Proactive Machine Learning Based Runtime Management of Heterogeneous MPSoCs

Erika S. Alcorta, Matthew Barondeau, and Advisor: Andreas Gerstlauer
{esalcort, mebarondeau, gerstl} @utexas.edu

I. Problem Statement

In modern heterogeneous SoCs, runtime management is a complex problem as runtime systems must continuously navigate an extremely large configuration space based on changes in dynamic workload behavior. For example, the scheduler has to decide the mapping of threads to heterogeneous accelerators and cores while the power manager further tunes each processor based on the workload. Despite extensive research into improving runtime management, current solutions have large inefficiencies with a recent study [13] showing that up to 45% reduction in energy consumption is possible using an offline oracle. This is due to existing runtime systems being reactive and thus inherently lagging in making optimal decisions. To close this gap, a runtime should anticipate workload changes and act proactively. *In this project, we aim to investigate advanced machine learning (ML) models for proactive runtime decision-making, specifically for scheduling and power management, to improve performance and save energy on heterogeneous SoCs.*

Runtime management of modern systems must address two difficulties: (1) responding quickly to changes in workload behavior and (2) rising scheduling complexity due to increased heterogeneity. Proactive scheduling can address these shortcomings, but it requires knowledge of future workload behavior and of differences in behavior across heterogeneous targets. In prior work, our lab has recently shown ML approaches that can predict both the characteristics and duration of upcoming workload phases with accuracy over 90% [2]. This advancement enables prediction of upcoming workload behavior, but does not address how to schedule these phases on a heterogeneous system. To efficiently map upcoming phases to processors, the system requires knowledge of how upcoming workload behavior will change depending on the processor it selects. Our lab has previously investigated ML-based cross-platform prediction that can predict the power and performance of a program on a target processor with 97% accuracy [22]. By coupling this platform predictor with workload phase classification and prediction, a runtime can quickly determine the best processor configuration on which to run the upcoming workload phase based on system requirements. These two ML based models, when combined and integrated into the system scheduler, directly address the shortcomings of existing runtime management and promise to offer significantly improved performance and reduced power utilization over existing solutions.

Ultimately, the ML models obtained after investigating the integration of workload phase classification and prediction with cross-platform predictions will guide proactive runtime scheduling and power management decisions in heterogeneous systems. These models must satisfy various requirements to be deployed efficiently on an SoC platform. First, their energy consumption should be low enough not to outweigh the benefits of proactive scheduling. Second, predictions must be delivered promptly; i.e., predictions should be ready before they are needed. And third, the predictors must rapidly adapt to workload changes. To meet these requirements, we will investigate methods that minimize the complexity of ML models, such as pruning and quantization, and state-of-the-art methods that reduce training overhead, such as few-shot learning [19].

II. Prior Work

In the context of computer systems, classifying workload phases is an unsupervised learning problem consisting of clustering intervals of workload execution with similar behaviors together. Some researchers have focused on clustering either similar blocks of code, e.g., basic blocks [16], [15] or instructions [20], or using performance metrics, e.g., hardware counters [7], [18], [2], [3]. We will focus our project on hardware counter based methods because clustering blocks of code typically requires access to the source code or augmenting the hardware. Classified phases are used by phase prediction methods to learn patterns between them and foretell future phase behavior. This is a supervised learning problem that can be formulated as a multi-class classification problem. Early work in phase prediction proposed table-based methods [7], [20]. In our previous work [2], we surveyed and evaluated existing and novel workload phase prediction techniques on a single-core platform, comparing table-based with machine learning approaches. We demonstrated that machine learning models can generalize unseen phase patterns and thus achieve much higher prediction accuracy, at the expense of fast adaptability. We further expanded that work to support multi-core systems [3]. However, we only studied a homogeneous system. *In this work, we will research ML-based phase classification and prediction in heterogeneous architectures to anticipate future workload behaviors. Additionally, we will investigate ways to enable low-overhead online learning to overcome the slow adaptability of ML models and further benefit their applicability to runtime management.*

The idea of cross-platform prediction was introduced by our group in [21], [22]. Cross-platform models learn the relationship of program executions on different platforms. The idea consists of profiling code on a host core and use collected statistics to predict expected program behavior on other targets, including CPUs, GPUs, FPGAs, and dedicated accelerators [1]. While our work demonstrated its applicability to rapid system design and prototyping, others have built on top of it to demonstrate its applicability to runtime task scheduling in heterogeneous architectures [12], where the

authors used a simple runtime heterogeneous scheduler to evaluate various cross-platform predictors. However, cross-platform predictions have not been studied in combination with phase prediction and advanced heterogeneous schedulers. *We will expand upon prior work in cross-platform prediction and apply it as part of the runtime on a heterogeneous system to proactively schedule and tune tasks across both workload phases and microarchitectures.*

