AndroBench: Benchmarking the Storage Performance of Android-Based Mobile Devices

Je-Min Kim and Jin-Soo Kim

School of Information and Communication
Sungkyunkwan University (SKKU), Suwon 440-746, South Korea
jmkim@csl.skku.edu, jinsookim@skku.edu

Abstract. The storage performance directly influences usability and overall user experience in mobile devices. In this paper, we present the design and implementation of AndroBench, a storage benchmarking tool for Android-based mobile devices. AndroBench measures the sequential and random I/O performance and the throughput of various types of SQLite transactions. We also analyze the AndroBench results we collect from hundreds of devices from all over the world. Our analysis shows that no single device outperforms another one in all categories we tested. We also find that the type of eMMC and the filesystem significantly affect the storage performance.

Keywords: AndroBench, Android, Storage performance, Benchmark.

1 Introduction

The performance of mobile devices is being continuously improved to meet end user's growing desire to run more resource-demanding applications. Among others, the storage performance of a mobile device is one of the most important factors which directly influences usability and overall user experience of the device [1]. For example, audio/video playback quality while a number of applications are downloaded in background, Internet web browsing speed, or the time to complete database queries all depends on the storage performance.

Obviously, the storage performance is mainly determined by the performance of the underlying storage media. When it comes to designing mobile devices, hard disk drives are bigger, easier to fragile, and consume more power than flash-based storage. For these reasons, almost every mobile device uses NAND flash memory for storing operating system image, applications, and user data [2]. In particular, many mobile devices are equipped with a special type of embedded storage solution called eMMC (Embedded MultiMediaCard). eMMC consists of NAND flash memory, controller, and MMC interface, all in a small ball grid array (BGA) package [3]. The optimization in the storage software stack, such as filesystems and I/O schedulers, also contribute to the storage performance.

This paper presents the design and implementation of AndroBench [4], a storage benchmarking tool for Android-based mobile devices. AndroBench measures the performance of sequential and random accesses to a file and the throughput of insert,

S. Sambath and E. Zhu (Eds.): Frontiers in Computer Education, AISC 133, pp. 667–674.
springerlink.com © Springer-Verlag Berlin Heidelberg 2012

update, and delete queries to a SQLite database. Currently, AndroBench is available in the Android market and can be downloaded freely by any Android-based mobile devices. The measurement results are automatically transmitted to our server. In this paper, we also present what we have found by analyzing those AndroBench results.

The rest of the paper is organized as follows. In Section 2, we briefly discuss the related work. Section 3 describes the methodology used in AndroBench in detail. Section 4 analyzes the benchmark results collected from various Android-based mobile devices. Section 5 concludes the paper.

2 Related Work

Many benchmarking tools, such as IOzone, IOmeter, and Postmark, have been developed to measure the storage performance and to analyze the characteristics of the underlying storage devices. IOzone is a microbenchmark program that supports a variety of file operations [5]. IOzone measures the I/O performance of accessing a single file with read, write, re-read, re-write, read backwards, fread, and fwrite operations. IOmeter is an I/O subsystem measurement and characterization tool for single and clustered systems, originally developed by Intel [6]. Postmark measures the performance of creating/reading/appending/deleting small and short-lived files, modeled after mail server or web-based commercial server workloads [7].

Although it is possible to port these existing storage benchmarks for the Android platform, there are several restrictions when they are run as a mobile application. First, as there is no way of getting superuser privilege in the Android-based mobile devices, some benchmarks which require superuser permission cannot be executed. For example, IOzone relies on unmounting and remounting the filesystem before each benchmark run to clear out any caches, which cannot be done in user-level applications. Second, the time needed to run the benchmark program should not be long, since the benchmark program will be executed not by system administrators, but by non-expert smartphone users.

Several benchmark tools specialized for the Android platform are already available in the Android market. Quadrant is one of the most famous benchmark applications, which measures the performance of CPU, memory, I/O, and 2D/3D graphics [8]. These results are sent to the server and the server calculates the final benchmark score. However, Quadrant does not provide any information on what are being measured and how the final score is calculated from the individual measurement result. The RL benchmark is a mobile application which focuses on the SQLite performance [9]. The RL benchmark reports the total elapsed time for processing a number of SQL queries. However, the specifics of the benchmarking methodology have not been known either. In comparison to these benchmarks, AndroBench is intended to be an open, versatile, and more objective benchmark. In AndroBench, users can measure the storage performance by changing configuration parameters freely. We also plan to provide the detailed information on our benchmarking methodology in the AndroBench homepage to ensure objectiveness of AndroBench.

