RESEARCH STATEMENT

Henry (Hank) Hoffmann (hankhoffmann@cs.uchicago.edu)

My research attacks the problem of intelligent energy management, a fundamental technical challenge for computing post Dennard-scaling and beyond the end of Moore's Law. Power and energy now constrain all computing, from mobile/embedded devices to the data centers supporting the cloud and the supercomputers driving scientific discovery. The current state-of-practice involves *ad hoc*, heuristic energy management solutions that offer no formally verifiable behavior and must be rewritten or redesigned wholesale as new computing platforms emerge. My research addresses energy management in a fundamental way, starting with formal, mathematical models and ending with real software and hardware that provides efficient control and rigorous understanding—enabling intelligent, scalable, and composable energy management solutions.

My work overcomes two fundamental challenges of intelligent energy management. The first is *complexity*: modern hardware and system software expose diverse configurable parameters whose complicated interactions have surprising effects on performance and energy. The second is *dynamics*: computing systems must reliably adapt to unpredictable changes in operating environment, input workload, and even user needs.

To meet the challenges of intelligent energy management, I proposed and developed the vision of *self-aware computing systems* [DAC 2012].[1] The software and hardware systems I build are *aware* of users' throughput, latency, power, energy, and accuracy goals; they continuously monitor themselves, adapting both their behavior and internal models to ensure these goals are met in complex, dynamic environments. My self-aware systems combine machine learning—to address complexity—and control theory—to handle dynamics. *One of my work's most significant benefits is that it learns new models at runtime, while still providing many of the formal guarantees of traditional control systems, making control theoretic benefits available to a wide range of systems and scenarios.* Given the work's potential to both save energy and fundamentally change users' interaction with computing systems, *Scientific American* named my self-aware computing model one of ten **World-Changing Ideas**.[2]

Self-awareness, as I define it, is an interdisciplinary pursuit combing traditional computing systems with machine learning and control theory, and I apply self-awareness at many levels of the computing stack from hardware to applications. My work improved fundamental control theoretic techniques [CDC 2010], [IEEE TCST 2013] and created new capabilities for computing systems through applications of control theory [ASPLOS 2011], [EMSOFT 2013], [ICSE 2014], [FSE 2015], [RTAS 2015], [FSE 2017], [ASPLOS 2018a (to appear)]; machine learning [ASPLOS 2015], [ASPLOS 2016], [MICRO 2017], [ICAC 2017]; and combinations of both [SOSP 2015], [ISCA 2016], [MICRO 2016], [ASPLOS 2018b (to appear)]. I have used self-awareness to implement automated resource management at the circuit-level [VLSIC 2014]; the hardware-level [MICRO 2016], [MICRO 2017]; the OS-level [EMSOFT 2013], [RTAS 2015], [USENIX ATC 2016], [ASPLOS 2018b (to appear)]; the application-level [ASPLOS 2011], [ICSE 2014], [FSE 2015], [FSE 2017], [ASPLOS 2018a (to appear)]; and across multiple layers of the system stack [ECRTS 2014], [SOSP 2015], [ISCA 2016]. Finally, because so many computing systems must tackle the challenges of complexity, dynamics, or both, my work naturally leads to many collaborations, and I am proud to have accumulated 160 co-authors in my pursuit of this work (as of November, 2017 according to DBLP).

This statement describes my research on self-aware computing systems, characterizing my specific goals of: (1) controlling system dynamics, (2) learning models of complex systems, and (3) combining learning and control to enable new capabilities that cannot be achieved by either in isolation. After describing these goals, I briefly review some earlier research projects and then elaborate my future plans for self-aware computing—especially the combination of machine learning and control theory.

Controlling Computing Systems Through Dynamic Fluctuations

Control theory is a powerful branch of engineering mathematics—especially useful for systems that must meet goals in dynamic environments. Control theory provides a set of formalisms for adjusting system parameters to ensure desired behavior while providing predictable responses to environmental fluctuations. A classic example is the cruise control in a car, which meets a user's speed requirement while seamlessly adapting to changing conditions in wind speed, incline, and other dynamic factors.

Control theory represents a general set of techniques, but its instantiations are almost always system-specific. While there is a rich history of applying control solutions to meet latency and throughput goals in computer systems, these

[1] All references in the [ConferenceAbbreviation Year] format refer to publications listed in my CV and all hyperlinks lead to the papers themselves.

