Dewco: Improving File System Performance of Mobile Storage Systems Using a Copyless Defragmenter

Sangwook Shane Hahn, Cheng Ji*, Sungjin Lee†, Li-pin Chang‡,
Jeongseok Ryoo, Liang Shi§, Chun Jason Xue*, and Jihong Kim
*Seoul National University, *City University of Hong Kong, †Inha University,*
‡National Chiao-Tung University, §Chongqing University

When a file system becomes highly fragmented, it has to allocate multiple storage areas, i.e., extents [1], for a single file more frequently. In an HDD-based file system, accessing such a highly-fragmented file degrades its performance significantly due to the increased time-consuming seek operations. In order to mitigate the performance impact of file system fragmentation, a file system commonly employs a defragmentation utility [2-4]. The defragmentation utility rearranges file-to-storage mappings so that files can be contiguously allocated using a small number of distinct extents, thus avoiding expensive future seek operations. Although the defragmentation process may take a long time, most HDD-based file systems recommend defragmentation to prevent gradual performance degradations caused by file system fragmentation.

Unlike for HDD-based file systems, defragmentation is generally not recommended for flash-based file systems [6-11]. Since flash-based storage does not require seek operations, it is commonly *believed* that the effect of defragmentation on the file system performance is rather negligible for flash-based storage. Furthermore, since a large number of files are copied during defragmentation, frequent defragmentation activities can significantly waste the limited flash lifetime. This negative view toward flash defragmentation has been widely accepted without any quantitative validation study. The initial motivation of this work, therefore, was to confirm whether defragmentation for mobile flash storage is really not needed for performance improvement.

In order to understand the impact of file system fragmentation on the performance of mobile flash storage, we quantitatively evaluated the performance impact of file fragmentation on the entire mobile I/O stack layers, from the file system layer to the flash device layer. Our evaluation results on representative smartphones, however, revealed somewhat surprising results over the common wisdom on flash defragmentation: *the performance of flash storage can be significantly degraded when files are highly fragmented, and defragging flash storage can substantially improve the performance of mobile devices.*

We observed that two main factors contribute the poor performance of a fragmented file system. First, when fragmented files are accessed, the number of block I/O requests is significantly increased. Although there is nothing new with this observation, as the number of block I/O requests increases, overhead increases in mobile flash storage had quite different characteristics from those in HDDs. In HDDs, an increase in the number of block I/O requests dominantly affects the time spent inside the HDD's storage medium, mainly from increased seek operations. On the other hand, in mobile flash storage, a processing time increase within a mobile flash storage was less dominant as the number of block I/O requests increases. Instead, processing time increases in the block I/O layer and device driver affected the overall I/O performance more significantly [12-15]. Second, when files were fragmented, the average I/O size of a block I/O request becomes smaller. Since most flash management techniques tend to exploit the spatial locality in flash accesses for higher performance, a large number of scattered small block I/O requests make it difficult for an FTL to achieve high performance because the internal I/O parallelism cannot be fully utilized [16-21].

Since we observed that file system fragmentation can significantly degrade the performance of mobile flash storage, as the next step, we investigated if file system fragmentation *actually occurs* on mobile flash storage. In order to quantitatively measure if and how files are fragmented on mobile flash storage, we collected several Android smartphones which have been used between 6 months and 2 years by engineering graduate students. Like typical smartphone users, these users' usage scenarios included popular applications for SNS messaging, web surfing, emails and cameras. Even though the Ext4 file system (which is used for our evaluated smartphones) employs several schemes for alleviating file fragmentation, our empirical study showed that files can be severely fragmented on mobile flash storage [22].

Motivated by our findings from both the empirical study and performance study on file fragmentation, we propose a simple but novel flash storage defragmentation technique, **de**fragger **w**ithout **co**pies (dewco), which takes advantage of the flash storage's internal logical-to-physical mapping table. Dewco performs flash storage defragmentation by remapping logical block addresses of a fragmented file within the flash translation layer's mapping table without physically moving data to new NAND pages. As a result, the proposed technique barely affects the lifetime of flash memories.

In order to validate the effectiveness of the proposed dewco technique, we have implemented dewco on an emulated mobile flash storage, `simeMMC` and `simUFS`. Both `simeMMC` and `simUFS` are based on an extended Samsung 843T SSD [23] which supports host-level FTLs. In order to emulate different mobile flash storage (eMMC [24] and UFS devices [25]), we limited the internal I/O parallelism level of the extended 843T so that it can effectively simulate the bandwidth of mobile flash storage. Our experimental results on `simeMMC` and `simUFS` show that dewco can improve the throughput of Ext4 by up to 108% without sacrificing the lifetime of flash storage.

References

[1] MANTHUR, A., CAO, M., AND BHATTACHARYA, S. The New ext4 File System: Current Status and Future Plans. In *Proceedings of Linux Symposium* (2007).

