Big Data for Little Cores

Combining Learning and Control for Mobile Energy Efficiency

Paper # 92

Abstract

Systems designed for mobile and Internet-of-things (IoT) must deliver performance to interactive applications (that monitor the environment and respond to users) and they must conserve resources to extend battery life. Meeting these conflicting goals is complicated by two issues: (1) hardware heterogeneity leading to complicated optimization spaces and (2) dynamic operating conditions including unpredictable changes in available resources and application behavior. Machine learning techniques handle complicated optimization spaces, but do not incorporate models of system dynamics; optimal control techniques provide formal guarantees of dynamic behavior, but cannot handle non-linear system models. In this paper, we propose a combination of learning and control techniques to meet mobile system's performance requirements in unpredictable environments. Specifically, we combine a hierarchical Bayesian model (HBM) with a lightweight control system (LCS). The HBM runs on a remote server, aggregating data from many separate mobile systems to learn customized models of performance/power tradeoffs. The LCS runs locally on the mobile system and tunes resource usage to meet performance requirements efficiently. We implement both the HBM and LCS and test their ability to learn and control ARM big.LITTLE systems. Compared to existing learning and control methods, our proposed combined system delivers more reliable performance – only XX% error compared to YY% for learning and ZZ% for control – and higher energy efficiency – achieving ZZ% of optimal and improving by XX% over learning and YY% over control. When used in a multi-application scenario these numbers improve: xx% error compared to yy% for learning and zz% for control and improvements of xx% and zz%, respectively, in energy efficiency.

1. Introducation

Mobile systems have clear requirements for correct operation: they must meet performance goals necessary for interacting with sensors and human users, but must also conserve energy to maximize battery life. To address these often conflicting requirements, hardware platforms have become increasingly diverse and complicated. Many such processors support, for example, different core types with different performance/power tradeoffs, which can be operated at a wide range of different speeds. Meeting performance requirements is further complicated by the dynamic nature of computing systems: application demands can vary widely as a function of input or application phase and multiple applications may compete for resources.

Thus, two central challenges arise to meeting performance requirements with minimal energy on mobile systems: (1) complexity and (2) dynamics. Each challenge has been addressed individually. First, machine learning approaches can identify the complicated, power/performance tradeoff spaces that arise on configurable, heterogeneous mobile systems []. Such approaches can learn complex models, identifying and avoiding local extrema that lead to inefficient resource usage. Second, optimal control theoretic techniques ensure performance is met with minimal energy by tuning resource allocation as applications run []. Control techniques provide a formal basis for reasoning about dynamics and can ensure performance requirements are met despite application, input, or workload fluctuations.

Learning and control techniques have complementary strengths and weaknesses. Learning approaches handle complexity, but have no established mechanism for managing system dynamics; more powerful learning methods also tend to incur higher computational cost making them ill-suited for runtime use on energy-constrained devices. Control approaches handle dynamics, but rely on linear models that are increasingly insufficient to capture the diversity of modern hardware.

We therefore propose combining learning and control techniques to manage both complexity and dynamics. Specifically, we use a hierarchical Bayesian model (HBM) to aggregate data across multiple devices and applications, creating accurate, customized models mapping application resource usage to performance and power. To mitigate overheads, the HBM runs on a remote server, allowing it to quickly collect data from a number of applications running on separate devices and amortize the cost of computing the models across all those devices. Once learned, the models are sent to individual devices where a lightweight control system (LCS) uses them to ensure that performance requirements are met with minimum energy even if the application changes phase, processes an unexpected input, or runs with other applications competing for resources. The control system is computationally efficient and provides formal guarantees that it will converge to the desired performance despite unpredictable system dynamics. These guarantees are a product of the combined learning and control framework. The accuracy

of the learner in the face of system complexity is essential for guaranteeing performance in highly dynamic systems **TODO: 1**.

While control and learning frameworks exist, the key to combining them is creating an interface between the learning and control systems. Specifically, learning frameworks for resource management map configurations (*e.g.*,resource allocations) into estimated performance and power. These mappings are discrete and non-linear, capturing the behavior of the underlying system. Controllers, in contrast, work with continuous linear models. Therefore, our proposed combination of learning and control requires an interface to convert the discrete non-linear learned models into continuous linear models. We address this challenge by forming the lower convex hull of points on the learned power/performance tradeoff space. Interpolating between these points gives us a piecewise linear function that is appropriate for control models, yet still captures the significant behavior of the underlying system. This interface allows us to combine the approaches studied in this paper, and we believe it is sufficiently general to apply to other combinations of learning and control as well.

