MobileBench: A Thorough Performance Evaluation Framework for Mobile Systems

Cholmin Kim, Jung-ho Jung, Tae-Kyeong Ko, Seung Woo Lim, Sunhye Kim, Kwanghee Lee, Wonju Lee
{cholmin.kim, jace.jung, tk.ko, sws.lim, sunhye.kim, kerri.lee, wonju.lee}@samsung.com

Abstract: Currently several benchmark tools for mobile platforms are available. However, because of the diversity of mobile devices and their components, test results from those tools are highly variable with low comparability between different devices. In addition to that, the results of previous mobile benchmark tools have little information about the user experience. In this paper, we introduce our new benchmark tool-set (MobileBench) which offers better comparability between tests and user experience evaluation that was not included in previous tools. The MobileBench consist of two parts: the physical performance benchmark and the user experience benchmark. We will explain the algorithms and methods that are used in the tools. The evaluation results and comparisons with other benchmark tools will also be provided.

Keywords: Mobile Platforms, Android, Benchmark, Performance Evaluations, User Experience.

1. Introduction

As smartphones become more popular, more people use them and many different devices from different vendors are available. Currently the number of active smartphones worldwide is more than 686 million [1]. Also, there are more than 8 manufacturers that produced more than 24 million smartphones between July 2011 and June 2012 [2]. If we also consider tablets, those values become larger.

Inside this great field of diversity, the criteria to select a good smartphone or tablet vary with individual preferences. Generally, hardware designs and features, operating systems, user interfaces and available software are considered important factors when selecting a smartphone [3]. If we exclude the design and eco-system correlation from those, the rest is related to physical performance and user experience. The physical performance can be reduced to the performance of each component that comprises the mobile device while the user experience is reduced to latencies that can be recognized by users in specific scenarios.

Previous mobile benchmark tools such as Quadrant [4], AnTuTu [5] and CFBench [6] concentrate on the physical performance. However, these tools are weak in comparability and stability. The results of these tools are hard to compare between OSes or have high deviations. The reason for these weaknesses comes from the difficulty of implementing an evaluation method with a high level of abstraction. Moreover, these tools do not provide general methods to measure user experience, or latencies in specific scenarios that users can frequently notice.

In this paper, we introduce a new mobile system benchmark tool called MobileBench. MobileBench can measure physical and user experience performances objectively and precisely. Since the evaluation algorithm of MobileBench is simple and we open it objectively, we can have those advantages comparing with previous tools.

MobileBench is convenient for the evaluator. In order to measure and monitor system performance with previous benchmark tools, evaluators had to install many applications for different purposes. However, MobileBench provides all in one solution.

The MobileBench consists of two sub-tools: a physical performance evaluation tool and a user experience performance evaluation tool. We have finished implementing both on Android systems. We named the first one as MobileBench and the second as MobileBench-UX. The name ‘MobileBench’ is used for both the whole tool-set and a sub-tool.

We will introduce our measuring algorithms and methods used in MobileBench, and also provide an evaluation result on Android smartphones and tablets. Several evaluation results from categories similar to previous benchmark tools will be compared with those of previous tools.

Since both tools use abstract algorithms, which are easy to port to different platforms, we are developing the same benchmark tools for WindowsPhone and iOS. We are expecting a platform independent benchmark between mobile devices after finishing those implementations.

2. Motivation

2-1. Previous Benchmark Suites

In this section, we will briefly describe several existing mobile benchmark tools and their pros and cons. These tools concentrated on benchmarking each component such as CPU (GPU), memory (RAM) and storage. They provide a score with the benchmark result and some of them provide an Internet page for relative score comparisons.

A. Quadrant

Quadrant is a benchmark for mobile devices. It covers CPU, memory, storage I/O, and 3D graphics performance. It has a convenient user interface that can run tests for CPU, I/O and 3D graphics with a single tap.

This tool is especially useful if the evaluator likes to change storage capacity or install apps that aim to improve the device's performance, and wants to check those possible improvements. But in objective comparison between mobile devices, it has less meaning. It provides little information for comparison between mobile devices. It shows the device rate compared on a bar graph with the average rate of other mobile devices, but not in detail. Another problem of this tool is that it cannot run if the target device does not have a GPU. For example, Samsung Nexus S cannot run this tool.

