CoW Link: Duplicated Data Block Management Scheme for Multi-user Consumer Electronics Devices

Sung Jae Park†‡, Dong Hyun Kang†, and Young Ik Eom†
†Sungkyunkwan University, Korea ‡Samsung Electronics, Korea
{nicesj82,kkangsu,yieom}@skku.edu

***Abstract*—In consumer electronics (CE) environments, as the needs for multiple user account grow, operating system for CE devices began to support *multi-user features*. Unfortunately, with this feature, the system installs software packages individually for each account, and it would waste the storage space of CE devices. In this paper, we propose a scheme, called *Copy-on-Write* (*CoW*) *Link*, that efficiently saves the storage space by eliminating duplicate blocks of the files belonging to software packages. Our evaluation results show that *CoW Link* reduces the storage space by up to 9 times, compared with the traditional scheme.**

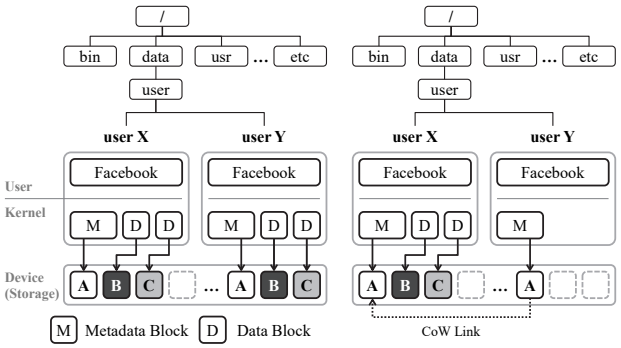
I. Introduction

Recently, operating systems of consumer electronics (CE) devices, such as smartphones, smart TVs, and tablet PCs, began to provide the *multi-user feature* that allows each user to have its own account within the CE devices, like traditional Linux operating system [1][2]. Especially, since the *multi-user feature* makes it much easier and safer to share one device, it has become a standard feature in consumer electronics. However, this feature potentially wastes the storage space of CE devices because each user maintains his own storage space and installs software packages independently in his storage space. Therefore, same files of a package may be downloaded and saved for each user account.

Figure 1a demonstrates the package installation process with the traditional *multi-user feature*. As shown in Figure 1a, there are two user accounts (user X and Y). We assume that user X and Y install the same social network service (SNS) application (e.g., Facebook) on their account, independently. As a result, some data blocks, which belong to the application, are duplicated in the storage of the device even though data blocks have exactly the same contents (e.g., B and C).

In this paper, we focus on this duplicated data blocks because of two reasons. First, it is extremely important to eliminate unnecessary write operations caused by the duplication of blocks. This is because most CE devices employ NAND flash storages, such as eMMCs, SD Cards, and SSDs, as their main storage, where the storages suffer from limited lifetime of NAND flash memory. Second, the duplicated data blocks can be eliminated through cooperative block management with the file system, such as Ext4. We propose a scheme, called *Copy-on-Write* (*CoW*) *Link*, which efficiently eliminates write operations to duplicated data blocks that belong to some files of the software packages. For evaluation, we have implemented a prototype of the *CoW Link* and compared it with the traditional multi-user feature. Our experimental results show that *CoW Link* reduces the wasted space of the storage by up to 9 times compared with the traditional scheme.

This work was supported by ICT R&D program of MSIP/IITP. [R0126-15-1065, (SW StarLab) Development of UX Platform Software for Supporting Concurrent Multi-users on Large Displays]. Young Ik Eom is the corresponding author of this paper.



(a) Traditional approach (b) CoW Link

Fig. 1. The concept of CoW Link

II. Background

There are two approaches to save the storage and memory space in the traditional operating systems (OS) such as Unix and Linux. One is the *file link* approach that is widely used to reduce the amount of redundant data blocks for the same file in the file system layer. There are two types of file link approaches: *hard link* and *soft link*. The hard link shares metadata of a file (e.g., inode), and it can give the positive effectiveness in terms of the storage space. On the other hand, the soft link has independent metadata and data block, but the data block keeps a reference to original file instead of file contents. With these two file link approaches, when the contents of the original file are updated, these changes also affect the shared file immediately.

Another is the Copy-on-Write approach, which is used to save both the process creation time and memory space in the kernel layer. When a new child process is created, it just shares the address space of its parent process. But, if a child or parent process tries to change a block in the shared address space, the kernel allocates new memory space for him and then copies the original block in the shared memory space to the newly allocated space. Afterwards, when the block in the parent or child memory space is changed, it never affects the memory space of the other process.

