To Cloud or Not to Cloud: Measuring the Performance of Mobile Gaming

Chun-Ying Huang
Department of
Computer Science and Engineering
National Taiwan Ocean University
Keelung, Taiwan
chuang@ntou.edu.tw

Yu-Ling Huang
Department of
Computer Science and Engineering
National Taiwan Ocean University
Keelung, Taiwan
ylhuang@snsl.cs.ntou.edu.tw

Yu-Hsuan Chi
Department of
Computer Science and Engineering
National Taiwan Ocean University
Keelung, Taiwan
yhchi@snsl.cs.ntou.edu.tw

Kuan-Ta Chen
Institute of Information Science
Academia Sinica
Taipei, Taiwan
swc@iis.sinica.edu.tw

Cheng-Hsin Hsu
Department of Computer Science
National Tsing-Hua University
Hsinchu, Taiwan
chsu@cs.nthu.edu.tw

ABSTRACT

Mobile cloud gaming allows gamers to play games on resource-constrained mobile devices, and a measurement study to quantity the client performance and energy consumption is crucial to attract and retain the gamers. In this paper, we adopt an open source cloud gaming platform to conduct extensive experiments on real mobile clients. Our experiment results show several insights that are of interests to researchers, developers, and gamers. First, compared to native games, mobile cloud games save energy by up to 30%. Second, the hardware video coders achieve higher frame rates but suffer from a small unnecessary buffering delay, and thus is less ideal for fast-paced games. Third, the frame rate, bit rate, and resolution all affect the decoders' resource consumption, while frame rate imposes the highest impact. Last, cellular networks incur 30%–45% more energy consumption than WiFi networks, and the event processing of touch screens is also energy-hungry. These findings shed some light on the further enhancements of the emerging mobile cloud gaming platforms.

Categories and Subject Descriptors

H.5 [**Information Systems Applications**]: Multimedia Information Systems

General Terms

Design, Measurement

Keywords

Mobile cloud gaming; remote rendering; live video streaming; real-time encoding; performance evaluation

1. INTRODUCTION

Cloud gaming companies offer on-demand cloud games to users. These games run on powerful cloud servers, and the game scenes are captured, encoded, and streamed to thin clients running on desktops, laptops, and TV set-top boxes. In contrast to the aforementioned devices, mobile devices, such as tablets and smartphones, have limited computation power and are battery-powered. Therefore, running mobile clients on these resource-constrained devices may lead to inferior performance and high energy consumption. For example, the gaming frame rate may become too low for smooth game play due to insufficient CPU power to execute video decoders. This results in degraded gaming quality and may drive the gamers away. On the other hand, when gamers play cloud games, the communication, computing, and display components on mobile devices all consume nontrivial energy, which may quickly drain the battery and prevent gamers from using their mobile devices for other purposes, such as making phone calls. Hence, carefully measuring the performance and energy consumption of mobile clients is critical to the success of the new mobile cloud gaming ecosystem.

In this paper[1], we adopt an open source cloud gaming platform [7, 8], called GamingAnywhere (GA), to setup a real mobile cloud gaming testbed. We conduct extensive experiments on the testbed to answer the following questions.

- **Does running cloud games save energy compared to native mobile games?** While in first glance, offloading games to cloud servers saves significant amount of computation energy, doing so however also increases the communication energy consumption. This is partially due to the nature of constant video streams incurred by cloud games, which prohibits the wireless interfaces from being turned off when idling. Hence, a carefully designed measurement study is required to compare their total client energy consumptions.

- **What are the pros and cons of the hardware video decoder?** The video decoder represents the majority of CPU workload on the mobile client, which in turn slows down the overall frame rendering and consumes more energy. One possible solution is to leverage the hardware video decoders

[1]This paper is an extended version of a workshop short paper that appeared as [6].

on modern mobile devices. However, hardware video decoders are usually less configurable than the software ones. Thus, detailed empirical comparisons may shed light for future mobile cloud gaming researchers and developers on the tradeoff between the hardware and software video decoders.

