Smartphones Exposed: Identifying Differences Between Seemingly-Identical Smartphones

Guru Prasad Srinivasa*, Scott Haseley†, Mark Hempstead‡, and Geoffrey Challen†
*University at Buffalo, †University of Illinois at Urbana-Champaign, ‡Tufts University

Abstract

Consumers, app developers, and many mobile systems researchers expect two identical smartphones to perform similarly. But transistor process variation can cause one device to heat up much more rapidly than another. As a consequence, this device may exhibit up to 20% performance and energy consumption difference despite the fact that consumers and experiments would have considered the two devices identical.

We make two important contributions to the understanding of temperature effects on today's mobile devices. First, we present results from carefully-controlled studies exploring the relationships between transistor quality, temperature, energy, and performance. Second, we have developed a novel way to expose normally-hidden information about transistor quality to consumers. Our approach is able to rank device transistor quality with a mean absolute error of less than 9 percentile-points for the Nexus 5 and 15 percentile-points for the Nexus 6. Furthermore, we are able to define equivalence classes based on transistor quality and classify devices within 96.00% accuracy at room temperature. It can be easily used by consumers to test the quality of their device, and by researchers to ensure that devices used in experiments are in fact equivalent.

1 Introduction

Your smartphone could be up to 20% worse in energy *and* performance than other devices of the same make and model. Variations in the underlying transistors may cause a device to exhibit different thermal characteristics. In turn, these thermal characteristics lead to devices heating up more quickly which forces them to slow down.

No two chips are produced equal. This is an inescapable fact of the chip manufacturing process. Increasing chip complexity and reducing transistor sizes has exacerbated this inequality, often referred to as process variation [21, 7, 8, 23]. Note that we are not referring to differences in battery life or flash performance caused over time due to wear. We are referring to differences that would separate two brand-new smartphones still in their original packaging.

Although process variation is well-known and well-understood, there is little evidence of it being considered by consumers, app developers and even mobile system researchers. Unlike desktop processors, where significant process variations result in frequency and pricing differences, the mobile market seems to paper over them. Consequently, good and bad chips find their way into devices that are identical in appearance and price.

How are they able to get away with this? By ensuring that all their CPUs operate at the same frequency—a process known as voltage binning [24]. In this process, all CPUs are configured to have the same operating frequency while their individual supply voltages are tweaked as necessary to ensure stable operation. Thus, to an unassuming consumer, their phone appears to be running just as fast as any other phone of the same model. Behind the scenes, however, the phone may be consuming more energy to do so, converting electrical energy to thermal energy and thus heating up in the users hand.

It does not end there. At their top frequencies, the heat generated by smartphone CPUs can reach their thermal limits within tens of seconds of beginning a compute-intensive task. Once these limits are reached, throttling strategies such as disabling cores or reducing CPU frequencies must be used, which in turn affect performance as well. As a result, consumers purchase their smartphones expecting their performance and battery life to be similar to those found in reviews and benchmarks, but may end up with a phone that can be up to 20% worse in terms of energy *and* performance [18].

For decades, system builders have effectively hidden these effects through the use of active cooling, careful case design, and controlled thermal environments. Unfortunately, smartphones frustrate all of the strategies es-

tablished to gain thermal control over hot CPUs. Unlike stationary devices and even larger mobile devices, smartphones get used in uncontrolled thermal environments—everywhere from hot cars to cold winter nights. Finally, smartphones are too small to incorporate the active cooling components commonly found on servers, desktops, and laptops, such as fans or large heat sinks.

Ambient temperature also plays a role through passive cooling since heat is transferred from the hot device to its surroundings by the second law of thermodynamics. Thus, lower ambient temperatures provide more passive cooling thereby giving the CPU with more thermal-headroom. Additionally, previous studies have shown that higher ambient temperatures can drain the battery 50% faster [16]. This is due to the fact that transistor current characteristics generally degrade at higher temperatures [20].

To understand the characteristics of their smartphones, users today resort to running benchmarks and comparing results. In the best case scenario, comparisons can be made against devices of the same model as that of the user's. In the worst case, users are left with a score and the scores of the top 50 device models overall [5]—a list in which their model may not even figure. Even in the best case scenario, the results are skewed in favor of lower ambient temperatures. The score of a good CPU would be no match to the score of a bad CPU if the bad CPU ran the benchmark at a significantly lower ambient temperature. Guo et al [9] discusses how putting a smartphone in a refrigerator could improve the overall score of Antutu [2], a popular Android benchmark, by more than 60%. Furthermore, running the same benchmark back-to-back would yield significantly different scores as the second run begins with the device already heated up as a consequence of the first run. To the best of our knowledge, none of the popular Android benchmarks [2], [3], [1] account for temperature while computing scores, be it temperature of the device or ambient.

The goal of our work is to establish a reliable technique with which users can rank their smartphones without having to explicitly account for temperature—both device and ambient. To that end, we implemented an Android application and published it on the Google Play-Store [6]. This study describes the core principles behind our technique and evaluates its accuracy.

2 Motivation

Our interest in studying process variations arose from our inability to reproduce performance results that we had earlier observed while running a CPU intensive benchmark. By swapping the SoC while keeping the workload, device casing, and battery constant, we confirmed that the SoC was the source of variation. Further experimentation revealed that the variations were caused due to intrinsic properties of the SoC.

The process of segregating CPUs based on its manufacturing quality and electrical characteristics is known as *CPU binning*. Note that because multiple cores that are part of a single CPU are drawn from the same patch of silicon, differences are between entire CPUs and not between cores. The two major binning techniques used by manufacturers are speed binning and voltage binning. When chips are manufactured, they are first tested to identify their stable operating frequencies. If a chip does not meet the necessary timing constraints or fails to operate at the expected frequency, the operating frequency is lowered until it passes the tests. The chips are then sorted into bins and labeled according to their speed. This process is called speed binning. They are then sold at price points proportional to their speed bin [24].

Speed binning labels chips according to their speed. Voltage binning keeps the frequency fixed across all chips and adjusts the voltage across bins. Voltage binning is based on the fact that both speed and leakage power of a transistor are a function of the supply voltage. Slow transistors—ones with larger gate lengths—leak less, while fast transistors—ones with shorter gate lengths—leak more. Manufacturers thus divide the chips into voltage bins where slower chips are binned at higher voltage so as to support the required operating frequency, while faster chips are binned at lower voltage in order to reduce their already high energy consumption. We believe that this is done in order to try and provide consistent performance (in terms of speed) across all devices using the same SoC.

Table 1 lists voltages used for multiple frequencies across bins on a Nexus 5 device. Bin-0 has the slowest transistors while Bin-6 transistors leak the most. Therefore, Bin-6 operates at lowest voltage while Bin-0 voltage is increased to enable equal performance as Bin-6. Manufacturers thus use this technique to attempt to enable consistent performance across all bins. Note that the process controls for speed, so both the Bin-0 and Bin-6 CPUs provide the same set of operating frequencies.

