The Role of the CPU in Energy-Efficient Mobile Web Browsing

The mobile CPU is starting to noticeably impact Web browsing performance and energy consumption. Achieving energy-efficient mobile Web browsing requires considering both CPU and network capabilities. Researchers must leverage interactions between the CPU and network to deliver high mobile Web performance while maintaining a low energy footprint. Designing future high-performance and energy-efficient mobile Web clients implies looking beyond individual components and taking a full system perspective.

Yuhao Zhu
Matthew Halpern
Vijay Janapa Reddi
University of Texas at Austin

Web technologies have transformed society, shaping how we think, communicate, and innovate. Recently, the Web has entered a new age that automatically recognizes, mines, and synthesizes user-specific information. The driving force behind this Web evolution is the ubiquity of mobile devices—today's most pervasive personal computing platform. The integration of mobile and Web enables strong connectivity, interactivity, and personalization, and is the focus of this article.

Mobile Web performance has significantly improved over the past several years. We demonstrate this improvement over six smartphone generations, each representing a top smartphone from 2009 to 2014. (See the next section for details.) Each smartphone runs the Google Chrome browser (version 33.0) downloaded from Google Play, and loads eight hot websites from the year corresponding to the device, which we obtained from Internet Archive (http://archive.org/web/web.php) as a representative Web snapshot of that year. Figure 1a shows the average webpage load time on the *y*-axis. Overall, webpage load time has improved from almost 40 seconds in 2009 to less than 5 seconds in 2014, which is effectively an 8× improvement in performance.

However, generational performance improvements began to slow down. Figure 1a shows that after 2011, the webpage load time improved only marginally. Meanwhile, smartphone power consumption has continued to steadily increase. Figure 1b shows the average device power consumption of the six smartphones, measured at the battery rail, while loading the webpages. We observe a 3× power increase over the past six years. The marginal performance improvement coupled with the steady increase in power consumption has resulted in the slowing down of energy consumption improvement. Figure 1a shows that the energy reduction plateaued out beyond 2011.

In this article, we answer the following question: what is the role of the mobile CPU in enabling energy-efficient Web browsing? We examine the two critical components that

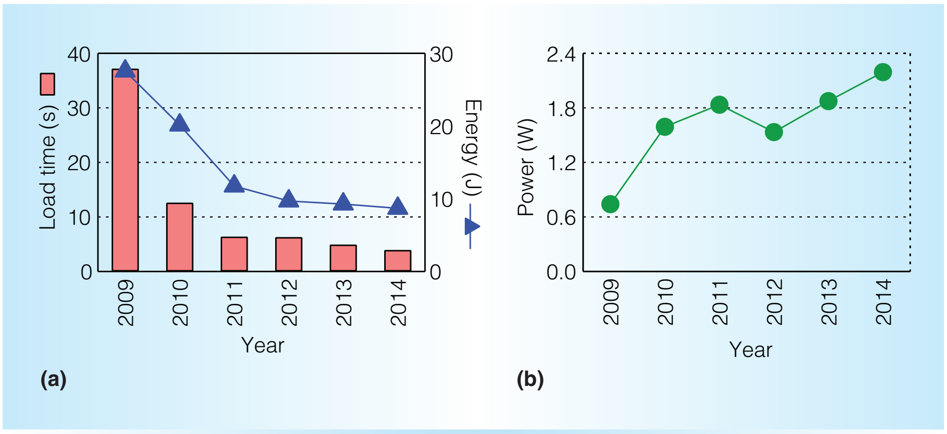

Figure 1. Average performance, power, and energy consumption trends from 2009 to 2014 for eight of the top websites, ranked according to Alexa. We pick one of the best-performing smartphones from each year to load all of the eight webpages' archived images from Internet Archive. (a) Performance and energy trend. (b) Power trend.

affect the energy efficiency of mobile Web browsing: the CPU and the network. Conventional wisdom suggests that network capability is the primary bottleneck in mobile Web browsing. A great portion of the research has focused on improving the network,[1-3] but little work has been done on understanding how the CPU impacts mobile Web browsing.