B. AnTuTu

AnTuTu Benchmark is a benchmarking tool for Android devices. It can run a full test of a key project through the "Memory Performance", "CPU Integer Performance", "CPU Floating Point Performance", "2D 3D Graphics Performance", "SD Card Reading/Writing Speed", and "Database IO" performance tests, and gives an accurate analysis for Android smartphones. AnTuTu Benchmark supports the latest quad-core CPUs. In reaching the overall and individual scores of the hardware, AnTuTu Benchmark can judge a phone by the scores of the performance of the hardware. By uploading the scores, it can view your device in world rankings, assigning points to let you know the performance level of the hardware.

However, there is no clear description for used algorithms. It is hard to user to interpret the result and some high

differences between trials.

C. CFBench

CF-Bench is (mainly) a CPU and memory benchmark tool specifically designed to be able to handle multi-core devices, produce a stable score, and test both native as well managed code performance. It tests specific device properties user do not regularly see tested by other benchmarks, and runs in a device timeframe.

All these tools provide physical performance evaluation results. But it is hard to compare those scores between mobile devices because the meaning of the score is hard to understand intuitively. These tools also provide the comparison via a web site, but it is only for relative comparisons on specific criteria. According to our experiments, these tools had relatively high fluctuations in their results when we tested the same target mobile device.

Moreover, these tools do not provide user experience analysis. In other words, when using these tools you cannot be certain that device A will be faster than device B in spite of A having a higher score than B.

As we consider the complexity and variety of current mobile devices, a stable and objective benchmark tool is needed. We have developed stable and objective measuring methods for MobileBench. Since the algorithm is not affected by OS features, we can also port the algorithm to any platform.

2-2. User Experiences on Popular Apps

Current usage of mobile devices is very complex. We can see that at a glance by just investigating popular app categories [7].

Figure 1 shows the top 30 categories and the number of available applications in the official Android market from December 2012. The statistics assume that the most frequently downloaded applications would run most frequently too on the mobile devices. Since the Android market categorizes the application into only rough categories, we use more detailed categories to analyze the user experience.

As we can confirm in the app categories, mobile usage is extremely complex and varies widely. We can measure the performance of each component of a mobile device, for example, CPU, memory, and storage. We will discuss this in greater detail in the next section. However, we cannot represent user behavior with a set of those performance results. Thus it is very hard to predict the performance for a specific user experience. In other words, we cannot precisely predict the latencies of user input in a specific scenario. Thus, it is safe to measure the real latencies in frequently used scenarios.

For this purpose, we define user experiences as those that are frequently observed. By automating these experiences we can measure latency. We implemented automation into MobileBench-UX. Currently, MobileBench-UX is the only tool that can measure the perceived performance of Android products by performing the actual user-scenarios on the device.

3. The MobileBench Suite

In this section, we describe the features and algorithms of our benchmark suites. As we mentioned in the previous section, our benchmark tool consists of 2 suites: MobileBench and MobileBench-UX. MobileBench is an app for Android devices, while MobileBench-UX consist of a PC application and an Android app.

