Characterization of User's Behavior Variations for Design of Replayable Mobile Workloads

Shruti Patil, Yeseong Kim(✉), Kunal Korgaonkar, Ibrahim Awwal, and Tajana S. Rosing

University of California San Diego, San Diego, USA
{patil,yek048,kkorgaon,iawwal,tajana}@ucsd.edu

Abstract. Mobile systems leverage heterogeneous cores to deliver a desired user experience. However, how these cores cooperate in executing interactive mobile applications in the hands of a real user is unclear, preventing more realistic studies on mobile platforms. In this paper, we study how 33 users run applications on modern smartphones over a period of a month. We analyze the usage of CPUs, GPUs and associated memory operations in real user interactions, and develop microbenchmarks on an automated methodology which describes realistic and replayable test runs that statistically mimic user variations. Based on the generated test runs, we further empirically characterize memory bandwidth and power consumption of CPUs and GPUs to show the impact of user variations in the system, and identify user variation-aware optimization opportunities in actual mobile application uses.

Keywords: Mobile device · User variation · Heterogeneous cores · GPU usage

1 Introduction

With growing expectations from mobile platforms, mobile SoCs utilize heterogeneous cores to deliver the desired performance within small power budgets. The compute cores in mobile SoCs comprise of CPUs, GPUs and custom hardware blocks (IP) such as DSPs, Multimedia Accelerators and Audio/Video decoders. Mobile CPU and GPU cores together are the second highest power hungry components after display components [3,17]. Architectural research efforts seek to optimize these cores, thus a way to characterize realistic workloads is needed so as to clarify the possible impacts of optimizations on actual device systems.

However, benchmarking them has been exceedingly challenging since mobile workloads are inherently interactive in nature. Real-world inputs must be user-initiated, therefore are subject to large variations in user behavior. User variations can arise from a number of factors, including differences in content type, speed and frequency of interactions. The subtle differences in these factors can affect the workload to a large extent. For example, in our initial study, Facebook use cases exhibit a 16–93% utilization range in GPU acceleration for different

S. Sigg et al. (Eds.): MobiCASE 2015, LNICST 162, pp. 51–70, 2015.
DOI: 10.1007/978-3-319-29003-4_4

factor combinations. Thus, to evaluate the effectiveness of an optimization, we should consider the variations in real user behavior. A popular approach is to prototype new ideas and deploy them in a user study. However this approach has several shortcomings. First, a before-and-after comparison is not possible due to replayability issues. Thus, ideas that cannot be easily prototyped (e.g. hardware optimizations) could not be rigorously evaluated. Moreover, it is not always possible to obtain representative samples of users from a user study.

Developing parameterized mobile benchmarks that can encompass a range of user variations is a viable solution to addressing such challenges. Mobile benchmark suites with replayable, interactive applications have been proposed recently [10,11,15]. These suites consist of popular mobile applications, and allow a small amount of parameterization to change user behavior. They still lack any model of realistic user behavior or user variation. Tools [9] have been also developed to capture and replay user touch interactions allowing repeatable and interactive runs. However, these alternative cannot reproduce same workload due to other important factors, e.g., time-dependent content changes of webpages.

In this paper, we propose an analytic way to characterize realistic workloads in the wild and generate representative workloads of real mobile applications for replayable evaluation. We first study utilization of compute cores as a way to study the amount of user variations that is experienced by the system during a typical user session. To account for actual user variations, we analyze CPU and GPU usage statistics from 33 real users for seven most frequently used mobile applications over all applications executed in a month-long study. We first study the usage of CPUs, GPUs, and their interactions with main memory to understand how much and when power-intensive compute processors are used in interactive mobile applications.

We also focus on GPU usage to indicate the intensity of user interaction in mobile workloads. The use of GPU acceleration in gaming applications has been well-characterized [13,14,16], but GPU usage in general applications on modern mobile platforms is not yet well understood. We show that the GPU is widely used for hardware acceleration for rendering text and images, and for enabling smooth and responsive user interaction (UI). Thus, the amount of GPU acceleration during user sessions can be a good indicator of variations related to both content and interactions. In this study, we find that non-gaming applications such as browser and facebook utilize GPUs average 51 % of the time with a standard deviation of 27.5 during an interactive session. This non-trivial range in GPU acceleration occurs due to user-behavior driven variations.

Using a clustering analysis on our data, we find that the variations can be reduced to relatively few characteristic groups. We use our analysis to develop a framework to generate representative test turns that are parameterized for the intensity of user interactions desired. We complement our automated benchmark generation efforts with more realistic, replayable test runs, that reproduce user interactions to actual mobile devices. These runs statistically cluster with the user runs, maintaining clustering 'goodness' within 0.1–3.3 % of the user clusters.

In order to present the practical value of the replayable test runs, we then show detailed user-accounted analysis on mobile platforms. Through power and

memory bandwidth evaluations, we find that user variations have a significant impact on these metrics. This shows that the use of the proposed replayable test runs allows realistic evaluations for debugging and testing of mobile systems, and also gives the guideline for user variation-oriented optimizations, for example, those that judiciously use mobile GPU acceleration and clock frequency adjustment depending on user behavior.

The overall contributions of this paper are as follows:

- We show that the differences in user behaviors result in significant variations in the utilization of the compute cores. This motivates user-based optimizations for interactive mobile workloads. This also shows the need to develop benchmarks that can incorporate and model user variations.
- We formulate an automated methodology to generate representative microbenchmarks, allowing us to study the specific effects of user variations. We also create realistic and replayable test runs that allow us to study the system under realistic usage patterns with user variations.

This paper is organized as follows: Sect. 2 discusses related work. Section 3 describes our experimental setup. Section 4 illustrates analysis from user study and Sect. 5 describes how test runs are created. Section 6 discusses the results from our characterization study. Finally, Sect. 7 concludes with the key take-away from our analysis.