- **How does the server configuration affect the client performance?** The cloud gaming servers are highly configurable, and selecting the best configuration itself is a challenging task. Therefore, a comprehensive set of experiments is required to throughly quantify the impacts of different server system parameters, such as frame rates and resolutions, on the client performance and energy consumption. The measurement results are useful to researcher, developers, and gamers to better configure their cloud gaming servers.
- **Does the mobile client suffer from any power-hungry software components other than video decoders?** Although the software video decoders incur the majority of the CPU workload, which consumes high energy, we also need to identify other software components, if any, that consume nontrivial energy. This is because, the battery technology is not improved as fast as other hardware components [3], and the overall energy consumption is crucial to retain mobile cloud gamers.

Our extensive experiments and in-depth analysis depict several insights that lead to design suggestions for future developments of mobile cloud gaming platforms. To the best of our knowledge, similar measurement studies have not been rigorously done in the literature.

2. RELATED WORK

Various optimization approaches have been proposed for mobile cloud gaming platforms, which can be classified into: specialized video codecs [2, 5, 13] and system adaptations [1, 11, 14], Specialized video codecs leverage the properties of computer rendered graphics to reduce the downlink bandwidth consumption of mobile cloud games. In particular, Hemmati et al. [5] selectively encode game objects to save network bandwidth and rendering power while maintaining gaming quality. Shu et al. [13] adopt 3D warping for light-weight post-rendering manipulation on mobile clients, in order to reduce the network bandwidth and cope with network delay. Chuah and Cheung [2] render low-quality game scenes on mobile devices while streaming the difference between low- and full-quality game scenes from cloud servers, so as to trade client computation complexity for communication complexity. System adaptations dynamically adjust the system resources across multiple distributed servers for better overall performance. Cai et al. [1] divide computer games into small components, and dynamically move these components across multiple distributed servers to meet the demands from mobile gamers. Wang and Dey [14] adjust the visual effect levels of computer games to trade off the server computation load and user-perceived quality. Liu et al. [11] present a subjective model to approximate the user experience under diverse video contents, coding parameters, and network conditions, in order to guide their adaptive rendering component for better gaming quality. These optimization techniques [1, 2, 5, 11, 13, 14] are complementary to the measurement studies presented in this paper.

Both objective measurement studies [12] and subjective user studies [9] on cloud gaming using desktops have been done in the literature. We focus on mobile cloud gaming, which has only been recently considered [4, 8, 10]. Lampe et al. [10] adopt three performance metrics: latency, energy, and cost, trying to demonstrate the

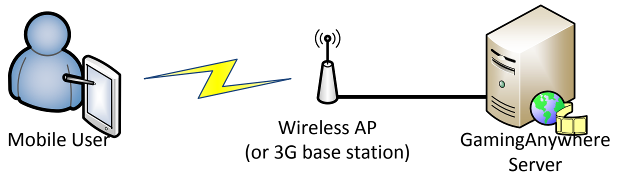

Figure 1: The GA experiment testbed used throughout this paper.

feasibility of mobile cloud gaming. In our earlier work [8], we conduct a user study to quantify the impact of different configurations on mobile gamer satisfaction. In contrast, the current paper concentrates on the objective measurements on mobile devices. Hans et al. [4] measure the energy consumption of Android smartphones running a cloud gaming client, and is probably the closest work to ours. While their findings on energy consumption are in-line with ours, Hans et al. [4] is different from the current paper for two main reasons: (i) we measure both client performance and energy consumption, while they focus on energy, and (ii) we deploy several actual games in our cloud gaming platform, while their CPU/GPU workloads are emulated.

The current paper is built upon our earlier work on developing the first open source cloud gaming platform [7], and porting the cloud gaming client to mobile devices [8]. Our mobile cloud gaming platform is transparent to existing PC games, and is of interests to researchers and developers for experiments and further enhancements.