III. Design of Copy-on-Write Link Scheme

In this section, we propose a scheme, called *CoW Link*, that completely eliminates write operations on duplicated data blocks that belong to same files. In order to avoid unnecessary

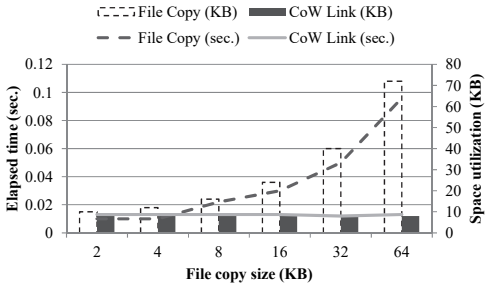


Fig. 2. Performance and space utilization

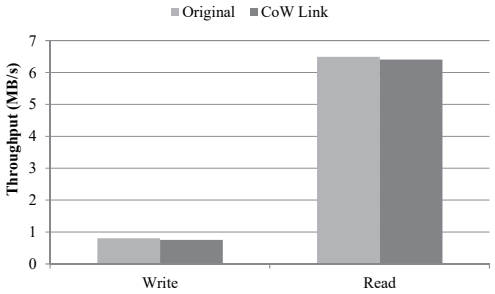


Fig. 3. Performance overhead

space wastage among the same files, *CoW Link* extends the traditional CoW approach into the file system layer.

This collaboration is very simple and straightforward, but it has a significant impact on the overall performance and storage lifetime of the CE devices.

Figure 1b depicts the overall process of *CoW Link* when a user installs an SNS application which is already installed by another user. As shown in Figure 1b, *CoW Link* first creates a metadata block for the SNS application and then it stores a reference to the inode of each original file into the created metadata block, as in the soft link approach. A data block for the shared file with *CoW Link* is allocated when a write operation occurs on either the original file or shared file, as in the CoW approach. If a read operation occurs on a data block that was never accessed by write operations, this operation is performed on the block of the original file. Otherwise, read operation is performed on the *CoW Link*ed data block. As a result, *CoW Link* can save the storage space and improve the overall performance, by sharing the blocks with same contents as much as possible and preventing the write operations on each duplicated data block that belongs to the same file until the data block is updated.

IV. Performance Evaluation

In order to verify the effectiveness of *CoW Link* scheme in terms of the performance and space utilization, we have implemented our scheme on top of the FUSE file system [3] and compared it with the traditional approach that creates both metadata and data blocks and then manages it independently even when two or more same files are created. All experiments were performed on a machine equipped with an Intel Core i7-6700 3.4 GHz and 4GB memory with the Linux Kernel 4.6.0 version. Also, we used the latest SD Card (Samsung Evo 64 GB) for comparison because most CE devices employ flash storage as their primary storage. In order to simulate an I/O intensive workload, we generated a workload that creates files by varying the file size from 2 KB to 64 KB and then copies the created files. We calculated the average elapsed time measured from five separate runs.

Figure 2 shows the evaluation results in terms of performance and space utilization. The X axis is in log scale and the Y axis presents the average elapsed time (left) and the space utilization (right). As we expected, the traditional approach reveals poor performance as the copy size increases. This is because it should copy all data blocks. On the other hand, *CoW Link* shows a steady performance in all cases. In addition, since *CoW Link* does not copy data blocks for the shared file immediately when a copy operation is triggered, it significantly reduces the elapsed time by up to 7 times compared to the traditional approach. Also, as mentioned in Section III, *CoW Link* can delay the copy operations until the shared block is updated. Therefore, it dramatically saves the storage usage by up to 9 times compared to the traditional one (Figure 2). These results are very meaningful for the storage of CE devices because most storages for CE devices suffer from the penalty caused by write operations, such as high-cost write operations, garbage collection, and limited lifetime.

Of course, *CoW Link* has the additional overhead to dynamically copy data blocks whenever a write operation occurs on either the original data block or the shared data block. In order to evaluate this overhead, we also measured the throughput of *CoW Link* while updating all shared blocks (i.e., *CoW Link* allocates new data blocks and then updates the contents of new blocks with CoW operations). Figure 3 compares the throughput of *CoW Link* with that of traditional approach. As shown in Figure 3, the overhead is negligible, while the overhead is 6% and 1.3% for write and read operation, respectively.

V. Conclusion

In this paper, we have proposed a *CoW Link* scheme which efficiently saves the storage space of CE devices by eliminating write operations to store duplicated data blocks and implemented the proposed scheme on the top of FUSE file system. Our evaluation results clearly show that *CoW Link* reduces the storage space by up to 9 times compared with the traditional approach. However, we need to consider the consistency issues also. If the original data blocks, which are shared, are crashed, all linked files can be crashed too. We will solve it in future works.

References

[1] Android official site. “Supporting Multiple Users.” Internet: https://source.android.com/devices/tech/admin/multi-user.html, Jul. 11, 2016 [Jul. 11, 2016].

[2] Tizen official site. “Multi-user Architecture.” Internet: https://wiki.tizen.org/wiki/Multi-user_Architecture, Jan. 30, 2015 [Jul. 11, 2016].

[3] B. Kahanwal, “File System Design Approaches,” *International Journal of Advances in Engineering Sciences*, vol. 4(1), pp. 16-20, 2014.