2 Related Work

Numerous user studies have been previously undertaken on mobile systems. Long-term studies [5,7,19,20,22,25] highlight the diversity of mobile users in terms of applications, frequency of use, duration, etc. These studies have uncovered traffic patterns, relationships between context and usage, power usage of mobile SoC units, which have aided optimization strategies. Our user study seeks to understand CPU and GPU utilization of mobile applications in short-term user interactive sessions. Previous studies lack information about simultaneous GPU acceleration used, which is the key to modeling our interaction and the focus of our study.

Unlike the workload of general multicore architectures [2], prior researches have shown that mobile systems are likely to be affected by how the user interacts with their devices [11]. Based on the interactive nature of the mobile device usage, architectural studies for mobile-specific workloads have also been recently proposed. Interactive applications covering genres such as browser, game, photo, video and chat have been designed with automatic scrolling or touch events [10,11,15]. Although these are interactive benchmarks, the interactions are not formally related to user behavior and there are no models for user variations. In this paper, we first study how these benchmarks can be adapted to the range of user variations that we have observed. For more accurate representation of user behavior, we create interactions with realistic speeds, which vary based on content, as expected from real users.

Table 1. Mobile Platform Specifications

	MSM8660 (45 nm)	MSM8960 (28 nmLP)
CPU	Up To 1.7 GHz, Dual-core Scorpion	Up To 1.7 GHz Dual-core Krait
3D GPU	Adreno 220	Adreno 225
Memory	Single-channel 500 MHz ISM, 333 MHz LPDDR2	Dual-channel 500 MHz LPDDR2
Android OS	Ice Cream Sandwich	Jelly Bean
Example Devices	HTC Evo 3D, LG Optimus, SS Galaxy Note	SS Galaxy S3, Nokia Lumia 1020, Sony Xperia V

Prior mobile core characterization studies have typically focused on single cores. Publications presented in [10,11,15,24] utilize proposed workloads for detailed CPU characterization with performance counters. One recent effort [8] studies thread-level parallelism in a mobile multi-core environment. Their findings suggest that while two CPU cores are sufficient to parallelize popular use cases in mobile benchmarks, GPU acceleration is limited. However, a single workload is executed for each mobile application to draw conclusions about its nature. In our study, we find that user variation significantly varies the utilization of multiple cores, in particular the GPU hardware acceleration significantly influenced by different factors such as content types and user interactions.

Mobile GPU studies have focused on gaming applications since they are known to be both GPU-intensive and power-intensive. A few publications [13,14,16] show the bottlenecks in GPU pipelines, while [1,4,18] propose power management schemes that trade-off on user experience. Recent efforts have studied general-purpose benchmarks such as image rendering and computer vision algorithms using GPGPUs in mobile systems [12,23]. In contrast, this paper characterizes CPU, GPU and memory interactions focusing on non-gaming applications. We show the large amount of GPU acceleration used, and its impact on power consumption. While power characterization could be performed in a field study, creating replayable test runs tied to user behavior allows comparison of metrics measured separately, as well as for future architectural research.

3 Experimental Setup

Development phones from two state-of-the-art mobile platforms, Qualcomm's MSM8660 and MSM8960 running Android OS (Table 1) were used in our study. These allowed fine-grained power and memory bandwidth measurements within the system, normally unavailable on commercial phones. Both phones support camera, bluetooth and sensors, and have dual-core CPUs, two 2D GPUs for

vector graphics and one 3D GPU for 3D graphics. 33 on-campus students[1] used the phones for a month, with full access to Amazon Android App Store. During the user study, the users were asked to use the device for one month without any detailed usage direction, so they could use the device and applications as usual. Sim cards were not used, instead wi-fi network access was available throughout campus. Since the focus was to observe the range of user variations experienced by the cores for mobile applications, lack of cellular usage does not significantly affect the results of compute core usages for the studied applications.

The rooted Android phones were instrumented to record utilization and frequency every 100 ms during short-term interactive sessions, i.e. when a user engages with the phone for a short duration of either seconds or minutes to complete activities using a mobile application. Unlike long-term user studies [20,22], e.g. targeted to daily application usage patterns, the short-term analysis allows to understand fine-grained system activities such as detailed CPU and GPU usages. The utilization and frequency of the CPUs and GPUs were profiled during these sessions, using information obtained from the `/sysfs` and `/proc` file system support. Power saving features were active on the phones, including DVFS (Dynamic Voltage and Frequency Scaling) and *mpdecision* daemon, which power-gates a CPU core when possible. Further detailed characterization was performed on the Qualcomm 8960 phone. Power measurements were obtained using Qualcomm's `Trepn` tool [21] which collects power consumption of system components such as per-core power using hardware sensors. Finally, memory bus monitoring was carried out using an internal bus monitor tool.

Users ran a total of 125 applications (or 'apps') on their devices during the one month study. Of these, we selected seven highest used applications for detailed analysis so that collected information of the selected apps sufficiently represent device usage of the app for each user. These include the interactive applications: Browser, Facebook and Email, streaming applications: Skype, Camera and Music and Templerun2 game. Browser, Email, Music and Camera represent widely used mobile applications. Similarly, Facebook, Skype and Templerun2 feature in the top 50 Android apps on Google Play.

4 Analysis of User Workloads

To explore the CPU and GPU usage of the applications, we analyze the temporal utilization data from the user study. Different sessions varied in duration, therefore we analyze the relative amount of time the CPUs and GPUs used within each session, and classify the quantified workload of compute cores of each session into different clusters. We first analyze usage into 16 combinations of the two cores (c0,c1), one 2D GPU (g2) and the 3D GPU (g3) for the target platforms described in Table 1 (Second 2D GPU showed negligible utilization and is neglected in the further study). This reduced each session to a 16-valued

[1] The study participants include undergraduate and graduate students. Even though we collected the data from the on-campus students, we could find a wide range of variations in mobile usages.

vector, where each value denotes the proportion of session time when a core combination had non-zero utilization. For example, combination (c0,g3) is the proportion of session time with active CPU0 and GPU3D and inactive CPU1 and GPU2D. Using these vectors, characteristic clusters for the sessions can be derived with k-means clustering analysis [6]. Each cluster is a closely-knit group of data points, with high separation from other clusters. This translates into a standard 'goodness' metric for clusters:

$$\text{Goodness} = \frac{\text{Variation between clusters}}{\text{Total Variation in Data}}$$

where the variation is computed as sum-of-square distances between groups of clusters and within each cluster. We select k, which specifies the number of clusters to identify, such that the clustering goodness is at least 80 % in all of our analyzes. Thus, we could capture at least 80 % of the variation by choosing this criterion. The analysis is performed at intra-app and inter-app levels to study the per application user variations, and overall degree of user variations.