To evaluate our approach, we implement the HBM in Matlab and run it on an x86 server. We implement the LCS in C and evaluate it on ODROID XU3 development boards featuring Samsung Exynos 5 Octa processors based on the ARM big.LITTLE architecture. We run 20 different benchmarks to test the HBM's ability to learn application specific models and the LCS's ability to deliver performance with near minimal energy consumption. We compare to published learning and control methods in a variety of settings. While many applications have inherent dynamics (*i.e.*,different processing phases), we explicitly test the ability to adapt to the unknown by running each application with other, random applications. We evaluate both the ability to deliver requested performance and the energy efficiency and find that the proposed approach:

- **Delivers Better Performance:** We quantify the ability to meet performance goals by calculating the error between the desired and delivered performance. In a single application setting, our approach achieves an average error of XXX%, compared to YYY% for existing learning methods and ZZZ% for existing control approaches (See Section **??**). In a multi-application setting, our approach avchieves an average error of YYY%, compared to WWW% and ZZZ% (See Section **??**).
- **Requires Lower Energy on Average:**
- **Performs Far Better in the Worst Case:**
- **Successfully Adapts to Dynamics: TODO: 1**

In summary, this paper makes the following contributions:

- Proposing the combination of a hierarchical Bayesian learning with a lightweight control system to meet the twin challenges of addressing complexity and dynamics to deliver performance with minimal energy on mobile systems.
- Demonstrating and implementing an interface for combining discrete learned models of resource usage with continuous control models of resource dynamics.
- Evaluating the implementation and comparison to existing, independent learning and control techniques.

2. Motivational Example

2.1. Machine Learning

2.2. Control Theory

2.3. Combining Learning and Control

3. Combined Learning and Control

Our goal is to combine learning with control to tackle the complexity and dynamics of modern mobile systems. We use a hierarchical Bayesian model to learn how resource allocation affects the power and performance of an application. This learning framework is computationally expensive so it runs on a remote server which sends learned models to the mobile system. Those models are used by a control system that dynamically tunes application resource usage to handle system dynamics. This section provides a brief overview of the relevant learning and control techniques used in this paper, and then describes the interface we propose to combine them.

3.1. Hierarchical Bayesian Learning

In general, all machine learning techniques take observations of some phenomena and turn them into a model to predict future outcomes. In our specific case, we want to take observations about application's performance and power given a resource allocation and predict how future application's will perform or how different resource allocations will affect the current application.

In this work, we use a hierarchical Bayesian model (HBM) to turn observations of applications' performance and power into predictions about how other, unobserved resource allocations will alter that performance and power. We use a HBM because it provides a statistically sound framework for learning across applications and devices. The HBM is also well suited to learning the complicated tradeoff spaces that arise from heterogeneity in modern mobile processors.

When selecting a learrning framework we must find a tradeoff between the specific and the general; *i.e.*,between frameworks that build application-specific models and frameworks that combine observations across applications. For example, the key to energy efficiency on heterogeneous mobile systems is knowing when to make use of