3 Methodology

3.1 Microbenchmarks

AndroBench uses four microbenchmarks to measure the sequential and random I/O performance. To measure the sequential read performance, AndroBench first creates a file (32MB in size by default) in the target partition (/data by default). Then, the file is sequentially read with the fixed buffer size (256KB by default). The sequential write performance is measured similarly except that the file size is reduced to 2MB by default. This is because otherwise the microbenchmark takes too long due to the slower write performance. For the random read/write performance, AndroBench measures IOPS or the number of I/O operations per second. Each read/write operation is 4KB in size by default and the default file size is 32MB for reads and 2MB for writes. While measuring the random read/write performance, the offset is chosen randomly and in a non-overlapping way in the 4KB boundary. AndroBench takes the average of three runs for each microbenchmark.

One of the important issues in measuring the storage performance is to minimize the effect of buffer cache. Unfortunately, Java does not provide any library that can bypass buffer cache. Therefore, our microbenchmarks are implemented in C using the Java Native Interface (JNI). Buffer cache is bypassed by opening a file with the O_DIRECT flag. When the filesystem does not support direct I/O (as in YAFFS2 and Ext2), AndroBench uses a brute-force method; it first creates a temporary file in the external SD card, whose size is equal to the main memory size. Then, the entire file is read to fill in the buffer cache, whenever AndroBench needs to clear the cached data in the buffer cache. Although the Linux kernel provides a very simple way to clear the buffer cache (using the /proc/sys/vm/drop_caches file), it requires superuser permission, hence cannot be used in AndroBench.

3.2 SQLite Benchmarks

SQLite is a small embedded database engine [10]. In the Android platform, SQLite is used as system and user databases storing such information as contacts, SMSes, and bookmarks of the web browser. Since SQLite stores the entire database as a single file on the host filesystem, the performance of SQLite is closely related to the storage performance.

The database schema used in AndroBench's SQLite benchmarks is modeled after the one used in the contact database. AndroBench creates a table consisting of 17 columns (12 integer types and 5 text types) in the /data partition, and performs three types of tests: insert, update, and delete. The number of transactions in each test is set to 300 by default. SQLite uses the page cache to process transactions. Before each benchmark, AndroBench calls SQLiteDatabase.releaseMemory() to release the allocated pages for SQLite in the page cache. The performance of SQLite benchmarks is measured as TPS or the number of transactions per second.

3.3 Changing Parameters and Collecting Measurement Results

AndroBench provides the Settings tab, where users can change various parameters such as the target partition, the file size and the buffer size used in the microbenchmarks, and the number of transactions in the SQLite benchmarks. The target partition can be any of internal (/data or /mnt/sdcard) or external SD card (such as /mnt/sdcard/external_sd) partition. Since AndroBench supports the external SD card, it can be also used to compare the performance of various microSD cards with different speed class ratings.

After successfully completing all the benchmarks, AndroBench transmits the measurement results into the central server. The information collected includes the result of each microbenchmark and SQLite benchmark, the model name, and the associated parameters. This information is used to analyze the storage performance of Android-based devices and to show the top 10 rankings for each benchmark category. When processing the submitted results, we use the 3σ rule; i.e., we only take the values within three standard deviations of the mean to get rid of the data with high variance for the given device. The use of the 3σ rule is known to provide approximately a 99.73% of confidence interval [11]. In our analysis in Section 4, we also exclude any device for which the number of results after applying the 3σ rule is less than 25.

4 Analysis

As of August 30th, 2011, AndroBench accumulates the total 2,628 microbenchmark results from 236 devices during the past three months. For the SQLite benchmarks, the total 845 results are collected from 124 devices during a month. In the following subsections, we analyze the sequential and random I/O performance, the SQLite performance, and other characteristics of the storage performance in detail.

4.1 Sequential I/O Performance

Table 1 and 2 list the top 5 devices which show the highest sequential read and write bandwidth, respectively. The international model (GT-I9100) and the domestic

Table 1. Sequential read (MB/s)

Model name	Average performance	Standard deviation
GT-I9100	43.76	2.9
SHW-M250K	41.89	2.26
SHW-M250S	41.78	2.18
GT-P1000	26.31	1.42
HTC Sensation	25.93	1.99

Table 2. Sequential write (MB/s)

Model name	Average performance	Standard deviation
GT-P1000	5.74	1.36
GT-I9100	5.28	0.99
HTC Sensation	5.21	0.79
Desire HD	5.04	1
SHW-M250K	4.71	1.21

models (SHW-M250K, SHW-M250S) of Samsung Galaxy S2 exhibit the sequential read bandwidth larger than 40MB/s. Samsung's Android-based Galaxy Tab (GT-P1000) shows the highest sequential write bandwidth. We can see that the sequential read is faster than the sequential write by almost a factor of 8 due to the characteristics of NAND flash memory.

