Freeze It If You Can: Challenges and Future Directions in Benchmarking Smartphone Performance

Yao Guo, Yunnan Xu, Xiangqun Chen
Key Laboratory of High-Confidence Software Technologies (Ministry of Education)
School of Electronics Engineering and Computer Science, Peking University
{yaoguo, xuyn14, cherry}@pku.edu.cn

ABSTRACT

Benchmarking the performance of mobile devices such as Android-based smartphones is important in understanding and comparing the performance of different devices. Performance benchmarking tools such as Antutu have been widely used in both academia and industry. However, one of the main difficulties when benchmarking smartphone performance is due to the fact that the performance cannot be measured accurately and steadily. This paper investigates the challenges on performance benchmarking for Android-based smartphones. We identify key factors affecting performance benchmarking, which include frequency scaling setting, temperature, running background services, etc. Experiments show that some of these factors may cause performance fluctuation by as high as 60%. We show preliminary results in controlling the performance benchmarking process to generate steady results, for example freezing a smartphone in a refrigerator could remove most of the fluctuations. Finally, we discuss the implications of our study and possible future research directions.

CCS Concepts

•General and reference → Performance; •Human-centered computing → Smartphones;

Keywords

Operating systems, smartphones, Android, performance benchmarking, frequency scaling

1. INTRODUCTION

Mobile devices such as smartphones have been widely adopted in the past decade. The number of smartphone devices shipped has reached over 1.4 billion in 2015, about four times than the number of PCs shipped in the same year. Despite the popularity of iPhones, Android-based smartphones have been dominating the market with more than 80% of the share.

HotMobile '17, February 21-22, 2017, Sonoma, CA, USA

DOI: http://dx.doi.org/10.1145/3032970.3032979

Benchmarking the performance of a computer system is important for many tasks, ranging from hardware design, OS optimization, to minimizing the overhead of security approaches. Many studies have focused on the performance measurement and optimization on traditional computers such as Windows and Linux based devices, including various benchmarks [8], measurement studies and optimization methods.

However, there are few research focused on the performance benchmarking of mobile devices, compared to much enthusiasm from the energy perspective. Nonetheless, performance of mobile devices is still an important issue. Similar to traditional computer systems, When developing new techniques or approaches for mobile systems, it is important to measure the performance, or compare the performance of mobile devices in order to understand the performance overhead and implications of new approaches.

Although there are many benchmarks available for traditional computer systems, there are no widely adopted performance benchmarks for smartphones. If you want to compare the performance of two smartphones, the most widely used methods are benchmarking applications such as Antutu [6] and PCMark [4].

For example, many security approaches use Antutu to assess their performance overhead on Android smartphones, for example Compac [13], SEAndroid [12] and RootGuard [11]. However, it is difficult to achieve consistent results in Antutu benchmarking. In order to evaluate the performance overhead of SEAndroid [12], the authors chose to run Antutu for 200 times![1] One of the reasons of running Antutu so many times is because the variation is high. For some performance score components, the standard variation is over 10% of the mean value, which indicates almost prohibitively high variance among the benchmarking results.

Meanwhile, many device manufactures have been accused of cheating on Android benchmarking tools [5], including many big brands such as Samsung, HTC, LG and Asus. For example, Samsung has been shown to use "benchmark boosters" such as overclocking to increase the benchmark scores by 20-50%.

This paper attempts to answer the following two questions: *Why is it hard to measure the performance of smartphones accurately? What are the challenges and how do we control the measurement process in order to achieve steady performance scores?*

We present a detailed study on performance measurements of Android smartphones with benchmarks such as An-

[1]Each run of Antutu typically takes about 5-10 minutes.

tutu, as well as microbenchmarks for storage and database performance. We have found that the performance measurement can be easily affected by various factors, sometimes generating results more than 50% apart in the value of performance numbers. *The key factors affecting performance measurements include: CPU frequency scaling settings, temperature, background services running on the device, available memory, etc.*

Based on these experiments, we observe that we can improve the benchmarking scores significantly by changing the parameters in the test environment. For example, putting a smartphone in a colder environment, such as in a refrigerator, could improve the overall Antutu score by more than 60%. Modifying the frequency scaling governor in Android or killing unused background services can also improve the benchmarking scores significantly.

However, the focus of this paper is not on how to improve (or cheat) benchmarking applications, instead on how to achieve stability in performance benchmarking results. In a normal test environment, some components of Antutu benchmarking scores can be varied by as much as 50% without intentionally changing any environment parameters. The variation can be reduced by fixing the CPU frequency, fixing the environment temperature, as well as killing background services that may generate performance noises. We show that with these practices enabled, we can improve the stability of benchmarking results by an order of magnitude for both Antutu and other microbenchmarks.