Heterogeneous schedulers are designed to detect changes in dynamically changing workload behavior and, based on configurable system metrics, modify where the workload is running. Early efforts in dynamic heterogeneous scheduling relied on heuristics [10], [4] to trigger migration between processors and greatly outperformed offline static predictions. The schedulers were refined to classify workload phase changes and save the scheduler's decision into a table alongside the phase information [14]. More recent work has applied ML to classify heterogeneous phases [17] and determine the processor on which to run the phase [6] [11], showing significantly higher accuracy than table-based approaches. However, these schedulers only react to workload changes, missing optimization opportunities. *In this project, we will deliver an advanced and proactive runtime management solution that further optimizes the energy and performance of heterogeneous MPSoCs.*

III. Detailed Tasks and Schedule

Our end goal is to create a runtime that can use accurate predictions of future workload behavior to better utilize a heterogeneous architecture. The final product is an adaptive ML-based runtime manager that schedules workload phases across heterogeneous components and tunes other core configuration options based on their future behavior.

A. Cross-Platform Prediction

Our lab has a phase-prediction model that works by sampling hardware counters, classifying, and then predicting upcoming workload phases. In a heterogeneous platform, such as an ARM big.LITTLE system, the little processor architectures will have different hardware characteristics attributed to each counter than the more powerful processors. An example of this is the difference in branch predictors, cache sizes, and issue width will affect the mispredictions, cache misses, and CPI, respectively. To predict upcoming behavior on a workload that has potentially switched many times between processors, the hardware counters will need to be transformed to have a single point of reference that can be learned. To do this for potentially many heterogeneous steppings, we can transform the performance data from several processors to that of a single reference processor, for instance, the most powerful processor in the system, and then predict future workload behavior using performance counters based on the single reference point.

To unify all performance counter streams into a single data source, we must modify existing cross-platform ML models to directly predict hardware performance counters across processors. Current models run a set of benchmarks on one processor and then, based on the hardware counters, predict the power and performance of running the same benchmarks on another architecture. Our lab has shown that the phase prediction models achieve the highest accuracy when looking at a history of hardware counters rather than a history of phases or interpreted history [3]. Therefore, for best accuracy, we need to create a model that, for the target heterogeneous platform, can scale the performance characteristics of a workload running on one processor to that of another processor. To train this model, we will first create a dataset by running a set of benchmarks that cover the breadth of execution characteristics on both the stronger and weaker processor cores in the system and record performance counter information. We will then label the data, align the workloads so that the model only learns the difference in execution of the same instructions, and apply supervised learning training techniques to generate a model. *In this stage, we will create an offline model, trained using hardware performance counters, that will transform performance information between heterogeneous processors into a unified trace with a single reference for performance.*

B. Heterogeneous Phase Classification & Prediction

Once we have a method of converting performance counter information in a heterogeneous system, we plan to use the set of workloads that we ran on the heterogeneous platform to create an oracle label for their phases. Our data set will be composed of an aggregation of unified traces (the output of cross-platform prediction). In this offline stage, we will use a clustering algorithm to group similar behaviors, i.e., vectors of hardware counter values, together. We anticipate some difficulty in tuning as we may introduce some additional noise by including the output of our modified cross-platform model in the input set. Therefore, sanity checks will be introduced to verify that workloads have the same definition of phases across their executions on different architectures. We will then use the generated phase labels to create the ground truth of the phase predictor's data set. These predictors observe a fixed-size window of hardware counter values as inputs and estimate a fixed-size window of upcoming phase labels. We will evaluate phase classification and phase prediction models in different combinations to find the model that provides the best trade-off in terms of prediction accuracy and time complexity. *At the end of the stage, we will have an offline phase classifier and phase predictor that can predict upcoming phases regardless of which processor in the system it has been executed on.*

C. Offline Proactive Heterogeneous Scheduling

In the third stage, we will extend the models we have developed thus far to inform scheduling decisions. At the end of stage two, our models could, given an execution trace of a workload, predict the upcoming phase. We will further extend the classification model here to classify not only the phase that is executing, but on which processor and with what voltage and frequency setting the phase should be executed. With the additional classification information,

whenever an upcoming phase is predicted, if the phase is deemed to best run a different processor, the runtime can migrate the workload. This migration introduces several additional difficulties when training the models, so at this stage, we will have to refine some of our work with the classifier. Once workloads can change voltage and frequency characteristics, in addition to the processor, many more potential execution traces are created. These additional options introduce difficulty in accurately classifying phases, and thus, we will have to expand the classifier to learn how to classify these new phases. While the model must get more complicated to learn these additional features, we must also balance runtime overhead and will investigate reducing the complexity of the ML models used in our classifier through strategies such as pruning and quantization [5].