4.2 Random I/O Performance

Table 3 and 4 present the top 5 devices for the random read and write performance, respectively. For the random read performance, Samsung's Galaxy S series such as SHW-M110S (domestic model) and GT-I9000 (international model) accomplish over 1420 IOPS. On the other hand, HTC Desire is a winner in the random write performance showing about 200 IOPS. Again, the random write is far slower than the random read due to the characteristics of NAND flash memory. We also note that the device with high sequential I/O performance does not necessarily show high random I/O performance.

Table 3. Random read (4K IOPS)

Model name	Average performance	Standard deviation
SHW-M110S	1430.14	173.53
GT-I9000	1421.59	101.68
GT-I9100	1366.78	73.39
SHW-M250S	1362.6	60.23
Nexus S	1210.8	51.17

Table 4. Random write (4K IOPS)

Model name	Average performance	Standard deviation
HTC Desire	196.64	153.77
GT-I9000	109.31	34.83
SHW-M180S	97.39	64.28
SHW-M110S	91.71	45.11
Desire HD	58.9	5.06

4.3 SQLite Performance

Table 5, 6, and 7 compare the top 5 results of the SQLite benchmarks. HTC Desire shows the best performance in all SQLite benchmarks, recording 54.66 TPS, 55.75 TPS, and 52.07 TPS in the insert, update, and delete benchmark, respectively.

Table 5. Insert performance (TPS)

Model name	Average performance	Standard deviation
HTC Desire	54.66	30.78
SHW-M110S	43.22	12.23
GT-I9000	38.5	5.55
Desire HD	27.68	7.64
GT-I9100	18.4	1.22

Table 6. Update performance (TPS)

Model name	Average performance	Standard deviation
HTC Desire	55.75	25.95
SHW-M110S	44.52	12.18
GT-I9100	42.6	3.37
GT-I9000	40.96	6.35
SHW-M250S	40.93	2.49

We observe an interesting point in the SQLite performance. The results of the update benchmark are highly correlated with those of the delete benchmark. We believe this is because both benchmarks have a similar database access pattern where records are randomly updated or deleted.

Table 7. Delete performance (TPS)

Model name	Average performance	Standard deviation
HTC Desire	52.07	28.76
SHW-M110S	45.19	12.51
GT-I9100	42.14	3.27
GT-I9000	40.67	5.47
SHW-M250S	39.38	2.04

4.4 Changes in the Storage Performance Over Time

Fig. 1 illustrates changes in the sequential and random I/O performance over time for four models of Samsung Galaxy series. SHW-M110S and SHW-M130L are domestic models of Galaxy S, and GT-P1000 is the 7-inch model of Galaxy Tab. GT-I9100 is the international model of the next-generation Galaxy S2. In Fig. 1, the x-axis is arranged in chronological order of release date of the corresponding model.

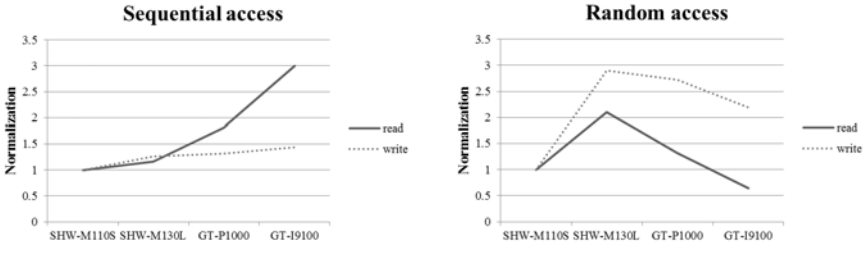

Fig. 1. Sequential and random access performance of Samsung Galaxy series

We observe that the sequential read and write performance have been improved over time, where the increasing rate of the sequential read is faster than that of the sequential write. This is because each model adopts a newer version of eMMC, which provides better sequential bandwidth. However, the random read and write performance of eMMCs are not significantly improved over time. The latest model of Samsung Galaxy S2, GT-I9100, shows the random I/O performance worse than its previous models. From these results, we can see that the random I/O performance of eMMCs is much harder to be improved.

4.5 The Effect of Filesystems

Two different filesystems, YAFFS2 and Ext4, have been used in HTC Desire. Fig. 2 compares the performance effect of using these filesystems on the same device. Ext4 shows the higher sequential I/O performance than YAFFS2. On the contrary, YAFFS2 is very strong on the random I/O performance. This is because YAFFS2 is based on LFS (Log-structured File System) [12], where filesystem updates are sequentially logged in the storage.