[2] Editors et al. "World-Changing Ideas: 10 new technologies that will make a difference". In: *Scientific American* 305.6 (Dec. 2011)

solutions require laborious profiling and tuning and must be reimplemented or redesigned wholesale when ported to a new computer system. Essentially, the great bulk of control theory applied to computing requires engineers to be experts in both control and computing. My research on self-aware computing creates general control implementations, allowing developers with no control background to build systems with formal guarantees that their goals will be met.

I generalized control solutions in two ways. First, I created a family of techniques for automatically synthesizing controllers to manage both application and system behavior. I developed a methodology for synthesizing controllers to meet any quantifiable goal of a software system [ICSE 2014]. I then created a framework for coordinating controllers at the application layer with those in the operating system to provide predictable timing, power, and accuracy for embedded signal processing applications [ECRTS 2014]. I then extended this work to manage any number of goals [FSE 2015]. Most recently, I developed a tool for synthesizing model predictive controllers (MPCs) that manage multiple interacting parameters to meet multi-dimensional goals [FSE 2017]. These MPC solutions capture the computer system's higher order dynamics, allowing the controller to proactively adapt to predicted future events instead of simply reacting to past behavior. This added ability, of course, comes at the cost of increased computational complexity. Together, this work allows a range of controllers to be synthesized, all of which provide guarantees that they meet user goals under different circumstances, with the more powerful guarantees coming from more computationally expensive controllers. This synthesis automates much of the most tedious and error-prone process of control design, making controllers available to non-experts in a wide array of software disciplines. The synthesis techniques, however, still require users to set some control-specific parameters.

My second line of research hides all control parameters from users and packages control solutions such that they can be embedded in applications or system software. I developed a compiler technique that automatically inserts control systems into approximate applications to provide predictable performance with maximum accuracy despite fluctuations in available power [ASPLOS 2011]. I developed runtime systems that manage embedded system resources to meet user-specified throughput constraints [EMSOFT 2013]. I demonstrated how difficult it is to achieve portable energy efficiency, due to the widely different performance/power tradeoffs on different systems [CPSNA 2015] and then developed a control library implementation that meets latency goals with near minimal energy across a wide range of systems [RTAS 2015]. Most recently, I designed and implemented a library for turning web infrastructure software's static configuration parameters into dynamically configurable parameters which are automatically tuned by a control system [ASPLOS 2018a (to appear)]. In this work, I found publicly posted bugs (such as out of memory errors) caused by improperly configured parameters in open source software (e.g., Hadoop MapReduce and HBase) and replaced existing static configuration parameters with control-based solutions that dynamically adjust parameters based on runtime conditions. The control-based approach keeps the system up in scenarios where even the patches posted by expert developers fail.

Whether automatically synthesizing controllers or embedding control systems through compilers, runtimes, and libraries, these solutions put control in the hands of non-experts. Thus, developers can focus on their area of application expertise while relying on the control system to ensure user goals are met despite dynamic fluctuations in operating environment, input, or even user-goals.

Learning Models of Complex Computer System Behavior

Until about the mid-2000s software developers relied on computer architects to reliably turn increasing transistor counts into increasing performance with no software changes. Due do the end of Dennard scaling and the physical limitations of power and energy dissipation, computer architects can no longer provide this "free ride." While additional hardware resources are available, they come at the cost of increased software complexity for determining how best to use those resources. As a simple example, many systems support heterogeneous processor types, with configurable processor and memory speeds. Such systems expose a wide range of parameters for software management. While my control systems are well suited to tune those parameters, they require accurate models of how the parameters affect the goals. As systems become more complicated, with an increasing diversity of parameters, their interaction becomes incredibly complex, and modeling those interactions is increasingly difficult. Even a few different resources can present a large search space which is infeasible to explore exhaustively at run-time. Machine learning techniques, however, are well-suited to estimating exposed hardware resource's affects on application performance and power without reverting to brute-force search.