We find that the CPU's impact on mobile Web browsing energy efficiency strongly depends on the network latency. Under low network latency, such as in an LTE cellular network, the best system energy efficiency is achieved under high CPU performance, which leads to significantly faster webpage load time while simultaneously conserving energy. In contrast, under an adverse connectivity with high network latency (for example, 3G), the webpage load time is insensitive to the CPU performance. Low CPU performance makes the system energy efficient by significantly saving energy without degrading Web browsing performance.

From these observations, we quantitatively demonstrate that existing dynamic voltage and frequency scaling (DVFS) governors in the OS are inadequate in deciding the CPU performance, because they do not consider the network latency, and therefore often lead to suboptimal energy-efficiency operating points. We describe a CPU-network interaction-based approach to adjust the CPU's voltage and frequency setting according to the operating network latency. Our findings suggest that designing future high-performance mobile Web browsers will require us to look beyond optimizing individual resource components and take an integrated full-system perspective.

Considering the past and present

Our study takes a historical perspective to understand Web performance trends, bottlenecks, and energy implications. Here, we introduce our investigative methodology for conducting such a study with representative smartphone and Web technology snapshots.

Smartphones

To study evolving CPU performance, we selected six top smartphones, each representing cutting-edge smartphone technologies for each year from 2009 to 2014. These are Motorola's Droid from 2009, and Samsung's Galaxy S, Nexus, and S3, S4, and S5 from 2010 through 2014, respectively. Each smartphone houses a CPU that differs in microarchitectural design, peak clock frequency, and number of cores. Chronologically, the six phones reflect how CPU processing capabilities have progressed over time. We used the Monsoon power monitor to measure the six smartphones' battery-level energy consumption.

Network

Network performance is typically evaluated in two metrics: latency and bandwidth. Prior work has shown that in the mobile context, network latency—typically evaluated by round-trip time (RTT)—has a much more significant impact than network bandwidth.[1,4] Therefore, we focused only on the latency aspect of network performance.

To study the impact of network latency on various cellular network generations, we hosted all the webpages on a Web server and manually injected delay into the server. We then used Wi-Fi on the smartphones to access the webpages. Because Wi-Fi has significantly lower latency than the current 4G/LTE network, the delay injection let us mimic a wide range of network latencies. This methodology is a well-established technique to control cellular network latency.[4] Although Wi-Fi has different behaviors compared to the cellular network,[2] our study focuses on network latency in general, rather than specific mobile network technologies.

Workloads

We carefully selected webpages that represent the evolution of the Web workloads. In particular, we used the webpage snapshots from Internet Archive to represent webpage evolution throughout the years. We mined through the top 10,000 websites as ranked by Alexa (www.alexa.com) and identified eight representative websites. Prior work has used these eight websites,[5] and they were statistically shown as representative. The eight websites are hot landing websites ranked among Alexa's top 40. They are CNN, Amazon, ESPN, Google, YouTube, 163, MSN, and Slashdot.

We considered not only the mobile version of the eight websites, but also their desktop counterparts, because many mobile users still prefer desktop-version websites for their richer content and experience. Moreover, many mobile devices, especially tablets, typically load the webpage's desktop version by default. As webpage sizes keep increasing, we must understand the performance and energy implications of complex webpages and not just simple mobile webpages.

The CPU's impact on mobile Web performance

Conventional wisdom suggests that mobile Web browsing performance is primarily limited by the network latency. In this section, we explain how the CPU impacts mobile Web browsing performance. We first perform a bottleneck analysis between the CPU and the network to show that webpage load time is sensitive to CPU performance under low network latency, such as the current LTE cellular technology. We then perform a combined study, varying both network latency and CPU performance simultaneously, to understand how mobile Web browsing performance is affected by CPU performance under different network latencies.

CPU versus network bottleneck analysis

To understand how the CPU and network impact the mobile Web browsing performance, we must first understand the mechanism of how browsers load webpages through the interaction of CPU computation and network access. Modern browsers, such as Chrome and Firefox, use an asynchronous execution model between CPU and network access.[4,6] CPU computation and network access are overlapped for performance optimizations. For example, the CPU doesn't wait for the network to return an image file; instead, it continues to process the rest of the webpage that is independent of the image. Such an asynchronous execution model implies that improving the performance of only one component will eventually make the workload bounded by the other component.