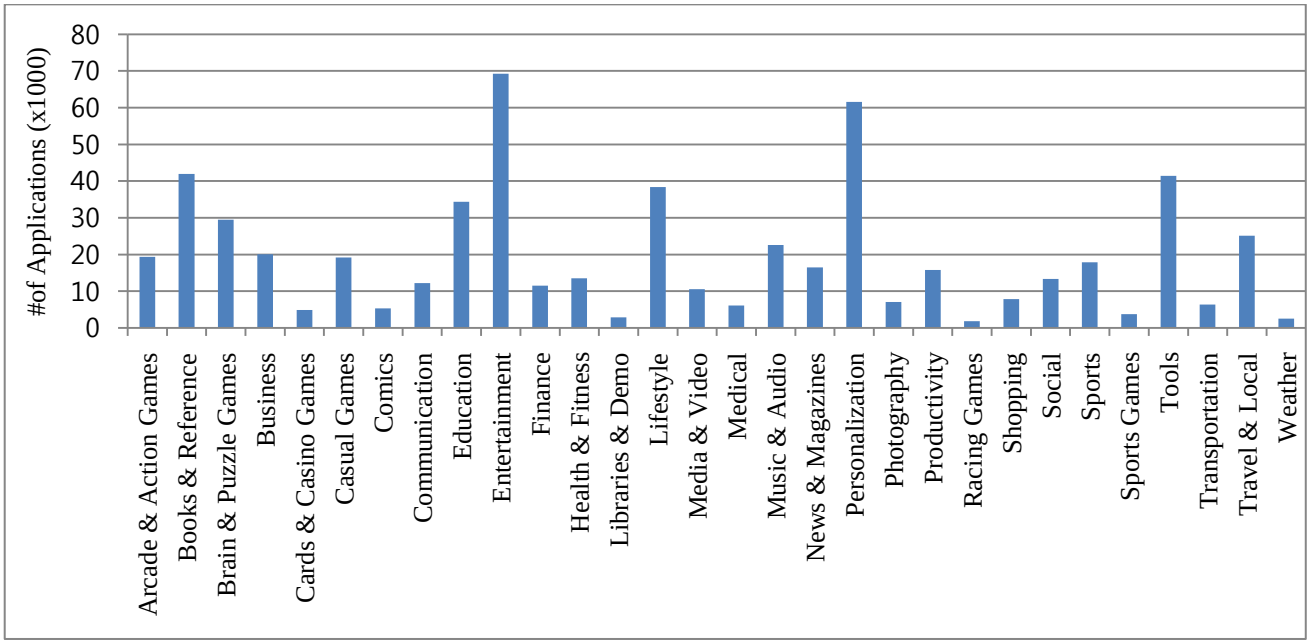


Figure 1. Top 30 Android Application Categories

Table 1. MobileBench Features with Components Correlations

Category	Features	CPU	Memory	Storage	Others
H/W Bench	Processor	O	X	X	X
	Random Access Memory	△	O	X	X
	Storage	△	△	O	X
Web Browsing	Java Script	O	△	X	X
	Web Loading	△	△	△	Communication Module
	HTML 5 Compatibility	X	X	X	Browser Compatibility
User Interface	UI Stress	O	△	X	X

O: High, △: Medium, X: No Correlation

3-1. MobileBench

MobileBench can measure the basic performance for each component and the overall Android device. We defined 3 categories consisting of 7 features for the measurement. Table 1 shows the MobileBench features.

The H/W Bench category is used to measure the pure performance of each H/W component. We selected 3 components (Processor, Random Access Memory and Storage) for these. To measure the processor performance, we used the algorithm shown in figure 2.

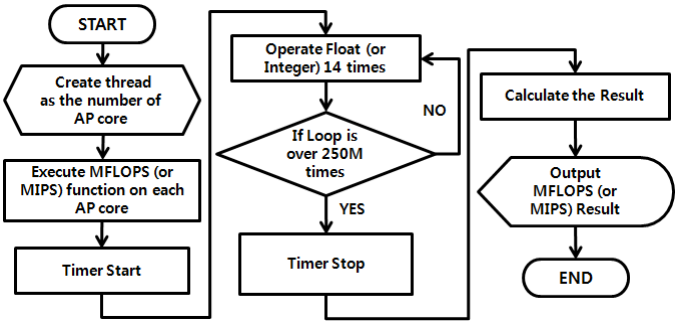


Figure 2. Algorithm to Measure Processor Performance

MobileBench retrieves the number of application processors (AP) from the target device then creates N threads, one for each AP core. Each thread runs 14 floating point (for MFLOPS) or integer (for MIPS) computations in a single loop. 7 was selected to avoid performance optimization by aligned array operation or lucky pipelining. The loop iterates 250 million times. With the elapsed time for looping to complete, we can calculate the MFLOPS or MIPS.

For memory evaluation, we used the algorithm shown in Figure 3.

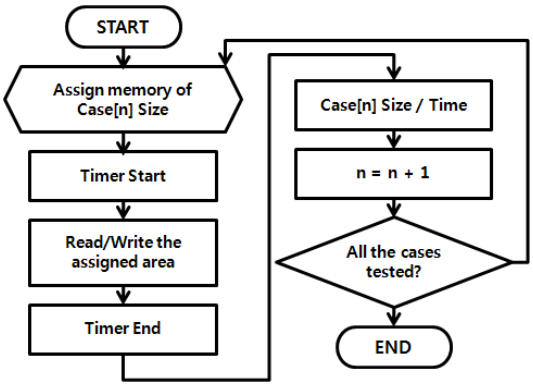


Figure 3. Algorithm to Measure Memory Performance

In memory performance evaluation, MobileBench first allocates 16 to 2048KB memory buffers (8 cases) for testing. In the read test, it fills the area with a random value, and then reads 8 variables of a single byte for (buffer size/8) times. In the write test, it writes the area with 8 variables of a single byte in a random pattern. We controlled the variables by ensuring that they were not cached. By measuring the duration for those operations, we can calculate the bandwidth of the target memory.

