Energy-Conscious QoS in the Mobile Web with Hardware Heterogeneity[*]

Yuhao Zhu Vijay Japana Reddi

1. The Rise of Mobile Web Browsing

Mobile and Web browsing together are emerging as a critical new application domain. Recent studies indicate that nearly every person in the U.S. now owns about 3 mobile devices [1] and that users spend more time on their mobile devices than on desktops. Accompanying the proliferation of mobile devices is the emergence of new Web technologies such as HTML5 that facilitate the rapid deployment of Internet content through Web browsers. As users rely more on mobile and Web technologies rather than desktop browsers to access the Internet, we can expect an even-faster penetration of mobile Web browsing.

In the user-centric and highly interactive mobile Web browsing context, webpage load time and latency are the most important criteria for end-users' quality-of-service (QoS) experience. The implications of poor Web QoS can be severe. For instance, Amazon concluded that a 1-second delay in webpage load time could translate to $1.6 billion lost in sales annually [2], simply because users abandon webpages that take too long to load. Similarly, Google calculated that four-tenths of a second delay in search results could lead to 8 million searches lost per day.

Although the demand for QoS is high, current mobile system designs already fall behind end-users' expectations. Studies in 2013 indicate that 71% of mobile Web users expect website performance on their mobile devices to be not worse than their desktop experience–up from 58% in 2009 [3]. It is becoming difficult to guarantee high QoS due to the increasing computational intensity of webpages. Using `www.cnn.com` as an example, in Fig. 1 we show that over the past decade, advancements in network technologies have enabled us to keep pace with network transmission overheads, increasing webpage sizes and so forth. However, the webpage computation time on the client side has increased by nearly a factor of 10X, which strongly emphasizes the growing demand for computational capability. But the battery-constrained nature of mobile devices prohibits the single-minded pursuit of a performance-oriented mobile system design strategy.

The challenge facing us is how to deliver the mobile Web browsing QoS requirement under the energy envelopment. Heterogeneous systems with big/little cores, each capable of performing DVFS, have been known as energy-efficient due to their large space for energy-delay tradeoff. Harnessing such

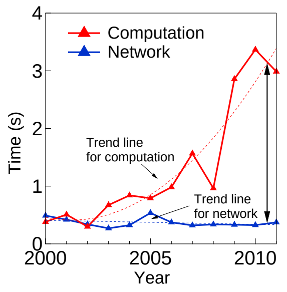

Figure 1: Increasing computational intensity of webpages.

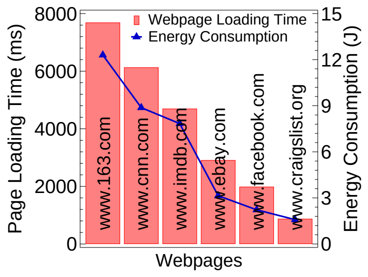

Figure 2: Webpage time and energy variances across websites.

heterogeneous systems requires us to exploit the inherent workload heterogeneity. Previous research has explored workload heterogeneity ranging from data centers [6] to desktop systems [5] with a special focus on exploiting task-level workload heterogeneity. Only more recently with the announcement of ARM's big.LITTLE architecture are we seeing heterogeneous architectures making a foray into the mobile processor market.

However, such heterogeneous systems have never been explored and deemed intractable for the mobile Web browsing workload due to its QoS sensitivity. In fact, the conventional wisdom has been to always supply the highest possible computational capability (i.e., voltage, frequency, core) to guarantee a satisfactory end-user browsing experience, which can lead to energy waste. We see the problem that the Web browser is typically treated as a stand-alone workload, such that the inherent workload heterogeneity is disregarded. We discover workload heterogeneity in the Web browser by treating individual webpages as the workload. Fig. 2 shows there is approximately 6X variance in webpage load time and energy consumption across six hot webpages, loaded on a ARM Cortex-A9 mobile processor. The presence of such workload heterogeneity makes heterogeneous systems strongly favorable for energy-efficient mobile Web browsing.

Through characterization of the workload heterogeneity, we take the position that *meeting the Web QoS requirement does not automatically necessitate burning excessive energy*. The key insight is to dynamically match the underlying heterogeneous hardware resources with different complexities in the webpages. We demonstrate a promising outcome of such insight on a big/little heterogeneous system through real software and hardware measurement. We show that dynamically selecting the big core or higher frequencies for complex webpages, and the small core or lower frequencies for simpler webpages leads to significant energy-efficiency improvement.

[*]The original article is "High-Performance and Energy-Efficient Mobile Web Browsing on Big/Little Systems" by Yuhao Zhu and Vijay Janapa Reddi, published in HPCA 19. It is also available for download at https://webspace.utexas.edu/yz4422/pubs/hpca13.pdf

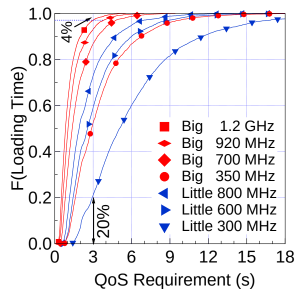

Figure 3: CDF for all configurations loading 5,000 webpages.