Therefore, my self-aware systems incorporate *learning* to estimate and model the behavior of complex, interacting resources. For example, I proposed a hierarchical Bayesian model for estimating application performance and power as a function of core, socket, hyperthreads, frequency, and memory controller usage on real server systems. This model samples less than 2% of the entire search space, but its performance and power estimations show 97% accuracy and result in almost 2$\times$ energy reduction compared to common heuristics like racing-to-idle [ASPLOS 2015]. I used software decision trees to model how performance is affected by the interaction of hardware resources (again including cores, sockets, hyperthreads, and memory controllers) with on-chip power management that controls clockspeed. My approach finds

resource configurations that consistently outperform Intel's hardware approach that only considers clockspeed [ASPLOS 2016]. Memory systems are also increasingly complicated, so I recently compared a variety of machine learning techniques for managing the huge design space of non-volatile main memory configurations. This work developed learning methods that find combinations of memory structures that are faster and more energy efficient than any individual proposal [MICRO 2017]. While most of this work models the effects of resource configuration on a single application, real systems run multiple applications simultaneously. Therefore, I designed a new regularized regression method for estimating the effects that multiple applications have on each other's performance. My method achieves comparable accuracy to non-linear techniques with performance much closer to linear methods [ICAC 2017].

These techniques handle complicated parameter spaces with 1000s to 100,000s of possible choices. While these spaces may not be large compared to some decision problems, computer resource management decisions must be made on a timescale of micro to milliseconds. My application of learning approaches to self-aware computing is essential for making timely and accurate predictions of how system configuration will affect measurable behavior.

Combining Machine Learning and Control Theory

Thus far the bulk of my research in developing self-aware systems applies control to ensure goals are met in dynamic environments and, separately, applies machine learning to model complex systems. I am now integrating these two approaches into a unified framework because each technique has tremendous benefits, but comes with distinct limitations, so neither completely addresses the challenges of intelligent energy management. Informally, control requires engineers to define the space of possible operating conditions, and the control system guarantees the desired dynamic behavior. All guarantees are lost, however, if the operational space is not correctly specified. In contrast, learning techniques require little to no prior knowledge and quickly define a model of possible operation. Learners—including reinforcement learning techniques—however, provide little to no guarantees that they will meet goals in dynamic environments. Ideally, the combination of control and learning should guarantee a computing systems' dynamic behavior in a complicated environment even with little prior knowledge.

The key challenge is bridging the gap between learned, non-linear models of discrete computing systems and the continuous, linear models used by control systems. While my work in this area is ongoing, early results show promise. I designed and demonstrated one of the first systems to provide formal energy guarantees (e.g., ensuring sufficient battery life to complete a video playback) by combining learned models of system energy efficiency with runtime control of application alternatives [SOSP 2015]. Additionally, this work shows that combining application- and system-level resource management improves energy efficiency by 1.5-3$\times$ compared to approaches that work at one level only. I also demonstrated that—when properly supported with fine-grain configurable hardware—combining learning and control delivers quality-of-service guarantees with minimum cost for infrastructure-as-a-service customers, providing as much as 70% cost reduction compared to existing heuristic techniques [ISCA 2016]. Finally, I designed and implemented a hardware combination of classification and control systems to meet performance guarantees with minimal energy on graphics processing units (GPUs) [MICRO 2016]. To the best of my knowledge this is the first work to provide hardware support for real-time processing with minimal energy on GPUs and it does not even require prior knowledge of the applications under control. I believe this work will become increasingly important as GPUs are tasked with mission critical processing and decision making tasks in self-driving vehicles.

While preliminary, this work shows how the combination of control and machine learning enables exciting new capabilities that are not possible with either in isolation. For example, formally guaranteeing energy consumption across multiple platforms is a new capability, but extremely relevant for mobile, embedded, and web services.

Prior Research Projects

I have a long standing interest in improving computing energy efficiency, which is reflected in my earlier research on *multi- and manycore architectures*, as well as using *approximate computing* to trade accuracy for energy savings.

Multicore Architectures: Raw and Tilera As a graduate student at MIT, I had the fortunate opportunity to turn academic research (the Raw processor [IEEE MICRO 2002], [ISSCC 2003], [JILP 2004], [ISCA 2004]) into a commercial product (the Tilera TILE family of processors [IEEE MICRO 2007], [HiPEAC 2010]). In addition to the excitement of seeing an idea transformed into something people could hold, the process of commercializing research opened my eyes to the challenges that real engineers (rather than academics) face in the current computing landscape. When meeting with customers, the following themes came up repeatedly: (1) the need to meet multiple—often competing—goals; e.g., high performance and low energy; (2) the burden that complex, modern hardware puts on the engineers who must use these systems; and (3) the requirement to meet these goals in dynamic environments where components fail and workloads fluctuate. So, my experience commercializing the Raw processor at Tilera had a great influence on my future work in self-aware computing.