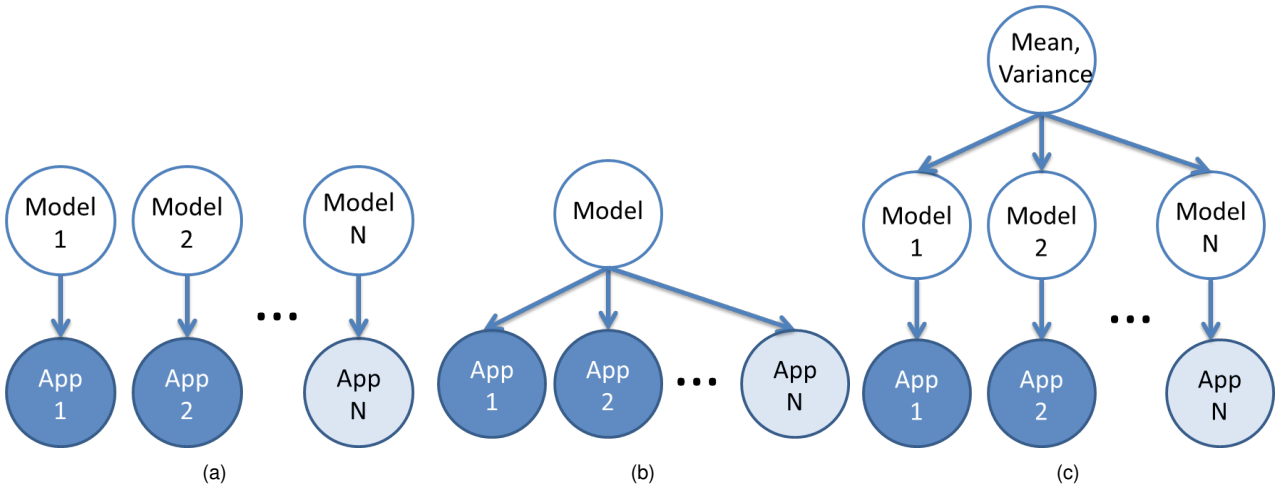

Figure 1: Comparison of online (a) offline (b) and hierarchical Bayesian models (c). The arrows represent dependences, circles are random variables, white circles are hidden variables that cannot be observed and must be learned, solid circles represent fully observed data, and shaded circles represent partially observed data. The online model is concerned only with the current application (labeled N), the offline model combines all observations into one model, and the HBM builds per-application models, but makes them conditionally dependent on one another so that prior observations can be used to increase the accuracy of the model built for application N.

the smaller, low-power cores []. An application-specific model will capture that precisely, but may require many observations before producing the correct model. A more general model will capture the trend, *e.g.*,when most applications should transition, but this general model might miss the key inflection point for some applications. We refer to application-specifc models as *online* because they build models for the current application and do not incorporate knowledge of other applicaitons. We refer to general models as *offline* as they use prior observations of other applications to predict the behavior of a new application.

The HBM provides a good balance between the online and offline approaches. The key differences between these three approaches are illustrated in Figure **??**. In this figure, circles represent random variables, directed edges show conditionally dependent relationships, and shading represents observability. A dark circle means we have seen all the data, a shaded circle means we have some partial observations, and the white circle means that the variable is unobservable. The models we are trying to learn areunobservable. A strictly online approach will handle each new application completely separately and ignore observations of previous applications. This approach never risks contaminating a model with unrelated observations, but it may take many observations of the current application to converge because it starts with no prior knowledge. The offline approach uses all observations from prior applications and will therefore converge very quickly; however, it is overly general and cannot learn features specific to a single application.

The HBM (illustrated in Figure **??**) is a good compromise. Each application has its own model, allowing specificity, but these models are conditionally dependent on some underlying probability distribution with a hidden mean and co-variance matrix. In practice, the HBM will estimate a model for a new application using a small number of observations and combining those observations with the large number of observations previously made of similar applications. Rather than over-generalizing, the HBM will use only similar applications to learn models. In addition, the HBM's accuracy increases as more applications are observed because more different types of behavior are represented in the pool of prior knowledge. Of course, the computational complexity of learning also increases with increasing applications, but this is why we offload the learnign to a remote server.

To provide intuition we offer a simple example. Suppose we have observed many prior applications, all of which are either completely compute-bound or completely memory-bound, and we have an equal number of both. The only resource we can allocate is clockspeed, which will increase the performance of compute-bound applications, but not memory-bound ones. When we need to work with a new application, we need to estimate its response to clockspeed. The online model will not use prior knowledge, but will observe many different clockspeeds for the new application, leading to high overhead. The offline model will predict the mean response of prior applications, meaning it will overallocate speed to memory-bound applications and under-allocate to compute-bound ones. The HBM will take a small number of observations

and then the prediction will combine those with prior knowledge: if the new observations show that clockspeed has no effect on performance the HBM will use only the prior memory bound applications to build its model, otherwise, it will use the compute-bound applications. In practice, the HBM can learn much more complicated tradeoffs by combining observations of the new application with prior knowledge of different types of behavior.

3.2. Controlling COmputing SYstems

Control theory provides a set of techniques for ensuring that a system meets some requirement and reasoning about the ability to meet that requirement in a dynamic environment where fluctuations cannot be predicted ahead of time.