With this understanding, we now quantify the impact of the CPU and network on the webpage load time. We experimentally compare how the webpage load time varies with different CPU performances and network latencies on today's high-end Galaxy S5. We use the delay injection technique described earlier to statically hold the network latency at 100 ms (a typical RTT in an LTE network) while investigating the CPU, and hold the CPU performance at its highest frequency while studying the network, to isolate each component's effect.

We first focus on the network. Figure 2 shows the webpage load time with respect to different network latencies. We superimpose the figure with different mobile network technologies' typical latencies derived from both technical specifications as well as real measurements in the field.[1,7] We observe that reducing the network latency from an adverse 3G connection at 2,000 ms to an LTE connection at 100 ms results in a 9.5× speedup in webpage load time from 38 to 4 seconds. As the network latency further improves within the range of LTE network latency (50 ∼ 100 ms), the network latency has only a marginal impact on the overall webpage load time. This is because at this point the fast network accesses are hidden behind CPU computations in the asynchronous execution model; the application is largely CPU-bound. Further reducing the network latency from LTE to Wi-Fi has almost no effect.

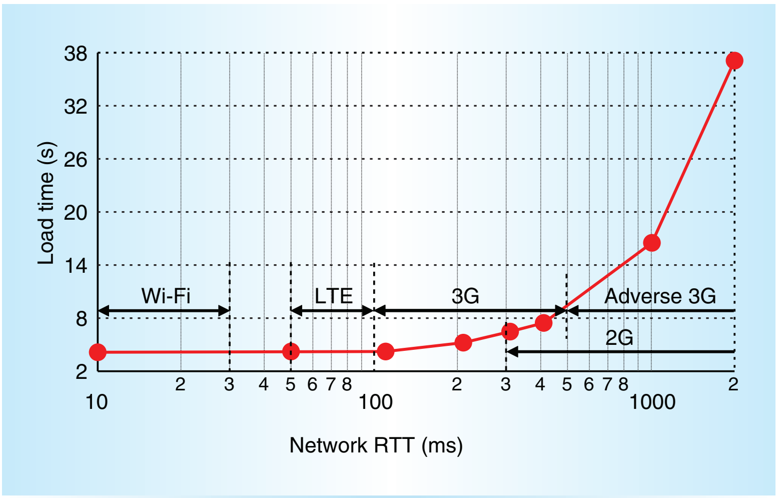

Figure 2. Average webpage load times with respect to changing network latency. Each marker corresponds to a round-trip time (RTT) value. We superimpose the RTT range for different cellular technologies to show that each technology has a spectrum of performance characteristics.

As the network latency becomes low (that is, small network RTT times), the CPU performance starts playing a significant role in the mobile Web browsing performance. To study how the CPU performance affects the webpage load time, we mimic a wide range of CPU performance capabilities by leveraging the S5's 14 frequency settings. Note that we use frequency only as a proxy for CPU performance; we do not intend to study the impact of a particular CPU's frequency itself. Figure 3 shows how webpage load time changes with CPU performance under a 100-ms RTT (LTE-like network connectivity). As the CPU frequency decreases from the highest to the lowest by about 6× (2.5 to 0.4 GHz), the webpage load time slows down by as much as 4.5×, from 4 seconds to about 18 seconds, indicating strong sensitivity to CPU performance.

An increase in the clock frequency between 1.6 and 2.4 GHz yields small performance benefits. One could then naively conclude that mobile CPU performance improvements yield marginal improvements in Web browsing performance. However, the marginal improvement is merely an artifact of using frequency as a performance proxy. At high frequencies, the processor's pipeline is already saturated with work, and the memory and interconnection become the microarchitectural-level bottlenecks.[8]

To overcome this artificial constraint, we perform the same experiment on a desktop CPU (Intel core i5 at 1.2 GHz) and assess the impact of future mobile CPU improvements on Web browsing performance. The average webpage load time on the desktop CPU is about 1 second, effectively a 4× speedup over the S5's peak performance. Mobile CPU performance today is still far from reaching a diminishing return point, and it can continue to have a significant impact on mobile Web browsing performance.

