Research Statement

Yuhao Zhu

We are in a golden age of mobile computing. Over the past two decades, mobile computing has evolved from communicating simple messages to personal supercomputers that are capable of recognizing, mining, and synthesizing user-originated "big data." Coupled with the increasing capability, mobile computing is also becoming more ubiquitous than ever. It is estimated that 6 billion mobile devices currently exist, and that number will reach nearly 20 billion by 2020 with the recent surge of cyber-physical systems and the Internet-of-Things. A key challenge as mobile computing hits the critical mass is *energy-efficiency*. Energy-efficiency fundamentally limits the computation capability of mobile devices, and is a major roadblock to the vision of Green Information Technology.

My research focuses on the core problem of improving the energy-efficiency of mobile computing via architecting better and practical computing systems. Within the broad scope of mobile computing, my dissertation work specifically focuses on *mobile Web* computing motivated by Web technologies' wide adoption and long-term applicability. I believe that addressing the energy-efficiency issue requires that we break the traditional software and hardware boundaries and design systems that perform synergistic cross-layer optimizations. To that end, I have developed hardware accelerators [12], runtime mechanisms [11, 18, 10], and programming language extensions [13], the combination of which forms a hardware/software co-designed computing substrate that enables an energy-efficient mobile Web.

When I first started my dissertation work back in 2011, the conventional wisdom was that mobile computing was primarily bottlenecked by network communication as opposed to computation. As a result, there was no mobile computing paper published in major architecture conferences in that year. After six years, mobile computing has become a regular session, and energy-efficiency is the key theme in virtually all mobile-related papers. In addition to impact within academia, my work has also raised industry attention. I received the Google Ph.D. fellowship to pursue research in mobile computing. The *ACM Queue* magazine and *Communications of the ACM* recently invited me to write an article to introduce the cutting-edge research in mobile Web computing to industry practitioners [6].

Dissertation Research: Energy-Efficient Mobile Web Computing

The landscape of mobile computing has experienced a tremendous transformation over the past decade. A key enabler is the advancement in Web technologies, which provide a platform-independent way (i.e., "write-once, run-anywhere") for mobile applications to interact with the Internet and to manage local computations, greatly improving development and deployment productivity. The integration of mobile and Web, i.e., the mobile Web computing, is crucial for not only today's needs but also for ushering in the next era of computing where all devices will rely on Web connectivity.

Overview In mobile Web computing, one of the most important concerns is performance as it directly impacts end-user quality-of-service (QoS) and often has severe economic consequences[1]. While mobile Web performance was previously network-limited, I observed that *compute* has become dominant—a trend propelled both by cellular network technology advancements and new Web technologies that are increasingly compute-intensive (e.g., HTML5, WebRTC) in quest of richer, real-time capabilities [19]. Improving the mobile compute capability, however, is fundamentally constrained by the tight energy envelope of mobile devices. Conventional solutions to improving mobile compute capability have largely focused on the mobile CPU design, especially by applying energy-hungry, desktop CPU-like design techniques [4]. Such strategies are extremely energy-inefficient and are unsustainable as the graceful Dennard Scaling ended a decade ago and the Moore's Law soon stops in just a few generations without a practical alternative.

My work develops new ways to improve the energy-efficiency of mobile Web computing while unleashing the compute capability of mobile systems. Specifically, I have taken a principled approach to understand two complementary problems: when and how to make a calculated trade-off between performance and energy consumption, and when and how to reduce energy while improving performance at the same time. To address these issues, my research takes a holistic view of the mobile Web computation stack spanning the application, Web browser runtime, and processor architecture layers. I believe that improving the energy-efficiency of mobile Web computing requires us to enhance the traditional system interfaces with *new abstractions* and to leverage the new interfaces for *cross-layer optimizations*. As such, my research forged a set of new abstractions that expose energy optimization opportunities in mobile Web computing, and developed various optimization mechanisms that are effective in practice.

