Controlling AI Engines in Dynamic Environments

Nikita Mishra Connor Imes Henry Hoffmann
University of Chicago
nmishra,ckimes,hankhoffmann@cs.uchicago.edu

John D. Lafferty
Yale University
john.lafferty@yale.edu

Abstract

A recent paper lists deploying AI in dynamic environments with unpredictable changes as a major research opportunity. To address this challenge we advocate the integration of AI and control systems. AI and machine learning are well-suited to deal with *complexity*, while control theory is specifically designed to manage *dynamics*. The integration of these approaches has several benefits: 1) fast reaction to dynamic changes, 2) error tolerance based on runtime feedback, and 3) robust design that guarantees that operating requirements will be respected when possible or the ability to report when those requirements cannot be met.[1]

1 The Challenge of System Dynamics

The last several years have seen a massive transition of Artificial Intelligence and Machine Learning technologies from academia and research labs into commodity products. AI systems are being deployed in all types of computing systems from image analysis in embedded systems to selecting and serving digital advertisements in the cloud. While this transition has been wildly successful, a recent survey notes several open research challenges in AI deployment, including operating "in dynamic environments, *i.e.,* , environments that may change, often rapidly and unexpectedly, and often in non-reproducible ways [27]."

Fortunately, control theory is an engineering discipline devoted to operating in dynamic environments. Furthermore, there is a rich history of deploying control theoretic solutions to computer management problems; *e.g.,* meeting quality-of-service requirements in web servers [8, 9, 23]. Control theory's appeal is that it provides a rigorous framework for designing systems that provably meet requirements despite unpredictable system dynamics. The drawback is that control deployment requires experts who can formulate difference models of the computer system, a discipline in which most programmers are not trained [8].

While it seems natural to combine AI engines with control systems to build AI that is robust in dynamic environments, this combination requires developers who are experts in AI, control systems, and the actual application area in which the system is to be deployed. This combination of expertise is an unrealistic burden to place on a developer, as even AI experts are in short supply at the moment. We therefore advocate a general framework for composing AI engines with control that provides many of control's guarantees without requiring control expertise from the user. The proposed combination of AI and control has three benefits:

- Formally analyzable dynamic response.
- Fast adaptation to unpredictable dynamic events.
- Increased robustness to errors in the learned model.

[1]This abstract is a summary of a full length paper to appear in ASPLOS 2018 [18].

Conference'17, Washington, DC, USA
2016. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

2 Allocating Heterogeneous Resource

As a motivational problem we consider allocating resources in a heterogeneous multicore processor, specifically Samsung's Exynos Octa processor used in mobile devices and some internet-of-things devices [24]. The processor is based on an ARM big.LITTLE architecture with four big, high-performance cores and four LITTLE, energy efficient cores. The big cores support 19 clock speeds and the LITTLE cores support 14.

The mobile and embedded systems running on these heterogeneous processors typically have latency requirements; *i.e.,* they must deliver reliable performance to effectively process (*e.g.,* video analysis) or produce (*e.g.,* gaming) a data stream. Additionally, any latency requirement should be met while minimizing energy to prolong battery life or reduce costs.

Finding the right combination of resources to meet a particular application's latency requirement while minimizing energy is a complex optimization process, and a number of AI and ML approaches have been proposed for this problem [1, 4–7, 13, 17, 19, 20, 25, 31, 33]. All these approaches estimate the performance and energy of some resource configuration, but they have limited ability to deal with dynamic changes. In contrast, control theoretic approaches dynamically adjust resource usage based on the difference between measured and expected behavior [2, 9, 11, 12, 14, 22, 26, 29, 30, 32]. Control formalisms, however, rely on "ground truth" models of application response to resource usage.

Intuitively, combining learned models of complex hardware resources with control-theoretic resource management should produce predictable latency and two recent research projects explore such a combination. Recht et al. have proposed several approaches for combining statistical learning models with optimal control theory [3, 28]. Simultaneously, Hoffmann et al. have developed OS- [10] and hardware-level resource management systems [21] that combine learning and control to provide both energy and latency guarantees in dynamic environments.

