

This work was supported by the Center for Integrated Smart Sensors funded by the Ministry of Science, ICT & Future Planning as Global Frontier Project (CISS-2011-0031863).

REFERENCES

[1] C.-R. Chang, J.-J. Wu, and P. Liu. An empirical study on memory sharing of virtual machines for server consolidation. In Proc. of the 2011 IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA'11, pages 244-249,2011.