A Case for Runtime Coordination of Accuracy-aware Applications and Power-aware Systems

Henry Hoffmann
Department of Computer Science, University of Chicago
hankhoffmann@cs.uchicago.edu

Abstract

We motivate runtime coordination of accuracy-aware applications, which expose accuracy/performance tradeoffs, with power-aware systems, which manage power/performance tradeoffs. This is a challenging problem because decisions made at one level affect, and often counteract, decisions made at the other, leading to unpredictable behavior. Our preliminary results indicate that coordination has the potential to not only avoid bad behavior, but also provide better outcomes than application or system adaptation alone.

1. Introduction

Power and energy constraints are dominating the design of modern computer applications and systems. Two complementary approaches have developed for dealing with these constraints. At the application-level frameworks have been developed for creating *accuracy-aware* applications, which can sacrifice the quality of their result for increased performance (or reduced energy usage) [1, 2, 8, 9, 14–16]. At the system-level frameworks have been proposed to create *power-aware* systems, which can respond to reduced computational demand by decreasing power or energy consumption [3–5, 10, 12, 17, 18].

These adaptive frameworks benefit users by automatically adjusting to meet constraints. For example, accuracy-aware applications may reduce accuracy to maintain performance despite reduced resource availability. Similarly, power-aware systems respond to reduced workload by reducing resource usage. Given the potential benefits of these adaptive approaches, it is increasingly likely that accuracy-aware applications will be deployed on power-aware systems. Therefore, it is important to study their potential interaction. *Based on preliminary results, this paper takes the position that decisions at the application level (such as switching to a higher-performance, lower-accuracy algorithm) should be dynamically coordinated with decisions at the system level (such as switching to a lower-performance, lower-power configuration).*

We identify at least three potential benefits of active coordination of accuracy-aware applications with power-aware systems. Specifically, active coordination:

- Avoids oscillations and destructive interference that can arise when application and system act independently.
- Allows control of multiple dimensions simultaneously.
- Produces better outcomes for the same constraints.
- Provides a richer tradeoff space for reacting to user goals and environmental changes.

2. Preliminary Results

We collect preliminary results by running an accuracy-aware video encoder (made with the PowerDial framework [9]) on a SandyBridge Linux x86 platform with a power-aware runtime [10]. The video encoder can reduce accuracy (adding noise to the encoded video) in exchange for increased performance. The system can increase resource usage to increase performance at a cost of increased power consumption. The following examples show the benefits that can be achieved through active coordination of the accuracy-aware application with the power-aware system.

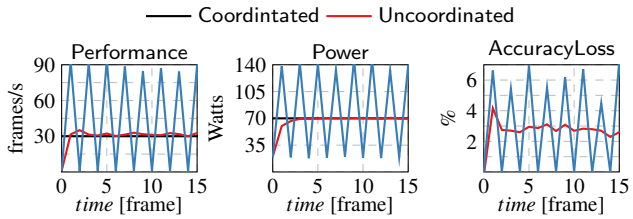

Figure 1: **Running an adaptive video encoder on an adaptive system with and without coordination. The goal is to maintain 30 frames per second performance. Without coordination, the performance goal is not met while power and accuracy fluctuate wildly. In contrast, the coordinated approach meets the performance goal while conserving power and accuracy.**

2.1 Avoiding Oscillations & Controlling Multiple Dimensions

Given a performance goal (*e.g.,* real-time or quality-of-service constraint), both the application and system are provably convergent (*i.e.,* they will achieve the goal) when run in isolation. When deployed concurrently, however, they may interfere with each other, resulting in constraint violations and unpredictable behavior. These problems arise because both application and system assume that changes in the other (*i.e.,* resources or workload) are rare, and will be long-lasting when they do occur. When application and system continuously react to each other, they miss performance goals, waste power, and lose accuracy.

Figure 1 shows this bad behavior and how it can be overcome through active coordination. The figures shows that the accuracy-aware application and power-aware system produce oscillating behavior when they act without coordination. However, coordination (in this case, achieved through a specially designed adaptive feedback control system [11]) drives performance and power to the desired targets (30 frames/s and 70 Watts, respectively) while avoiding oscillations.

2.2 Better Outcomes for the Same Constraints

In addition to preventing bad behavior, our preliminary results indicate that coordination can achieve better outcomes for the same constraints as shown in Figure 2. We first consider the energy required to perform video encoding with an accuracy constraint. As shown in the figure (left side) coordination produces lower energy for the same accuracy constraint. The results on the right side of the figure show that for an energy efficiency goal, coordination produces a smaller accuracy loss.

2.3 Adapting to Application Phases

We use our video encoder to compress a video with three distinct scenes. The top row of charts in Figure 3 illustrates the behavior of these scenes using the application's and system's default settings (*i.e.,* encoding with the highest accuracy and all system resources). The three different scenes (each 500 frames) are demarcated by

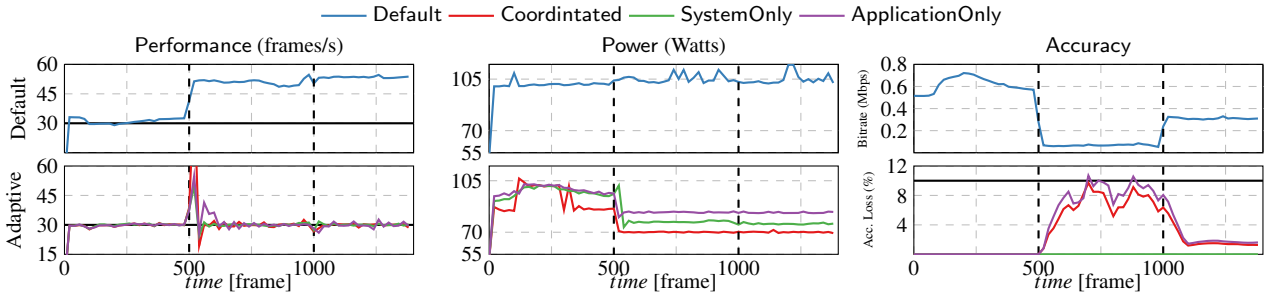

Figure 3: **Comparison of non-adaptive behavior and several adaptive schemes adjusting to phases in x264.**