This prior work, however, still requires expertise in both learning and control methods to effectively deploy the proposed solution. Recent work, however, defines abstractions that allow a number of AI and learning techniques to be combined with an adaptive controller, maintaining control-theoretic formal guarantees [18]. This paper reviews those abstractions and demonstrates their benefits.

3 Combining AI Engines and Control Systems

Figure 1 shows the proposed approach of splitting resource management into learning and control sub-tasks and then composing their solutions. When a new application enters the system, an adaptive controller allocates resources using a generic model and records latency and power. The recorded values are sent to a learner, which estimates the application's latency and power in all other resource configurations. The learner extracts those that are predicted to be Pareto-optimal and packages them in a data structure called the performance hash table (PHT). The PHT and the learner's estimated error are sent to the controller, which tunes its internal parameters

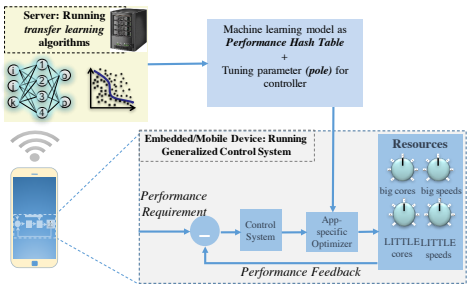

Figure 1. Combining learning and control to manage heterogeneous resources.

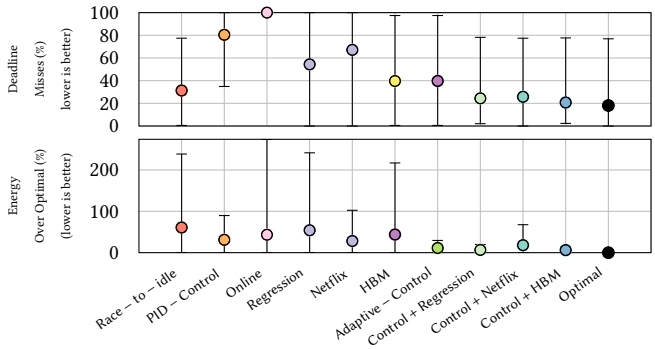

Figure 2. Summary data for multi-app scenario.

to tolerate that error and then selects an energy minimal resource configuration. The only external, user-specified parameter in this approach is the latency requirement, meaning users can deploy this framework with no control knowledge at all, but just knowledge of an application's performance requirements.

While the mathematical details are beyond the scope of this document, the key to the proposed approach is that the control system is *abstract*. While traditional controllers for computing manage physical quantities—*e.g.,* cores and clockspeed [22]—the proposed control system manages *speedup*. This abstraction creates a layer of indirection between the behavior the controller is enforcing and the specific physical mechanism that achieves it. The mapping of the desired behavior (speedup in this example) to specific resources is done using learned models of the system's performance and power as a function of resource usage. This abstract control system is thus quite general, allowing different AI/ML engines to be paired with the controller and allowing the combination to be easily ported to many different computing systems; *e.g.,* ARM embedded and Intel server systems [16].

Additionally, while the proposed controller works at a higher level of abstraction than typical controllers, it still provides formal guarantees that it will converge to the desired behavior. In this case, the guarantees are probabilistic and based on the error estimates (or confidence intervals) provided by the learners. Thus, while any AI/ML approach could be paired with the controller, the best results will come from those which provide accurate confidence intervals.

4 Reacting to Dynamic Events

To demonstrate the benefits of combining AI/ML and control, we allocate cores and clockspeed in a dynamic environment. Specifically we choose 12 applications from embedded and mobile processing. Each processes a stream of inputs, and we set a per input latency goal based on the worst case input. We measure the number of missed deadlines and the energy consumption over optimal for each application. Many applications have inherent dynamics due to input dependent processing. To create an additional dynamic burden, one fifth of the way through each application's execution we launch a second application on a single big core. This new application disrupts the system and any resource allocator must adjust to ensure the original application's latency.