3. METHODOLOGY

3.1 Environment Setup

Figure 1 shows the testbed used in our experiments. The testbed consists of a server and a mobile client connected via a wireless access (a campus WiFi or a 3G cellular network). We install five games of different genres on the GA server: Super Smash Bros, Limbo, Batman, Mario Kart, and Zelda. Super Smash Bros is a fighting game, Limbo is a 2D scrolled adventure game, Batman is a 3D adventure game, Mario Kart is a 3D racing game, and Zelda is an RPG. We study the GA mobile client's performance and energy consumption using these five games, and report the sample results from Super Smash Bros if not otherwise specified. For comparing cloud and native games, we adopt a cross-platform OpenGL game: GLTron, which runs on both the server and Android devices. GLTron is a 3D snake-like game.

The server has an Intel Q6600 2.4 Ghz quad-core CPU and runs Windows 7. We consider two mobile devices: an ASUS Nexus 7 tablet and a Sony Xperia Z smartphone. The tablet has a Nvidia Tegra 3 1.2 quad-core process with 1 GB ram, and the smartphone has a Qualcomm Snapdragon APQ8064 1.5 GHz quad-core processor wth 2 GB ram. Both mobile devices run Android 4.4.2. We adopt the tools, *UseMon* and *Current Widget*, to collect measured CPU utilization and power consumption of a device, respectively. During the experiments, we set the screen brightness to medium, and always keep the battery level above 70% to avoid noises due to battery's nonlinear discharging characteristics (details are given in Section 3.3).

3.2 Controlled Parameters

Table 1 lists the controlled parameters during the experiments. First our mobile client supports both software and hardware video decoders. The software decoder is provided by the *ffmpeg* project,

Table 1: Controlled parameters

Parameter	Value
Decoder	hardware, software
Controller	disabled, enabled
Video codec parameter[‡]	
Resolution	**640x480**, 960x720, 1280x720
Bitrate	1Mbps, **3Mbps**, 5Mbps
Frame rate	10fps, **30fps**, 50fps

‡ Default values are highlighted in boldface.

and the hardware decoder is accessed via Android's *MediaCodec* framework. We use the popular H.264 coding standard, which is supported by all the chosen implementations. Second, we selectively disable and enable the controller on mobile devices, which is a transparent overlay over the video surface. When the controller is disabled, we play the games on the server. This is to isolate the additional energy consumption due to: (i) activating touch screens and (ii) handling the user input events. The remaining three parameters, resolution, bitrate, and frame rate, are for video codecs. In each experiment, we fix two video codec parameters, and vary the other one. We let 640x480, 30 fps (frame per second), and 3 Mbps be the default settings, if not otherwise specified. The goal is to quantify the impacts of different parameters on client performance and energy consumptions.

3.3 Baseline Energy Measurement

We measure the baseline energy consumptions before conducting the experiments. We close all irrelevant applications and services, turn on the display, and set brightness to medium. We find that the CPU utilization is close to zero. We measure the current and voltage for each mobile device, sampled at 1 Hz. The results are shown in Figure 2. On both devices, we observe that when the battery level reduces, the voltage gets lower and the current gets higher. When the battery level is lower than 60%, the current exceeds the average. For fair comparisons, we only conduct experiments when battery level is higher than 70%. Based on the measurements, the baseline power consumption for Nexus 7 and Xperia Z are 1.7 W and 1.1 W, respectively.

4. MEASUREMENT RESULTS

4.1 Software vs. Hardware Video Decoders

Table 2 shows the frame rates achieved by the software decoders. This table reveals that while software decoders work well when the resolution, frame rate, and bitrate are low, they fail to achieve the configured frame rate when these video codec parameters are high. This can be attributed to the limited CPU resources on the mobile devices. We then switch to the hardware decoders, which run faster but do not report the achieved frame rate. We observe fairly constant frame rate under different video codec parameters. We make an interesting observation: the hardware decoders on both mobile devices buffer 1 or 2 decoded frames, which lead to unnecessary delay. Such limitation renders the hardware decoders less suitable to mobile cloud gaming platforms with longer network latency and fast-pace games.