Finally, we show that with our recommended benchmarking environment and parameters, the variations of both Antutu and microbenchmarks can be controlled within 1% of the mean value, which can be considered statistically stable.

This paper makes the following main contributions:

- To the best of our knowledge, this is the first explorative study on how to measure the performance of Android-based devices accurately and steadily.
- We have identified a list of factors that may cause fluctuation in smartphone benchmarking results and measured their effects through experiments on both benchmarking apps and microbenchmarks.
- We propose some best practices that can be applied in performance benchmarking, which can reduce the variation of Android performance measurement to a minimal degree.

2. BACKGROUND AND RELATED WORK

2.1 Background

Android has been the most popular mobile platform with more than 80% market share. Due to the openness of Android, there are thousands of different Android-based smartphone devices on the market.

Benchmarking the performance of a computing system (smartphones included) has always been an important issue. For smartphone designers, it is important to understand the performance bottlenecks in a system and fix the issues before shipment. For smartphone users, it is important to know the performance numbers before choosing a device. For researchers, whenever the system has been modified, either for improving security or optimizing for power, it is always important to test the performance implications to make sure it does not cause significant performance degradation.

Another very important issue, particularly for Android-based devices, is to compare the performance of different Android devices. Besides comparing hardware parameters, many evaluation and review reports will include some kind of performance measurement results, for example with tools such as Antutu. As many devices nowadays are equipped with processors of similar configuration and the same amount of memory, the performance testing results will demonstrate which device has been better optimized.

2.2 Existing Tools and Methods

Typically, performance evaluation of a computing system requires both benchmarks and testing tools. For example, SPEC CPU Benchmarks [8, 9] have been widely used for benchmarking the performance of computer systems, especially at the architecture level. The Linpack benchmarks [7] are also often used to measure a system's floating point computing power.

For a mobile system, the most widely used methods are themselves mobile applications, many of which can be downloaded directly from Google Play, including Antutu [6], GeekBench [2], PCMark [4], etc.

For example, Antutu [6] is one of the most widely used performance measurement tools, which is able to not only produce a set of performance scores, but also comparing the performance scores with similar devices to help users understand the advantages and limitations of the device. Besides an overall performance score, Antutu (as of version 6.0) also provides a breakdown including "3D (Graphics)", "User Experience (UX)" (including data security, data processing, I/O and gaming), "CPU" (including mathematics, common algorithms and multi-core computation), and "RAM" performance tests.

Other researchers also studied mobile performance issues. Yoon [14] studied the performance of the Android platform using a benchmark application, as well as public profile software. MobileBench [10] is a set of mobile benchmarks including both performance and user experience benchmarks.

In comparison to these existing work that focusing on how to evaluate or improve benchmarking scores, our main goal is to identify the reasons why performance numbers fluctuate and how to achieve steady performance results during benchmarking.

3. EXPERIMENTAL SETUP

We first evaluated different benchmarking methods on an Android smartphone to examine the extent of variation in the performance benchmarking results. we mainly use a Nexus 5 smartphone, which installs Android version 4.4 and Android version 6.0, respectively.

We perform experiments with the following benchmarks:

- We use the latest Antutu Version 6.0 as one of the representative benchmarking tools. Antutu is a comprehensive performance benchmarking application considering performance from different aspects including CPU, User Experience, Storage and RAM.
- We use GeekBench 3, which is an Android application focusing on both single core and multi-core performance benchmarking.

- We also wrote a microbenchmark that ports FIO [1], a flexible I/O testing application, to Android, to test the I/O, especially file and storage, performance.

- Finally, we use a database benchmark application for Android, ORM [3], to test the database (including the default SQLite and other database modules) performance of Android.

For most of the experiments, we restart the phone and wait for about 5-10 minutes for the device to reach a steady state before we conduct the experiments. This procedure is in accord with most practices recommended by benchmark producers and also widely used in research papers.

4. BENCHMARKING RESULTS

4.1 Original Antutu Results

Testing Method #1 (Normal): *We first show how the performance scores may vary in a normal benchmarking environment. We did not try to change the environment or control the testing process. Instead we run the benchmarks or benchmarking tools normally at room temperature, without intentionally changing any parameters.*

We ran the Antutu benchmark on a Nexus 5 with Android 4.4 for 15 times. Table 1 shows the analysis of the performance results, focusing on the distribution of the performance results. To expose the variation, we show the highest and lowest scores for each score element, the average scores (Mean), standard deviation (SD), as well as the coefficient of variance (CV)[2], which is used to indicate the degree of variation in the results.