It is important to note that while desktop manufacturers typically have different price tags associated with each of their bins, reflecting the difference between good and bad silicon, mobile SoCs do not. To the best of our knowledge, mobile SoC manufacturers appear to be assigning CPUs of all bins with the same label which are then eventually sold to the consumer at the same price.

Different transistor properties combined with varying operating voltages leads to differences in the thermal characteristics between various CPU bins. These thermal characteristics in turn result in variations in both energy consumption and performance. Figure 1 describes the energy characteristics of the different CPU bins on

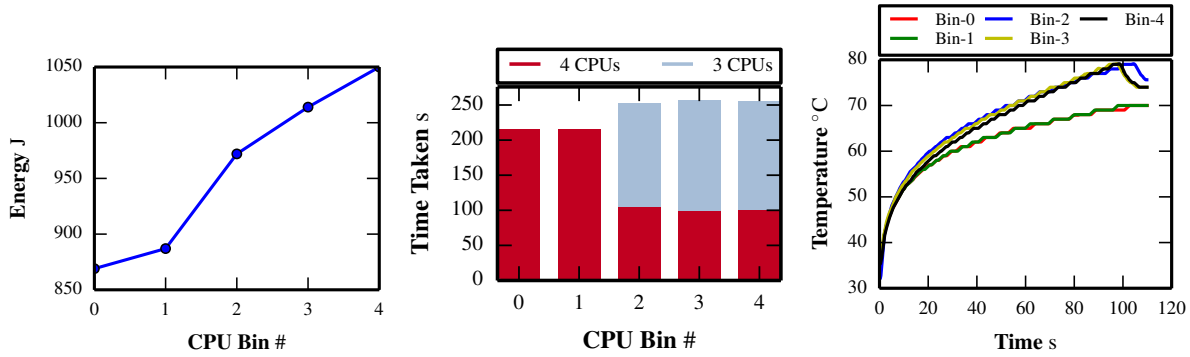

Figure 1: **Energy, Performance and Temperature Variation Across CPU Bins of Nexus 5**. Bin-4 consumes 20% more energy while also taking 18% longer due to thermal throttling. Once thermal limits of 80°C are reached, one CPU core is shut down.

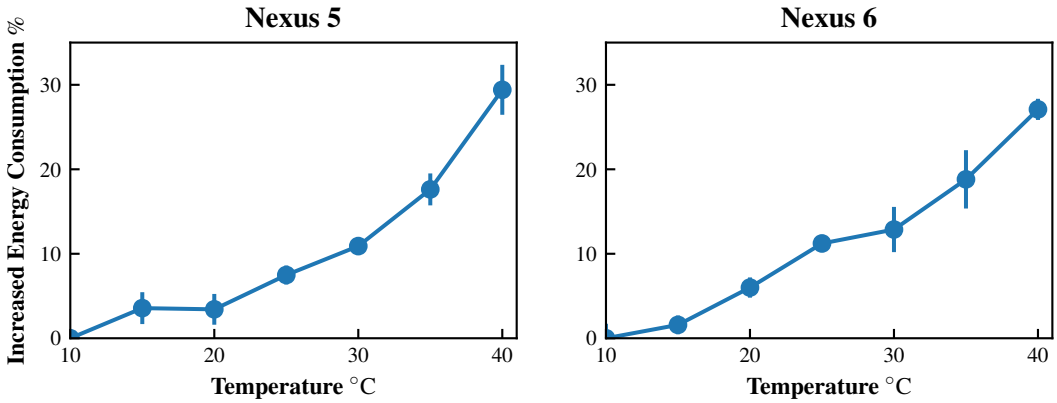

Figure 2: **Energy Scaling on Two Different Devices at Max Frequency**. Differences in ambient temperature can cause an increase of 25% or more energy consumption to do the same work. This effect is observed across devices.

Voltage	Frequency (MHz)				
(mV)	300	729	960	1574	2265
Bin-0	800	835	865	965	1100
Bin-1	800	820	850	945	1075
Bin-2	775	805	835	925	1050
Bin-3	775	790	820	910	1025
Bin-4	775	780	810	895	1000
Bin-5	750	770	800	880	975
Bin-6	750	760	790	870	950

Table 1: **Voltage vs. Frequency Across Bins.** Voltages for various frequency levels across bins for Nexus 5.

the Nexus 5. It plots the energy consumption of various Nexus 5 bins while performing a fixed CPU intensive workload. From the figure, we see that Bin-4 consumes about 20% more energy than Bin-0 while also taking ≈20% more time to do the same amount of work due to thermal throttling events.

Ambient temperature also plays a crucial role in determining the amount of energy consumed to do a certain amount of work. The leakage current of transistors is proportional to temperature [11]. Transistors that leak more also generate heat at faster rate compared to those that have lower leakage current. To make matters worse, in cases where the cooling rate is not increased, the higher heat dissipation increases the temperature of the device which in turn creates a feedback loop that increases leakage current. Figure 2 describes this trend for two different devices. Both devices consume up to 30% additional energy to do the same work at higher ambient temperatures.

Being aware of the differences between seemingly-identical devices of the same make and model is important, but being able to identify them is paramount. Identifying these differences will help consumers filter out bad devices and select those that are similar to online reviews and benchmarks. Detecting these differences will also benefit researchers who run experiments on a small set of devices and extrapolate their results to larger sets.

3 Design

The goal of our technique, which we refer to as ACCURANK, is to expose the underlying transistor differences of the CPU. We expose transistor variations by running a CPU intensive workload and comparing its results with

those from other devices. The idea being that a CPU with bad transistors would generate more heat and thereby yield lower performance thus scoring less in our CPU intensive workload.

The ACCURANK technique can be broken down into the following steps:

- Warm up the CPU for fixed time T_{warmup}
- Perform cooldown for fixed time $T_{cooldown}$
- Record device temperature θ_{expt}
- Run workload for fixed time $T_{workload}$

One of the problems with existing benchmarks is that they produce very different results on the same CPU depending on whether the CPU was previously idle or under use. The warmup phase is designed to mitigate this by synthetically generating some heat and warming up the CPU. Thus, CPUs that were idle become warm while CPUs that were previously warm continue to remain warm. The cooldown phase serves two purposes. First, it provides the hot CPUs sufficient time to cool down, and second, it enables us to estimate the ambient temperature of the device's surroundings. At the end of the cooldown phase, the temperature of the CPU is recorded and serves as a proxy for ambient temperature. Finally, the main CPU-intensive workload is executed.

The entire technique is packaged into an app that we have published on the Google PlayStore [6].

At its core, our app uses a WebView and all of the core functionality is written in JavaScript. This JavaScript code uses APIs exposed by the app to perform restricted operations such as reading the CPU temperature, acquiring wakelocks, logging and storing experimental logs, and in case a device is rooted, the app also tries to access the CPU-bin information of the device. The benefit of writing the app in JavaScript is that, the app can be easily updated in the backend, without waiting for each user to update their instance of the app. With this approach, the latest JavaScript code is pulled as part of the web page and executed every time the user opens the app.