To attest to the fact that CPU performance has consistently contributed to overall mobile Web browsing performance improvement, we also take an evolutionary perspective and compare the five earlier smartphones released from 2009 to 2013 with the S5, which came out in 2014. Each of the earlier smartphones possesses a peak performance equivalent to a certain S5 frequency. For example, the S has a peak frequency of 1 GHz, whose performance is equivalent to the S5 at about 650 MHz. We superimpose the webpage load time and the equivalent S5's frequency of earlier smartphones in

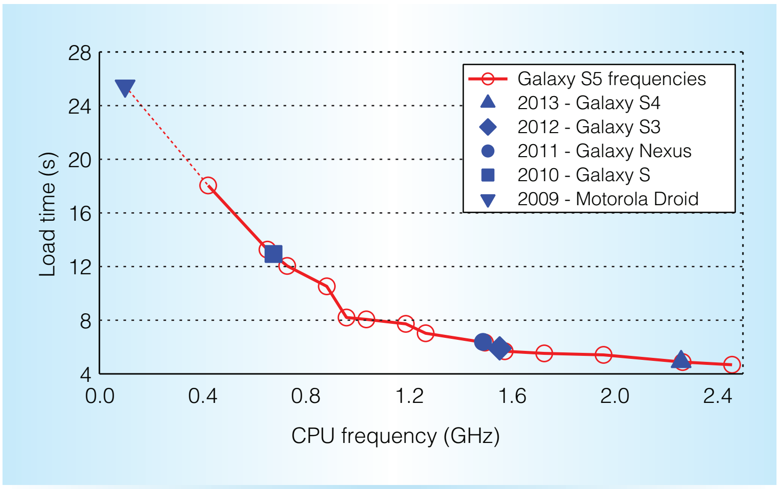

Figure 3. Webpage load time with respect to CPU frequency on a Galaxy S5 smartphone. Empty markers represent CPU frequencies (from 0.4 to 2.5 GHz). Each dark marker represents the best phone from 2009 to 2013; each phone has peak performance that corresponds to an equivalent S5 frequency. The Motorola Droid's time exceeds the S5's lowest frequency; therefore, we extrapolate the curve.

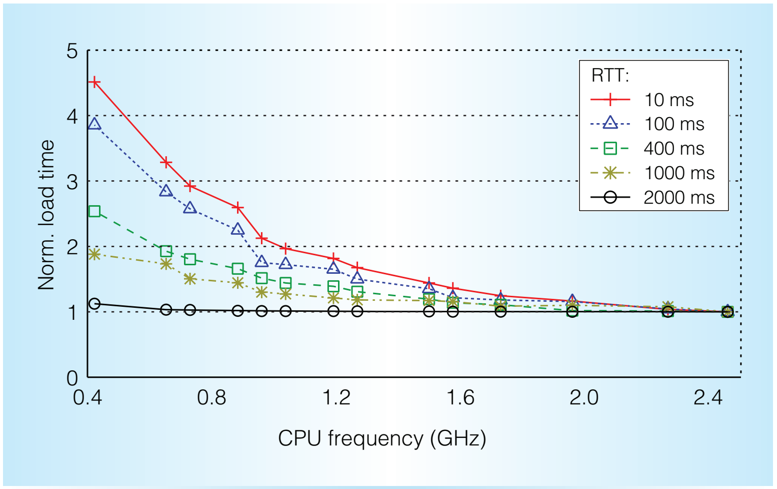

Figure 4. Average webpage load times with respect to CPU frequencies, operating at different network RTTs. Each line corresponds to a unique RTT value, and the markers correspond to different CPU frequencies. The webpage load time at different frequencies under a particular RTT is normalized to the lowest load time under that RTT.