4.1 Intra-App

Figure 1 shows examples of two user runs in each cluster generated for Browser. This presents the eleven representative clusters of core combinations in order of the right stacked labels while the other rare combinations are grouped as 'others'. 105 sessions were grouped into 5 clusters with 81 % clustering ratio. Since CPU0 is the master core, it is utilized throughout the active portions of the runs. Browser runs used one or more GPU cores (as 'GPU Used' label denotes) an average 46 % of the session time with a standard deviation of 21. This shows that mobile browser activities are highly using GPU acceleration, which varies dramatically with the variation in user behavior. Figure 2 shows the average proportion of time which the core combinations are utilized in clusters for the other six most used applications. There are up to 36 sessions within a cluster. Facebook, Templerun, Skype and Email use GPUs average 50 % time in sessions. The variance in their use of 3D GPUs show the impact of user behavior on GPU acceleration exercised. Music and Camera, due to their use of DSP and Audio/Video accelerators instead of GPU, appear to be CPU dominant.

Typical core usage of applications illustrates their expected energy draw. Music tends to use a single CPU core (and an accelerator) while Skype or Templerun require 2–4 cores for majority of the time in their respective sessions. Thus, a 30-min Skype or Templerun session draws more battery power than a 30-min Music session. We study the power usage in Sect. 6. The user variation as experienced by the cores is significant. The number of sessions per application ranged from 24 to 105, with the exception of Camera, which had 9. The user sessions are distilled into relatively few clusters representing the dominant variations. We leverage these representative groups to study the applications closely through detailed characterization and to develop our automated generation of replayable test runs.

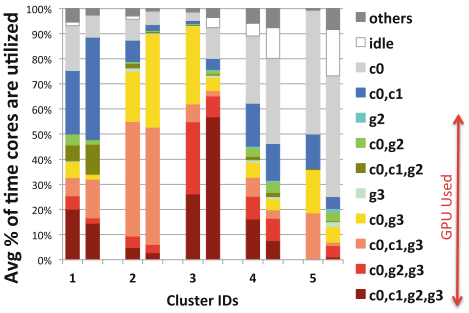


Fig. 1. Browser clusters with two different runs for each cluster

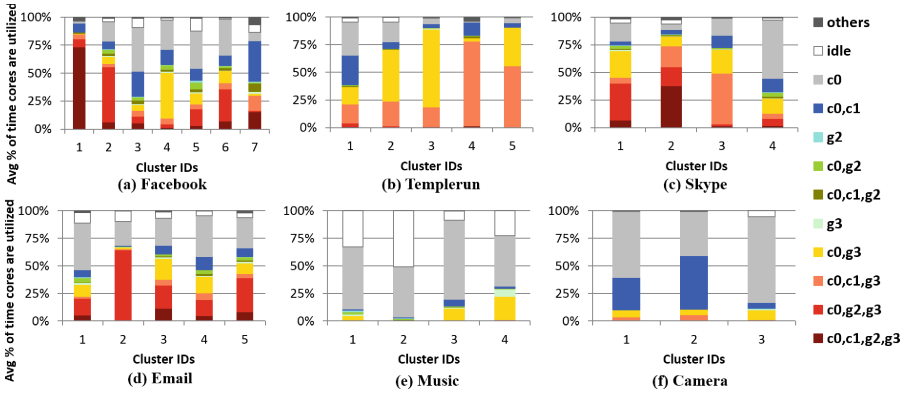


Fig. 2. Clusters for user sessions for 6 most used apps out of 125 total

4.2 Inter-App

We next perform clustering analysis on all user sessions to extract patterns in the resource usage independent of the applications. Goodness ratio of 80 % divides data into seven clusters using k-means analysis. Figure 3 shows the split of the sessions of the seven applications in seven clusters. Sessions from different applications show many similarities with each other. For example, Cluster 1 captures heavily GPU-accelerated sessions (80 % or more GPU usage) from Browser, Templerun and Skype. Cluster 4 has the sessions that used GPU3D for an average 76 % of the time, while Cluster 5 grouped sessions that used GPU2D for an average 52 % of the time. The rest of the clusters represent subtle differences in their proportions of GPU acceleration.

We make two key observations from these results: First, the clusters with gaming sessions also contain sessions from non-gaming applications. This shows that GPU acceleration in non-gaming applications is not negligible. Second, Templerun, Facebook and Browser sessions split into 5–6 of the total 7 clusters showing the prevalence of user variations. Thus, in order to thoroughly evaluate

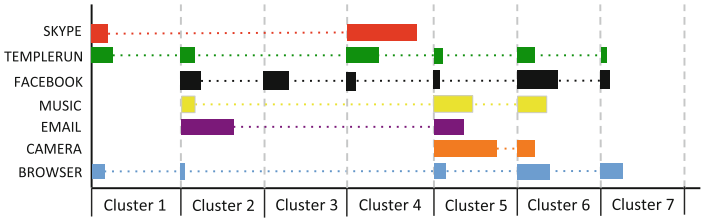


Fig. 3. Inter-app clustering analysis for 7 applications

actual workloads interacted with users, this shows that we must take into explicit account the variation of user behaviors.

4.3 Utilization Values

Next, we analyze CPU and GPU3D utilization to study how loaded the cores are during short-term interactive sessions. Since DVFS was active on the phones during the experiment, we first normalize the utilization to the maximum frequency in order to account for the frequency changes. To study CPU utilization, we add the utilization values from the two cores per sampling period and normalize the total utilization to 100 %. We then analyze histograms of utilization values observed during all user runs.

