PROFS: CPU Profile-based Tiering File System for All Flash Storage System

Yu Nakanishi[1,2], Steven Swanson[1]

[1] UC San Diego, [2] KIOXIA

Abstract- Solid State Drive (SSD) plays an essential role in modern computing systems. In recent years, the new kinds of SSD, such as the low-latency high-cost SSDs and the low-cost high-latency SSDs, have been emerged. However, when integrating different types of SSD in a storage system, it can be more challenging to make a balance between cost and performance due to the conflicting features of SSDs. For example, not only does redundant data migration cause overhead, but it also accelerates wearing out of Flash SSDs. We propose the CPU Profile-based Data allocation Management (CPDM) scheme, and the CPU PROfile-based tiering File System (PROFS), which is a dynamic tiering file system based on CPDM. CPDM and PROFS are designed to provide not only performance-efficiency but also low and stable write amplification. We have performed an experimental evaluation and compared it with the popular caching system. Experimental results demonstrate that PROFS provides both a low average and low variance of write amplification.

1 INTRODUCTION

In recent years, the technological progress of memory leads to the development of a new kind of SSDs. Some vendors have already shown the low-latency high-cost SSDs, such as Optane SSD from Intel-Micron, Z-NAND SDD from Samsung and XL-Flash SDD from Kioxia. Additionally, the low-cost high-latency SSDs, such as QLC SSDs from Intel-Micron, Samsung, and Kioxia, have also emerged. However, it requires a challenge to integrate different types of SSDs into a storage system due to the conflicting features of SSDs. For example, not only does redundant data migration cause overhead, but it also accelerates the wearing out of Flash SSDs. So it is important to keep write amplification low to provide stable performance and a long lifetime of flash devices.

In this work, we propose the CPU Profile-based Data allocation Management (CPDM) scheme, which captures necessary information from CPU profiles for deciding the property of data. Based on CPDM, we further develop the CPU PROfile-based tiering File System (PROFS), which is a dynamic tiering file system. PROFS focuses on performance-efficient, keeping write amplification low and stable write amplification.

We have performed an experimental evaluation by using several workloads. Experimental results demonstrate that PROFS provides both a low average and low variance of write amplification. Besides write amplification, the result also shows that PROFS got better performance improvement in 27 out of 39 cases against DM-Cache and L2ARC of ZFS.

2 CPU PROFILE-BASED TIERING FILE SYSTEM

In this work, we propose the CPU Profile-based Data allocation Management (CPDM) scheme and the CPU PROfile-based tiering File System (PROFS) based on CPDM to provides a cost-performance effective solution for multiple storage systems, especially for those with different types of SSDs inside.

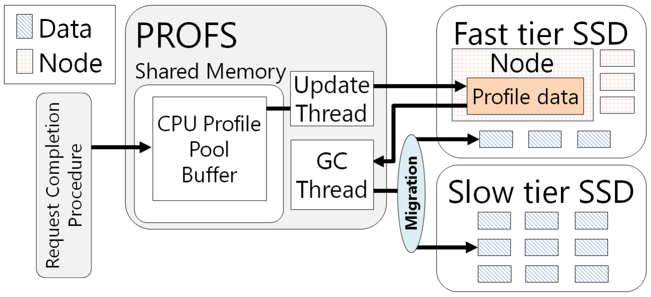

Figure 1: Overview of PROFS

2.1 CPDM

The basic idea of CPDM is using the CPU profile to classify a data type. CPDM defines *Cost* as the property of data that can be used by PROFS. For a data block, in brief, *Cost* indicates the expected time that CPU will be stalled by I/O process of the data. We can determine proper data allocation basically according to *Cost* to keep the amount of a stalling time short. It could remove some bottlenecks, which make CPU wait for I/O completion of essential events such as fsync and read requests for launch application.