The app requires no special permissions other than the `READ_PHONE_STATE` permission which allows us to log the unique IMEI of the device. We use the IMEI distinguish between multiple devices of the same model. In cases where the user does not provide this permission, we use the less-reliable `Settings.Secure.ANDROID_ID`.

The app allows users to view detailed information about their device such as number of CPUs, RAM size, and a comprehensive history of the device's CPU temperature, among others. The core functionality of the app, however, is the ability to test the current device. When the user chooses to test her device, the app does the following:

1. **Acquire wakelock** to ensure all operations complete as expected without the device entering sleep state
2. **Check if battery** $> 80\%$ to ensure that the operating system does not affect device performance
3. **Track changes in charging state** to ensure the experiment can put the device to sleep during cooldown
4. **Run ACCURANK**

Both the warmup and workload consist of running a CPU intensive task on all available CPU cores, for a fixed duration of time. In our experiments, T_{warmup} was configured to run for 3 minutes to try and allow an idle CPU to heat up to the same state as a busy CPU. A busy CPU on the other hand, would throttle and continue to maintain its heated state. We chose a 5 minute duration for $T_{workload}$ to ensure that devices have ample time to heat up and exhibit their thermal differences. The CPU intensive workload consists of computing the digits of π in a loop on all available CPUs. Specifically, we compute the first 4,285 digits of π. This number was chosen as it was estimated to take roughly 1 second to compute at the highest frequency on the Nexus 6. The number of CPU cores is determined via JavaScript's `navigator.hardwareConcurrency`. In our experience with several different devices, this API always returned the number of available cores accurately. Performance is measured by the number of iterations the device is able to complete across all cores within $T_{workload}$.

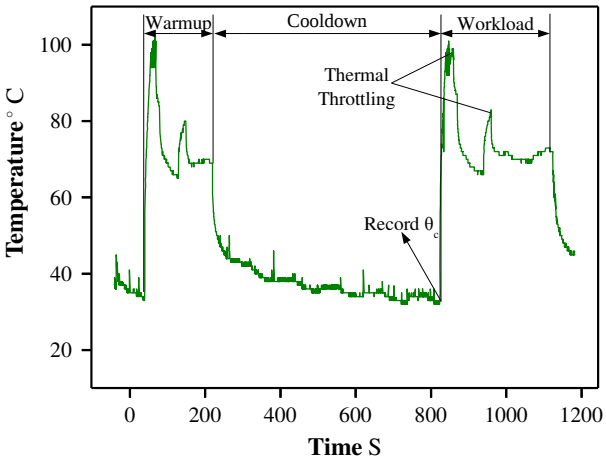

Figure 3: **Various Stages of ACCURANK**. The warmup and cooldown phases together act as a proxy for ambient temperature. The sudden drop in temperature during the warmup and workload phases are due to thermal throttling.

When the user triggers the experiment, the app acquires a wakelock to ensure the device does not sleep and begins the CPU warmup phase. As soon as the CPU warmup is completed, the device releases the wakelock and starts the cooldown phase. In this phase, the device

enters into a sleep state and wakes up momentarily every 5 seconds to poll the temperature sensor. This cooldown phase was configured to last for 10 minutes. We log the temperature at the end of the cooldown phase, which we refer to as θ_{expt}. Overall, the test currently takes $\approx$18 minutes to complete. Figure 3 depicts the various events that occur during our ACCURANK technique. It is important that the device stay unplugged during the cooling phase as plugging in the device prevents it from entering the sleep state. This, in turn, prevents the device from cooling down and introduces significant error in our ambient temperature estimate as the CPU may be doing work and actively generating heat while awake.

The rate at which an object cools down is exponential in nature and is determined by the difference in temperature between the object and its surroundings. This applies to smartphones as well. By allowing the CPU to cool down for a long period of time (10 minutes), we are attempting to ensure that the exponential curve has sufficient time to reach near-zero slope. Once the curve is at near-zero slope, the rate of decrease of temperature is low enough to assume a state of equilibrium with the surrounding ambient temperature, and can thus be considered as a reasonable proxy for ambient temperature. The relationship between the cooling curves and ambient temperature are studied in detail in Section 5.3.

The θ_{expt} temperature is a crucial feature that we use in conjunction with the observed performance of the device, while comparing it to other devices. This ensures that only experiments that were run under similar ambient conditions are being compared.

Once the experiment has finished, the θ_{expt} and performance are uploaded to the backend where the device is then ranked in relation to other experiments from other devices of the same model. The θ_{expt} acts as a filter to ensure that the resulting set of experiments all had similar device state and ambient temperature.

To ensure that we are strictly measuring hardware differences and not software, we also collect the Android build information. This includes the kernel commit as well as a fingerprint of the current Android image that is running on the device. Additional filters are then applied during the ranking stage to ensure that comparisons occur only between devices running the same software stack.

4 Experimental Methodology

Our experiments were performed on the LG Nexus 5 and Motorola Nexus 6 handsets, running the Android Open Source Project (AOSP) version 5.1 (Lollipop) and Cyanogenmod 13.1 (Marshmallow) respectively. The reason for selecting Cyanogenmod over AOSP for the Nexus 6 were purely based on a simpler building and flashing experience.

The Nexus 5 was released in 2013 and has a 4 core 2.26 GHz ARM CPU and 2 GB of RAM in a smartphone form factor. The Nexus 6 was released in late 2014 and consists of a 4 core 2.7 GHz ARM CPU and 3 GB of RAM in a larger "phablet" form factor. Since the operating system may alter device behavior under changing battery conditions, we decided to eliminate this source of variance by powering our devices via the Monsoon Power Monitor [4]. We configured the Monsoon to provide 3.8 V supply voltage—the standard voltage as specified by the device manufacturer.

As earlier studies such as [9] and [18] have shown, ambient temperature can play a crucial role in determining device performance. Following the best practices laid down by previous studies, all our experiments were performed in a controlled thermal environment which we refer to as THERMABOX. Temperature inside the THERMABOX was controlled using a RaspberryPi which measured the current temperature via a temperature probe. This RaspberryPi controller was also connected to a heating and cooling element which enabled it to regulate temperature within the THERMABOX. Heating and cooling the THERMABOX was achieved by power cycling a compressor and a 250W halogen lamp respectively. Figure 4 shows this setup. The controller was configured to ensure that the temperature inside the THERMABOX always stayed within $\pm 0.5^{\circ}$C of the target temperature. This setup was necessary to be able to produce reproducible results, particularly given how sensitive CPU performance can be to temperature.

Our first requirement as part of evaluating our ACCURANK technique, was that the results be repeatable at any given ambient temperature across a range of ambient temperatures. This mandated that we run multiple iterations of the experiment at each ambient temperature. The entire process was automated by our app which was able to communicate with our temperature controller to configure the temperature of the THERMABOX. Upon firing a particular intent on the device, the app first communicates with the THERMABOX and configures it to maintain a specific ambient temperature. It then periodically polls the THERMABOX state until it reports that it is stable for a consecutive period of 3 minutes, a process that may take up to 30 minutes. Once stable, the app performs 6 iterations of our ACCURANK workload back-to-back. Since the device is being actively cooled or heated by the THERMABOX for a significant duration while it attempts to stabilize at the configured ambient temperature, we discard the first iteration and use the remaining 5 iterations in our evaluations.