3.3. Interfacing Control and LEarning

4. Experimental evaluations

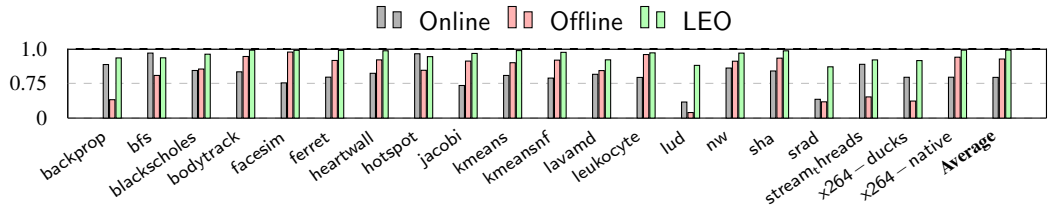

Figure 2: Accuracy performance

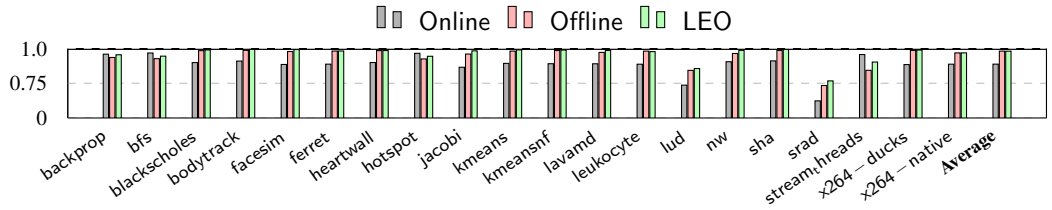

Figure 3: Accuracy power

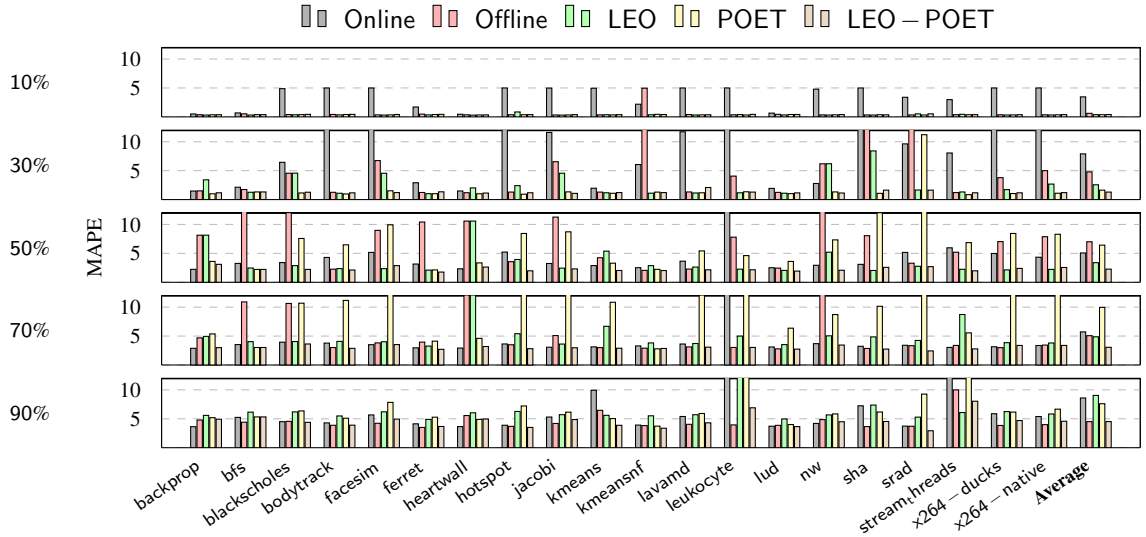