For storage evaluation, we used the algorithm described in figure 4.

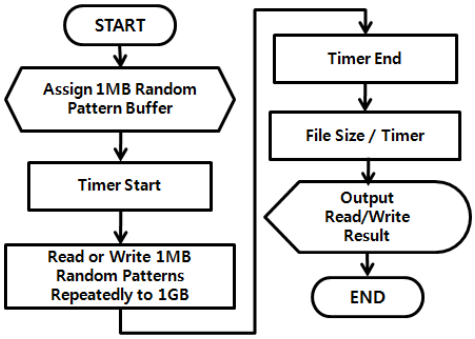


Figure 4. Algorithm to Measure Storage Performance

In storage performance evaluation, MobileBench generates 1,000 × 1 MB patterns. It reads or writes these patterns to storage and measures the elapsed time. Using the elapsed time we can calculate the IOPs (Input or Output Per Second). For example, if the test takes 20 seconds for write, the result is 50 write IOPs.

We have inserted Web Browsing test items in MobileBench to confirm browser compatibility. Since those items just check the compatibility, we omitted the test result of these items in the paper.

To measure physical performance when processing a user interface area object, we added a UI stress test to MobileBench. It creates a 2D view and object in the display. For each second, it inserts a new object into the view and checks the frame rate (frames per second) during the view update (Figure 5). By measuring the fps, we can find the point at which it drops below certain fps. At that point, MobileBench returns the number of objects that were inserted during the iterations. A large number represents a high speed in processing objects on the view since the device should process all views simultaneously.

Figure 5. Inserting 2D Objects for UI Stress Performance

3-2. MobileBench-UX

MobileBench-UX measures the user experience on mobile devices. MobileBench-UX consists of one program that runs on the mobile device and another that runs on a connected PC. The PC program controls the mobile device behavior via a USB connection. It measures the time needed to replay a specific user experience on the mobile device. It triggers an activity on the mobile device and calculates the time between the start and end of that activity. We have developed tests for 8 popular scenarios. Table 2 shows these scenarios and the related user experiences.

Table 2. MobileBench-UX Scenarios with Related User Experiences

Scenario	Description	Related User Experiences
Reboot	Reboots the device and measures the time it takes to boot up.	Booting
External File I/O	Reads and writes files between the PC and the device to measure read/write speed.	Data Sharing with PC
Internal File I/O	Measures read/write speed within the device.	File backup, loading applications, syncing
Media Scanning	Mounts media, broadcasts it, requests media scanning and measures scanning time.	Booting, mounting storage
Video Trimming	Measures the time for editing and saving the middle portion of a video.	Video recording, video editing
Camera Testing	Activates the device's camera and logs the time to activate the camera and save files.	Taking a picture
Installing Applications	Installs APK files from the PC to the device and measures the time it takes.	App installation
Loading Applications	Loads installed applications and measures loading times.	App execution, app switching

Table 3. Evaluation Methods for MobileBench-UX Scenarios

Features	Evaluation Methods
Reboot	1. Start timer with ADB (Android Debug Bridge) reboot. 2. Stop timer after getting connection complete message from logcat.
External File I/O	1. Generate 3 folders with random files on the PC. 2-1 Write: Measure the time to write files from the PC to the device using the ADB push command. 2-2 Read: Measure the time to read files from the device to the PC using the ADB push command.
Internal File I/O	1. Generate a 512 MB file. 2. Fill a buffer with an 8 KB random pattern. 3. Measure the time to fill the entire test file with the 8 KB pattern.
Media Scanning	1. Start timer on media scanning request to platform 2. Stop timer on an acknowledge of event completion
Video Trimming	1. Select a video to trim. 2. Select the trim region (15sec from start, 1 minute length). 3-1. Decode the trimmed area using ffmpeg. 3-2. Encode the trimmed area using ffmpeg. 4. Save the encoded video. 5. Measure the time from the ffmpeg start message to ffmpeg complete on logcat.
Camera Testing	1. Measure the time to take and save 10 pictures using the onTakenPicture event.
Install Applications	1. Install an APK file using the ADB command. 2. Get the average installation time for n APK files.
Loading Applications	1. Run a background process. 2. Run APKs. 3. Wait for the completion message from logcat and measure the time.