Figure 4 shows the histogram of CPU utilization values for all sessions. There are three peaks around 0 %, 50 % and 100 % utilization. The CPU distributions for Browser and Facebook are similar to this distribution. Email is also similar in that it shows the three peaks, but has higher idle values than the other peaks. The result presents the interactive nature of mobile workloads. For example, the highest peak for idle periods represent that there were no user interactions to be processed. Thus, this implies that workloads of compute cores are significantly affected by how long users interact with applications, reaffirming the observation that user behaviors highly influence to the system usage.

The GPU distribution of all applications except Templerun are similar to each other, shown by the example of Skype in Fig. 4c. Skype shows a small spike at 100 % utilization, not seen in the other applications. This presents that Skype-like video chat applications use relatively high GPU acceleration compared to other interactive apps, and therefore chat applications similar to Skype should be included in GPU characterization studies along with gaming and graphics applications. Templerun shows a more interesting GPU3D distribution (Fig. 4d). Although it peaks at 0 %, the game exhibits non-significant utilization over the full range of GPU3D utilization.

When all applications are considered together, we observe that GPU3D utilization seldom exceeded 60 % as shown in Fig. 4b. Further, the average GPU utilization across the seven applications was 16 %. The results present that, although our analysis of active cores shows an average 50 % use of GPUs during interactive sessions, the actual load experienced by the GPU3D is low. Thus, in

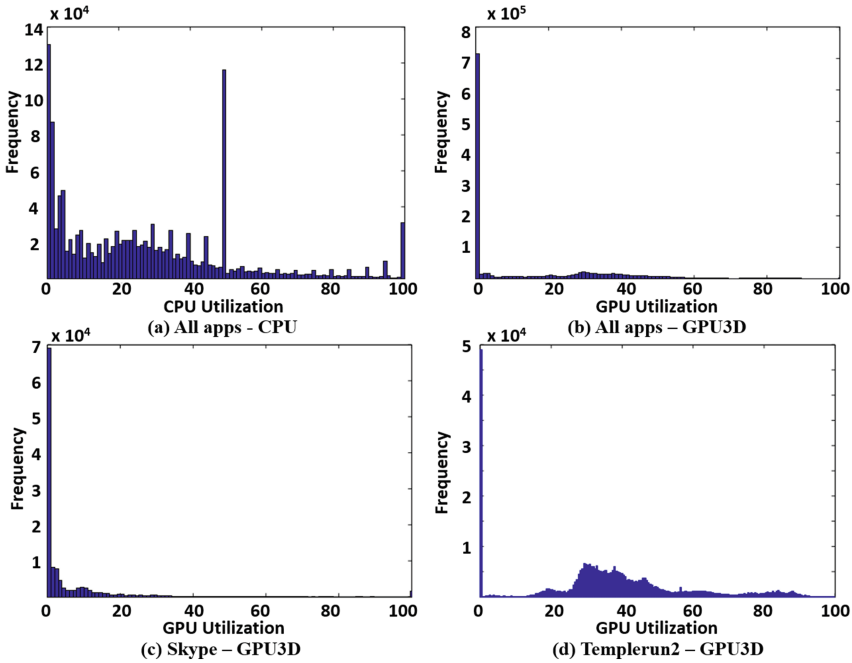


Fig. 4. Histograms of core utilizations for all user sessions

order to save GPU power, more intelligent power gating techniques for GPUs will be effective, such as turning off individual streaming cores and allowing shorter time scale in power gating.

5 Automated Test Run Generation

To reliably analyze the impact of user behavior on heterogeneous processing, we require replayable test runs that are representative of the observed diversity. Previous efforts in mobile benchmarking have proposed workloads with popular applications [10,11,15], but their inputs have not been associated with user behavior. We first explore how these interactive benchmarks can be adapted to exhibit our observed user variations. Then, to simulate more realistic user behavior, we generate new test runs that match the principal characteristics of user clusters with more than 80 % goodness with an aggregate clustering analysis. This provides a real user basis and creates representative variations for more accurate evaluation.

5.1 Automated Generation of Test Runs

The test runs are designed to be replayable using a record- and-replay utility [9]. This allows statistical rigor with multiple runs and collecting data from various

counters in separate and low-overhead runs. We first generated test runs in the lab. Since the analyzed clusters represent usages of compute cores, the key goal is to generate a replayable test run which closely matchs the cluster of an actual user session. Through multiple attempts, we identified runs that mimic the clusters observed so that a small set of test runs can be used as representative of our larger user study. The runs are more 'realistic' in that they are recorded as a trace of real human interaction with the system.

However, these hand-operated generations are cumbersome and expensive if we need to mimic user behaviors of more interactive applications, whose clusters exhibit a wide range of variations, such as Browser and Facebook in our case. Thus, it is required to generate mobile workload with user variations in an automated way. In order to understand how such interactive app reacts to user interactions, we investigate core utilizations of Browser by using Bbench3.0 [10]. The Bbench3.0 browser benchmark offers *scroll delay*, *scroll size* and *page delay* parameters, which may be exploited to create user variations. Figure 5 shows the utilization breakdown when scroll delay and scroll size is varied. As shown in the result, changing the parameters results in the variation of core usage. For example, a 34–78 % range in GPU acceleration is observed. However, the different core combinations cannot be tuned easily, thus we could not cover all the variations that we observed in the user profiles. In reality, typical mobile user behavior is not restricted to one type of interactions such as scrolling, but often switches between multiple activities depending on content.