Figure 4: Single Applications MAPE

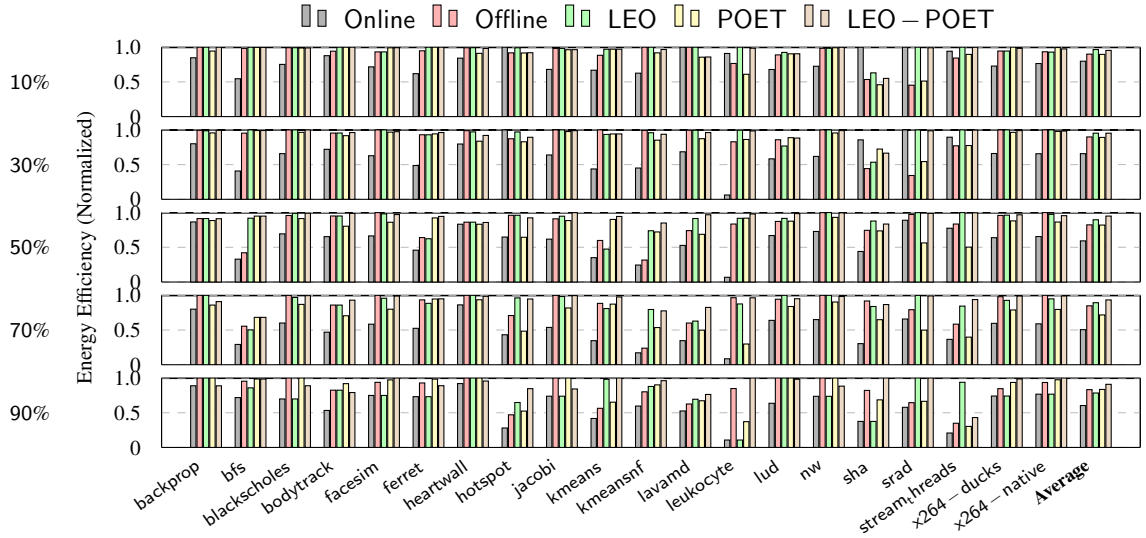

Figure 5: Single Applications energy

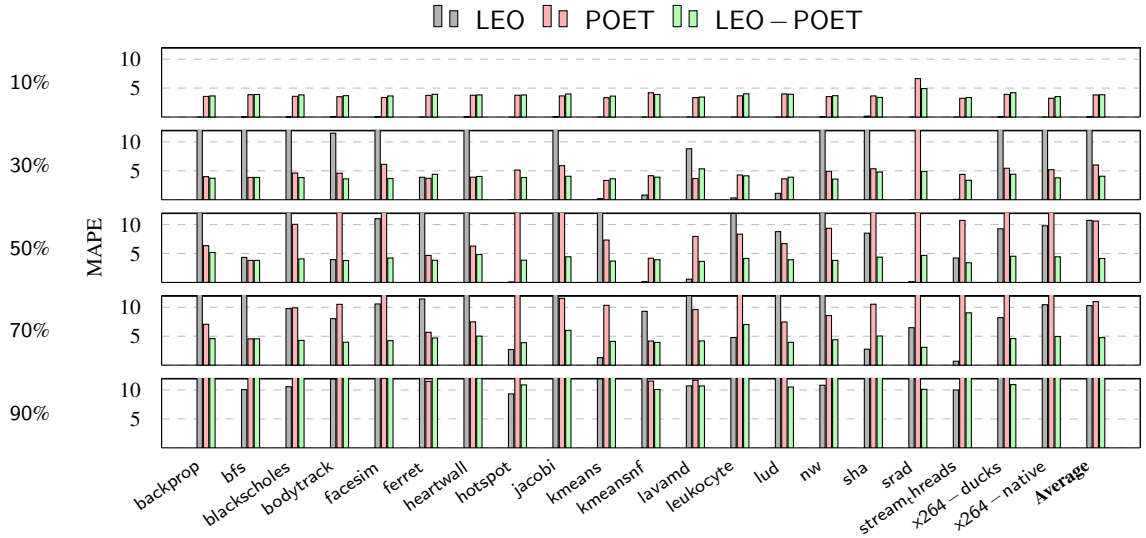

Figure 6: Multi Applications MAPE

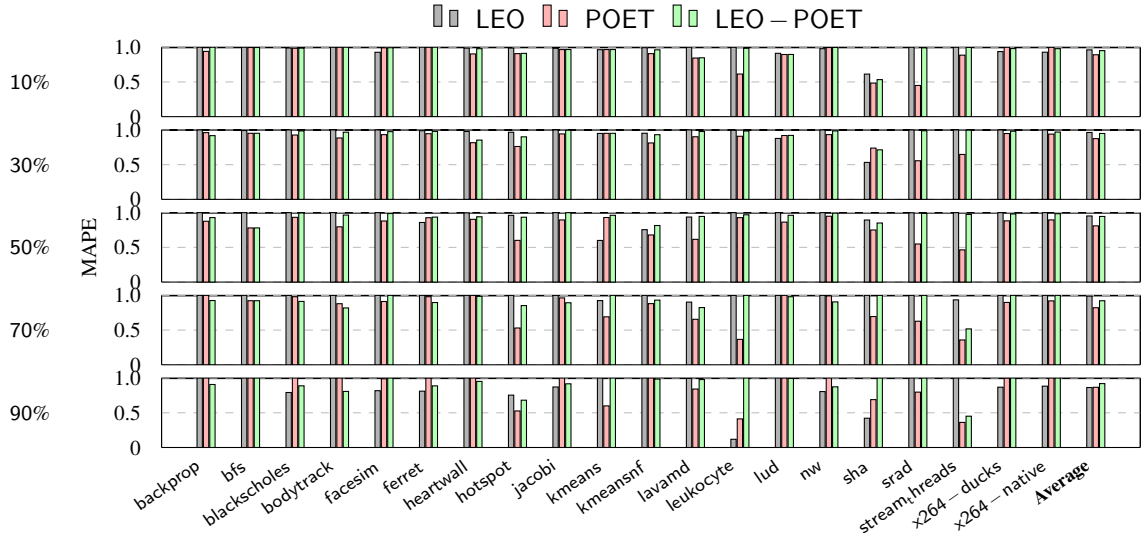

Figure 7: Multi Applications Energy