Although the standard deviation of the overall scores is only 4% of the mean value, the difference between the highest and lowest overall scores in 15 runs is close to 14%, which indicates that the variance is pretty high.

When we look at the detailed score breakdown, we can see that for some UX (User Experience) and CPU scores, the differences between the highest and lowest scores are over 55%, while the standard deviation is also more than 15% of the mean value. In such cases, the high fluctuation in these scores means that the results cannot be used as an accurate indication during performance comparison. Running the benchmarking applications for many times does not solve this issue because the numbers are unsteady in nature.

4.2 Variations Due to Frequency Scaling

From the results in Table 1, it appears that the main factors affecting the final performance are mostly CPU-related scores. Because the default voltage scaling policy is set as "ONDEMAND", it may change the frequency in a different way every time Antutu runs, resulting in the performance fluctuation.

Testing Method #2 (Fixed Frequency): *In order to confirm whether frequency scaling increases the variance, we modified the frequency scaling (DVS) policy from "ONDEMAND" to "PERFORMANCE", which means that the smartphone will run at a fixed high frequency to achieve the best possible performance.*

[2]CV is calculated as SD/Mean.

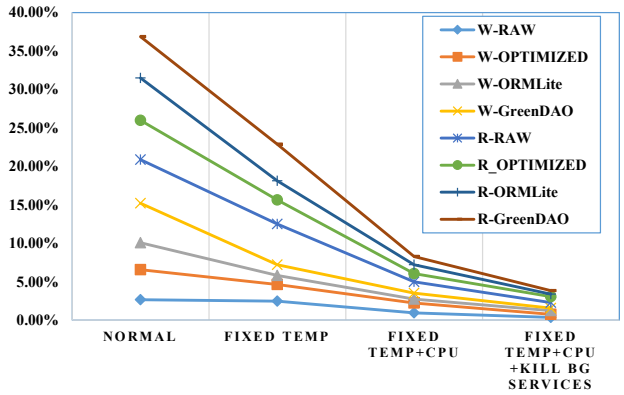

Figure 1: Database benchmarking results, showing only the coefficient of variance (CV) of each score components from 15 test runs. *Each line shows the CV of read or write performance using different database access modules.*

Besides Antutu, we also tested other benchmarks including PCMark and GeekBench, receiving similar results. Table 2 compares the results of GeekBench with and without controlling the CPU frequency. When we change the CPU scaling policy to "Performance", the overall results improve and the variance is reduced by almost an order of magnitude. At the same time, we observe that the overall performance has been improved by 40% simply by changing the CPU frequency scaling policy.

Besides benchmarking apps, we also tested the FIO microbenchmark. Table 3 shows the benchmarking results with FIO in 40 test runs, comparing the cases with different CPU governor settings. Originally with the "ONDEMAND" CPU setting, the variance is pretty high as the best and worst performance difference is more than 13%. When we fix the CPU frequency, the variance can be reduced by 3X.

Because FIO focuses on file and storage performance, it is not CPU-intensive, such that fixing CPU frequency at a high level only improves its performance by about 10% on average. However, with a fixed CPU frequency, we can control the performance variation much better.

4.3 The Effect of Temperature Control

Controlling the CPU frequency at the highest level may overheat the CPU and board, thus it should not be used as a recommended practice to conduct benchmarking. The temperature of the Nexus 5 smartphone has sometimes become extremely hot during our tests.

One of the reasons why the performance drops is because the frequency governor has to scale down the frequency when temperature readings are higher than a threshold. If we can control the temperature of the environment, we can improve the performance and probably reduce the variation as well.

Testing Method #3 (Fixed Temperature): *To measure the effects of environment temperature, we conduct experiments at a fixed lower temperature. During testing, without changing any configuration on the phone itself, we put it in a refrigerator that has a fixed temperature of* 4 °C.

