Effective SSD Caching for High-Performance Home Cloud Server

Dongwoo Lee, Changwoo Min and Young Ik Eom
Sungkyunkwan University, Suwon, Korea
{lightof, multics69, yieom}@skku.edu

***Abstract*— In this paper, we propose a novel SSD caching scheme and virtual disk image format, named SVD (Sequential Virtual Disk). Our proposed techniques are based on workload characteristics in home cloud server, in which synchronous random writes dominate. Unlike previous studies, we use SSD as a read-write cache in order to effectively mitigate performance degradation of synchronous random writes.**

I. Introduction

Virtualization has been utilized restrictively for personal purpose, whereas it has been widely used in many enterprise fields in the past few years. However, as greeting an IoT (Internet-of-Things) era, recent trends show that some appliances such as smart TVs and home gateways are expected to evolve into home cloud server [1], [2]. The home cloud server governs all connected devices and provides them with the computing resources through a virtualization technology.

Despite its valuable virtualization functionalities such as multiplexing of physical resources and fair management of them, one of the major concerns is the performance degradation due to the virtualization latency. The constant research efforts made performance of processor and memory nearly bare-metal in virtualized systems, but, unfortunately, storage system affects the performance of applications in unanticipated ways [3]; especially, the synchronous interface in SQLite, which is used for ease of development, incurs massive random write traffics with plenty of force-unit-access (FUA) operations. These synchronous operations more significantly degrade performance of applications running on the home cloud server since virtual storage is accessed in a costly and complex way than the physical one. Therefore, improving the performance of virtual storage can expedite applications of the home cloud server.

SSD caching schemes have been widely studied and applied to conventional systems as a cost-effective storage model to bridge the speed gap between fast memory and slow disks. Many previous studies [4], [5], however, used SSD just as a read-only cache to reduce access time on read-most data. In contrast, the write operation is unwilling to be cached in SSD because NAND flash has two well-known limitations for the write operation: erase-before-write and relatively small write endurance. Nevertheless, the enormous writes induced by frequent FUAs should be cached in order to reduce their latency, which is critical to the performance of applications.

In this paper, we propose a novel SSD caching system which can caches the write operations effectively as well as the read operations by making up for the asymmetric I/O characteristics of the NAND flash storage. Our approach mitigates the effect of the storage on the application performance in virtualized systems, thereby improving the overall performance of the home cloud server.

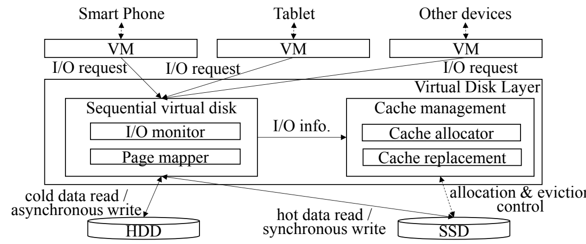

Fig. 1. Architecture of the proposed SSD caching system

II. Designs of Our SSD Caching System

Our SSD caching system is designed for improving the performance of storage I/O in the home cloud server. Fig. 1 shows the overall architecture of the proposed system. In the rest of this section, we will describe our design in detail addressing the following technical challenges. The first is on considering the NAND flash characteristics. Our system must be able to cache the write operations in the SSD without harming its performance or lifetime. The second is on managing the cache efficiently. Our system must be adaptive to reflect cache demands of each VM in order to fairly utilize the cache space shared among the VMs. The third is on the low overhead. SSD caching can obviously benefit storage I/O performance, but it can harm the overall system performance if its operational cost is too high.

A. Sequential Virtual Disk

In order to cache the write operations in SSD, reducing the number of random writes is important since it is about ten-fold slower than the sequential write and can cause excessive internal fragmentation. The fragmentation increases garbage collection cost inside SSD by incurring more block erases and thus it also reduces the lifetime of the SSD drastically. The problem of the random writes can be addressed by transforming them to the sequential writes as like a log-structured file system [6]. We adopt the same idea to the virtual disk layer. In virtualized environment, the virtual disk layer is a crucial component determining the write patterns of each VM since it manages host file as a virtual disk and provides mapping between them. Therefore, we propose a Sequential Virtual Disk (SVD) image format that uses the SSD fully sequentially.

The SVD uses a mapping method similar to QCOW [7], except data can be stored into either the virtual disk image or the SSD cache. Due to the space limitation, we focus only on the use of the SSD, considering the I/O characteristic of the NAND flash. The SVD partitions the SSD to chunks having the same size as the block that is a unit of the erase operation,

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)

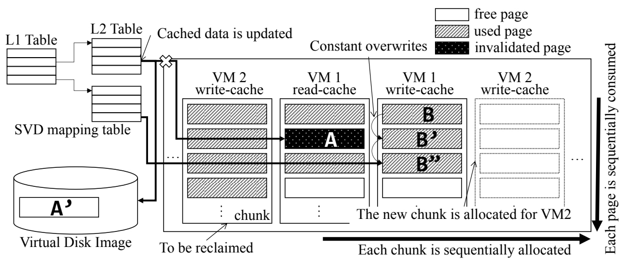

Fig. 2. Overview of the caching process and management in SVD

and allocates them to the VMs sequentially. Each chunk is divided by the page size of the SSD and each page is also consumed sequentially when the I/O data needs to be cached. As Fig. 2 shows, no page updates are happened in our system in order to prevent fragmentation; the read cache only invalidates the modified page instead of updating it, and the write-cache allocates a new page for each successive update.