This process was repeated on bins 0–5 on the Nexus 5 and bins 2–7 on the Nexus 6.

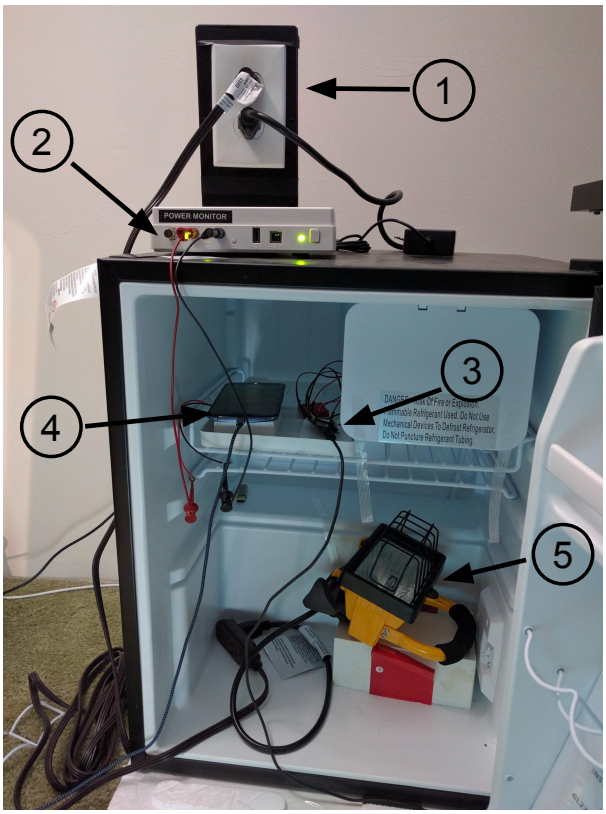

Figure 4: **Controlled thermal environment**. All our experiments were run inside a controlled thermal environment with an ambient temperature of 10±0.5°C. 1) Temperature Controller (RaspberryPi), 2) Monsoon, 3) ESP-8266+Thermistor (Temperature Probe), 4) Device, 5) Heating Element.

5 Evaluation

Recall that the primary question we are interested in answering is, 'How good is my smartphone?'. The answer can be formulated in a couple of ways, and we will describe both in this section and evaluate each answer's accuracy. Note that throughout our evaluation, we only compare devices of the same make and model. We next evaluate the importance and effectiveness of ambient temperature and θ_{expt} in ensuring similar device and environmental characteristics. Finally, we evaluate and discuss the impact of existing thermal management software on energy and performance on smartphones.

In the interest of full disclosure, we would like to point out that bin-4 of the Nexus 6 is not considered in the results presented here. Although we lack concrete proof, we have reasons to believe that there is something fundamentally different about this device. One visually discernible difference between this and other Nexus 6 devices was that this device exhibits a different boot logo. But in terms of temperature, this particular device consistently exhibited significantly different θ_{expt} compared to other Nexus 6 bins. We suspect that this particular bin-4 device may contain an SoC of a different revision.

5.1 Device Ranking

One approach to answering the core question is to provide an overall rank to a device in comparison to other devices that also ran our workload under similar device and environmental conditions. Once the results from other devices are filtered based on temperature, the device rank is simply represented as a percentile rank of the performance measured on this device in comparison to the filtered results. We evaluated this ranking algorithm by picking each experiment 'X' that we ran in our controlled environment, and querying our database for all other experiments that had similar θ_{expt} as that of 'X'. Since the temperature sensors only report integer values, our filter accepts a range of ± 1°C. For the ground-truth dataset, instead of querying the database for experiments with similar θ_{expt}, we query the database for all experiments that were performed at the same ambient temperature as that of the current experiment 'X'.

In this approach of ranking, variations in θ_{expt} is the primary source of error between the algorithm and ground truth. Experiments that were performed at the same ambient temperature as 'X' but had variations larger than 1°C in θ_{expt} would not be considered by our algorithm during ranking. However, this same experiment would feature in the ground-truth dataset since the experiment was recorded as having run at the same ambient temperature as that of 'X'. Similarly, experiments that were performed at ambient temperatures close to that of 'X' may still have their θ_{expt} falling within the ± 1°C threshold. These experiments would be considered by the algorithm during ranking, whereas the ground-truth dataset would filter out these experiments due to the ambient temperature being different. In cases where one device contributed more data points than other devices due to some experiments being repeated, we oversampled the remaining devices until each device contributed an equal number of samples for ranking.

Using this approach, we were able to achieve a predictor with a maximum Root Mean Square (RMS) error of 12 percentile points for the Nexus 5 and 17.2 points for the Nexus 6. Figure 5 and Table 2 summarize these results. Although we have several large outliers, the medians are low and the distribution is reasonably tight.

Despite our best attempts at trying to carefully control the thermal state of the device during the warmup and cooldown phases, we still had a few outliers where the θ_{expt} showed variations greater than ± 1°C. We believe that this happened due to a couple of reasons. First, the heating/cooling element of the THERMABOX may have been active towards the end of the cooling phase which

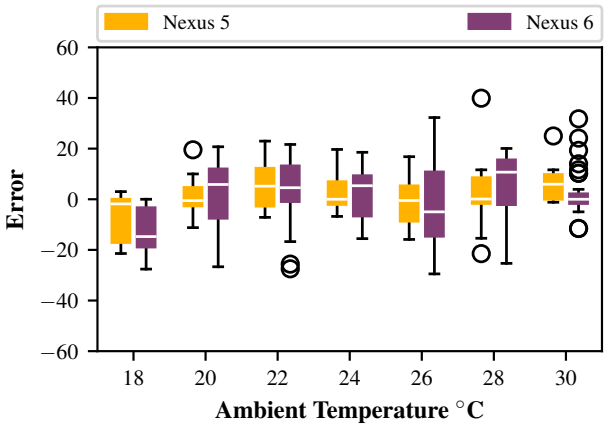

Figure 5: **Errors in Ranking Algorithm at Different Ambient Temperatures**. The median error is low across all ambient temperatures, with tight distributions in most cases. The outliers are due to significant differences in start temperature possibly caused by THERMABOX's cooling element.