Figure 3. Comparing the peak performance of the Motorola Droid from 2009 and the S5 from 2014, six years of CPU innovation can decrease the webpage load time by 6×, from more than 24 seconds to 4 seconds.

The takeaway from our observations is that the continuous improvement to network latency will eventually take us to a point where further Web performance improvement will be unattainable without improving CPU performance. We are not suggesting that network latency is no longer relevant. Instead, our data leads us to conclude that under low network-latency conditions, the CPU plays a vital role in mobile Web browsing performance. When the network deviates from an ideal low-latency network condition, high-latency network access is prevalent, and under such circumstances mobile Web performance is indeed constrained by network performance.

CPU–network interaction

Having identified the CPU's impact on mobile Web performance at a low network-latency range, we must also understand how that impact changes under various network latencies. This is because network-latency variation in the real world is large.[9] For example, field measurements show that network latency can be anywhere between 100 ms to more than 20 seconds in a 3G network depending on the time of day and the location.[10] The difference across mobile carriers exacerbates the latency variation issue even further. Here, we show that the network latency can strongly affect the mobile CPU's impact on the Web browsing performance. Such an interaction between CPU and network casts strong energy-efficiency implications.

We use Figure 4 to show how network latency affects the CPU's impact on webpage load time. We continue to use the Galaxy S5 with Wi-Fi for our experiments and manually inject delay into the Web server, as we explained earlier. Each line represents a particular network latency, ranging from 10 to 2,000 ms. The *y*-axis shows the webpage load time normalized to the best performance (that is, using the highest frequency) of each particular network latency.

When we compare the different RTT trend lines, at higher network latency, the corresponding line has a sharper curve, indicating that the CPU performance has a larger impact on webpage load time. For example, at an adverse 3G network where the latency is about 1,000 ms, loading the webpage at the lowest frequency (0.4 GHz) leads to only

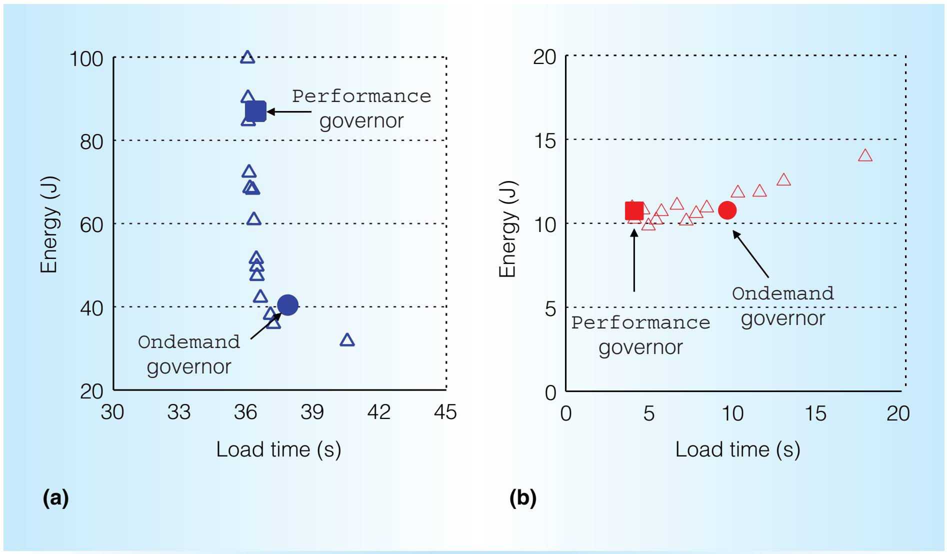

Figure 5. Load time versus energy consumption at different frequencies (empty markers), operating under two network RTTs: (a) 2,000 ms and (b) 100 ms. At each RTT, the CPU frequency decreases from 2.5 to 0.4 GHz from left to right. We also show the load time and energy consumption of two OS governors: `performance` and `ondemand`.

a less than 2× slowdown as compared to running the highest performance (2.5 GHz).

In contrast, under a 100-ms latency, typical for LTE, the lowest frequency leads to almost 4× slowdown in performance. On the other hand, as network latency increases, the corresponding curve becomes flatter, indicating that webpage load time becomes less sensitive to CPU performance. For instance, Figure 4 shows that CPU performance has little impact on webpage load time at a 2-second network latency.