[2] E4defrag - Online Defragmenter for Ext4 File System. http://manpages.ubuntu.com/manpages/trusty/man8/e4defrag.8.html.

[3] Condusiv Diskeeper. http://www.condusiv.com/products/diskeeper/.

[4] Auslogics Disk Defrag. http://auslogics.com/en/software/disk-defrag/.

[5] Samsung SSD Performance Enhancement & Maintenance. http://www.samsung.com/semiconductor/minisite/ssd/support/faqs-03.html.

[6] Frequently Asked Questions for Intel Solid State Drives. http://www.intel.com/content/www/us/en/support/software/000006110.html.

[7] Crucial SSD and HDD Support & Maintenance. http://www.crucial.com/usa/en/support-system-maintenance-defragment-hard-drive.

[8] KEHRER, O. O&O Defrag and Solid State Drives. http://www.oo-software.com/en/docs/whitepaper/ood_ssd.pdf.

[9] LIND, A. Auslogics: How to Defrag Disk Drives The Right Way. http://www.auslogics.com/en/articles/how-to-defrag/.

[10] Windows 8 TRIM SSD Instead of Defragmentation. http://www.eightforums.com/tutorials/8615-optimize-drives-defrag-hdd-trim-ssd-windows-8-a.html.

[11] Windows 10 TRIM SSD Instead of Defragmentation. http://www.tenforums.com/tutorials/8933-optimize-defrag-drives-windows-10-a.html.

[12] T10, TECHNICAL COMMITTEE OF THE INTERNATIONAL COMMITTEE ON INFORMATION TECHNOLOGY STANDARDS. SCSI TEST UNIT READY Command. http://www.t10.org/ftp/t10/document.06/06-022r0.pdf.

[13] T10, TECHNICAL COMMITTEE OF THE INTERNATIONAL COMMITTEE ON INFORMATION TECHNOLOGY STANDARDS. SCSI Block Commands - 3 (SBC-3). http://www.t10.org/ftp/t10/document.05/05-344r0.pdf.

[14] ANDERSON, D. C., CHASE, J. S., GADDE, S., GALLATIN, A. J., AND YOCUM, K. G. Cheating the I/O Bottleneck: Network Storage with Trapeze/Myrinet. In *Proceedings of the USENIX Annual Technical Conference* (1998).

[15] AHMAD, I., GULATI, A., AND MASHTIZADEH, A. vIC: Interrupt Coalescing for Virtual Machine Storage Device I/O. In *Proceedings of the USENIX Annual Technical Conference* (2011).

[16] AGRAWAL, N., PRABHAKARAN, V., WOBBER, T., DAVIS, J. D., MANASSE, M., AND PANIGRAHY, R. Design Tradeoffs for SSD Performance. In *Proceedings of the USENIX Annual Technical Conference* (2008).

[17] KANG, J.-U., KIM, J.-S., PARK, C., PARK, H., AND LEE, J. A Multi-channel Architecture for High-performance NAND Flash-based Storage System. *Journal of Systems Architecture: the EUROMICRO Journal* (2007).

[18] PARK, S.-H., HA, S.-H., BANG, K., AND CHUNG, E.-Y. Design and Analysis of Flash Translation Layers for Multi-channel NAND Flash-based Storage devices. *IEEE Transactions on Consumer Electronics* (2009).

[19] HU, Y., JIANG, H., FANG, D., TIAN, L., AND LUO, H. Performance Impact and Interplay of SSD Parallelism Through Advanced Commands, Allocation Strategy and Data Granularity. In *Proceedings of the ACM International Conference on Supercomputing* (2011), pp. 96–107.

[20] JUNG, M., AND KANDEMIR, M. T. An Evaluation of Different Page Allocation Strategies on High-Speed SSDs. In *Proceedings of the USENIX Workshop on Hot Topics in Storage and File Systems* (2012).

[21] JUNG, M., WILSON III, E. H., AND KANDEMIR, M. T. Physically Addressed Queueing (PAQ): Improving Parallelism in Solid State Disks. In *Proceedings of the International Symposium on Computer Architecture* (2012), pp. 404–415.

[22] JI, C., CHANG, L., SHI, L., WU, C., LI, Q., AND XUE, C. J. An Empirical Study of File-System Fragmentation in Mobile Storage Systems. In *Proceedings of the USENIX Workshop on Hot Topics in Storage and File Systems* (2016).

[23] SAMSUNG 843T Data Center Series. http://memorysolution.de/mso_upload/out/all/SM843T_Specification_v1.0.pdf.

[24] Embedded MultiMediaCard (e.MMC). http://www.jedec.org/standards-documents/technology-focus-areas/flash-memory-ssds-ufs-emmc/e-mmc.

[25] Universal Flash Storage (UFS). http://www.jedec.org/standards-documents/focus/flash/universal-flash-storage-ufs.