Approximate Computing: Loop Perforation and Dynamic Knobs Approximation increases application flexibility; e.g., when resources are scarce, rather than stop computing, an approximate application produces a slightly less accurate result OOPSLA 2010. While other work in this area requires programmer changes to application code (e.g., specifying functions or data amenable to approximation), my work developed two techniques that automatically find approximate variations for existing programs. First, *loop perforation* discards loop iterations and evaluates whether the resulting program still produces an acceptable output [MIT TR 2009], [ICSE 2010], [FSE 2011]. Second, dynamic knobs transform existing statically configured parameters into a data structure allowing dynamic response to changing resources [ASPLOS 2011]. While my prior work focused on new approximation techniques, my latest work focuses on coordinating approximation at the application-level with system-level resource management. Much of this work has already been mentioned in this document (e.g., [ECRTS 2014], [SOSP 2015]), but another example includes using approximation to ensure hard real-time guarantees with minimal energy [USENIX ATC 2016].

Summary and Future Work

My research studies self-awareness as a fundamental property of computing systems. Self-aware systems automatically adapt behavior to meet user-specified goals in complex, dynamic computing systems. While I am primarily a systems builder, I base my work on sound mathematical models providing formal guarantees that they will meet user goals, and—perhaps more importantly—providing understanding of when those goals are unreachable.

In the future I plan to extend this work in several ways. First, I am extremely excited about the further possibilities of combining control and learning systems. My work in this area has created custom solutions for the individual problems investigated. I have just begun generalizing those results to create an interface that allows many different learning systems to be combined with different types of controllers [ASPLOS 2018b (to appear)]. This work should make the combination of control and learning accessible to application developers who are not familiar with either. In addition, the preliminary results show that the combined approach produces much more predictable latency and lower energy than either learning or control alone.

Second, and perhaps even more exciting, is the idea of using control to provide formal guarantees for learning systems that currently have none. I originally envisioned the combination of learning and control as a way to further generalize my control systems. I now view it as a way to add predictability to learning systems. For example, the last few years have seen an explosion in the capability and deployment of deep neural networks (DNNs). DNNs are powerful inference engines, but they provide almost no behavioral guarantees. I plan to investigate control strategies for managing the accuracy and latency of DNN-based inference engines. Such work could make DNN inference predictable when operating in complex, dynamic environments such as autonomous vehicles.

Finally, I plan to extend my work on self-aware computing beyond the goals studied so far and incorporate resilience [IEEE CAL 2015] and security [ASPLOS 2018c (to appear)] into the decision process. The ultimate goal is to build systems where users or operators simply specify latency, energy, accuracy, resilience, and security goals, while the self-aware system tunes application, system software, and hardware parameters to ensure these goals are met despite unpredictable dynamic disturbances.

Impact

I have been fortunate enough to work on several high-impact research projects that affected both computing and society.[3] As noted above, I began graduate school working on the Raw processor, an early multicore. At the time, the idea of integrating multiple cores onto a chip was still very new—the term *multicore* would not be recognized until after we had commercialized the Raw processor at Tilera. Now, of course, multicores are standard and directly impact every computer and phone user. Tilera itself was sold for $130M. When I began working on approximate computing, that was an extremely controversial topic (perhaps that is why the first reference I listed is an unpublished technical report from 2009). Now, it is common for major conferences (e.g., ASPLOS, MICRO, and PLDI) to have entire sessions devoted to new approximation techniques and management of the same. Even when the session is not on approximation directly, many recently proposed techniques for hardware optimization of deep learning rely on trading accuracy for reduced energy or decreased runtime.

Indeed, I hope the sudden explosion of deep learning—and machine learning in general—will help propel my most recent work on self-aware computing towards a similar impact as my previous projects. One of the great challenges of deploying learning is trusting these systems to make good decisions in dynamic environments. For example, critical sensing and decision making in essential cyber-physical systems (e.g., autonomous vehicles, intelligent homes, and the smart grid) are being trusted to mechanisms like DNNs. My most recent work on combining learning with control to preserve the latter's formal guarantees has the potential to make it much safer to employ learning mechanisms in safety critical systems that interact with humans daily.

[3] Google Scholar lists my citations as over 5K with and h-index of 32.