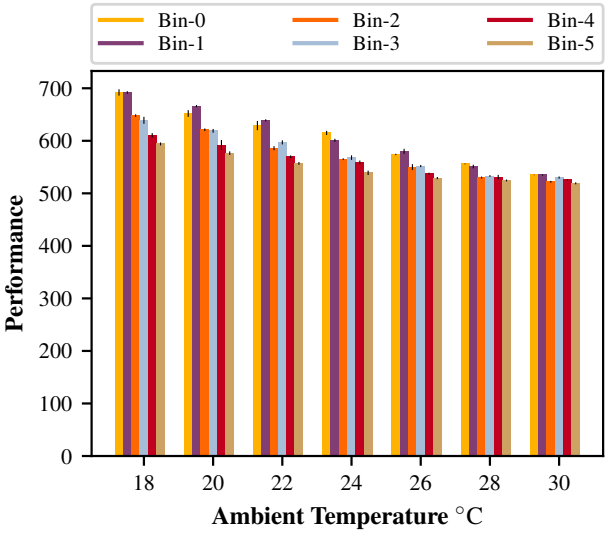

Figure 6: **Nexus 5 Performance**. There is a clear distinction between bins [0,1], [2,3], and [4,5] at lower temperatures.

may have in turn influenced the temperature of the device. Second, the device may have had an unexpected burst of activity close to the end of the cooldown phase which may have caused a small spike in CPU temperature. Finally, we picked a long cooldown period of 10 minutes hoping that the device would reach a near-zero slope in the cooling curve and attain a stable temperature. However, in some cases, this may not be the case due to excess heat in the system caused by differences in throttling behavior. On the Nexus 5, due to the nature of underlying transistors and thermal throttling rules, bins 4, 5 throttle much faster than the other bins and spend most of their warmup periods throttled. As a result, they generate significantly less heat than the other bins which in turns causes them to attain a lower θ_{expt} than the other bins.

5.2 Identifying Performance Groups

Manufacturers bin CPUs based on their transistor quality, and this bin information can be obtained on rooted devices. On some device models—such as the Nexus 5—we see a correlation between performance and CPU bin. We also observe that some bins exhibit similar performance characteristics, for example bins [0–1] and [2–3] on the Nexus 5, as described in [18]. Devices that exhibit similar performance characteristics belong to an equivalence class which we call a *performance group*.

Another approach to answering the question 'How good is my smartphone?' is to identify which performance group the phone belongs to for its device model. Knowing how many such performance groups exist, the distribution of devices, and which group a device fits into would provide consumers with a sense of how good their device is compared to other devices of the same model. In the sections that follow we evaluate our ability to correctly label performance groups for the Nexus 5 and Nexus 6, and discuss how we might determine performance groups for new devices.

5.2.1 Nexus 5 Performance Groups

In Figure 6, we observe a few trends with regards to performance, temperature, and CPU bin for the Nexus 5. First, the performance of bins 0 and 1 is very similar across all ambient temperatures, and consistently better than the other bins. Bins 2 and 3 also perform similarly across all ambient temperatures and consistently better than bins 4 and 5, though the difference is more pronounced at lower ambient temperatures. Finally, the performance of all devices regardless of bin tends to converge as ambient temperature increases. As described in Section 5.4, at higher temperatures these devices are already thermally throttled and so performance is very similar. Based on these observations, we define three performance groups, Group 0 containing bins [0,1], Group 1 containing bins [2,3] and Group 2 containing [4,5].

Our goal is to build a classifier that can accurately label a device's performance group. We use the θ_{expt} and performance (iterations) collected in our experiments as our feature vectors. Figure 8 shows each experiment plotted in this feature space, and we see a clear separation between performance groups. Because of this, we use a Support Vector Machine (SVM) classifier to classify the performance groups, as they appear to be either linearly or quadratically separable.

To evaluate the performance of the SVM classifier, we used 10-fold cross-validation to split the experiment data

RMSE	18	20	22	24	26	28	30
Nexus 5	12.0	6.6	10.0	8.0	9.8	10.4	8.0
Nexus 6	16.0	14.5	12.6	9.6	17.2	13.9	9.5

Table 2: **RMS Errors in Our Ranking Algorithm at Different Ambient Temperatures**. Note that RMS is strongly influenced by outliers.

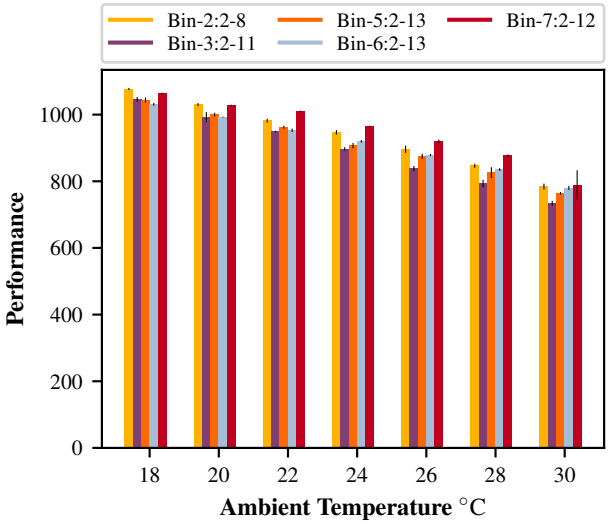

Figure 7: **Nexus 6 Performance**. The difference between bins is minimal.

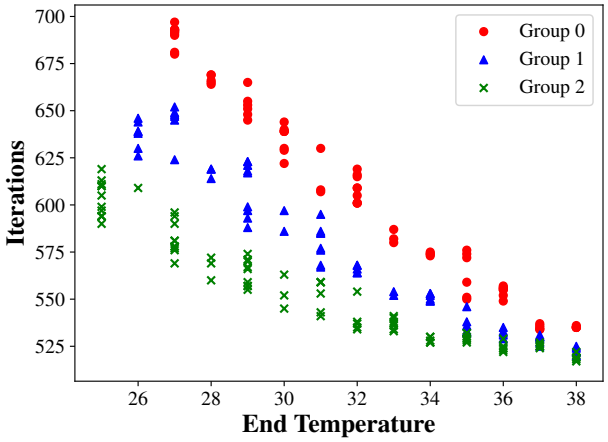

Figure 8: **Nexus 5 Performance Groups**. At lower temperatures the performance are visibly separable.

into multiple testing and training datasets. We found that the classifier performs very well at lower ambient temperatures, but accuracy decreases as ambient temperature increases, which we expected due to the performance convergence at higher temperatures. At ambient temperatures of $\leq 26°$—which is in the range of room temperature [10]—the classifier performs with 99.38% mean accuracy and 1.88 standard deviation with a polynomial (quadratic) kernel, and 99.38% mean accuracy but with a standard deviation of 8.88 with a linear kernel. When considering all ambient temperatures, the accuracy and standard deviation are 88.79% and 13.34 with the quadratic kernel, and 80.53% and 21.57 with the linear kernel.