[1]Kit Eaton: How One Second Could Cost Amazon $1.6 Billion In Sales. http://goo.gl/qG0M2Q

Web Language Extensions. Today's Web programming languages mainly express structure, style, and functionality of an application. End-user QoS information, however, is largely unaccounted for. Without the QoS information, the underlying Web runtime (typically a Web browser) may make uninformed decisions that lead to severe QoS violations or energy waste. I designed GREENWEB, a set of Web language extensions that let Web developers express user QoS expectations as compiler pragma-like directives [13]. The philosophy behind GREENWEB is that developers provide minimal yet vital QoS information to guide the runtime energy optimizations. Empowering a new generation of energy-conscious Web application developers necessitates new programming abstractions at the language level. GREENWEB proposes two new language abstractions, QoS type and QoS target, to capture the critical aspects of user QoS experience. With the developer-assisted QoS information, the runtime substrate of GREENWEB could then dynamically determine how to ensure the target QoS specifications while minimizing the energy consumption.

Historically, the Web community has a long tradition of providing language extensions to guide *performance* optimizations, link prefetch being a prime example. GREENWEB is the first Web language support that specifically targets *energy-efficiency*. Ironically, major Web platform vendors have not been very enthusiastic in supporting previous language hints due to concerns about adversarial attacks where malicious developers intentionally place hints that lead to inefficient system decisions. Critically, GREENWEB demonstrates that the alternative to language hints is not necessarily no hints at all; rather we must carefully design the language abstractions such that the underlying runtime can intervene and reject application behaviors that are deemed harmful or undesired. Specifically, GREENWEB by design let developers only express QoS expectations without direct control over energy consumption. The runtime retains the control over system behaviors by deciding how to deliver the QoS in an energy-efficient manner.

Energy-Efficient Mobile Web Runtime. While GREENWEB empowers developers to guide the underlying runtime system to save energy, it does not impose any constraints on *how* the runtime should save energy. I designed a mobile Web runtime called WEBRT that takes advantage of an emerging hardware platform—asymmetric chip-multiprocessor (ACMP). ACMP is an energy-efficient heterogeneous architecture that is widely adopted by today's mobile hardware vendors such as Samsung, Qualcomm, and just recently Apple. The heterogeneities of ACMP, including different core types and frequency settings, are exposed as new hardware abstractions to WEBRT, which then intelligently provisions hardware resources to consume "just enough" energy while meeting user QoS expectations.

The key insight of WEBRT is that mobile applications are event-driven, and thus energy optimization is most effective when conducted at an event granularity. Accordingly, the nugget of WEBRT is the event-based scheduler (EBS) that dynamically schedules event executions on the ACMP hardware to save energy. EBS has two key components that exploit different event characteristics. The first one targets events that occur repetitively during a usage session (e.g., finger swiping) [18]. It takes an adaptive approach that tunes the scheduling decision based on previous occurrences of the same event. The other one targets events that occur once per usage session (e.g., application loading) [11, 10]. It uses a proactive mechanism that makes the scheduling decision purely based on characteristics of the incoming event.

The event-based scheduler in WEBRT is a fundamentally new scheduling mechanism, analogous to the canonical thread-based scheduler in an Operating System or a memory scheduler in a memory system. EBS is unique in capturing the event-driven nature of mobile applications and in optimizing for the objective of minimizing energy under QoS specifications. I integrated the EBS-based WEBRT into Google's Chrome Web browser engine, and achieved 30% energy savings as compared to Android's default scheduling policy. As event-driven is an established, fundamental execution paradigm for mobile applications, EBS has a long-term applicability in energy-efficient mobile computing.

Web-Specific Mobile Processor Architecture. While GREENWEB and WEBRT demonstrate the benefits of making trade-offs between performance and energy consumption, next-generation "always-on" mobile applications will demand a much higher compute-intensity. Therefore, it is important to design mechanisms that can simultaneously improve performance and energy consumption. Domain-specific architectural specialization has long been deemed as one such approach. The key challenge to specializing for Web computing, however, is to retain *general-purpose programmability*. The general-purpose programmability is an absolute necessity to the Web because the Web is constantly evolving with emerging software frameworks and changing user-experience expectations. Sacrificing the programmability of the Web would hurt the Web as a universal mobile development platform in the long run.

I designed WEBCORE, a general-purpose processor architecture specialized for the mobile Web [12]. WEBCORE strikes a balance between general-purpose programmability and specialization efficiency. Specifically, WEBCORE uses a general-purpose processor as the baseline, thus retaining the programmability. On top of the general-purpose baseline, WEBCORE incorporates specialized hardware to improve energy-efficiency and performance. WEBCORE's specializations include a hardware accelerator for critical computation kernels as well as fast, low-power scratchpad memories that capture the unique data locality of Web applications' principal data structures (e.g., the DOM tree).