We define *Cost* as follows.

$$Cost = \frac{\sum_{i=0}^{n-1} I/OwaitingTime_i}{\sum_{i=0}^{n-1} Contribution_i}$$

$$Contribution_i = \frac{BusyTime_i + I/OwaitingTime_i}{Duration}$$

$$i : CoreID, \quad n : Number\ of\ cores$$

If all cores did nothing due to waiting for I/O completion, *Contribution* and *Cost* becomes 1.0. In the case that all cores computed other tasks during waiting for I/O completion, *Contribution* becomes 1.0, but *Cost* becomes zero. In the other case that just one core was waiting for I/O completion, and other cores were just idling, then both of *Contribution* and *Cost* becomes 1.0. It is intended to consider the possibility that other cores are waiting for other resources such as a semaphore. Therefore the last case has a higher priority than the case that other cores were busy.

In addition, CPDM uses *Reuse Distance* as a secondary metric. A reuse distance widely used in the cache algorithm is how much data access is between the reusing and the last using. It implies the frequency that the data is accessed. We use *Reuse Distance* to make a decision for data with close *Cost* values.

2.2 PROFS

We build the tiering file system PROFS that implements CPDM based on the existing F2FS log-structured file system[5]. Figure 1 shows the overview of PROFS. As with F2FS, PROFS classifies a type of data into DATA and NODE. DATA indicates actual user data of files. NODE contains metadata of a file and pointers to each DATA block. We divided DATA type into Fast Tier DATA and Slow Tier DATA. PROFS identifies a data type and decide which tier a write data should be stored in. On the other hand, a data of NODE should be written to Fast Tier because the impact of accessing metadata is significant[1]. We added a profiling data structure to inode.

PROFS employs its own request issue procedure and completion procedure to put an identification tag on each request. When I/O request is completed, the completion procedure stores the request's tag and *Cost* into the shared buffer, after which the internal thread processes the dozen inode updating at one time to prevent frequent lock operations and losing scalability. A node page, incidentally, is made dirty only once in several updating to avoid serious write-back overhead. Although there is a possibility that updating is lost, we expect an impact is slight.

The migration process is executed by the garbage collection thread, which is inherited from F2FS. It means the migration process is implemented as a part of the garbage collection thread. F2FS has two kinds of garbage collection mode. One is called foreground mode, which is only triggered if there is no enough free space. Another is called background mode, which is triggered periodically and executed only under the idle state. We added the migration mode, the execution of which is exclusive in other GC modes. PROFS invokes the migration process every two seconds while a background garbage collection is kicked every 30 seconds(it is configurable).

The migration process decides an amount of migration data in first. An amount of migration data is limited adaptively by a free space of each tier and a recent amount of data moving to reduce the effect for performance. Next, the migration process gets a migration candidate and checks all pointers to the data block and tries to move the data block to the proper tier. However, PROFS does not issue any write-back operations at this timing. Instead, PROFS only issues a read request to cache a data in page caches and makes it dirty, and these will be written back by the kernel background thread. This mechanism is almost the same implementation of background garbage collection to avoid the serious performance impact.

3 EVALUATION

We evaluated PROFS about performance, write amount, and write amplification. Write amplification is represented by the ratio of a write amount with multiple storage systems such as tiering and a write amount with only single storage. We can use this value to evaluate how much extra data was moved by the multiple storage system. Although a data movement is needed for good performance, keeping write amplification low is important to avoid the acceleration of wearing out.

To evaluate PROFS on several kinds of workload, We used Intel Xeon E5 server equipped Intel Optane SSD as a fast SSD, and Western Digital SATA SSD as a slow SSD. We compared PROFS with several combinations of file systems and multiple storage systems. We chose F2FS, EXT4, and ZFS as a file system. Also we used DM-Cache[3] and L2ARC[4] as a multiple storage system.

Figure 2 shows the result of MongoDB[6]. We use YCSB[2] as a workload to evaluate MongoDB. We selected two types of ratio about Read and Update; 10:90(updating main), 90:10(reading main). The results indicate PROFS could achieve a good performance for both access pattern in spite of low write amplification. In particular, the scaling of PROFS about the updating main workload is outperformed than other multiple storage systems, whereas other multiple storage systems could not keep a scaling.

