Write Performance Analysis with Buffered FUSE

Chanhyun Park, Youjip Won

Department of Computer and Software
Hanyang University, Seoul, Republic of Korea
E-mail: {parkch0708, yjwon}@hanyang.ac.kr

Abstract

With the development of technology, the size of files handled by smart devices is increasing. This phenomenon can cause the storage to be the biggest bottleneck of smart devices. There is a study that presents a major factor in this phenomenon is overhead in user level filesystem (FUSE). And it proposed buffered FUSE (bFUSE) and increased size of FUSE I/O as a solution of this problem. In this paper, we reproduced it in ODROID-Q2 and measured write performance. As a result, we can raise the write performance up to 96% compared to raw device bandwidth with FUSE.

Keywords: Android, Storage, User level filesystem, Write buffer

1. Introduction

Penetration rate of smart devices is increasing every year. And with the development of technology, utilization range of smart devices is widening. Latest smart devices can take and retouch a picture of 10 million pixels or more, and record and playback video of ultra HD class. This means that the size of the file handled by smart devices is increasing. According to Kim et al. [1], this phenomenon can cause the storage to be the biggest bottleneck of Android-based smart devices.

The study of Jung et al. [2] proposed buffered FUSE to solve this problem. They analyzed I/O workload of user partition on Samsung Galaxy S3, and discovered an overhead of user level filesystem (FUSE). For the reasons of the overhead, they pointed out the fact that FUSE I/O's unit of 4KB caused the I/O fragmentation and excessive context switch. They proposed to use buffered FUSE (bFUSE) and increased size of FUSE I/O to solve this problem and to raise the write performance up to 99% compared to raw device bandwidth with FUSE.

In this paper, we reproduced bFUSE environment on ODROID-Q2 that has lower performance than Samsung Galaxy S3 and measured sequential buffered write performance. As a result, with the use of specific filesystem and I/O scheduler, we can raise the write performance up to 96% ratio of raw device bandwidth with FUSE.

2. Evaluation Design

Table 1. Specifications of ODROID-Q2

Processor	Samsung Exynos4412 Prime Cortex-A9 Quad Core 1.6Ghz with 1MB L2 cache
Memory	2GB LP DDR2 880Mega data rate
Storage	16GB eMMC
OS	Android 4.1 Jelly Bean (3.0.51 Kernel)

Hardware specifications of ODROID-Q2 which are used in experiment are shown in Table 1. ODROID-Q2 uses Samsung Exynos4412 Prime as a processor. It has 4 cores and 1.6GHz clock speed. Memory of total 2GB is installed. And 16GB eMMC was used for storage. OS was Android 4.1 Jelly Bean and Kernel is 3.0.51 Kernel.

To measure I/O performance, benchmark tool MobiBench [3] was used. During the measurement of performance, the size of file was set to 100MB and the size of I/O was set to 512KB. We measure performance of sequential buffered write for ten times, and calculate minimum, maximum and average value.

We use EXT4 and XFS filesystem. And the experiments were performed in six environments. The basic environment of ODROID-Q2, **BASE** (FUSE with EXT4 filesystem, BFQ scheduler, 4KB FUSE I/O unit) and environment capable of maximum performance, **MAX** (raw device, NOOP scheduler, without FUSE) was the standard environment. We changed I/O scheduler to the Deadline, which was suggested as the most appropriate scheduler from previous studies [2]. After changing size of FUSE I/O, we conducted the experiment with the same workload in two different environments with EXT4 and XFS respectively – **512KB-EXT4** (FUSE with EXT4, Deadline scheduler, 512KB FUSE I/O unit) and **512KB-XFS** (FUSE with XFS, Deadline scheduler, 512KB FUSE I/O unit). Finally, we conducted an experiment in two different environments with 8MB FUSE buffer applied to them – **bFUSE-EXT4** and **bFUSE-XFS**.

3. Experiment

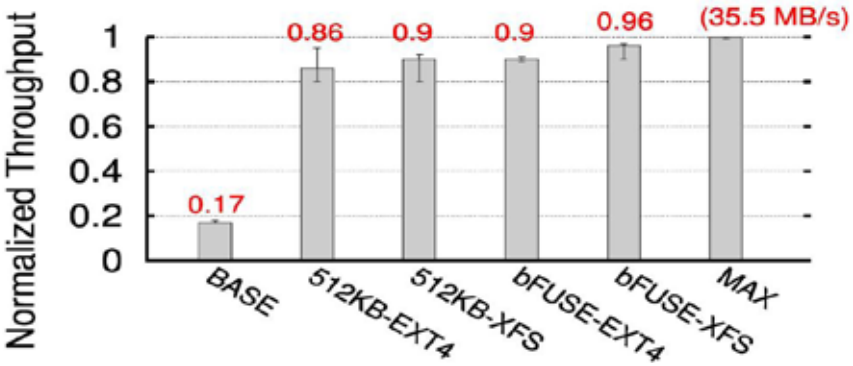

Figure 1. Result of sequential buffered write (100MB file, 512KB I/O size)

Figure 1 shows sequential buffered write performance in six environments which Chapter 2 presented. **MAX** environment shows 35.5MB/sec throughput. **BASE** environment shows only 17% throughput of **MAX** environment. When changing I/O scheduler to Deadline, and increasing size of FUSE I/O to 512KB, buffered write performance showed 86% of performance compared to that of raw device in EXT4 filesystem. In XFS filesystem, showed 90% of performance compared to that of raw device. And two environments with 8MB FUSE – **bFUSE-EXT4** and **bFUSE-XFS** environments show 90% and 96% of performance compared to that of raw device.

4. Conclusions

As a result, changing I/O scheduler and size of FUSE I/O could increase performance more than five times. And the use of buffered FUSE could show an additional performance improvement of 5%. Than we can raise the buffered write performance up to 96% compared to that of of raw device bandwidth with FUSE. This technology is expected to be able to minimize storage bottlenecks of Android-based Smart device in the future.

References

[1] H. Kim, N. Agrawal, and C. Ungureanu, “Revisiting storage for smartphones” In Proc. Of the 10th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, Feb. 2012

[2] S. Jeong, Y. Won, “Buffered FUSE: optimizing the Androi IO stack for user-level filesystem”, International Journal of Embedded Systems (IJES), Special issue for Embedded and Ubiquitous Computing, IEEE EUC-13, Zhangjiajie, China, November 13-15, 2013

[3] S. Jeong, K. Lee, J. Hwang, S. Lee, Y. Won, "Framework for Analyzing Android I/O Stack Behavior: from Generating the Workload to Analyzing the Trace", Future Internet 2013, 5(4), Special Issue for "Mobile Engineering², MDPI(ISSN 1999-5903), 591-610; doi:10.3390/fi5040591