Next, we zoom into two video codec configurations: (i) *light*, with 640x480, 10 fps, and 3 Mbps and (ii) *heavy*, with 1280x720, 30 fps, and 3 Mbps. We run each experiment for 15 minutes, and plot the CDF curves of per-core CPU utilization in Figure 3, where

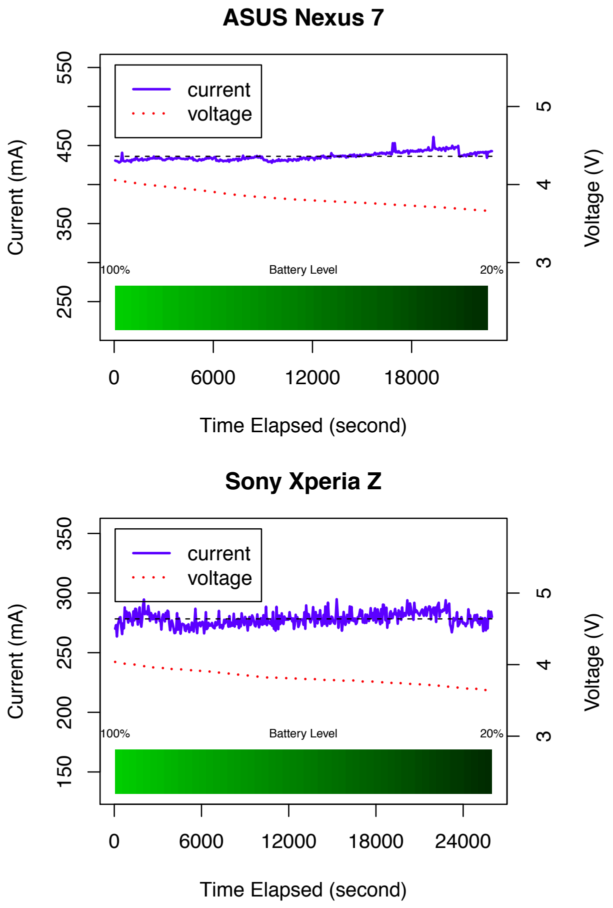

Figure 2: The voltage and current levels measured under the baseline configuration.

N7 and XZ represent the tablet and smartphone respectively. We first observe that, for each mobile device, the two curves (light and heavy) of the hardware codecs are very close. This validates the aforementioned observation: the hardware decoders achieve the same frame rate under different video codec parameters. In contrast, for each mobile device, the gap between two curves of the software codec is much larger. Last, our measurements on the power consumption leads to similar observation (figure not shown due to the space limitations). The hardware decoders consume power levels that are about two times of the baseline, which is independent to video codec parameters. In contrast, the software decoders are sensitive to video codec parameters, and draw power consumptions that are between two and three times of the baseline.

4.2 Video Codec Parameters

We next present the CPU utilization and power consumption under different video codec parameters. We repeat the 3-minute experiment 5 times, collect samples at 1 Hz, and give the average results with minimum and maximum in Figure 4. This figure reports average CPU utilization, i.e., 25% CPU utilization is equivalent to a fully-loaded CPU core. We make two observations: (i) higher bitrates, frame rates, and resolutions consume more resources and (ii) the software decoders consume more resources. Next, we take a closer look at how each codec parameter affects the performance of the hardware decoders. We do not consider the software de-