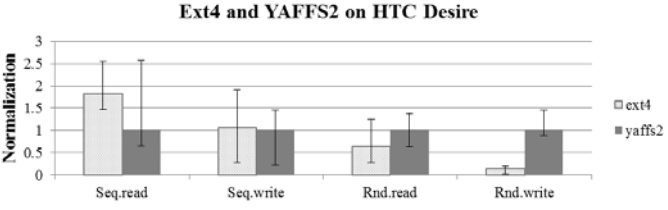

Fig. 2. The effect of using different filesystems (Ext4 and YAFFS2) on HTC Desire

The results of sequential and random read performance shown in Fig. 2 require careful attention. It is known that the log-structured design impairs the read performance as the data blocks belonging to a file can be spread out over the storage. Especially, the random read performance has no benefit in YAFFS2 as each 4KB block will be fetched randomly in both YAFFS2 and Ext4. In spite of this, YAFFS2 shows the significantly higher random read performance compared to Ext4. We believe this is because the kernel readahead is enabled in YAFFS2 which prefetches up to 128KB of data to the page cache. Unlike Ext4, we could not make YAFFS2 bypass the page cache as it does not support direct I/O.

4.6 The Effect of eMMC Upgrade on the Same Device

Fig. 3 plots the sequential read and random write performance of the submitted results for GT-I9000. Interestingly, we can observe that all the measurement results are clustered into two groups: around 18MB/s and 13MB/s for sequential reads, and around 150 IOPS and 50 IOPS for random writes. The reason of these results is due to the internal eMMC upgrade in the same model. The GT-I9000 model is known to use two kinds of eMMCs, one from Samsung and the other from Sandisk.

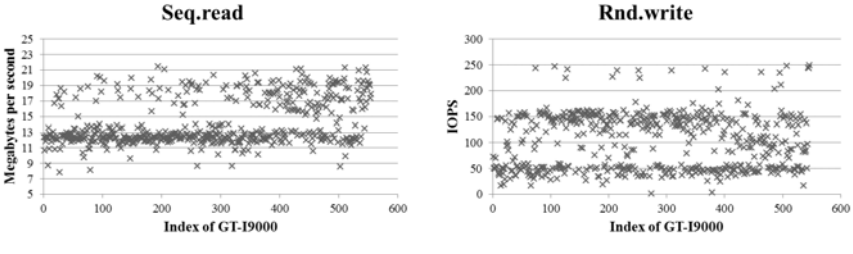

Fig. 3. The sequential read and random write performance of GT-I9000

5 Conclusion

This paper presents the design and implementation of AndroBench, which measures the storage performance of Android-based mobile devices. AndroBench measures the sequential and random I/O performance and the throughput of various types of SQLite transactions. We also analyze the AndroBench results we collect from hundreds of devices from all over the world.

Our analysis shows that no single device outperforms another one in all the benchmark tests we performed. Samsung Galaxy S2 shows the best sequential read performance. Samsung Galaxy Tab appears to be better than other devices at the sequential write performance. HTC Desire demonstrates the strong performance in the random write performance and all of SQLite benchmarks. We also find that the type of eMMC and the filesystem significantly influence the storage performance.

The actual storage access patterns in the Android-based mobile devices are far more complex than the benchmark tests we have performed in AndroBench. We plan to characterize the important storage access patterns in the Android-based mobile devices and to enhance AndroBench so that it can measure the real-world workloads in a multithreaded environment.

Acknowledgements. This work was supported by Mid-career Researcher Program (No. 2010-0026511) and by Next-Generation Information Computing Development Program (No. 2010-0020730) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology.

References

1. Kusnetzky, D.: Application Performance Requires Better Storage Performance, Document #20110110, The Kusnetzky Group (2011)
2. Heger, D.A.: Mobile Devices - An Introduction to the Android Operating Environment Design, Architecture, and Performance Implications., DHTechnologies (2011), http://dhtusa.com
3. MultiMediaCard (2011), http://en.wikipedia.org/wiki/MultiMediaCard
4. AndroBench (2011), http://www.androbench.com
5. IOzone, http://www.iozone.org/
6. IOmeter, http://www.iometer.org/
7. Katcher, J.: PostMark: A new filesystem benchmark. Tech. Rep. TR3022, Network Appliance (1997), http://www.netapp.com/tech_library/3022.html
8. Quadrant, http://www.aurorasoftworks.com/products/quadrant
9. RL benchmark, http://redlicense.com/
10. SQLite, http://www.sqlite.org/
11. Sapsford, R., Jupp, V.: Data Collection and Analysis. Sage Publications Ltd.
12. Rosenblum, M., Ousterhout, J.K.: The Design and Implementation of a Log-structured File System. ACM Trans. on Computer Systems 10(1), 26–52 (1992)