Table 4. Characteristics of the Evaluated Devices

Devices / Features	GS3(Galaxy S3)	G-Note10.1 (Galaxy-Note10.1)	Nexus7	G-Note(Galaxy Note)	Optimus LTE	GS2 HD (Galaxy S 2 High Definition)	Raider
AP	Exynos4412	Exynos4412	Tegra3	Exynos4212	APQ8060	APQ8060	APQ8060
AP spec	4-Core 1.4GHz Cortex-A9	4-Core 1.4GHz Cortex-A9	4-Core 1.3GHz Cortex-A9	2-core 1.4HGz Cortex-A9	2-core 1.5GHz Scorpion	2-core 1.5GHz	2-core 1.2GHz
GPU	Mali-400MP	Mali-400MP	ULP GeForce	Mali-400MP	Adreno220	Adreno220	Adreno220
RAM	2GB	2GB	1GB	1GB	1GB	1GB	1GB
RAM B/W	2CH LP2 800	2CH LP2 800	1CH DDR3 1333	2CH LP2 800	1CH LP2 800	1CH LP2 800	1CH LP2 800
Storage	16/32/64GB	16/32/64GB	8/16GB	16/32GB	2GB	16/32GB	16GB
Storage spec	Samsung eMMC (DDR), P1 Patch	Samsung eMMC (DDR) P7 Patch	Kingstone	Samsung eMMC (SDR)	Not accessible to internal Storage	Samsung eMCP	Samsung eMCP
Android Version	4.0	4.0	4.1	4.0	4.0	4.0	4.0

Each test item has specific measurement scenario which described in Table 3.

All scenarios are initiated by the PC program. It starts the timer and activates each feature. By monitoring the logcat logs through the USB connection, the PC program can detect the start and end events of each scenario.

File I/Os and Installing Applications scenarios can be represented in MB/s. The others are measured in ms for elapsed time for the behavior. In those scenarios, the elapsed time represents latency for the user. Thus, lower is better.

4. Evaluations

In this section, we provide the test result for MobileBench suite.

4-1. Environments

Table 4 shows the mobile devices that were used for the evaluation. All devices were used for MobileBench evaluation. 3 of them were used for the MobileBench-UX evaluation. All devices run Android as their operating system.

4-2. MobileBench

We tested each item with MobileBench 10 times on each mobile device and took the average value for the results.

Figure 6 shows the test results for the processor test.

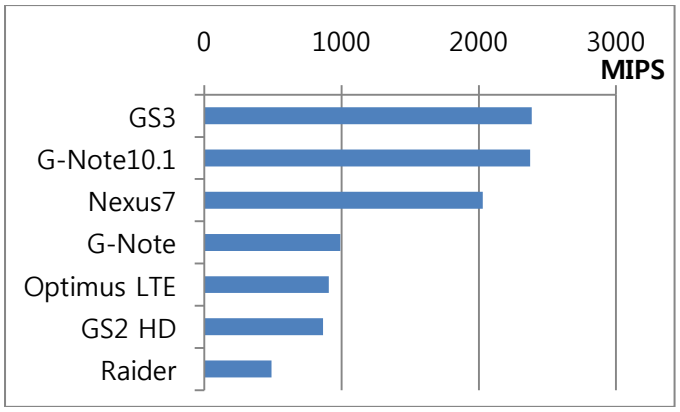


Figure 6. Processor (CPU) Benchmark Result

In the figure, the value of the X axis is in MIPS, since the processor benchmark measures MIPS in these experiments. As we can see in the graph, the GS 3 and G-Note 10.1 that use the Exynos 4412 quad core processor show the highest and similar value. Since the processor benchmark only measures CPU, the results for the GS3 and G-Note10.1 should be similar. The results of the Nexus7 are a little lower than the GS3 or G-Note10.1. These 3 devices use the same CPU core, ARM cortex-A9. But the Tegra 3 CPU in the Nexus 7 runs on a 1.3GHz clock while the Exynos 4412 runs in 1.4GHz mode. We assume that the differences are due to the core speed.