We compare the proposed combination of learning and control to several published approaches. We compare to ML-based resource allocators including: an offline approach that averages across all prior behaviors [25], online multivariate regression [15, 17, 20], the Netflix algorithm modified for resource management [4, 5], and a hierarchical Bayesian model (HBM) [17]. We compare to two control approaches including: a proportional-integrative-derivative (PID) controller tuned for best average case behavior [9] and a self-tuning controller for embedded systems [12]. We compare to the heuristic of racing-to-idle, where all resources are allocated and the system transitions to a low-power idle state if an input is not worst case. Additionally, we compute an optimal resource schedule through brute force search.

Figure 2 shows the average (over all applications) deadline misses and energy over optimal. A deadline is missed if an input takes longer than the latency target. The error bars show the worst and best case deadline misses of any application. Some latencies are unachievable for some applications; thus, even the optimal allocator has some deadline misses. Race-to-idle misses more deadlines than optimal because it cannot use LITTLE cores to do some work, it simply continues using all big cores despite the degraded performance. Most approaches do badly in this dynamic scenario—even adaptive control has 40% deadline misses. *All combinations of learning and control, however, produce better outcomes than the learners alone because these learning approaches do not adapt to the interfering application (or they do so too slowly to make a difference).* The combination of HBM and control produces the fewest deadline misses with an average of 20%, which is only 2 points more than optimal and almost half of the best prior approaches. This combination also produces the lowest energy, just 6% more than optimal. Detailed, per-application results are available in the full paper [18].

5 Conclusion & Future Work

These results demonstrate the two claims from the introduction. First, the combination of learning and control quickly adapts to the dynamics of the new application launch. Second, the combination is always better then the learning approach alone because it corrects for errors in the learned model. This result is even more apparent when looking at the behavior of applications without interference, where learning plus control still outperforms learning by itself (for details see the full paper).

We believe the proposed framework is quite general, and we plan to test it on additional systems with different goals. For example, to meet a target power requirement on a server while maximizing performance. We believe this framework is applicable for tasks other than managing computing resources and we hope to deploy it with other AI systems that must solve constrained optimizations in dynamic environments, such as logistics systems, and autonomous vehicles.

References

[1] R. Bitirgen et al. "Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach". In: *MICRO*. 2008.

[2] J. Chen and L. K. John. "Predictive coordination of multiple on-chip resources for chip multiprocessors". In: *ICS*. 2011.

[3] S. Dean et al. *On the Sample Complexity of the Linear Quadratic Regulator*. Tech. rep. 1710.01688v1. arXiv, 2017.

[4] C. Delimitrou and C. Kozyrakis. "Paragon: QoS-aware Scheduling for Heterogeneous Datacenters". In: *ASPLOS*. 2013.

[5] C. Delimitrou and C. Kozyrakis. "Quasar: Resource-efficient and QoS-aware Cluster Management". In: *ASPLOS*. 2014.

[6] Z. Deng et al. "Memory Cocktail Therapy: A General Learning-Based Framework to Optimize Dynamic Tradeoffs in NVM". In: *MICRO*. 2017.

[7] C. Dubach et al. "A Predictive Model for Dynamic Microarchitectural Adaptivity Control". In: *MICRO*. 2010.

[8] A. Filieri et al. "Control Strategies for Self-Adaptive Software Systems". In: *TAAS* 11.4 (2017), 24:1–24:31. DOI: 10.1145/3024188. URL: http://doi.acm.org/10.1145/3024188.

[9] J. L. Hellerstein et al. *Feedback Control of Computing Systems*. John Wiley & Sons, 2004. ISBN: 047126637X.

[10] H. Hoffmann. "JouleGuard: energy guarantees for approximate applications". In: *SOSP*. 2015.