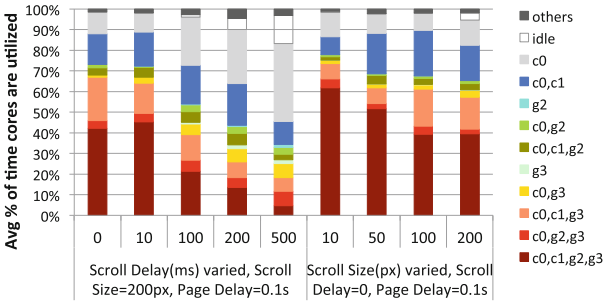
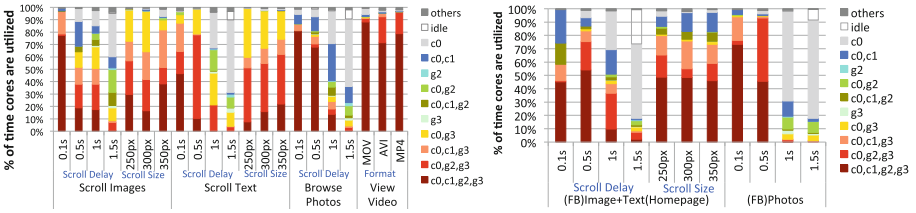


Fig. 5. Core utilization of BBench with different scrolling parameters

Thus, we further profile more detailed characteristics for diverse common interactions in Browser and Facebook by creating different microbenchmarks. We explore the actions of scrolling (down), browsing through photos (scrolling horizontally) and video viewing, and vary the speed of interaction (scrolling delay, scrolling size) and contents being viewed (images, text, images and text). Figure 6 describes the experimental results of clusters for different microbenchmarks in the Browser app. The result shows that the speed of interaction (scroll delay) impacted the amount of GPU acceleration most, however core combinations were impacted by the variation in the content. In addition, the profiled



(a) Browser **(b) Facebook**

Fig. 6. Clusters for common interactions of two mobile applications

microbenchmarks exhibit a wide range of variations in compute cores. In Fig. 6b, we show the example of Facebook use cases. Although common interactions vary for different applications, we found the similar observation that the speed of user interactions significantly affect GPU usage. We also observed the wide variations, for example, 10–90 % in GPU acceleration.

Using the profiled microbenchmarks which allow creating different workloads, we develop a framework to automatically generate mobile test runs to with user variations. We exploit the CPU-GPU profiles of microbenchmarks as a library of scenarios. The library creation can be repeated for other applications of interest. These are then combined in a calculated mix to generate a desired proportion of CPU-GPU interaction, that represents a target level of user interactions. This can be framed as the following optimization problem (1):

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \|Ax - b\| + \lambda|x| \\ \text{subject to} \quad & 1 \geq x_i \geq 0 \\ & \sum_i x_i = 1 \end{aligned} \tag{1}$$

where **A** is the matrix representing the library of interactions, **b** is the desired core utilization profile, and **x** is the proportion of the test run that each microbenchmark is used. λ is a regularization constant used to encourage sparsity. In our case, the profiled compute core usage of each microbenchmark, which is a vector for the time percentage of each cluster, is given by a column of the matrix **A**, while a cluster profile of an actual user workload is set to **b**. Then, this optimization problem computes a vector **x** that contains the proportion of each scenario to be executed.

These proportions form the parameters of a MonkeyRunner script that consists of the automated inputs for each microbenchmark. The resultant script drives the workload on the Android system. Figure 7 shows an example of generated CPU-GPU profiles for a range of desired GPU acceleration. GPU usage observed is within ± 3.3 % of the desired usage. High GPU acceleration characterizes high user engagement and vice-versa. Thus, the range of inputs can model the behavior of a vast range of users, from a power user to a light user.

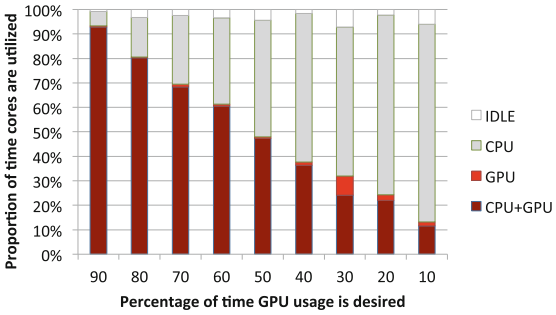


Fig. 7. Generating runs with varying intensity of user interactions

5.2 Comparison with User Workloads

Table 2 lists the interactions that comprise each test. The test runs use a combination of common interactions within a 3–6 min session for the apps. Templerun is not replayable due to the game dynamics, instead the test constitutes a 4-min play. We also include the BBench (BB) benchmark in the characterization, which matches a Browser cluster closely as shown.

Table 2. User Interactions in Test Runs

		Interaction
Browser	$B1$	Search for a Video, Play a Youtube video(A), Scroll while video is playing(B), Scroll while video is stopped(C).
	$B2$	Search for pictures, swipe through full-screen pictures from Google images(A), Scroll through the image results(B).
	BB	Display and scroll through webpages from 11 sites provided in BBench3.0 [3], with 1 s page delay, 0.5s scroll delay, 200px scroll size in 5 iterations.
Facebook	$F1$	View albums and pictures in a profile
	$F2$	View a video that plays within the app(A), view photos(B), view a video that plays on an external website in Browser(C).
Email	$E1$,$E2$	Quick scrolling through four emails multiple times (two text emails, one email with inline photos and one email with a single link that is viewed but not clicked).
Skype	$S1$	5 min Skype call: 1 min guest video on, 2mins host front camera on, 1 mins host back camera on.
Templerun	$T1$	A 4 min play, with 4–5 instances of lost game lives.
Music	$M1$	Play-Pause-Play music for 2mins each.
Camera	$C1$	3 min video capture with zoom-in/out

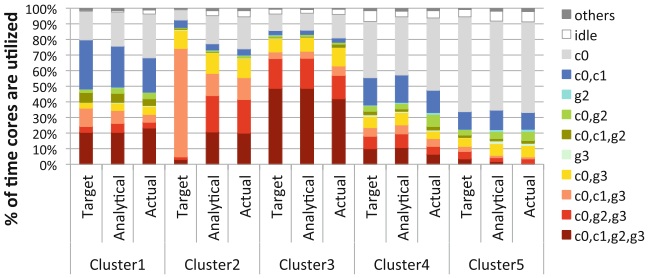


Fig. 8. Comparing automated test run generation for clusters

Figure 8 illustrates the use of the framework to generate the average clusters observed with the Browser app in our user study. The GPU usage in the analytical profile is within $\pm 2\,\%$ of the desired profile, while that in the observed profile is within $\pm 5\,\%$ of the desired profile, for all clusters except Cluster-2. In these clusters, the library of microbenchmarks was also able to closely match the various core combinations in the desired profile. In Cluster-2, with our current library profiles, the total GPU usage in the best library combination was 13 % lower than the average GPU usage in the cluster. With a richer library, such clusters can be matched better.