We run the ORM database benchmarking application [3] under different temperatures: the first time under room

Table 1: Overall benchmarking results using Antutu 6.0 for Nexus 5 (Android 4.4) in 15 runs. Higher is better. ***(Diff: difference between highest and lowest scores; SD: Standard deviation; CV: Coefficient of Variance.)***

Antutu Score	Highest	Lowest	Diff	Mean	SD	CV
Overall	**33792**	**29719**	**13.71%**	**31233.27**	**1246.99**	**3.99%**
3D	**4030**	**3909**	**3.10%**	**3989.53**	**32.10**	**0.80%**
3D [Garden]	1980	1796	10.24%	1841.27	40.52	2.20%
3D [Marooned]	2177	2050	6.20%	2148.27	30.76	1.43%
UX	**9229**	**7964**	**15.88%**	**8722.93**	**408.07**	**4.68%**
UX Data Secure	2209	1885	17.19%	2029.87	80.27	3.95%
UX Data Process	736	627	17.38%	650.93	24.87	3.82%
UX Strategy Games	1261	807	56.26%	1078.87	165.33	15.32%
UX Image Process	3415	2999	13.87%	3278.27	146.99	4.48%
UX I/O Performance	1852	1507	22.89%	1679.00	112.45	6.70%
CPU	**15612**	**12467**	**25.23%**	**13597.53**	**979.70**	**7.20%**
CPU Math	4731	3428	38.01%	3636.07	306.49	8.43%
CPU Common	5227	3311	57.87%	4053.80	552.75	13.64%
CPU Multi-Core	6365	5532	15.06%	5907.67	307.44	5.20%
RAM	**5298**	**4673**	**13.37%**	**4923.33**	**162.92**	**3.31%**

Table 2: Benchmarking results with GeekBench 3. Higher is better.

GeekBench	Scores	Mean	SD	CV
Normal	Single Core	667.05	127.57	19.13%
	Multi-Core	2069.43	405.65	19.60%
Fixed Freq	Single Core	925.73	18.38	1.98%
	Multi-Core	2851.13	65.63	2.30%

Table 3: Testing results for FIO in 40 test runs. Lower is better. (Unit: millisecond.)

FIO	Normal	Fixed Freq
Highest	4820	4175
Lowest	4238	3993
Diff	13.73%	4.56%
Mean	4544.7	4047
SD	123.8	37.5
CV	2.72%	0.93%

temperature at about 25 °C, and then at a fixed temperature of 4 °C.

Table 4 shows the comparison of performance results of the ORM benchmark. The ORM application compares the performance of the write (W) and read (R) performance of different database modules, including the default SQLite (-RAW), the optimized SQLite (-OPT), ORMLite (-ORM) and GreenDAO (-GDAO).

We can see that when we fix the temperature at 4 °C, the overall performance does not increase significantly, but the variance has been reduced to about half (See Figure 1). When we fix both temperature and CPU frequency, the performance improves by 20-30%, while the variance has been further reduced.

4.4 The Effect of Killing Background Services

Besides temperature and CPU frequency, other factors might also affect performance numbers, including the number of running background services, the amount of free memory, the I/O or network activities running in the background, etc. We have tested various combinations, finding that background services and activities cause significant performance noises during performance benchmarking.

Next, we show the influence on performance results by killing background services. In contrast, running extra processes or services in the background will influence performance results in the opposite direction.

Testing Method #4 (Killing Background Services): *There are many processes and services running in the background when an Android smartphone boots up, depending on the number of pre-installed applications and services provided by the manufacture. We kill all user-mode processes running in the background before we run the benchmarking application. Note that some of the processes might restart automatically after being killed. We wait for 3-5 minutes for the system to reach a steady state before we run benchmarks.*

As an example, we still use the ORM database benchmarking results shown in Table 4 and Figure 1. In the last configuration, we show performance results after killing all background processes, besides fixing the temperature and CPU frequency. The results show that although we can only improve the overall performance marginally, we can reduce the performance variance further by about 50% after killing unused background services.

4.5 Antutu Revisited

Finally, we show Antutu benchmarking results again, with the same Nexus 5 smartphone running Android 6.0[3].

Table 5 shows the method we used to control the test environment such that we can reduce the variation in benchmarking results. The absolute performance numbers are similar to previous scores shown in Figure 1.

The results show that once we fixed the temperature at 4 °C (freeze the phone in a refrigerator), the variance has been reduced by more than half to less than 2%. After we remove the influence of background services, the variance is reduced further to about 0.5%. In this case, even without setting the CPU frequency at a fixed high level, we can still achieve very steady benchmarking results.

[3]Similar results can be achieved for Android 4.4 as well.

Table 4: Database benchmarking results with the ORM application, showing comparison for each score components from 15 test runs. Each column shows the values for Write or Read performance for each database module. Lower is better.