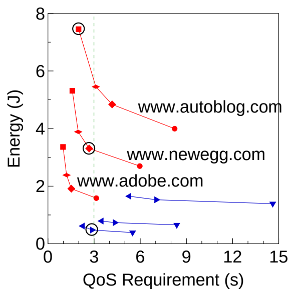

Figure 4: Energy vs. QoS tradeoff for three hot webpages.

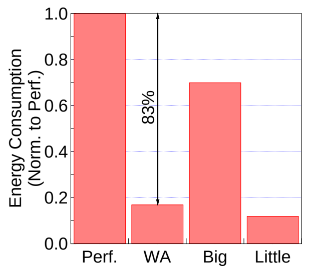

Figure 5: Energy savings.

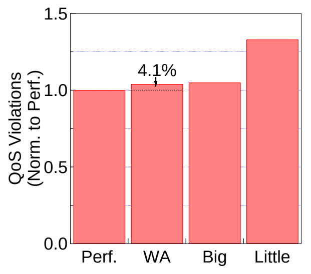

Figure 6: QoS violations.

2. Heterogeneous Systems for Web Browsing

We find a strong amenability of the big/little heterogeneous system to the mobile Web browsing workload. In particular, a heterogeneous system provides different computational capabilities (i.e., core and frequency configurations) for different webpages to *minimize the processor's energy consumption while still meeting the QoS requirement*. We demonstrate this on a big/little system with an ARM Cortex-A9 and A8 processor, each capable of performing DVFS (1.2 GHz, 920 MHz, 700 MHz, and 350 MHz on A9; 800 MHz, 600 MHz, and 300 MHz on A8). The Cortex A9 is a three-issue, out-of-order processor, and regarded as the big core. The Cortex A8 is a dual-issue, in-order processor, and regarded as the little core.

The benefits of the big/little system stem from our observation that different architecture configurations exhibit different capabilities to meet the QoS requirement. Fig. 3 shows the cumulative distribution functions of all seven configurations loading the hot 5,000 webpages listed in `www.alexa.com` using Mozilla Firefox. We observe a great dispersion across configurations. For example, assuming the 3-second QoS tolerance rule commonly used for loading webpages [4], only 4% of the webpages violate the QoS under 1.2 GHz on the big core. In contrast, 300 MHz on the little core fails to deliver the 3-second requirement for nearly 80% of the webpages.

Such a pronounced difference in computational capability leads to a wide range of energy-QoS tradeoffs. For example, Fig. 4 shows the energy-QoS distribution of three hot webpages under different architecture configurations. The seven configurations cover a QoS range from 2 seconds to almost 15 seconds, where the energy consumption ranges from less than 1 Joule to almost 8 Joules. The large range indicates the capability of a heterogeneous system to budget the energy consumption according to the various QoS constraints.

On one hand, under a certain QoS requirement–say, 3 seconds–the big/little system can flexibly provide configurations that minimize the energy consumption for different webpages. For instance, `www.autoblog.com` is a complex website, and it requires 1.2 GHz on the big core, which, however, is over-pumped for simpler websites such as `www.newegg.com`, for which 700 MHz on the big is sufficient. Running on the latter configuration results in 40% lower energy consumption than in the best performance mode. Similarly, `www.adobe.com` only requires 600 MHz on the little core, which saves 85% energy as compared to the performance-oriented 1.2 GHz on the big core. Both are significant savings.

On the other hand, the big/little system also flexibly adapts to varying QoS requirements expected by end-users. Typically, users expect lower QoS when mobile devices are in the battery-saving mode, in which case saving energy is more important than pure webpage load time speed. With a more relaxed QoS requirement such as 10 seconds, the big/little system can adapt to and provide the power-efficient little core configurations for all three webpages. In contrast, to meet a higher QoS expectation, such as 2 seconds, various big core configurations are utilized to deliver the QoS requirement.

3. Energy-Conscious QoS with Heterogeneity

We implement a webpage-aware mechanism in the Mozilla Firefox engine to harness our observations. We first mine the hottest 5,000 webpages and show that webpage load time and energy consumption are strongly correlated with various characteristics of webpage primitives (e.g. HTML and CSS). We then derive regression models to predict webpage load time and energy consumption based on webpage primitives. Our models can be executed within 1% of the entire webpage loading. The load time and energy consumption models have 5.7% and 6.4% median prediction error rates, respectively.

We apply the derived prediction models for different core and frequency configurations. On the basis of these predictions, our dynamic mechanism selects the configuration for each webpage to minimize the loading energy consumption while still meeting the specified QoS requirement.

We evaluate our mechanism on real hardware (Cortex A9 and A8) and software (Mozilla Firefox) for the 5,000 webpages. Fig. 5 and Fig. 6 shows that compared to using the *typical* QoS-oriented high-performance mode (Perf.), i.e., loading all the webpages at 1.2 GHz on A9, our webpage-aware mechanism (WA) achieves 83.0% energy savings while slightly impacting QoS for only 4.1% webpages. We also compare to two other schemes that utilize the existing OS-level DVFS scheme running on the big core (Big) and little core (Little) individually. Since the OS-level DVFS is agnostic to webpage characteristics and QoS requirements, our mechanism achieves significant energy saving and have much less QoS impact as compared to Big and Little.