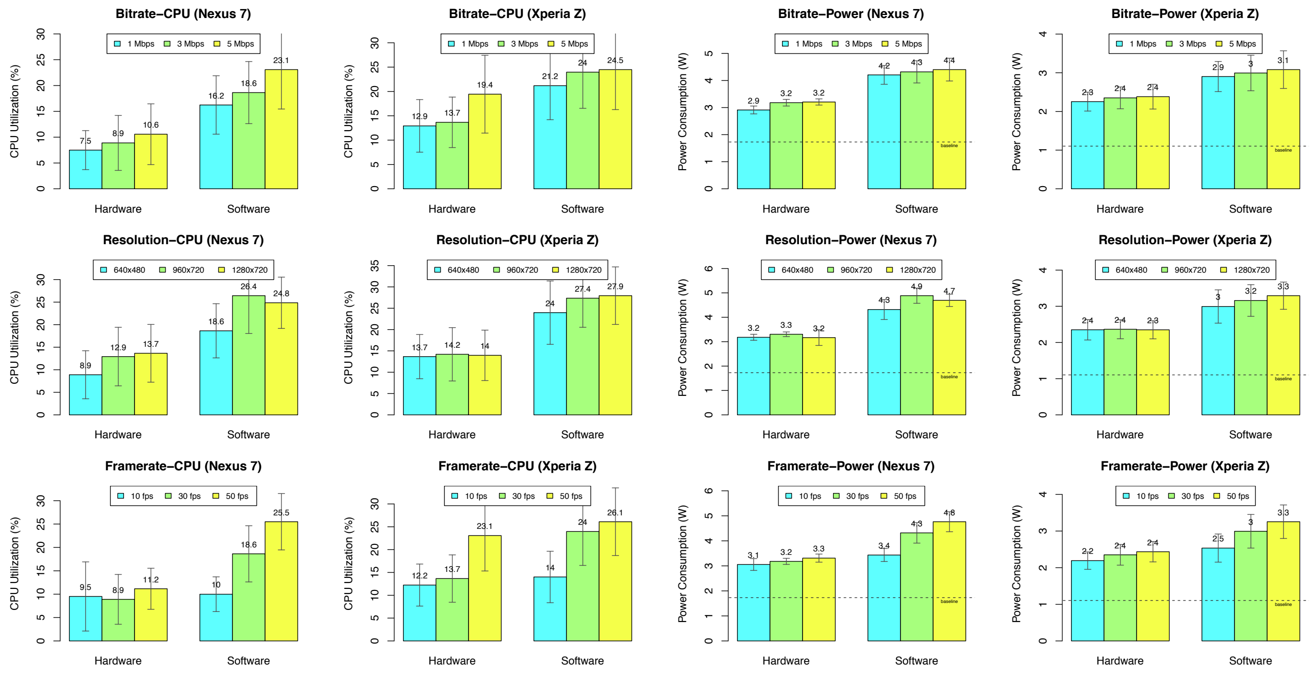

Figure 4: Measured CPU utilization and power consumption under various codec parameters.

Table 2: Achieved decoder frame rate (in fps)

	Nexus 7			Xperia Z		
Configured	10	30	50	10	30	50
1280x720, 5 Mbps	10	13	13	10	14	13
1280x720, 3 Mbps	10	14	13	10	14	13
1280x720, 1 Mbps	10	27	19	10	17	13
960x720, 5 Mbps	10	13	13	10	14	13
960x720, 3 Mbps	10	18	15	10	16	13
960x720, 1 Mbps	10	30	24	10	24	14
640x480, 5 Mbps	10	30	30	10	30	22
640x480, 3 Mbps	10	30	46	10	30	27
640x480, 1 Mbps	10	30	45	10	30	45

coders, because neither of the considered mobile devices can keep up with the high frame rate. We define the parameter impact factor as follows. Given a parameter p and a function f_p that quantifies the load of p based on its parameters. Suppose p is altered from c_i to c_j, we write the increased load L_p as $L_p = \frac{f_p(c_j)-f_p(c_i)}{f_p(c_i)}$. We also measure the battery level differences m_i and m_j for c_i and c_j, respectively. The increased overhead O_p for p is defined as $O_p = \frac{m_j - m_i}{m_i}$. Last, the impact factor for parameter p is written as $\frac{O_p}{L_p}$. Note that we measure the battery level difference to define the parameter impact factor because CPU utilization does not fully reflect system loads, as some workload is offloaded to the hardware decoders. Table 3 gives the parameter impact factors. This table shows that the frame rate has the highest impact, and the resolution has the lowest.

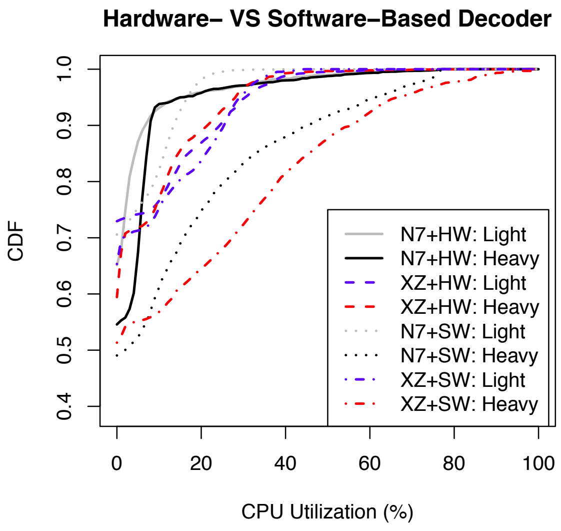

Figure 3: Comparison of CPU utilization with the hardware and software decoders under different loads.