Figure 7 shows the benchmark results for memory.

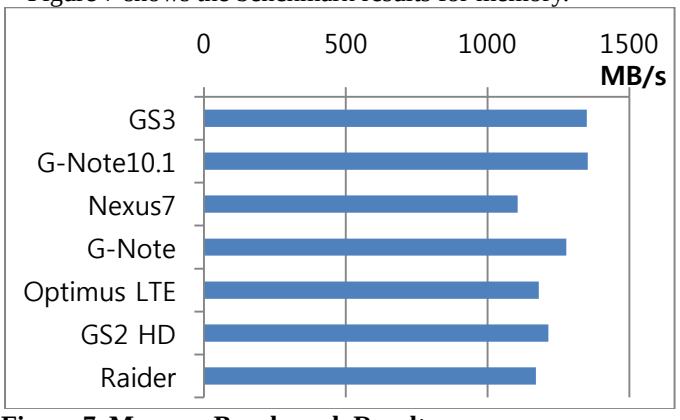


Figure 7. Memory Benchmark Results

In the figure, the X axis value indicates the average bandwidth (MB/s). The GS3, G-Note 10.1, and G-Note have the same and fastest Memory configuration. However, the G-Note has a slightly lower value than the other 2 devices. We believe that this small difference is due to the different CPU. The G-Note uses a dual core CPU while the other 2 use a quad-core CPU. The Nexus7 has the lowest value since it uses single channel of DDR3 at 1333MHZ while the others use one or 2 channel of LP DDR2 800.

From the results of this test, we can compare the bandwidth of each device without knowing the Memory component for a specific device.

Figure 8 shows the benchmark results for storage in the mobile devices.

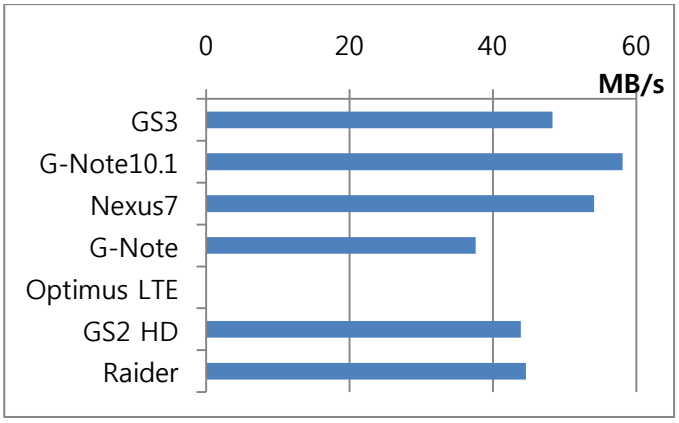
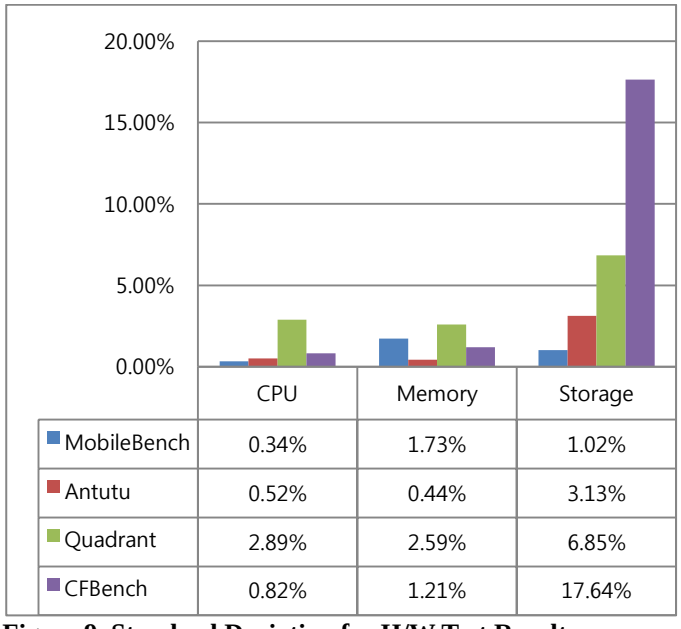


Figure 8. Storage Benchmark Result

In the figure, the X axis value indicates average bandwidth for read and writes. For example, the GS 3 shows almost 50 MB/s in the test case. Since we used a 1 MB unit for each I/O, 50MB/s can be converted to 50 IOPs (Input Output Per Second). The Optimus LTE has no test result since we cannot access its internal storage. Generally, internal storage in smartphones or tablets is eMMC.