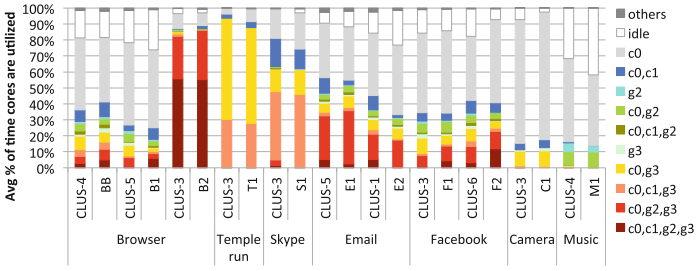


Fig. 9. Replayable test runs for 7 apps and clustered with user runs

We cluster replayable runs of all test apps along with user runs for the same number of clusters as before, ensuring that test runs cluster with user data while maintaining $\geq 80\,\%$ goodness ratio. Figure 9 shows the replayable test runs that match different clusters as denoted by 'CLUS' in short. Test runs are compared to closest matching data runs according to distance criterion to show similarity with real user runs. For each application, the difference in goodness ratio after test runs were added was 0.1–3.3 %, meaning that the replayable test runs can represent the use of compute cores.

6 Analysis with Test Runs

6.1 Resource Characterization for Scenarios

With generated replayable test runs, detailed system analysis becomes possible under realistic mobile workloads. We first characterize how the system resource of compute cores are used with respect to different user scenarios. In the experiment, we exploited the distinct scenarios (A, B, C) described in Table 2 to replay different user interactions.

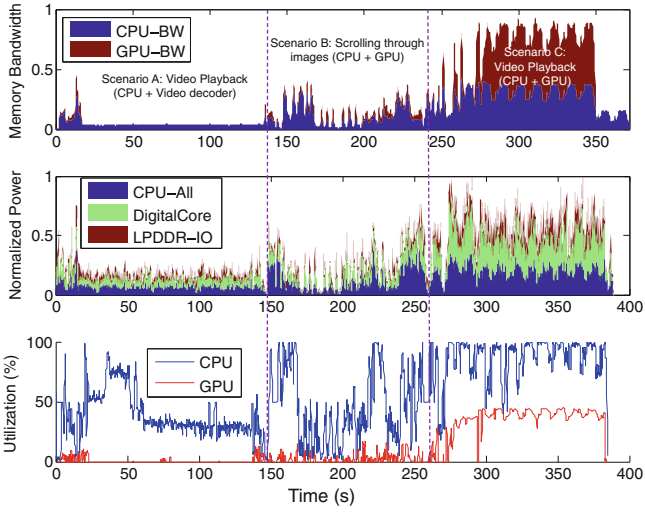


Fig. 10. Power, Bandwidth and Utilization in F2

Figure 10 illustrates the correlation between power, bandwidth and utilization, best seen with the example of $F2$. While video playback occurs in both scenarios A and C, the system experiences lower core usage in A (in-app playable video) than C (youtube video playback in browser). This translates into lower memory bandwidth requirements and power dissipation of the cores. GPU memory bandwidth scales with the GPU utilization. Power measurements include the power consumption from other digital core components such as video decoder, but show a similar scaling.

6.2 Memory Bandwidth and User Interaction

We then analyze the memory bandwidth requirements of the applications. Our systems embed LPDDR memory shared between compute cores, enabling CPU and GPU interactions. Figure 11 shows the temporal memory bandwidth usage of test runs measured on LPDDR links with the CPU and GPU. The bandwidth

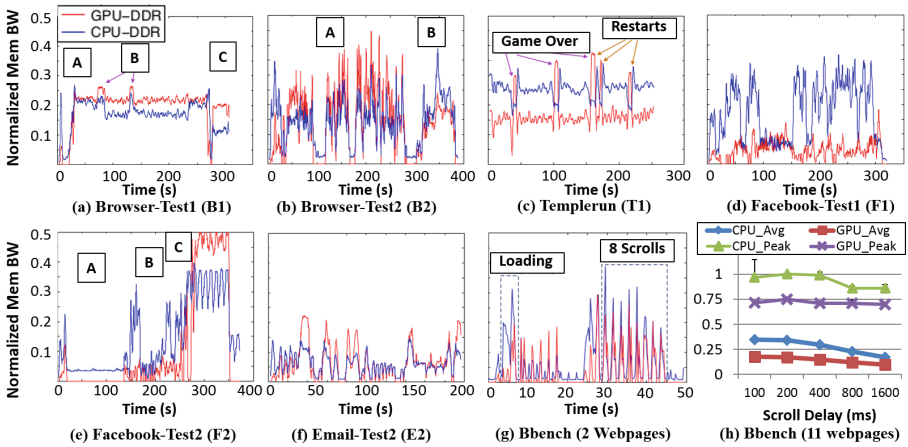


Fig. 11. Memory bandwidth usage by CPU and GPU for six applications and Bbench scrolling

is normalized to the peak total bandwidth experienced by the system across all runs. Specific interaction scenarios described in Table 2 are annotated.

We observe that while Templerun game shows a constant use of GPU acceleration, even the non-gaming applications show frequent memory transfers. This suggests fine-grained interactions with the CPU. To probe deeper, we profile memory bandwidth during slow browser scrolling (delay = 2 s) in Bbench. Figure 11g shows an experiment with two webpages (bbc and slashdot). Each scroll generates a CPU and GPU spike, highlighting its use of GPU acceleration. User 'wait' events in Bbench, $B2$, $E1$ and $E2$ are experienced as bandwidth dips on GPU ports. When scrolling was profiled at varying speeds (Fig. 11h), the GPU peak bandwidth did not change significantly, but average GPU bandwidth experienced by the system decreased by ∼14 % as scroll delays doubled. This suggests that predicting scroll action and its delays could allow for shaving off GPU bandwidth spikes for slow interactions without affecting user experience.