4.3 Game Genres

We report the CPU utilization and energy consumption of different game genres in Figure 5. All the games are configured to stream game scenes at 1280x720, although only Windows games (Limbo, Batman, and GLTron) support 1280x720: the game scenes captured from N64 emulator (Super Smash Bros, Mario Kart, and Zelda) have to be up-sampled to 1280x720 before being streamed. Therefore, the game scenes from the N64 emulated games contain less details, which in turn consume less resources. The power consumption fluctuations caused by all game genres are within $\pm 5\%$.

Table 3: The parameter impact factors for hardware decoders

	Nexus 7			Xperia Z		
Param. Change[†]	a→b	b→c	a→c	a→b	b→c	a→c
Bitrate	+0.14	+0.02	+0.13	+0.07	+0.03	+0.07
Frame rate	+0.06	+0.10	+0.10	+0.11	+0.09	+0.14
Resolution	+0.07	-0.17	-0.01	+0.01	-0.03	-0.01

[†] a, b, and c are the minimal, median, and maximal values of each parameter.

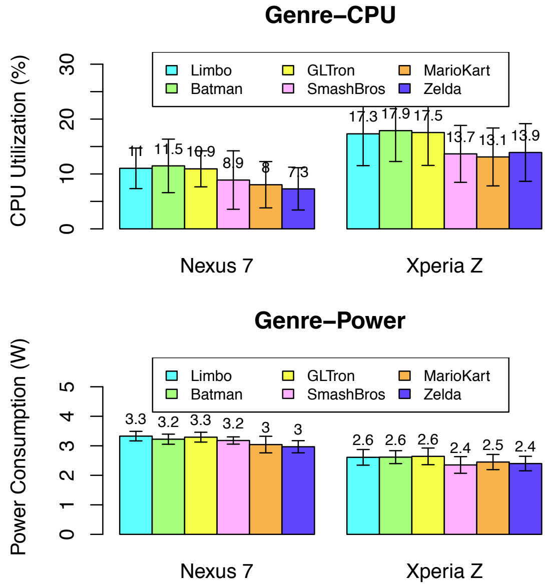

Figure 5: CPU utilization and power consumption for different game genres.

If we separate the Windows and N64 games, the fluctuations are about ±2%. We conclude that game genre has very little impact on cloud gaming CPU utilization and power consumption.

4.4 Cloud versus Native Games

Next, we play Super Smash Bros and GLTron as cloud games and GLTron as a native game. Cloud games are configured to stream at 1280x720. The results are shown in Figure 6. In the figure, “Cloud#1" and “Cloud#2" correspond to Super Smash Bros and GLTron, respectively. “Native" is the Android version of GLTron. It is clear that the native game consumes much more resources than cloud games: the CPU consumption is doubled and the power consumption is also increased by more than 30%. We emphasize that GLTron is not very visually-rich, but running it natively incurs nontrivial resource consumption. The resource consumption gap between the cloud and native games will be even larger for modern 3D games. Last, we make another observation: enabling the controller results in additional resource consumption. We take a deeper look at this observation below.

4.5 Other Components

We study the impact of other components on resource consumption, including the wireless access links and touch screens. Figure 7 shows the resource consumptions by using different wireless access links. We only report results from Xperia Z because Nexus 7 does not have a 3G module. We work with the default codec configuration (640x480, 30fps, 3Mbps). Both the software and hardware decoders can decode at the configured frame rate. The measurements also show that the 3G module consumes additional 30%–45% of power.