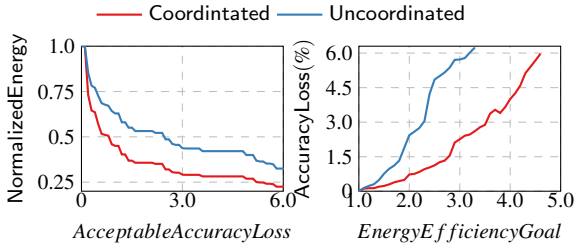

Figure 2: **Coordinating application and system provides lower energy for the same acceptable accuracy (left) and lower accuracy loss for the same energy target (right).**

the vertical dashed lines. Each scene has distinct characteristics. The first is the slowest, has lowest average power consumption and produces the largest bitrate. The second scene is faster and naturally has lower bitrate. The third scene is the fastest, but occupies a middle ground in bitrate. To show the benefits of coordination, we set a soft real-time performance goal (of 30 frames per second) and deploy the encoder with a coordinated runtime, a system-level runtime, and an application-level runtime.

The bottom row of charts in Figure 3 shows how coordinated, application-only, and system-only approaches react to the changing scenes. Clearly all three approaches meet the performance goal (30 frames/s) (with some short, deviations appearing at the scene changes); however, each approach achieves the goal differently, resulting in different power consumptions and accuracies. Clearly, the coordinated approach produces the lowest power consumption of the three, achieving almost half the power consumption of the application-only approach. Furthermore, coordination also produces slightly better accuracy than an application only approach.

3. Related Work

Some prior work has studied application and system coordination. Flinn and Satyanarayanan build a framework for coordinating operating systems and applications to meet user defined energy goals [8]. This system trades application quality for reduced energy consumption, providing up to 30% increase in battery life. The Truffle architecture [6] supports applications, like those written in EnerJ [14], which explicitly mark some computations and data as "approximate," allowing accuracy to be exchanged for reduced energy consumption. A similar approach, Parrot, replaces approximate regions of an application with a neural network implementation, which is then executed on a special neural processing unit in hardware [7]. Flikker, allows applications to explicitly mark some data as "non-critical," storing this data in a DRAM that trades accuracy for energy savings [13]. Of these approaches, Flinn and Satyanarayanan's approach is closest to to that advocated here, because it provides dynamic management. However, it supports only energy goals and cannot support real-time performance or accuracy bounds. Furthermore, the adaptations are based on heuristic approaches. We believe that techniques should be developed to provide runtime guarantees in any dimension (accuracy, power, performance, or energy). In addition, well-founded analytical techniques should be developed for providing these guarantees.

A runtime management system, like that proposed here, complements static analysis. The runtime would benefit from static analysis and worst case bounds, while a system which provides static analysis could benefit from runtime management to tailor control to the specific characteristics of the current input.

4. Conclusion & Future Work

The results in this position paper indicate great potential for coordinating the adaptation of accuracy-aware applications and power-aware systems. Therefore, we propose these results should be extended by 1) exploring additional applications and systems, 2) developing formal models to describe and control the interaction between these two components, and 3) integrating frameworks which perform static analysis of these interactions with other approaches that perform dynamic runtime coordination.

References

[1] J. Ansel et al. "Siblingrivalry: online autotuning through local competitions". In: *CASES*. 2012.

[2] W. Baek and T. Chilimbi. "Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation". In: *PLDI*. 2010.

[3] R. Balasubramonian et al. "Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures". In: *MICRO*. 2000.

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

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

[6] H. Esmaeilzadeh et al. "Architecture support for disciplined approximate programming". In: *ASPLOS*. 2012.

[7] H. Esmaeilzadeh et al. "Neural Acceleration for General-Purpose Approximate Programs". In: *MICRO*. 2012.

[8] J. Flinn and M. Satyanarayanan. "Managing battery lifetime with energy-aware adaptation". In: *ACM Trans. Comput. Syst.* 22.2 (May 2004), pp. 137–179.

[9] H. Hoffmann et al. "Dynamic Knobs for Responsive Power-Aware Computing". In: *ASPLOS*. 2011.

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

[11] H. Hoffmann et al. *CoAdapt: Predictable Behavior for Accuracy-Aware Applications Running on Power-Aware Systems*. Tech. rep. TR-2014-02. University of Chicago, 2014.

[12] T. Horvath et al. "Dynamic Voltage Scaling in Multitier Web Servers with End-to-End Delay Control". In: *Computers, IEEE Transactions on* 56.4 (2007).

[13] S. Liu et al. "Flikker: saving DRAM refresh-power through critical data partitioning". In: *ASPLOS*. 2011.

[14] A. Sampson et al. "EnerJ: approximate data types for safe and general low-power computation". In: *PLDI*. 2011.

[15] S. Sidiroglou-Douskos et al. "Managing performance vs. accuracy trade-offs with loop perforation". In: *ESEC/FSE*. 2011.

[16] J. Sorber et al. "Eon: a language and runtime system for perpetual systems". In: *SenSys*. 2007.

[17] A. Verma et al. "Server workload analysis for power minimization using consolidation". In: *USENIX Annual technical conference*. 2009.

[18] Q. Wu et al. "Formal online methods for voltage/frequency control in multiple clock domain microprocessors". In: *ASPLOS*. 2004.