Setups	Value	W-RAW	W-OPT	W-ORM	W-GDAO	R-RAW	R_OPT	R-ORM	R-GDAO
Normal	Mean	2388.67	1109.27	5047.93	1482.47	602.07	488.73	1464.00	883.47
	SD	63.72	43.14	176.72	76.39	34.16	24.92	80.81	47.47
	CV	2.67%	3.89%	3.50%	5.15%	5.67%	5.10%	5.52%	5.37%
Fixed Temp	Mean	2327.80	1131.80	5098.40	1487.47	596.20	483.40	1439.93	863.20
	SD	57.29	24.62	59.48	20.79	31.59	15.06	36.14	40.97
	CV	2.46%	2.18%	1.17%	1.40%	5.30%	3.12%	2.51%	4.75%
Fixed Temp +CPU	Mean	1619.47	903.87	4141.13	1204.93	407.80	307.47	1122.40	598.80
	SD	15.45	11.48	20.91	9.27	6.10	3.26	12.78	6.39
	CV	0.95%	1.27%	0.50%	0.77%	1.50%	1.06%	1.14%	1.07%
Fixed Temp +CPU +Kill BG Services	Mean	1620.53	897.73	4080.53	1198.73	403.40	305.87	1088.20	598.93
	SD	5.94	3.47	19.28	4.02	2.92	2.33	3.60	2.64
	CV	0.37%	0.39%	0.47%	0.34%	0.72%	0.76%	0.33%	0.44%

Table 5: Antutu benchmarking results in different controlled environments for Nexus 5 with Android 6.0. We run Antutu for 5 times in each case.

Setups	Avg Score	SD	CV
Normal @ Room Temp	31833	1363.05	4.28%
Fixed Temp @ 4 °C	50716	970.92	1.91%
Killing BG Services	52013	276.29	0.53%

Although the results are still preliminary, it indicates that the variation of smartphone performance can be reduced in a controlled testing environment. As a result, there is no need to run Antutu for hundreds of times because the variation can be controlled within a small enough range.

5. DISCUSSIONS

5.1 Accurate and Steady Performance Benchmarking is Possible

Based on the results presented above, we recommend the following practices in minimizing the fluctuation in performance benchmarking. One should consider control the following parameters in the testing environment:

- The temperature in the testing environment should be controlled at a stable level. Running benchmarks on a smartphone will definitely increase its temperature (especially on inside boards), which affects the decisions made by the frequency scaling governor. Controlling the temperature at a lower level than normal room temperature will improve the stability of benchmarking results significantly.

  Note that although the overall performance may be improved when we conduct testing in low temperature, it does not affect the results because our focus is to compare the performance between different scenarios, instead of absolute performance numbers.

- Keeping the background activities at the minimal will help reduce the benchmarking noise. This includes killing unnecessary processes and services running in the background and keeping an eye on the available free memory to check whether there are any irregularities. Removing the influence of background activities also helps compare two different smartphones more fairly as different smartphone makers pre-install a different set of apps that might run in the background.

- In order to achieve real low variance, one should consider disable dynamic scaling and set the CPU frequency at a fixed level if necessary. For example, changing the scaling governor from ONDEMAND to PERFORMANCE will make the results more stable. However, this should be used with care. On one hand, because in reality frequency scaling is unavoidable, we risk creating an unrealistic testing environment. On the other hand, one may also risk the possibility to burn his/her phone if the temperature reaches a very high level.

Although far from an ideal solution, putting your smartphone inside a refrigerator or close to the AC vent will help stabilize the testing results.

5.2 Limitations

Benchmarking the performance of mobile devices is a complicated issue because many factors might affect the performance and its stability during benchmarking. The main goal of this paper is to raise the importance of this issue and inspire more concerns and ideas to better performance benchmarking approaches.

We have mainly focused on one aspect of mobile performance benchmarking, i.e., *the stability of benchmarking results*. If the result is far apart every time we take a test, it will be difficult to use the results to measure the performance overhead, which is typically very small. Thus if the performance overhead is only a couple of percentage, it only makes sense when the variance of testing results should be much lower than that.

However, there are more complicated issues surrounding mobile performance benchmarking. For example, frequency scaling is typically used to prevent overheating and saving battery. From this angle, it will difficult to tell whether we should disable frequency scaling to achieve steady performance, as a device has good reasons to scale down the frequency to respond to the running environment.

For the purpose of this work, we believe achieving steady results is important to compare the performance overhead of many research implementations, for example, security

or energy optimization approaches. Our main goal is to help researchers obtain steady results while evaluating the performance impact of newly applied mechanisms. However, comparing the performance of different smartphones would need to consider many other factors, which is out of the scope of this paper.