To measure the impact of touch screens, we develop an Android application that does nothing but accepting gestures from a user. The accepted events are dropped immediately. We run this *gesture* application on both mobile devices for 3 minutes and collect the CPU utilization and power consumption. We continuously slide the touch screens during the second minute, and leave the mobile devices idle in the first and last minutes. To isolate the resource consumption due to touch screens and event processing, we also write *replayer* application that injects the touch screen events to the gesture application. We run the gesture application and log all the timestamped events. We then inject the events to the gesture application using the replayer application, and compare the two measurement results. Figure 8 gives the CPU utilization and power consumption over time. This figure shows that the touch screen (including event processing) and event processing (only) consume similar amount of resources. This indicates that the event processing is energy-hungry, while the touch screen consumes negligible amount of additional resources. That is, touch screen and event processing is not free, although it may be overlooked in the past. Our measurements show that event processing consumes additional 6%-10% power. More in-depth analysis along this line is among our future tasks.

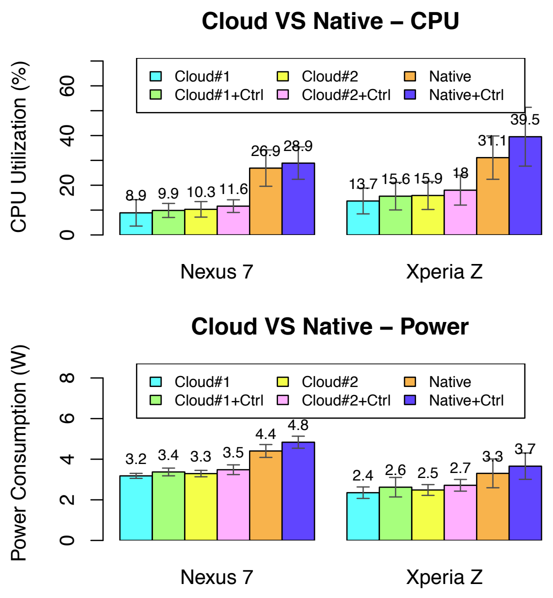

Figure 6: CPU utilization and power consumption for cloud and native games.

5. CONCLUSION

In this paper, we implement a testbed using a real mobile cloud gaming platform [7, 8] developed by us. We conduct extensive experiments to measure the client performance and energy consumption. Our measurement results lead to the following main findings.

- Running mobile cloud games is more energy efficient than native mobile games. Our experiments indicate that mobile cloud games reduce the CPU utilization by half, and save energy by 30%.

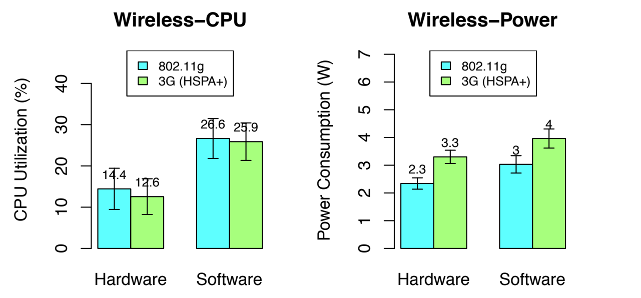

Figure 7: CPU utilization and power consumption for different wireless access links.

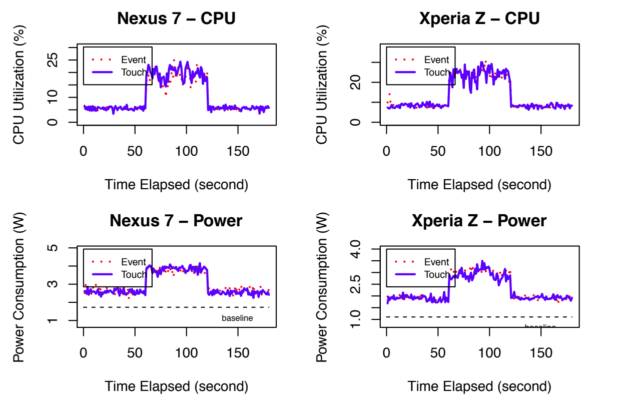

Figure 8: CPU utilization and power consumption for handling control events.