[11] H. Hoffmann et al. "A Generalized Software Framework for Accurate and Efficient Managment of Performance Goals". In: *EMSOFT*. 2013.

[12] C. Imes et al. "POET: A Portable Approach to Minimizing Energy Under Soft Real-time Constraints". In: *RTAS*. 2015.

[13] E. Ipek et al. "Self-Optimizing Memory Controllers: A Reinforcement Learning Approach". In: *ISCA*. 2008.

[14] B. Li and K. Nahrstedt. "A control-based middleware framework for quality-of-service adaptations". In: *IEEE Journal on Selected Areas in Communications* 17.9 (1999).

[15] J. Li and J. Martinez. "Dynamic power-performance adaptation of parallel computation on chip multiprocessors". In: *HPCA*. 2006.

[16] N. Mishra. "Statistical Methods for Improving Dynamic Scheduling and Resource Usage in Computing Systems". English. PhD thesis. 2017, p. 126. ISBN: 9780355077810. URL: https://search.proquest.com/docview/1928485902?accountid=14657.

[17] N. Mishra et al. "A Probabilistic Graphical Model-based Approach for Minimizing Energy Under Performance Constraints". In: *ASPLOS*. 2015.

[18] N. Mishra et al. "CALOREE: Learning Control for Predictable Latency and Low Energy". In: *ASPLOS*. 2018.

[19] P. Petrica et al. "Flicker: A Dynamically Adaptive Architecture for Power Limited Multicore Systems". In: *ISCA*. 2013.

[20] D. Ponomarev et al. "Reducing Power Requirements of Instruction Scheduling Through Dynamic Allocation of Multiple Datapath Resources". In: *MICRO*. 2001.

[21] M. H. Santriaji and H. Hoffmann. "GRAPE: Minimizing energy for GPU applications with performance requirements". In: *MICRO*. 2016.

[22] A. Sharifi et al. "METE: meeting end-to-end QoS in multicores through system-wide resource management". In: *SIGMETRICS*. 2011.

[23] S. Shevtsov et al. "Control-Theoretical Software Adaptation: A Systematic Literature Review". In: *IEEE Transactions on Software Engineering* PP.99 (2017), pp. 1–1. ISSN: 0098-5589. DOI: 10.1109/TSE.2017.2704579.

[24] Y. Shin et al. "28nm High- Metal-Gate Heterogeneous Quad-Core CPUs for High-Performance and Energy-Efficient Mobile Application Processor". In: *ISSCC*. 2013.

[25] D. C. Snowdon et al. "Koala: A Platform for OS-level Power Management". In: *EuroSys*. 2009.

[26] D. C. Steere et al. "A Feedback-driven Proportion Allocator for Real-rate Scheduling". In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. OSDI '99. New Orleans, Louisiana, USA: USENIX Association, 1999, pp. 145–158. ISBN: 1-880446-39-1. URL: http://dl.acm.org/citation.cfm?id=296806.296820.

[27] I. Stoica et al. *A Berkeley View of Systems Challenges for AI*. Tech. rep. 1712.05855v1. arXiv, 2017.

[28] S. Tu and B. Recht. *Least-Squares Temporal Difference Learning for the Linear Quadratic Regulator*. Tech. rep. 1712.08642v1. arXiv, 2017.

[29] V. Vardhan et al. "GRACE-2: integrating fine-grained application adaptation with global adaptation for saving energy". In: *IJES* 4.2 (2009).

[30] W. Yuan and K. Nahrstedt. "Energy-efficient soft real-time CPU scheduling for mobile multimedia systems". In: *SOSP*. 2003.

[31] H. Zhang and H. Hoffmann. "Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques". In: *ASPLOS*. 2016.

[32] R. Zhang et al. "ControlWare: A middleware architecture for Feedback Control of Software Performance". In: *ICDCS*. 2002.

[33] Y. Zhu and V. J. Reddi. "High-performance and energy-efficient mobile web browsing on big/little systems". In: *HPCA*. 2013.