	CPU	Memory	Storage
MobileBench	0.34%	1.73%	1.02%
Antutu	0.52%	0.44%	3.13%
Quadrant	2.89%	2.59%	6.85%
CFBench	0.82%	1.21%	17.64%

Figure 9. Standard Deviation for H/W Test Results

Figure 9 shows the standard deviation of H/W test results and comparisons with previous benchmark tools. We selected the GS 3 for the test mobile device and repeated the tests 10 times for each benchmark tool. We then calculate the standard deviation for 10 iterations. In the figure, the standard deviation is represented as percentage of the average of the 10

iterations.

As we can identify in the figure, MobileBench shows lower standard deviations than other benchmark tools. AnTuTu has lower standard deviations than MobileBench in the memory test, but has a higher standard deviation in the CPU and storage tests. Quadrant has higher standard deviations than MobileBench for all H/W tests. CFBench shows very high standard deviation. It is hard to interpret that high value since every test is done for same devices.

Lower standard deviations means that the fluctuations in the values of each test are low, thus we can say that MobileBench provides more stable test results than previous benchmark tools.

4-2. MobileBench-UX

We tested MobileBench-UX on 3 Android mobile devices: the GS3, G-Note 2, and GS2 LTE HD. Figure 10 shows the results of 5 tests. The result values for these tests indicate the response times of the relevant user scenario in millisecond unit.

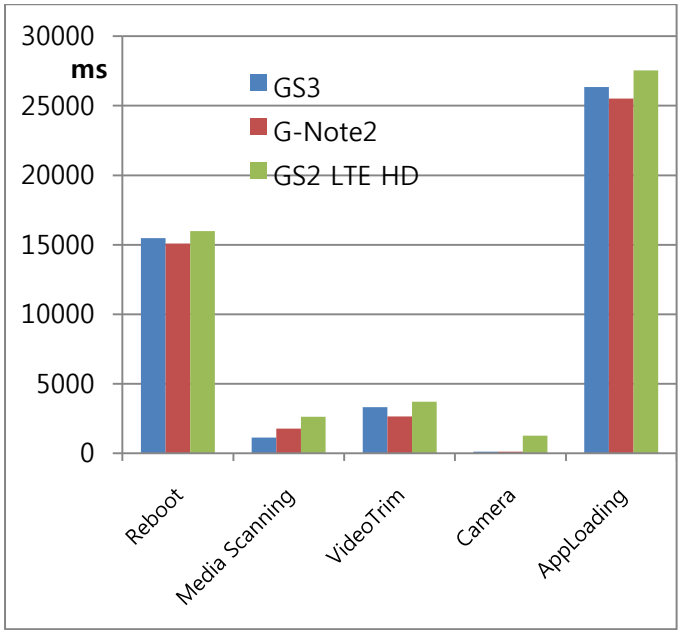


Figure 10. Elapsed Times for Scenarios

In the results we can see that the 3 devices have almost the same boot time. However, in media scanning, the GS3 is much faster than the GS2 LTE HD. Thus, users can expect faster responses when mounting devices, such as SD cards, with the GS3.

In app loading, the GS2 LTE HD takes about 2 seconds longer than other devices. Since it is about 8% of the average app loading time, users will notice more latency when starting apps. In the camera test, the GS2 LTE HD had a 10 times larger value than the GS3 or Galaxy Note 2. Thus, if a user frequently takes continuous shots, the GS3 or G-Note 2 is recommended.

Figure 11 shows the results for the other tests in MobileBench-UX.