- The hardware video decoders are not sensitive to higher video codec parameters, such as resolutions and bitrates. However, the hardware video decoders always buffer 1–2 frames, which are less ideal to fast-pace games.
- The video codec parameters (bitrate, frame rate, and resolution) impose different degrees of impacts on CPU utilization and energy consumption. The frame rate affects the most, while the resolution affects the least.
- Two other energy-hungry components are identified: (i) the 3G cellular module consumes 30%–45% more energy and (ii) the event processing of touch screens consumes nontrivial resources as well.

These insights lead to design recommendations for future researchers and developers of the emerging mobile cloud gaming platforms.

Acknowledgment

This work was supported in part by the Ministry of Science and Technology of Taiwan under the grants MOST 103-2221-E-019-033-MY2 and MOST 103-2221-E-001-023-MY2. We would also like to thank the reviewers for their insightful comments.

References

[1] W. Cai, C. Zhou, V. Leung, and M. Chen. A cognitive platform for mobile cloud gaming. In *Proc. of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom'13)*, pages 72–79, Bristol, UK, December 2013.

[2] S. Chuah and N. Cheung. Layered coding for mobile cloud gaming. In *Proc. of ACM International Workshop on Massively Multiuser Virtual Environments (MMVE'14)*, pages 1–6, Singapore, March 2014.

[3] G.Perrucci, F. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. In *Proc. of IEEE Vehicular Technology Conference (VTC Spring'11)*, pages 1–6, Budapest, Hungary, May 2011.

[4] R. Hans, U. Lampe, D. Burgstahler, M. Hellwig, and R. Steinmetz. Where did my battery go? quantifying the energy consumption of cloud gaming. In *Proc. of IEEE International Conference on Mobile Services (MS'14)*, pages 63–67, Anchorage, AK, June 2014.

[5] M. Hemmati, A. Javadtalab, A. Shirehjini, S. Shirmohammadi, and T. Arici. Game as video: Bit rate reduction through adaptive object encoding. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'13)*, pages 7–12, Oslo, Norway, February 2013.

[6] C.-Y. Huang, P.-H. Chen, Y.-L. Huang, K.-T. Chen, and C.-H. Hsu. Measuring the client performance and energy consumption in mobile cloud gaming. In *Proceedings of IEEE/ACM NetGames*, December 2014.

[7] C.-Y. Huang, C.-H. Hsu, Y.-C. Chang, and K.-T. Chen. Gaminganywhere: An open cloud gaming system. In *Proc. of the ACM Multimedia Systems Conference (MMSys'13)*, pages 36–47, Oslo, Norway, February 2013.

[8] C.-Y. Huang, C.-H. Hsu, D.-Y. Chen, and K.-T. Chen. Quantifying user satisfaction in mobile cloud games. In *Proceedings of ACM Workshop on Mobile Video Delivery (MoVid'13)*, pages 4:1–4:6, 2014.

[9] M. Jarschel, D. Schlosser, S. Scheuring, and T. Hobfeld. Gaming in the clouds: QoE and the users' perspective. *Mathematical and Computer Modelling*, 11-12(57):2883–2894, June 2013.

[10] U. Lampe, R. Hans, and R. Steinmetz. Will mobile cloud gaming work? Findings on latency, energy, and cost. In *Proc. of IEEE International Conference on Mobile Services (MS'13)*, pages 960–961, Santa Clara, CA, June 2013.

[11] Y. Liu, S. Wang, and S. Dey. Content-aware modeling and enhancing user experience in cloud mobile rendering and streaming. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 4(1):43–56, March 2014.

[12] R. Shea, J. Liu, E. Ngai, and Y. Cui. Cloud gaming: Architecture and performance. *IEEE Network Magazine*, 27(4):16–21, July/August 2013.

[13] S. Shi, C. Hsu, K. Nahrstedt, and R. Campbell. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proc. of the ACM international conference on Multimedia (MM'11)*, pages 103–112, Scottsdale, AZ, November 2011.

[14] S. Wang and S. Dey. Adaptive mobile cloud computing to enable rich mobile multimedia applications. *IEEE Transactions on Multimedia*, 15(4):870–883, June 2013.