5.3 Future Research Directions

Performance benchmarking of mobile devices such as Android smartphones is an important issue that has not been studied extensively. Although we have shown preliminary results that performance benchmarking can be conducted accurately and steadily with careful control on the testing environment, we need a more systematic approach to benchmarking mobile performance.

A systematic study on factors resulting in performance variations. For example, how do the different factors affect each other in performance variations? How should we perform trade-offs between performance speedup and performance stability? What is the extent of performance variations on different smartphone devices? Besides smartphones, do these factors affect performance stability of PCs or servers as well? We plan to expand our research to cover these details to reveal more insights on smartphone performance benchmarking.

More comprehensive approaches to achieving performance stability. We have shown several methods that can be used to achieve performance stability, however methods such as "freezing the smartphone" can only be used as an ad-hoc solution, disabling frequency scaling should not be recommended as a best practice either. We need more comprehensive and practical solutions that can not only achieve performance stability, but also represent real-world testing scenarios as well.

Mobile benchmarks. The research community needs mobile benchmarks that can be used to evaluate the performance of a mobile device, but there are no good benchmark suites available at this point. Performance-oriented benchmarks can also be used to evaluate the power/energy aspects of a smartphone. Compared to traditional benchmarks that are computing-intensive, a mobile benchmark suite should consider the special characteristics of mobile devices, including user interaction, mobile gaming and different network configurations.

Performance testing tools. Many researchers agree that tools such as Antutu is not a good indication of device performance. Nonetheless, Antutu is still widely used in both academia and industry, because we do not have a better substitute. We, as a community, should be able to create a performance testing tool that reports accurate and steady results, such that we can keep ourselves and other consumers/researchers from being cheated by smartphone manufactures.

6. CONCLUDING REMARKS

This paper presents an explorative study on how to measure and compare the performance of Android-based devices steadily. Through a series of experiments, we show that measuring the performance of Android smartphones can be inaccurate and difficult to control due to various reasons. We also show that it is possible to generate stable performance results by controlling the testing environments and parameters.

Acknowledgment

This work was partially supported by the National Key Research and Development Program. 2016YFB1000105 and the National Natural Science Foundation of China under Grant No.61421091. We also want to thank the anonymous reviewers for their constructive feedbacks.

7. REFERENCES

[1] FIO: A Flexible I/O Tester. https://github.com/axboe/fio.

[2] GeekBench 3. https://play.google.com/store/apps/details?id=com.primatelabs.geekbench.

[3] ORM: Android application for benchmarking ORMLite and GreenDao. https://github.com/daj/android-orm-benchmark.

[4] PCMark. https://play.google.com/store/apps/ details?id=com.futuremark.pcmark.android.benchmark.

[5] AnandTech. The State of Cheating in Android Benchmarks. http://www.anandtech.com/show/7384/state-of-cheating-in-android-benchmarks.

[6] Antutu Labs. Antutu Benchmark. https://play.google.com/store/apps/details?id=com.Antutu.ABenchMark.

[7] Dongarra, J. J., Bunch, J. R., Moler, C. B., and Stewart, G. W. *LINPACK users' guide*. Siam, 1979.

[8] Henning, J. L. Spec cpu2000: Measuring cpu performance in the new millennium. *Computer 33*, 7 (2000), 28–35.

[9] Henning, J. L. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News 34*, 4 (2006), 1–17.

[10] Kim, C., Jung, J., Ko, T.-K., Lim, S. W., Kim, S., Lee, K., and Lee, W. Mobilebench: A thorough performance evaluation framework for mobile systems. In *The First International Workshop on Parallelism in Mobile Platforms (PRISM-1), in conjunction with HPCA-19* (2013).

[11] Shao, Y., Luo, X., and Qian, C. Rootguard: Protecting rooted android phones. *Computer 47*, 6 (June 2014), 32–40.

[12] Smalley, S., and Craig, R. Security Enhanced (SE) Android: Bringing flexible MAC to Android. In *NDSS* (2013), vol. 310, pp. 20–38.

[13] Wang, Y., Hariharan, S., Zhao, C., Liu, J., and Du, W. Compac: Enforce component-level access control in android. In *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy* (New York, NY, USA, 2014), CODASPY '14, ACM, pp. 25–36.

[14] Yoon, H.-J. A study on the performance of android platform. *International Journal on Computer Science and Engineering 4*, 4 (2012), 532.