B. Cache Space Management

In a virtualized system, the cache space is shared among VMs and the cache manager distributes it to each VM on demand. To maximize the utilization of the cache space, existing systems [4], [8] monitor the I/O requests of the VMs and make an effort to predict the cache demand precisely. In contrast, our caching system allocates the cache space sequentially without partitioning to each VM. Information about the cache demand is only required for eviction; if there is no available chunk in the cache, cache manager selects a victim chunk in the VM which has least demand. Not only we can easily achieve the high cache space utilization without costly monitoring, but it is also possible to use cache utilization statistic to eagerly reclaim the cache space for reducing on-time eviction overhead.

C. Data Placement & Eviction

Our system places two cases to the SSD cache: hot data to the read cache and data for synchronous writes to the write cache. In order to determine the hotness of data or identify the type of writes, the I/O request of VMs should be monitored. There are few options for monitoring the I/O at different levels of hierarchy. By exploiting I/O virtualization hierarchy, our system can easily capture the I/O requests without costly monitoring since all I/O requests in the VM are processed through the virtual disk layer.

Eviction can be conducted more simply since our system uses chunks, each of which has the same size as the unit of erase operation. When the cache is full, cache manager first evicts all fully-used chunks for write cache to maintain the read cache for a long time. If the ratio of the read cache exceeds a threshold (75% in our current prototype), chunks for the read cache can also be evicted. In contrast with write-cache, victims of the read cache are selected by cache replacement algorithms, most of which are LRU and its variants. Though our system adopts CLOCK [9] due to its low operational cost, other algorithms can be applied for more optimization.

III. Performance Evaluation

To verify the proposed caching system, we implemented a prototype using QEMU providing the virtual disk layer to VMs on KVM in Linux. Our experimental environment is configured by combining the 7500RPM 1TB HDD and high-end 32GB SSD. We measured the performance of sequential and random I/O ten times with same workloads in the VM with three different configurations: storing data only to the HDD with QCOW (HDD-QCOW), using the SSD as a primary storage ignoring its space limitation (SSD-QCOW), and our caching system (Caching-SVD). As Fig. 3 shows, our system outperforms HDD-QCOW and attains approximately to SSD-QCOW on 10th iteration case, though it initially cannot benefit from the read performance due to the absence of locality. Especially, in comparison with SSD-QCOW, ours significantly improves performance of random writes by transforming them to the sequential writes.

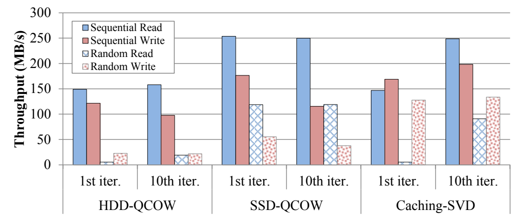

Fig. 3. Comparison of I/O throughput in configurations of combining virtual disk image format (QCOW, SVD) and underlying physical storage (SSD, HDD)

IV. Conclusion

This paper proposed a novel SSD caching system for improving storage performance as also improving the performance of applications in the home cloud server. Our system effectively caches not only the read operations but also the write operations, which are generally reluctant to be cached in the SSD due to the characteristics of the flash storage, by using the SSD fully sequentially. Experimental result shows that our caching system can significantly improve the performance of the random writes by transforming them to the sequential ones.

References

[1] M. R. Cabrer, R. P. D. Redondo, A. F. Vilas, J. J. Pazos Arias, and J. G. Duque, "Controlling the smart home from TV," *IEEE Trans. Consum. Electron.*, Vol. 52, No. 2, pp. 421-429, 2006.

[2] H. Park, I. Lee, T. Hwang, and N. Kim, "Architecture of home gateway for device collaboration in extended home space," *IEEE Trans. Consum. Electron.*, Vol. 54, No. 4, pp.1692-1697, 2008.

[3] H. Kim, N. Agrawal, and C. Ungureanu, "Revisiting storage for smartphones," *ACM Trans. Storage*, Vol. 8, No. 4, Article. 14, 2012.

[4] T. Luo, S. Ma, R. Lee, X. Zhang, D. Liu, and L. Zhou, "S-CAVE: effective SSD caching to improve virtual machine storage performance," In *PACT*, pp.103-112, 2013.

[5] H. Jo, Y. Kwon, H. Kim, E. Seo, J. Lee, and S. Maeng, "SSD-HDD-hybrid virtual disk in consolidated environments," In *Euro-Par*, pp.375-384, 2010.

[6] C. Min, K. Kim, H. Cho, S. W. Lee, and Y. I. Eom, "SFS: random write considered harmful in solid state drives," In *USENIX FAST*, p.12, 2012.

[7] M. McLoughlin. (2008, Sep. 11) The QCOW2 Image Format [Online]. Available: https://people.gnome.org/~markmc/qcow-imageformat.html

[8] F. Meng, L. Zhou, X. Ma, S. Uttamchandani, and D. Liu, "vCacheShare: Automated Server Flash Cache Space Management in a Virtualization Environment," In *USENIX ATC*, pp.133-144, 2014.

[9] S. Jiang, F. Chen, and X. Zhang, "CLOCK-Pro: An Effective Improvement of the CLOCK Replacement," In *USENIX ATC*, pp.323-336, 2005.