We also run other 11 workloads(MondoDB's another workload, fio's 4 workloads, Filebench's 3 workloads, MySQL's 2 workloads,

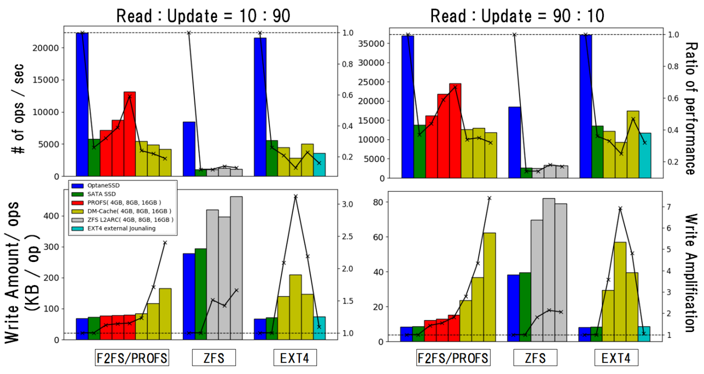

Figure 2: MongoDB: Result of the performance(Above), and the amount of write(Bottom). The bars indicate the performance and the write amount, the lines indicate the performance ratio of only fast SSD value and the write amplification.

GraphChi's 1 workload) and 3 fast tier conditions, which are 4GB, 8GB, and 16GB. The results (not shown for brevity) show that PROFS achieves better performance in 21 out of 33 cases (11 workloads $\times$ 3 fast tier sizes, 4GB, 8GB, 16GB), while EXT4-DM-Cache got 6 cases, F2FS-DM-Cache got 2 cases, ZFS-L2ARC got 4 cases, and EXT4 in external Journal mode got no case. Beside, the average write amplification of PROFS is 1.31 while F2FS-DM-Cache is 2.14, EXT4-DM-Cache is 1.96, ZFS-L2ARC is 2.33. Further, PROFS' variance (it is presented as $\frac{1}{n} \cdot \sum (x - \overline{x})^2$ *x: Write Amplification, n: The number of the evaluation results.*) is only 0.091, while EXT4-DM-Cache is 0.28, F2FS-DM-Cach is 0.46, and ZFS-L2ARC is 0.27. It indicates PROFS could provide stable and low write amplification which makes a lifetime of storage device longer.

4 CONCLUSION

We have proposed a new scheme called CPDM and developed the tiering file system, PROFS based on CPDM. We evaluated PROFS about performance, write amount, and write amplification. Experimental results demonstrate that PROFS achieves the low average, the low variance of write amplification, and the better performance. The resulting average write amplification and variance values of PROFS is 1.31 and 0.091 accordingly, which are much lower than 1.96 and 0.28, average write amplification and variance values of the popular EXT4 on DM-Cache, while 2.14 and 0.46 are the values of the F2FS on DM-Cache, and 2.33 and 0.28 are the values of the ZFS-L2ARC. Furthermore, PROFS got better performance improvement in total 27 out of 39 cases against DM-Cache and ZFS-L2ARC.

REFERENCES

[1] Nitin Agrawal, William J Bolosky, John R Douceur, and Jacob R Lorch. A five-year study of file-system metadata. *ACM Transactions on Storage (TOS)*, 3(3):9, 2007.
[2] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154. ACM, 2010.
[3] Introduction of dm-cache. https://www.kernel.org/doc/Documentation/device-mapper/cache.txt, 2018.
[4] Explanation of ARC and L2ARC. https://www.zfsbuild.com/2010/04/15/explanation-of-arc-and-l2arc/, 2010.
[5] Changman Lee, Dongho Sim, Jooyoung Hwang, and Sangyeun Cho. F2fs: A new file system for flash storage. In *13th {USENIX} Conference on File and Storage Technologies ({FAST} 15)*, pages 273–286, 2015.
[6] MongoDB: A cross-platform document-oriented database program. . https://www.mongodb.com/.