The significance of WEBCORE is that it identifies the Web as a key application *domain* to customize and specialize for in mobile computing. As we enter the dark silicon era, one important question to address is to identify application domains that benefit the most from dedicated hardware resources and design efforts. In retrospect, software kernels that made their way into today's hardware have had a strong usage base and long-standing impact. WEBCORE identifies the Web computation stack as a lucrative target for specialization in mobile computing, and takes a principled approach to provide a forward-looking mobile processor substrate in anticipation of future compute-intensive mobile applications.

Beyond Mobile: Managed-Language-based Event-driven Server Design

In my last two years of graduate study, I noticed a unique confluence of managed languages and event-driven programming model in today's cloud applications that pose interesting challenges to future server systems. Using the popular *Node.js* framework as a case-study, I investigated two specific issues regarding microarchitecture and runtime design.

I found that today's server processor microarchitecture is ill-designed for managed-language-based, event-driven applications. This is because such applications rely on repeated, but flat event callback functions that result in large instruction reuse distances, thus violating a fundamental assumption in microarchitecture design: code locality. Critically, I observed that the lost code locality can be regained by exploiting the ample code reuse across event callbacks. Accordingly, I designed a microarchitectural mechanism that orchestrates the instruction cache insertion policy with the instruction prefetcher to capture the unique inter-event code reuse pattern in such applications, and thus improves the overall performance [16]. The observations are independently validated by Intel engineers, and the workload suite that I constructed for this research [1] is adopted by Intel as a start point for benchmarking event-driven servers.

I recently mentored a junior graduate student to study the tail latency issue in *Node.js*. We developed a latency analysis framework called Event Dependency Graph, which let us identify Garbage Collection (GC) as a major source of tail latency. We then designed a GC-oriented runtime optimization mechanism that reduces the tail latency by 40%.

GPGPU Programming and Microarchitecture. My prior research focused on enabling modern graphics processing units (GPUs) for general-purpose computing. I researched at both the software layer (parallel EDA algorithms [3, 15, 9]) and the hardware layer (IP router microarchitecture [20, 8]).

Future Research Directions

The computing industry is experiencing a renaissance with a key theme of leveraging machine learning techniques to create values from big data generated by massive numbers of Web-connected devices. I am excited to continue my research in computer architecture within this context, with a focus on rethinking the roles and goals of computing systems as well as realizing the new roles and goals by designing practical hardware and software prototypes. I explained my research vision in two recent invited position articles [17, 14]. Below, I highlight a few exciting directions.

From Mobile Computing to Edge Computing. While today's mobile devices are more or less a portal to powerful data centers in the cloud where most of the computations are carried out, I believe that a great portion of the compute work will soon be shifting toward network edges devices, ushering in the era of the so-called *edge computing*. Edge devices include both end-user mobile devices (e.g., wearables, sensors) as well as network gateways and routers that are just one hop away from end-users. These devices are uniquely suited for many application scenarios because they are much more energy-proportional than data centers, have low-latency, high-bandwidth access to massive raw data, and are better at preserving user privacy by keeping data local and avoiding information transfer altogether [17].

Edge computing is closely related to mobile computing with unique challenges. I see many opportunities to use my experience with cross-layer mobile computing research to address open research questions in edge computing that will inevitably cross the hardware/software boundaries:

- My work on WEBRT suggests the benefits of exposing hardware details to the runtime system design. However, future edge hardware will be extremely complex, with tens, if not hundreds, of vendor-customized and application-specific hardware Intellectual Property (IP) blocks (e.g., sensors, beacons, accelerators). Today's mobile runtime systems such as the Web browser are unscalable to manage the hardware complexity because it is a monolithic design that appeals to all existing IP blocks. For instance, whenever a new sensor is designed, the Web platform is extended with a new set of APIs to support it. The code size, bugs, performance and energy overheads accumulate as the number of IP blocks surges.

I plan to design a scalable runtime that can flexibly support device-specific hardware components without creating a gigantic software blob. The key design philosophy is similar to the principle of a microkernel-based OS: a minimal runtime kernel equipped with extensible modules, each dealing with one or one group of hardware IPs. The end result is a clear separation of concerns between runtime kernel designers who maintain a lightweight runtime kernel that is distributed to all devices and hardware vendors who design extensible runtime modules specific to their own specialized IPs. I will develop new abstractions at the kernel-module interface by evaluating how tasks are best split between the kernel and various modules in quest of optimal overall system efficiency.