5.2.2 Nexus 6 Performance Groups

For the Nexus 6, the inter-bin performance difference is not nearly as pronounced as for the Nexus 5, as shown in Figure 7. We see that bins 2 and 7 tend to perform slightly better and so we create 2 performance groups consisting of bins [2,7] and [3,4,5,6]. Figure 9 shows that there is some separation between these groups, but they perform much closer even at lower temperatures than for the Nexus 5. We do find, however, that the SVM classifier is able to distinguish between these performance groups with reasonable accuracy, especially at lower ambient temperatures close to room temperature. Running the SVM classifier on the Nexus 6 dataset, we achieve a mean accuracy of 96.00% using the linear SVM classifier for ambient temperatures $\leq 26°$, and 91.25% for all temperatures, with standard deviation of 6.80 and 8.92, respectively. With a quadratic kernel, the SVM classifier performed slightly worse with mean accuracy of 92.00% for ambient temperatures $\leq 26°$, and 87.61% for all temperatures, with standard deviation of 10.24 and 10.64, respectively.

5.2.3 Performance Groups for Additional Models

It is a more difficult problem to identify performance groups for arbitrary device models with data collected in the wild. Given enough samples with ground truth bin data, taking a similar approach as with the Nexus 5 and 6 would likely work well. For popular devices, this is quite feasible as bin information can be obtained on rooted devices. But in the absence of labeled data, creating a model that properly clusters the data is a difficult task. We attempted to use the popular k-means clustering for our Nexus 5 data, but as expected it did not identify the performance group. The shape of the performance groups does not lend itself well to this technique, as there is no obvious center of mass in the performance groups. We leave the exploration of new clustering techniques to differentiate performance groups for datasets collected in

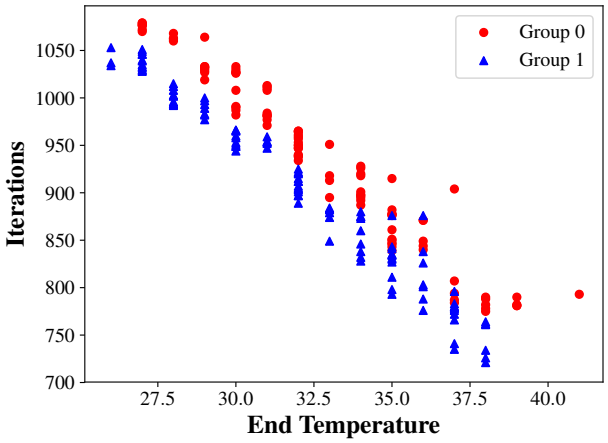

Figure 9: **Nexus 6 Performance Groups**. Inter-bin performance is a lot closer on the Nexus 6 than the Nexus 5, but still 2 performance groups emerge.

the wild as an interesting future direction.

5.3 θ_{expt} Analysis

One other aspect that remains to be proved is how effective θ_{expt} is at modeling ambient temperature. In our experiments, we observed that there is a clear correlation between θ_{expt} and ambient temperature despite influence from the differences arising from process variations.

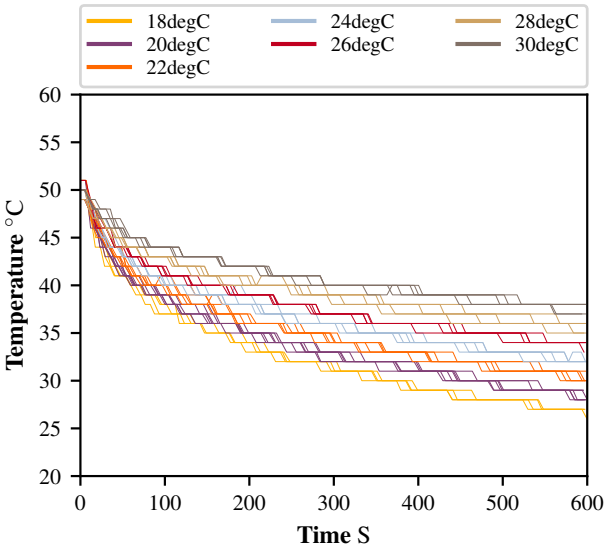

Figure 10: **Nexus 6 Cooling Curves at Different Ambient Temperatures**. The curves are consistent and reproducible at each ambient temperature.

The cooling curve for a Nexus 6 device at various ambient temperatures is shown in Figure 10. Notice that despite differences in performance results and individual thermal characteristics, different devices of the same

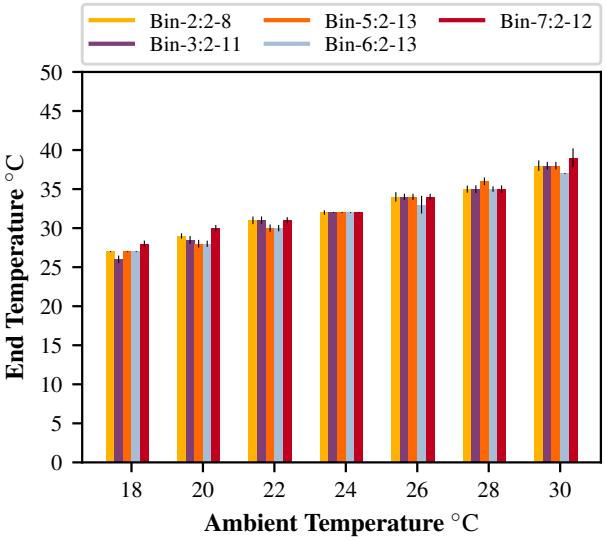

Figure 11: **Nexus 6 End Temperatures**. There is a strong correlation between θ_{expt} and ambient temperatures.

model end up with almost identical end temperatures given sufficient time for cooling down.

Figure 10 shows the correlation between θ_{expt} and ambient temperature. The correlation coefficient between the two variables was computed to be 0.86 while the spearman's correlation was found to be 0.98. Similar results were obtained for the Nexus 5 as well.

5.4 Importance of Thermal Management

Finally, to round out the thermal aspects of our study, we describe existing thermal management approaches and explore its importance and effects.

Linux's thermal subsystem includes throttling mechanisms to ensure that the device temperature is always within safe limits. Lowering CPU frequency and disabling cores are the mechanisms that the kernel uses to throttle the CPU and control rising temperature. Naturally, these actions affect performance.

However, Android devices using Qualcomm based chips contain a proprietary userspace binary called THERMAL-ENGINE that runs with elevated privileges and is responsible for thermal management. While certain parts of the THERMAL-ENGINE application can be configured via a configuration file, many of the rules are embedded and hard-coded. For example, although there are several rules in the configuration file that specify how the CPU should be throttled on the Nexus 6 when the temperature sensor reports values of 90°C or higher, in our experiments, throttling began as early as 80°C.

One interesting feature of THERMAL-ENGINE on the Nexus 5 is the use of the additional temperature sensor `xo_therm_pu2`. The labeling of the rule in the configu-

ration file as SKIN_THERMAL_management_1 and its low thermal threshold of 40°C suggest that, this sensor reports the device's case temperature. This sensor is not present on the Nexus 6. This low threshold of 40°C is the reason why performance between various Nexus 5 bins was indistinguishable at higher ambient temperatures. At ambient temperatures of 28°C and above, it takes only a few seconds for devices to hit the 40°C threshold and thus all devices spend almost the entire $T_{workload}$ duration throttled.