We make several other observations from these results: One, browser session $B1$ and Templerun, $T1$ show comparable memory bandwidth use. $B1$ is actively playing a video, but shows clear peaks in bandwidth when scrolling (events B,C). Similarly, the GPU bandwidth of email $E1$ and $E2$ approach the average bandwidth of $B1$ and $T1$ during routine scrolling events. This indicates that the GPU bandwidth requirements of UI engine are often higher than those needed for the gaming application, which presents an opportunity for optimization. Two, the memory bandwidth of CPU and GPU were quite different during the video and photo viewing actions in Browser and Facebook. F1 used 3x higher CPU than GPU bandwidth for photo viewing, while with B2 they were comparable. This may be imputed to both content and the application differences, however it is dramatic for essentially a similar type of user interaction. The interactive Email application also uses 3D UI to render emails, requiring GPU bandwidth

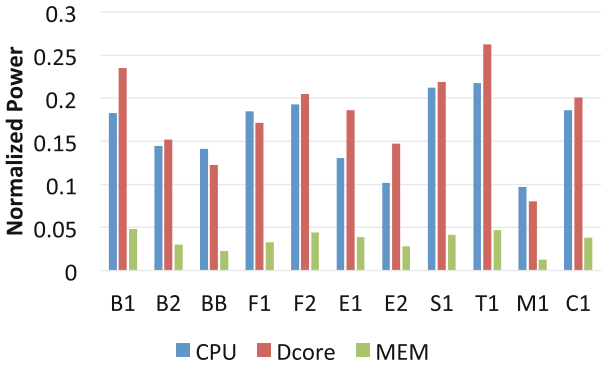


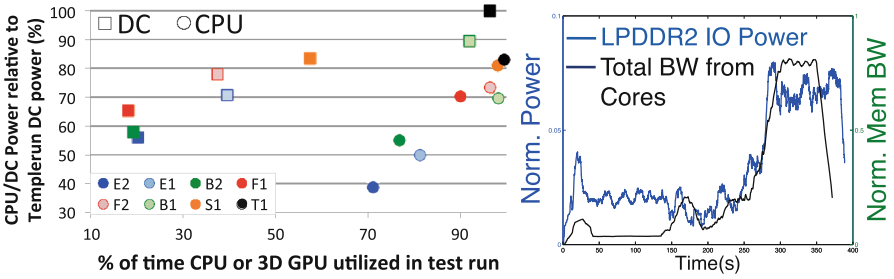
Fig. 12. Average power consumption of test runs

comparable to that of the CPU. Lastly, sudden bandwidth jumps are observed in Templerun at the end of a game, where the GPU bandwidth shoots up for the entire duration of the *'play again'* screen, although it displays static content. This suggests that game computations could be optimized for their use of CPU and GPU during this time. In all, due to the frequent GPU acceleration during the highly interactive applications, any optimization of the CPU-GPU interaction data path would be useful for performance or power improvements.

6.3 Impact on Power and Optimization

Figure 12 shows the relative power expended in the CPUs, GPUs and memory I/O in test runs. GPU power draw is inferred from Digital Core power rail which includes the GPUs, video decoder and modem digital core. Power is normalized to the maximum average battery power observed in all our tests. This occurred during Skype run due to wi-fi for video call. As conjectured in Sect. 4.1, the power consumption of the system depends on the types of cores used by the mobile applications (e.g. Templerun compared to Music) and is impacted by the user variations within an app.

The power consumption trends are well explained from the usage analysis of cores. As shown in Fig. 13a, the CPU and Digital Core (DC) power consumption of test runs is correlated with the proportion of time the CPUs and the 3D GPU are active respectively. By design, test runs from the same application showed distinct CPU-GPU usage based on user variation, and these differences clearly scaled the power consumption. Thus, the large range of GPU utilization observed due to user variation resulted in significantly large range of power consumption in GPU (45–100 % with respect to Templerun power.) This motivates the need to uncover more user behavior based optimizations for better power efficiencies. Applications at same utilization proportion require deeper analysis of the utilization levels to understand power consumption. For example, $B2$, $F1$ and $E2$ each show about 20 % 3D GPU usage, yet they consume varying amounts of power due to inter-app differences.



(a) Power trends correlate to % time cores are active (b) $F2$ memory I/O power & bandwidth

Fig. 13. Impact of user variation on optimization

Furthermore, the power consumption of memory I/O is also highly affected by user interactions. As shown in Fig. 13b, the LPDDR2 IO power is correlated to the amount of IO accesses. Since memory bandwidth trends are changed by the speed of user interactions as discussed in Sect. 6.2, this suggests that the memory system can utilize the remained bandwidth by considering user interactions.

When normalized to their respective average battery power, CPUs, digital cores and memory consume about 50 % of the average battery power across all test runs. Since these power measurements include the effect of the on-demand Linux governor for DVFS and mpdecision, we further investigate how the user interaction variations affect the optimization policy behavior. Figure 14 shows three example benchmarks with varied user interaction intensity. Using the Opt-OFF case, which the frequencies of two cores are maintained at the maximum level with Performance Governor, as the baseline, we compared to the Opt-ON case that the two cores are controlled by Android Linux default CPU management policy, i.e., Ondemand Governor and Mpdecision daemon. Power measurements on MSM8660 showed $8-15$ % lower core power due to the effect of DVFS and mpdecision. This is again an example where replayability helped to compare power savings given 'real user-like' interactive sessions.