4. Significance for Architecture and the Web

The mobile Web has become an integral part of our society. Increasingly, we are surrounded by interactive Web applications that pose a high demand for QoS because they are latency sensitive. Delivering low-latency services that meet end-users' QoS expectations could soon become the new measure of a system's performance and energy efficiency. For instance, if a user abandons a webpage just one second short of completion, then all the energy consumed by the device's resources, including CPU and display, are a waste of the limited energy.

There is a need for us to design a high-performance and energy-conscious mobile computing system for future Web applications. Computer architects have long been good at building latency- and throughput-oriented systems that have a continuous power source. However, optimizing for latency is significantly harder in the context of mobile systems due to the battery-constrained nature of these devices. Simply throwing additional hardware resources will not solve the problem.

The Vision To achieve the objective of building a high-performance and energy-conscious mobile Web computing system, we must develop a synergistic approach that spans the boundaries of application, runtime system, and energy-efficient hardware. To that end, understanding the performance and energy characteristics of the fundamental building blocks in Web technologies can provide valuable insights into how we can meet the mobile QoS requirements. In the original paper, we demonstrated how application knowledge (i.e., HTML and CSS) can be leveraged by the Web runtime system to intelligently manage the heterogeneous hardware resources.

We believe our study of Mozilla Firefox browser and big/little heterogeneous system can motivate further research along this direction because the Web browser engine is the substrate for a wide spectrum of Web applications. Many Web applications are developed using HTML, CSS, and JavaScript and are wrapped by a native "shell." Internally, the applications leverage the Web browser engine (e.g., Firefox and WebKit) for processing and rendering contents. Examples include the iOS e-mail client Mail, Walgreens, and Banana Republic applications on both iOS and Android. On iOS, Apple provides Object-C APIs that expose browser features for easy integration of the browser engine into mobile Web applications.

Predictable Web Performance and Energy We demonstrate that the webpage load time and energy consumption can be predicted by inspecting the characteristics of fundamental Web primitives, such as HTML and CSS, at the webpage level. We show that different HTML and CSS elements have different processing overhead and energy implications. We also correlate the performance and energy characteristics of these HTML and CSS primitives with the load time and energy consumption of the entire webpage, which can readily extend to other Web applications built using the Web technologies.

We see a wide range of future applications for our prediction approach. First, Web developers can use our predictive models to reduce the rate of application abandonment by improving interactivity and load times. Web developers can use the predictive models to gather quantitative and fine-grained optimization results before deployment. Second, our work is the first step in the process of integrating prediction and scheduling techniques into a mobile device with significant room for improvement. We can extend our scheme for system-level predictions with a more comprehensive and holistic understanding of the interactions among the different Web browser components and underlying hardware resources. Finally, future mobile Web browsers can optimize their execution on the basis of a Web application's characteristics and their users' unique requirements. A browser is like a traditional runtime system that continuously orchestrates its hardware resources to maximize some user-specific optimization objective.

Bridging Energy Efficiency and Soft-QoS The Web offers an interesting trade-off space between performance, energy consumption, and user expectation. Specifically, we find that mobile Web applications do not have hard cut-off latencies, entailing a unique notion of energy efficiency. For example, different end users have different expectations of loading a webpage. As Fig. 4 shows, an "energy-efficient" system configuration for one user may fail to deliver the QoS expectation of another user, becoming an energy-inefficient system. Similarly, a system optimally designed for one webpage or web application may over-provision performance for another application, effectively wasting energy. Our analyses shows that no one fixed design is suitable for all cases. We must integrate such end-user QoS experience into the system design/optimize loop. In contrast, conventional workloads have hard QoS metrics that are easily quantifiable. A transaction either completes or does not when the deadline expires. Optimizing energy efficiency under such circumstance is simply minimizing energy consumption within a fixed deadline.

Toward Application-Aware Heterogeneity Heterogeneous systems have long been known to be energy efficient due to the flexibility in adapting to different energy-performance trade-offs. However, the key to maximize their utility is to combine the heterogeneous system with application-level or domain-specific knowledge. In our paper, we demonstrate this on one form of heterogeneous system that has both big and little cores. In the long term, our results encourage designing Web workload-specific architectures. Integrating such energy-efficient architectures into the system can increase system heterogeneity and allow for more flexible scheduling to meet the mobile Web's demanding QoS requirements.

References

[1] Everyone is Carrying Too Many Mobile Devices. http://goo.gl/FaZgIZ

[2] How 1s Could Cost Amazon $1.6 Billion in Sales. http://goo.gl/jJO1l

[3] Fact Sheet: Gomez Mobile Monitoring. http://goo.gl/TW8HUr

[4] RD2: "The three second rule". http://goo.gl/pynBl

[5] T. Cao *et al.*, "The yin and yang of power and performance for asymmetric hardware and managed software," in *Proc. of ISCA*, 2012.

[6] M. Ferdman *et al.*, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," in *Proc. of ASPLOS*, 2012.