- I am interested in expanding the performance-energy trade-offs demonstrated in my thesis to considering end-user privacy, a serious concern in today's ever-connected world. In particular, I plan to co-design the application privacy policy with hardware system capabilities to enable flexible trade-offs between privacy, performance, and energy-efficiency in edge computing. Intuitively, an edge-only execution model ensures the strongest privacy but also poses the highest performance and energy requirements to edge devices, whereas a cloud-centric model has the highest risk of privacy leakage but puts the lowest computation pressure on edge devices.

  To help reason about the trade-off between privacy, performance, and energy-efficiency, I believe the key is to build abstractions that specify different privacy levels at different compute hierarchies. I call these abstractions *privacy rings*, analogous to the privilege rings in conventional computer systems. I envision a set of GreenWeb-like language support that express the privacy rings, which guide the runtime system to schedule application executions while making calculated trade-offs. Through this project, I hope to lay the foundations for how privacy is treated in edge computing under practical system considerations (e.g., performance, energy).

I expect my research interests in edge computing to create ample collaborations with faculty in areas such as networking, security and privacy, distributed computing, and programming languages.

Taming the Tail Accuracy Toward Safe Computing. Just as traditional data centers suffer from the long tail latency issue, many of today's systems suffer from "tail accuracy," where a small fraction of computations exhibit extremely poor accuracy—due to the statistical behaviors of the machine learning models or approximation techniques that these systems rely on—and form a long accuracy tail distribution that is hard to curtail. Tail accuracy leads to poor worst-case accuracy guarantees, which often raise serious safety concerns in mission critical systems (e.g., self-driving cars).

My long-term research goal is to tame the tail accuracy issue and help shape the landscape of safe computing. Safe computing is undoubtedly a multi-disciplinary problem, where my experiences in cross-layer computer architecture research offer unique values. In particular, I see a lot of values in treating the safety issue as a new class of resiliency issue, to which computer architects are no stranger. After all, both share a same goal of guaranteeing expected system behaviors in the event of a rare (worst-case) error. Recent work has started relating accuracy in approximate computing to resiliency [7]. I plan to generalize that and investigate how the experience of building resilient architectures could be transferred to building safe computing systems, and how to design new solutions to overcome new safety challenges.

To achieve that goal, we need a framework to reason about computing safety and to understand where computing "unsafety" comes from. In a recent position paper [14], I use machine learning as an example to introduce the notion of "safety-gap" as one such framework. The safety gap refers to the gap between the worst-case accuracy guarantee that an oracle system demands and the actual accuracy that a particular implementation provides. The safety gap is the composition of two "meta" gaps: the learning gap and the execution gap. The "learning gap" refers to the gap between the oracle and the best-trained machine learning model. The learning gap exists because even the best machine learning model is not a perfect representation of the objective reality. The "execution gap" is introduced during the hardware execution. The execution gap exists mainly because computer architects often rely on "unsafe" optimizations such as weight compression and data quantization that intentionally trade-off accuracy with execution efficiency. To bridge the learning gap as well as the execution gap, my research agenda is two-fold:

- First, I plan to build systems that provide the performance and energy benefits of hardware specializations *with a minimal execution gap*. For instance, although one accelerator might introduce the execution gap, an ensemble of them might not. I plan to investigate how to build a safe whole out of less safe parts. One promising approach is to build a cluster of accelerators where some implement simple models that are interpretable but provide less average-case accuracy (e.g., linear regression, decision tree) while others implement complex models that are difficult to interpret but more accurate on average (e.g., neural networks). Such a system improves the worst-case accuracy by routing requests to the simple models when they can be interpreted as safe on simple models.

- Second, I plan to build systems that help *reduce the learning gap*. For instance, I am interested in practical formal verification on large-scale machine learning systems. I have started building tools to formally verify machine learning robustness, a core aspect of safety [2]. While the verification overhead for a naive approach might be high, there are many research opportunities to reduce the overhead and make formally verifiable machine learning practical. For instance, rather than first train a machine learning model and then, once the model is trained, verify the trained model, we should treat verification as a first-class consideration. I see opportunities to design custom loss functions to train models such that they are amenable to formal verification by design. This is analogous to how architects design cache coherency protocols that are inherently verifiable [5]. Also, I am interested in providing hardware support for formally verifying machine learning systems. The key is to realize that machine learning is a particular application domain relying heavily on linear algebra. Thus, it might be possible to specialize the hardware support for formal methods in that particular domain to maximize efficiency.