The Mobile CPU's energy-efficiency implications

We study how the CPU impacts the energy consumption of mobile Web browsing, and how such an impact varies with different network latencies. Such interactions between the CPU and network shed light on energy-efficiency optimization opportunities. On the basis of these observations, we describe an approach to coordinate mobile CPU performance with network latency to optimize Web browsing energy efficiency (that is, saving energy without compromising performance). Our analysis, based on oracle knowledge, suggests that such an approach can provide better energy efficiency over traditional OS DVFS governors that currently do not leverage the interplay between the CPU and network.

Energy-efficiency optimization opportunities

Because the CPU's impact on Web browsing performance varies with different network latencies, studying the CPU's energy consumption impact also requires us to consider various network latencies. For simplicity without loss of generality, we consider two network scenarios: high network latency with RTT of 2,000 ms, such as with adverse 3G connectivity; and low network latency with RTT of 100 ms, such as with good LTE cellular connectivity. Figures 5a and 5b show the total device energy consumption against various CPU frequencies under the two network latencies, respectively. Within each network latency, each empty marker represents a CPU frequency that decreases from 2.5 to 0.4 GHz from left to right.

Under adverse network connectivity (such as 2,000 ms latency), webpage load time is extremely insensitive to CPU performance (see Figure 4). Therefore, there is an opportunity to exploit slack, meaning that during slow network performance there is no need for high CPU performance. We can lower the CPU's operating frequency, and consequently reduce the voltage, and thus reduce device power consumption without causing any noticeable performance impact. Figure 5a shows that using lower frequencies causes little performance degradation compared to the highest frequency, while achieving at most 70 percent energy savings.

Under a well-behaved network (such as a 100-ms network latency), however, the webpage load time is sensitive to CPU performance, as Figure 4 shows. As a result, Figure 5b shows that using the highest frequency loads the webpages about 4× faster and consumes more than 20 percent less energy than using the lowest frequency.

Our results indicate that the mobile CPU casts two implications on Web browsing energy efficiency. First, under high network latency, lowering CPU performance is more energy efficient for the overall system, because lower CPU performance can achieve significant energy savings with little performance degradation. Second, under low network latency, provisioning high CPU performance is more energy efficient for Web browsing because it avoids excessively long webpage load time and therefore consumes less energy.

Current OS governors

Current mobile operating systems do not consider network connectivity, and therefore do not always choose the ideal CPU performance for energy management. We measure the webpage load time and energy consumption of two OS DVFS governors commonly adopted in Android, `performance` and `ondemand`, taken from the Android CPUFreq Governor (http://goo.gl/8pkyri), and show the opportunity to improve them. Comprehensively evaluating a new OS DVFS governor is beyond the scope of this article, especially because our focus is to characterize the CPU's role in enabling energy-efficient mobile Web browsing.

The solid markers in Figures 5a and 5b show the measured results of the two governors under the network latency of 2,000 ms and 100 ms, respectively. The `performance` governor statically sets the frequency to the highest, and therefore guarantees application interactivity. The Galaxy S5 uses the `performance` governor by default. The `ondemand` governor attempts to save energy by changing the CPU frequency according to CPU use. This governor is commonly adopted for conserving energy.

Under high network latency, Figure 5a shows that the `performance` governor biases toward high CPU performance, consuming excessive energy without providing much performance gain. Under low network latency, Figure 5b shows that the `ondemand` governor biases toward lower frequencies, and therefore loads webpages 2× slower than the highest frequency while consuming slightly higher energy.

We see the feasibility of new heuristics for future governor designs. For example, it is possible to design a hierarchical governor that first detects the network latency (RTT) when the first network request comes back, which typically takes less than 5 percent of the webpage load time, and selects the `performance` governor if the RTT is below a certain threshold, or the `ondemand` governor if the RTT is above the threshold. A more complex approach is to train offline models that predict the webpage load time and energy consumption under various network latencies,[11] and at runtime select the ideal frequency that minimizes the energy consumption without exceeding a certain load time threshold.