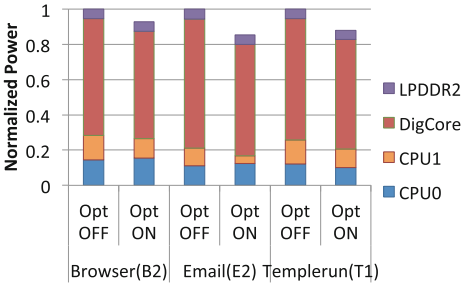


Fig. 14. Comparing savings due to power optimizations

6.4 Changes in Devices and Platforms

The test run can reproduce the same workload to different devices and platforms. Figure 15 shows the utilization breakdown of the test runs replayed on three devices: (A) MSM8660 with Icecream Sandwich(ICS) (B) MSM8960 with ICS (C) MSM8960 with JellyBeans (JB). The different MSM platforms used different combinations of accelerators, but the OS differences were not very significant. The most differences in platforms were seen in the way they handled Browser tests. Where MSM8660 used other accelerators, MSM8960 used the GPU in conjunction with the other accelerators. This is also seen in the MobileBench benchmarks of PhotoView and VideoView. MSM8960 also utilized the GPU cores more efficiently in that it avoided use of 2DGPU in Email and Bbench in contrast to MSM8660. Overall, the experiments show that different devices may use accelerators in different combinations, however, more recent platforms used higher amounts of GPU acceleration in most of the test runs.

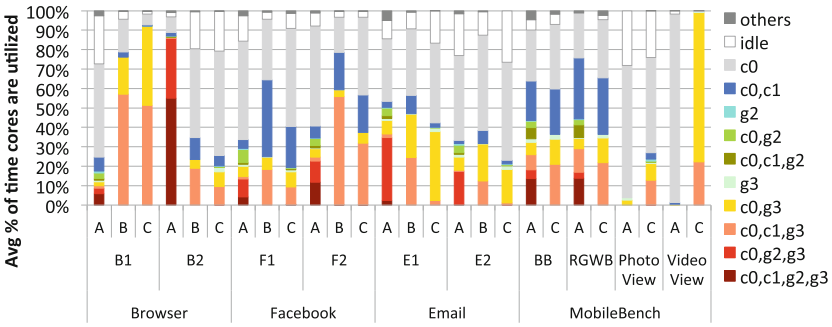


Fig. 15. Utilization breakdown of test runs on three devices

7 Conclusion

Mobile systems must provide rich user experience in extremely low power budgets. The demands placed on these heterogeneous cores in typical user scenarios are unclear, limiting usage behavior driven power and performance optimizations. In this paper, we analyze how real users utilize CPUs and GPUs and show the large amount of GPU acceleration used in non-gaming, interactive mobile applications. With a detailed study of the CPU-GPU-memory interactions under user behavior variation, we developed a framework that automatically generates replayable test runs. Using the replayable test runs which reproduce realistic user behaviors, we show the impact on memory bandwidth and power consumption, which suggests the need to optimize GPU acceleration for common interaction tasks.

Acknowledgments. This work was supported in part by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA, National Science Foundation (NSF) award 1344153 and Qualcomm.

References

1. Arnau, J.-M., et al.: Parallel frame rendering: trading responsiveness for energy on a mobile GPU. In: PACT 2013 (2013)
2. Bogdan, P., Marculescu, R.: Workload characterization and its impact on multicore platform design. In: CODES+ISSS, pp. 231–240. ACM (2010)
3. Carroll, A., Heiser, G.: An analysis of power consumption in a smartphone. In: USENIXATC (2010)
4. Cho, C.-W., et al.: Performance optimization of 3D applications by opengl es library hooking in mobile devices. In: ICIS (2014)
5. Do, T.M.T., et al.: Smartphone usage in the wild: a large-scale analysis of applications and context. In: ICMI (2011)
6. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Willey, New York (1973)
7. Falaki, H., et al.: Diversity in smartphone usage. In: MobiSys, pp. 179–194 (2010)
8. Gao, C., et al. A study of mobile device utilization. In: ISPASS, pp. 225–234 (2015)
9. Gomez, L., et al.: RERAN: Timing- and touch-sensitive record and replay for Android. In: ICSE (2013)
10. Gutierrez, A., et al.: Full-system analysis and characterization of interactive smartphone applications. In: IISWC, pp. 81–90 (2011)
11. Huang, Y., et al.: Moby: a mobile benchmark suite for architectural simulators. In: ISPASS (2014)
12. Kim, S., et al.: Computing energy-efficiency in the mobile GPU. In: ISOCC, pp. 219–221 (2013)
13. Ma, X., et al.: Characterizing the performance and power consumption of 3D mobile games. In: Computer (2013)
14. Mochocki, B., et al.: Signature-based workload estimation for mobile 3D graphics. In: DAC (2006)
15. Pandiyan, D., et al.: Performance, energy characterizations and architectural implications of an emerging mobile platform benchmark suite. In: IISWC 2013 (2013)
16. Park,J.-G., et al.: Quality-aware mobile graphics workload characterization for energy-efficient DVFS design. In: ESTIMedia (2014)
17. Pathak, A., et al.: Where is the energy spent inside my app?: fine grained energy-accounting on smartphones with eprof. In: EuroSys, pp. 29–42 (2012)
18. Pathania, A., et al.: Integrated CPU-GPU power management for 3D mobile games. In: DAC (2014)
19. Peters, J.F.: Topology of digital images. In: Peters, J.F. (ed.) ISRL, vol. 63, pp. 1–76. Springer, Heidelberg (2014)
20. Shepard, C., et al.: Livelab: measuring wireless networks and smartphone users in the field. SIGMETRICS Perform. Eval. Rev. **38**(3), 15–20 (2011)
21. Trepn profiler. https://developer.qualcomm.com/mobile-evelopment/increase-app-performance/trepn-profiler
22. Trestian, I., et al.: Measuring serendipity: connecting people, locations and interestsin a mobile 3G network. In: IMC 2009 (2009)

23. Wang, G., et al.: Accelerating computer vision algorithms using opencl framework on the mobile GPU - a case study. In: ICASSP, pp. 2629–2633 (2013)
24. Wang, R., et al.: Architectural characterization and analysis of high-end mobile client workloads. In: ICEAC (2013)
25. Xu, Q., et al.: Identifying diverse usage behaviors of smartphone apps. In: IMC, pp. 329–344 (2011)