To stress the importance of good thermal management, we attempted to perform our own experiments with and without THERMAL-ENGINE. We were able to run these experiments on the Nexus 5 and the results are presented below. The Nexus 6 is not part of this study as all Nexus 6 devices we tested shut down at some point during the CPU intensive warmup when THERMAL-ENGINE was disabled.

At the beginning of the experiments, Android's log (logcat) buffer was cleared and at the end of the experiment, the logs were dumped. These logs give us a peek into how the CPU is throttled as the CPU temperature increases. We also combined these logs with the energy data gathered from the Monsoon power monitor. Both experiments were performed at an ambient temperature of 22°C.

Figures 12 and 13 show us the impact of THERMAL-ENGINE on a bin-3 Nexus 5 device. Without thermal management, the kernel toggled cores on and off while maintaining the temperature at or below 80°C. When online, cores always ran at the maximum frequency. Turning cores off impacts performance significantly while a high temperature of 80°C has a severe impact on efficiency. As a result, this setup only completed 291 iterations with an efficiency of 2.6J/iteration.

With thermal management enabled, we see that after an initial temperature spike up to about 100°C, the rest of the workload was executed well below 80°C with all cores running at about half their maximum frequency. This resulted in a total of 560 iterations with an efficiency of 1.8J/iteration. This represents a 93% improvement in terms of performance and 30% in terms of efficiency.

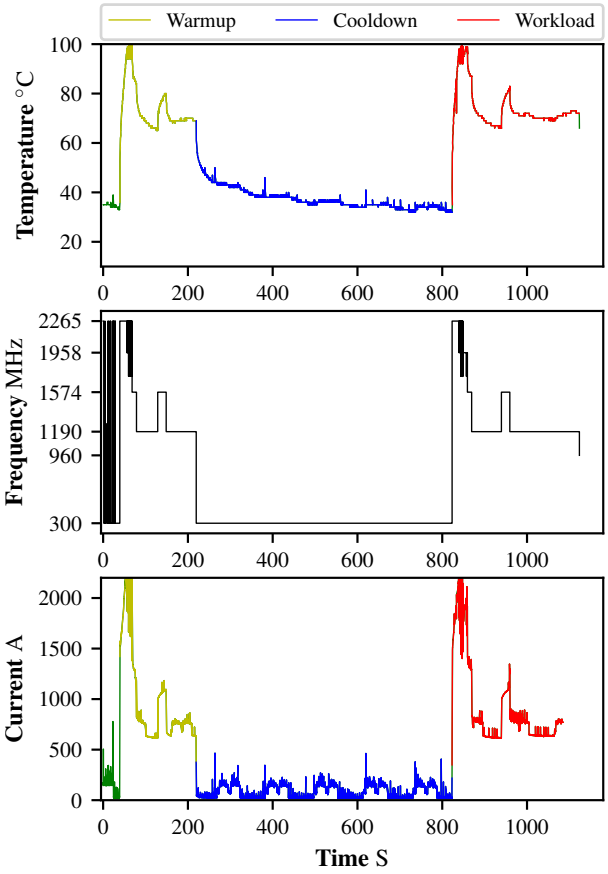

Figure 12: **Workload Analysis With THERMAL-ENGINE Enabled**. The device throttles down quickly to pre-configured frequencies. Frequencies during cooldown were very noisy and have thus been omitted.

6 Related Work

Characterizing smartphone behavior in terms of temperature, energy, and performance has been attempted by numerous academic and industry researchers. However, to the best of our knowledge, we are the first to attempt to use these characteristics to determine the quality of the underlying silicon.

Sekar describes the challenges faced in thermal aware power management of mobile devices [17]. The paper discusses how power management policies are typically temperature agnostic, despite temperature having a significant impact on leakage and dynamic power. Other works such as [19] and [14] characterize the impact of power dissipation on different aspects of user experience such as skin temperature and performance. They don't, however, consider the effects of ambient temperature. Multiple efforts have also modeled thermal behavior of smartphones. Lee et al. developed three dimensional finite element thermal models using the size and power dissipation data of commercial hand-held devices [12]. Therminator is a full device thermal analyzer for smartphones that is capable of generating accurate temperature maps for chip containing multiple hardware components and the skin of the device [22]. There have also been several efforts to model and measure energy consumption of CPUs on mobile devices. For example, Nachiappan et al. present a multi-component energy management of mobile devices for frame-based applications [15]. They use simulation models that use multiple activity counters to estimate energy consumption of cores. Lee et al. present an energy management approach for mobile

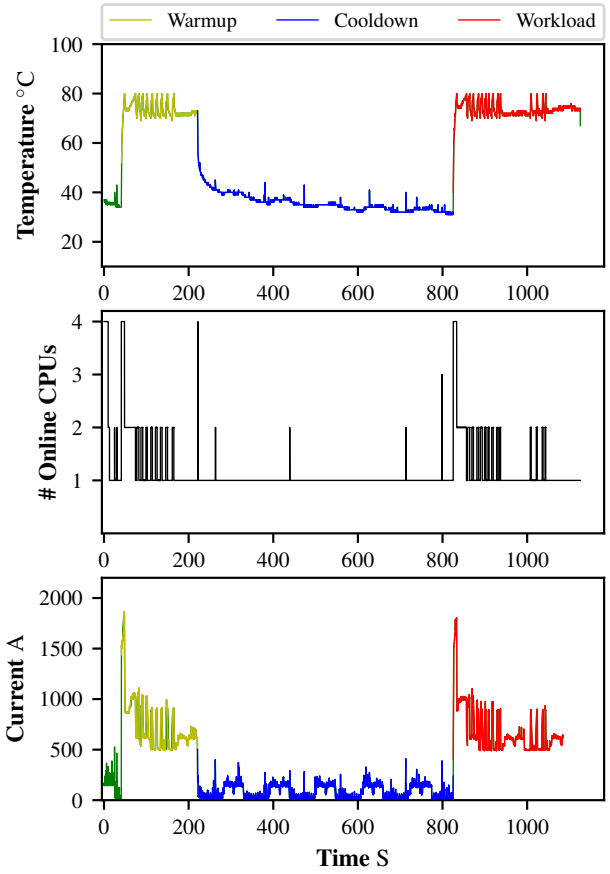

Figure 13: **Workload Analysis With THERMAL-ENGINE Disabled**. Most of the workload is executed with only 1 or 2 cores online. Frequency of online cores is fixed at 2265 MHz.

interactive web workloads to maintain clouded guided QoS [13]. They measure energy consumption of Cortex-A7 and Cortex-A15 cores using onboard energy sensors on ODROID-XU3 board. We record the power consumption of mobile devices under test by measuring the current drawn by these devices using Monsoon, an external power monitor.

7 Future Work

We have published our app on the Google PlayStore [6]. Our aim is to gather sufficient data from devices of various smartphone models and help users determine the ranking of their smartphone, in comparison to other smartphones. The app also records CPU bin information from devices that are both rooted and have the bin information readily accessible via sysfs. Thus, with the information collected from this app, not only can the devices be ranked on the absolute scale with respect to one another, but the gathered information can also be used to understand how the manufacturers are binning their CPUs and the distribution of various bins. This information is currently not openly available to researchers or consumers.