Although this article focuses on coordinating CPU processing capability with network performance, the general takeaway is that similar coordinated optimization opportunities can exist that extend beyond the CPU and include other major mobile components as well. Mobile systems on chips (SoCs) offer several other computation resources that can be adapted to network connectivity. For instance, many heterogeneous SoCs (such as Samsung's Exynos Octa) contain little cores that have lower performance than main cores but consume less energy. Loading webpages on the little core[11] in observance of high network latency would likely achieve an even lower energy consumption compared to simply throttling the main core's operating frequency, which is what we demonstrate in this article.

Moreover, contrary to adapting the CPU's performance to network latency, the network latency could also be adapted to the CPU's performance. Web performance is less sensitive to network latency on a slow CPU, and therefore under circumstances where the CPU is running at low performance, or simply on a low-end feature phone, it is possible to throttle the network latency, or even switch to a lower-generation cellular network connection to save energy.[2]

MICRO

References

1. I. Grigorik, *High Performance Browser Networking*, O'Reilly, 2013.
2. J. Huang et al., "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," *Proc. 10th Int'l Conf. Mobile Systems, Applications, and Services* (MobiSys 12), 2012, pp. 225–238.
3. J. Huang et al., "An In-Depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance," *Proc. ACM SIGCOMM*, 2013, doi:10.1145/2486001.2486006.
4. Z. Wang et al., "Why Are Web Browsers Slow on Smartphones?" *Proc. 12th Workshop Mobile Computing Systems and Applications*, 2011, pp. 91–96.
5. Y. Zhu and V.J. Reddi, "WebCore: Architectural Support for Mobile Web Browsing," *Proc. 41st Ann. Int'l Symp. Computer Architecture* (ISCA 14), 2014, pp. 541–552.
6. X.S. Wang et al., "Demystifying Page Load Performance with WProf," *Proc. 10th USENIX Conf. Networked Systems Design and Implementation* (NSDI 13), 2013, pp. 473–486.
7. "3G/4G Wireless Network Latency: How Do Verizon, AT&T, Sprint and T-Mobile Compare?" *Fierce Wireless*, 5 Nov. 2013; http://goo.gl/L9ieoy.
8. V. Agarwal et al., "Clock Rate Versus IPC: The End of the Road for Conventional Microarchitectures," *Proc. 27th Ann. Int'l Symp. Computer Architecture* (ISCA 00), 2000, pp. 248–259.
9. W3C, *Improving Web Performance on Mobile Browsers*, Nov. 2012; www.w3.org/2012/11/webperf-slides-greenstein.pdf.
10. J. Bixby, "How I Learned that 3G Mobile Performance is Up to 10× Slower than Throttled Broadband Service," *Web Performance Today*, 26 Oct. 2011; http://goo.gl/XuO8Of.
11. Y. Zhu and V.J. Reddi, "High-Performance and Energy-Efficient Mobile Web Browsing on Big/Little Systems," *Proc. IEEE 19th Int'l Symp. High Performance Computer Architecture* (HPCA 13), 2013, pp. 13–24.

Yuhao Zhu is a PhD student in the Department of Electrical and Computer Engineering at the University of Texas at Austin. His research focuses on improving the energy efficiency of mobile Web computing holistically, spanning processor architecture, runtime systems, and language extensions. Zhu has a BS in computer science and engineering from Beihang University. Contact him at yzhu@utexas.edu.

Matthew Halpern is a PhD student in the Department of Electrical and Computer Engineering at the University of Texas at Austin. His research interests include runtime systems and computer architecture for mobile computing. Halpern has a BS in electrical and computer engineering from the University of Texas at Austin. Contact him at matthalp@utexas.edu.

Vijay Janapa Reddi is an assistant professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin. His research interests include processor architecture and system software design and implementation for addressing performance, power, thermal, energy, and reliability issues for mobile and high-performance computing. Janapa Reddi has a PhD in computer science from Harvard University. Contact him at vj@ece.utexas.edu.