I expect my long-term research interests in safe computing to create many strategic collaborations with faculty in areas such as reliability, fault-tolerance, machine learning, and software engineering.

References

[1] "Node.js Benchmarks." https://github.com/nodebenchmark/benchmarks

[2] "Verdict: Formal Verification and Optimization Toolkit for Robust Machine Learning." https://github.com/yuhao/verdict

[3] Bo Wang, Yuhao Zhu, and Yangdong Deng, "Distributed Time, Conservative Parallel Logic Simulation on GPUs," in *Proc. of DAC*, 2010.

[4] Matthew Halpern, Yuhao Zhu, and Vijay Janapa Reddi, "Mobile CPU's Rise to Power: Quantifying the Impact of Generational Mobile CPU Design Trends on Performance, Energy, and User Satisfaction," in *Proc. of HPCA*, 2016.

[5] Meng Zhang, Jesse D. Bingham, John Erickson, and Daniel J. Sorin, "PVCoherence: Designing Flat Coherence Protocols for Scalable Verification ," in *Proc. of HPCA*, 2014.

[6] Peter Bailis, Jean Yang, Vijay Janapa Reddi, and Yuhao Zhu, "Research for Practice: Web Security and Mobile Web Computing," in *ACM Queue*, 2016.

[7] Radha Venkatagiri et al., "Approxilyzer: Towards A Systematic Framework for Instruction-Level Approximate Computing and its Application to Hardware Resiliency," in *Proc. of MICRO*, 2016.

[8] Yangdong Deng, Xiaomemg Jiao, Shuai Mu, Kang Kang, and Yuhao Zhu, "NPGPU: Network Processing on Graphics Processing Units," in *Theoretical and Mathematical Foundations of Computer Science*, 2011.

[9] Yangdong Deng, Yuhao Zhu, and Bo Wang, "Asynchronous Parallel Logic Simulation on Modern Graphics Processors," in *GPU Solutions to Multi-scale Problems in Science and Engineering*, 2013.

[10] Yuhao Zhu, Aditya Srikanth, Jingwen Leng, and Vijay Janapa Reddi, "Exploiting Webpage Characteristics for Energy-Efficient Mobile Web Browsing," in *Computer Architecture Letters*, 2014.

[11] Yuhao Zhu and Vijay Janapa Reddi, "High-Performance and Energy-Efficient Mobile Web Browsing on Big/Little Systems," in *Proc. of HPCA*, 2013.

[12] Yuhao Zhu and Vijay Janapa Reddi, "WebCore: Architectural Support for Mobile Web Browsing," in *Proc. of ISCA*, 2014.

[13] Yuhao Zhu and Vijay Janapa Reddi, "GreenWeb: Language Extensions for Energy-Efficient Mobile Web Computing," in *Proc. of PLDI*, 2016.

[14] Yuhao Zhu and Vijay Janapa Reddi, "Cognitive Computing Safety: The New Horizon for Reliability," in *IEEE Micro*, 2017.

[15] Yuhao Zhu, Bo Wang, and Yangdong Deng, "Massively Parallel Logic Simulation with GPUs," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2011.

[16] Yuhao Zhu, Daniel Richins, Matthew Halpern, and Vijay Janapa Reddi, "Microarchitectural Implications of Event-driven Server-side Web Applications," in *Proc. of MICRO*, 2015.

[17] Yuhao Zhu, Gu-Yeon Wei, and David Brooks, "Cloud No Longer a Silver Bullet, Edge to the Rescue," 2016.

[18] Yuhao Zhu, Matthew Halpern, and Vijay Janapa Reddi, "Event-based Scheduling for Energy-Efficient QoS (eQoS) in Mobile Web Applications," in *Proc. of HPCA*, 2015.

[19] Yuhao Zhu, Matthew Halpern, and Vijay Janapa Reddi, "The Role of the CPU in Energy-Efficient Mobile Web Browsing," in *IEEE Micro*, 2015.

[20] Yuhao Zhu, Yangdong Deng, and Yubei Chen, "Hermes: An Integrated CPU/GPU Microarchitecture for IP Routing," in *Proc. of DAC*, 2011.