Newer Qualcomm chips are also being binned differently. Instead of having a static number or label associated with the device—as observed on the Nexus 5 and Nexus 6—which in turn determines its dynamic voltage-frequency (DVFS) mappings, the devices appear to be configuring their voltage-frequency tables dynamically at runtime. This is referred to as Adaptive Voltage Scaling (AVS). The voltage-frequency table is computed by using on-chip sensors that detect which frequency-voltage pairs operate the processor without errors. Note that this approach only improves the accuracy of the DVFS tables. There is still an intrinsic difference between processors due to process variations which can be measured and quantified with our ACCURANK technique. In such cases, where there is no clear bin label, we aim to create our own labels which are unique to each table configuration and try to use this as a hint for the number of clusters in some unstructured learning algorithms. The goal of this approach would be cluster devices based on their 'bins' and rank them based on their performance.

All of the app data that we collect is stored in a MongoDB instance and we plan to provide open access to this database to consumers and researchers.

8 Conclusion

In this work, we described a problem that currently plagues smartphone consumers and researchers, that differences in smartphone CPUs affect both energy and performance. We introduced our technique of identifying underlying transistor quality which allows users to determine the rank of their smartphone relative to other smartphones of the same device and model. The app accounts for thermal characteristics of both environment and device while ranking thereby allowing untrained users to conduct meaningful experiments in uncontrolled settings. We evaluated our technique on the Nexus 5 and Nexus 6 smartphone models. We were able to rank Nexus 5 devices accurately with an RMS error less than 10.5. On the Nexus 6, which did not exhibit much performance difference, our technique was able to rank smartphones with a maximum RMS error of 17.2. In terms of classifying devices based on performance, we were able to accurately identify performance groups at room temperature with 99% and 96% accuracy on the Nexus 5 and Nexus 6 respectively.

We have published our app implementing our ACCURANK technique on the Google PlayStore. We hope to collect data from many users and devices and thus help users determine the quality of their smartphone CPU.

References

[1] 3dmark - the gamer's benchmark. https://play.google.com/store/apps/details?id=com.futuremark.dmandroid.application&hl=en.

[2] Antutu benchmark. https://play.google.com/store/apps/details?id=com.antutu.ABenchMark&hl=en.

[3] Geekbench 4. https://play.google.com/store/apps/details?id=com.primatelabs.geekbench&hl=en.

[4] Monsoon power monitor. https://goo.gl/rlizsj.

[5] Ranking - antutu benchmark. http://www.antutu.com/en/ranking/rank1.htm.

[6] Smartphones exposed. Link removed for double-blind review.

[7] BORKAR, S., KARNIK, T., NARENDRA, S., TSCHANZ, J., KESHAVARZI, A., AND DE, V. Parameter variations and impact on circuits and microarchitecture. In Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451) (June 2003), pp. 338–342.

[8] BOWMAN, K. A., DUVALL, S. G., AND MEINDL, J. D. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. IEEE Journal of Solid-State Circuits 37, 2 (Feb 2002), 183–190.

[9] GUO, Y., XU, Y., AND CHEN, X. Freeze it if you can: Challenges and future directions in benchmarking smartphone performance. In Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications (New York, NY, USA, 2017), HotMobile '17, ACM, pp. 25–30.

[10] JOHN SNOW, I. I. C. W. T. W. H. O. Guidelines for the Storage of Essential Medicines and Other Health Commodities. John Snow, Inc./DELIVER, for the U.S. Agency for International Development, 2003.

[11] KIM, N. S., AUSTIN, T., BAAUW, D., MUDGE, T., FLAUTNER, K., HU, J. S., IRWIN, M. J., KANDEMIR, M., AND NARAYANAN, V. Leakage current: Moore's law meets static power. computer 36, 12 (2003), 68–75.

[12] LEE, J., GERLACH, D. W., AND JOSHI, Y. K. Parametric thermal modeling of heat transfer in handheld electronic devices. In Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008. 11th Intersociety Conference on (2008), IEEE, pp. 604–609.

[13] LEE, W., SUNWOO, D., GERSTLAUER, A., AND JOHN, L. K. Cloud-guided qos and energy management for mobile interactive web applications. In Mobile Software Engineering and Systems (MOBILESoft). (2017).

[14] MERCATI, P., ROSING, T. S., HANUMAIAH, V., KULKARNI, J., AND BLOCH, S. User-centric joint power and thermal management for smartphones. In Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on (2014), IEEE, pp. 98–105.

[15] NACHIAPPAN, N. C., YEDLAPALLI, P., SOUNDARARAJAN, N., SIVASUBRAMANIAM, A., KANDEMIR, M. T., IYER, R., AND DAS, C. R. Domain knowledge based energy management in handhelds. In High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on (2015), IEEE, pp. 150–160.

[16] PELTONEN, E., LAGERSPETZ, E., NURMI, P., AND TARKOMA, S. Energy modeling of system settings: A crowdsourced approach. In Pervasive Computing and Communications (PerCom), 2015 IEEE International Conference on (2015), IEEE, pp. 37–45.

[17] SEKAR, K. Power and thermal challenges in mobile devices. In Proceedings of the 19th annual international conference on Mobile computing & networking (2013), ACM, pp. 363–368.

[18] SRINIVASA, G. P., BEGUM, R., HASELEY, S., HEMPSTEAD, M., AND CHALLEN, G. Separated by birth: Hidden differences between seemingly-identical smartphone cpus. In Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications (2017), ACM, pp. 103–108.

[19] STRAUME, A., OFTEDAL, G., AND JOHNSSON, A. Skin temperature increase caused by a mobile phone: a methodological infrared camera study. Bioelectromagnetics 26, 6 (2005), 510–519.

[20] TSIVIDIS, Y., AND MCANDREW, C. Operation and Modeling of the MOS Transistor. Oxford Univ. Press, 2011.

[21] UNSAL, O. S., TSCHANZ, J. W., BOWMAN, K., DE, V., VERA, X., GONZALEZ, A., AND ERGIN, O. Impact of parameter variations on circuits and microarchitecture. IEEE Micro 26, 6 (Nov 2006), 30–39.

[22] XIE, Q., DOUSTI, M. J., AND PEDRAM, M. Therminator: a thermal simulator for smartphones producing accurate chip and skin temperature maps. In Low Power Electronics and Design (ISLPED), 2014 IEEE/ACM International Symposium on (2014), IEEE, pp. 117–122.

[23] ZHANG, L., BAI, L. S., DICK, R. P., SHANG, L., AND JOSEPH, R. Process variation characterization of chip-level multiprocessors. In 2009 46th ACM/IEEE Design Automation Conference (July 2009), pp. 694–697.

[24] ZOLOTOV, V., VISWESWARIAH, C., AND XIONG, J. Voltage binning under process variation. In Proceedings of the 2009 International Conference on Computer-Aided Design (2009), ACM, pp. 425–432.