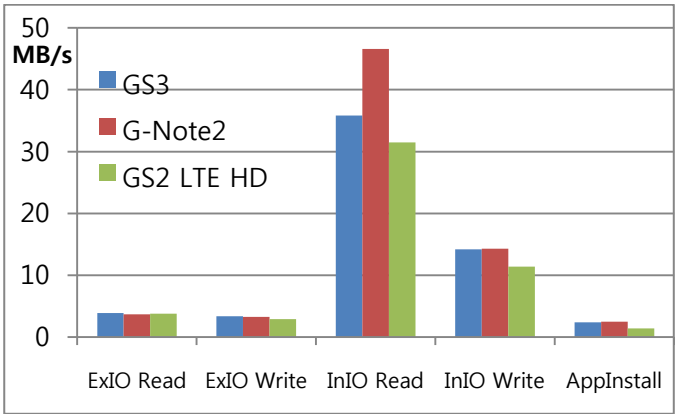


Figure 11. File IO and App Installation Evaluation Results

Since the Y axis in this graph indicates MB/s, higher values are better. In the external file I/O test, the 3 devices show little difference, but in the internal I/O case, the G-Note 2 shows higher bandwidth than the others. This means a user who updates the contents of his mobile device frequently is better off with a G-Note 2. Since the contents update includes internal file moving and re-ordering, Business, Mail, Personalization, Whether, News, and Magazine app categories can be included in this area.

5. Conclusions and Future Work

In this paper, we introduced MobileBench, a Mobile Benchmark suite. The benchmark suite includes 2 sub-tools: MobileBench and MobileBench-UX. In MobileBench, we introduced 7 features for physical performance evaluations. In MobileBench-UX, we developed 8 scenarios for user experience performance evaluation. Using MobileBench, we evaluated 7 different Android devices for evaluation. A portion of the results was used for comparison with previous benchmark tools, and we showed that the results for MobileBench are more stable than previous tools. Using MobileBench-UX, we evaluated 3 devices and showed that the tool can be used to measure user experience performance. The result of MobileBenchUX is user friendly since we can show the test result by latencies which can be understand by user intuitively. The user experience evaluation on MobileBenchUX is the first trial for mobile benchmarks. We also get the objectivity and reliability since the algorithm is opened, for user, and simple

Currently, there is no test item for 3D performance and pre-conditioned storage performance in the MobileBench and no multitasking test item for MobileBenchUX. We will include these features in the next version.

Both tools are ready for the Android platform and we are porting the tools for iOS and Windows 8. We expect that after finishing the port, we can compare the results between devices that run different OSes.

References

[1] “Smart Phone Shipments,” http://www.idc.com/getdoc.jsp?containerId=prUS23523812.

[2] “World's Top Selling and Most Profitable Smartphones,” http://www.rediff.com/business/slide-show/slide-show-1-tech-worlds-biggest-selling-profitable-smartphones/20120914.htm.

[3] “Smartphone Selection Criteria: What Factors Matter Most to You?,” http://www.techrepublic.com/blog/smartphones/smartph

one-selection-criteria-what-factors-matter-most-to-you/1674.
[4] "Quadrant," http://www.aurorasoftworks.com/products/quadrant.
[5] "AnTuTu Benchmark," http://www.antutulabs.com/AnTuTu-Benchmark
[6] "CF-Bench v1.2," http://forum.xda-developers.com/showthread.php?t=1134768.
[7] "Most popular Android market categories," http://www.appbrain.com/stats/android-market-app-categories
[8] "EEMBC," http://en.wikipedia.org/wiki/EEMBC.
[9] "Android Developers Home Page," http://developer.android.com/index.html, Google
[10] "The Art of Computer Systems Performance Analysic ," Raj Jain, John Wiley & Sons, INC., 1991.
[11] "Technical Note for Samsung Mobile ," http://www.samsung.com/global/business/semiconductor/html/mobile-memory-ecosystem/mobile-memory-technical-note.html.
[12] "The AM-Bench: An Android Multimedia Benchmark Suite ," Chayong Lee et al, GIT-CERCS, 2012.
[13] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on, dec. 2001, pp. 3 – 14.
[14] A. Gutierrez, R. G. Dreslinski, T. F. Wenisch, T. Mudge, A. Saidi, C. Emmons, and N. Paver, "Full-system analysis and characterization of interactive smartphone applications," in In IEEE International Symposium on Workload Characterization, Nov 2011.
[15] "GLBenchmark," http://www.kandroid.org/online-pdk/guide/instrumentation testing.html, Kishonti Information Ltd.