Finally, we will also examine the impact of phase duration on scheduling decisions. In addition to upcoming phase behavior, our phase predictor is able to determine the duration of the phase. Given information about the workload characteristics and phase duration, the best scheduling decision can be to not migrate tasks in the case of short alternating workload phases. *At the end of this stage, we will have integrated offline-trained ML models with the heterogeneous runtime manager to give information about upcoming phase behavior and enable proactive scheduling of workloads.*

D. Online Proactive Heterogeneous Scheduling

Our final stage will focus on developing an online learning solution for phase prediction and classification that can adapt to dynamically changing workloads. Prior to this stage, all of the models were trained offline and evaluated online, but this limits the models as new workloads may exhibit characteristics that the models are not well trained for. To remedy this issue, we will add online feedback to our prediction and classification so we can learn the best behavior while workloads change. However, the most accurate models can be costly to train in both computation time and memory complexity. Recent advancements in continual learning research have proposed methods, such as importance sample [8], [9] and few-shot learning [19], that minimize the training cost of machine learning models by using fewer samples to train. We aim to use these reduced-cost methods to integrate proactive policies into the runtime scheduler that will lower overall system energy use via timely, lightweight, and accurate heterogeneous workload predictions. *At the end of this stage, we will have achieved our goal of delivering a proactive runtime task scheduler and power manager for real-world heterogeneous MPSoCs.*

TABLE I
ONE YEAR PROJECT TIMELINE

Months 1 - 3	Train the cross-platform model to predict hardware counters on heterogeneous architecture
Months 4 - 6	Create and train a heterogeneous phase classification & prediction model
Months 7 - 9	Integrate the offline ML models into scheduler
Months 10 - 12	Extend ML models for online training

IV. TIMELINE, TEAM, AND EXPERTISE

A brief outline of our one-year plan is shown in Table I. This team is eminently qualified to execute this plan based on their applicable experience listed below.

Erika S. Alcorta is an expert in machine learning for system design, having published a book chapter in the field [1] and multiple papers [2], [3], including a best paper award for her work in ML-based workload phase classification and prediction. Her current research involves a collaboration with Ampere Computing on developing an ML-based runtime system extension to optimize the performance of next-generation ARM-based many-core server platforms.

Matthew Barondeau's expertise is in computer architecture, hardware-assisted threading and scheduling. He holds the Virginia & Ernest Cockrell Jr. Fellowship and collaborates with Google researching efficient utilization of processor architectures. Matthew is also an expert in heterogeneous architectures due to his prior experience at ARM, Tactical Computing Labs, and Nvidia working on architectures ranging from embedded to exascale.

Andreas Gerstlauer (PI) is a Professor, Engineering Foundation Endowed Faculty Fellow and Associate Chair for Academic Affairs in the Electrical and Computer Engineering Department at UT Austin. Dr. Gerstlauer's research expertise and publications span IoT, embedded and edge computing, heterogeneous system-on-chip design, and ML-based system modeling. He is the leader of the system-level architecture and modeling (SLAM) research group, where he has advised over 30 students across multiple projects. Work in his group was recognized with the 2021 MLCAD, 2016 DAC and 2015 SAMOS best paper awards, several best paper nominations from, among others, DAC, DATE and HOST conferences, and as one of the most influential contributions in 10 years at DATE in 2008.

V. CONCLUSION

Heterogeneous runtime management is a complex problem due to the large system configuration space and dynamic workload behaviors. Existing solutions observe the current state or phase of the system to select an optimized configuration. However, these solutions can only react to workload changes, diminishing the optimization opportunities. Conversely, we propose a proactive solution, which makes decisions based on estimations of future workload behaviors. We proposed to generate such estimations by integrating cross-platform prediction with workload phase prediction.

The team is actively participating in presentations and discussions with industry researchers and have already produced research in the realm of proactive and heterogeneous systems. With Susy's machine learning and phase prediction expertise, Matthew's knowledge of threading and scheduling, and Dr. Gerstlauer's leadership in performance modeling and heterogeneous architectures, the team has the right skill-set to develop a proactive heterogeneous scheduler that can best utilize increasingly heterogenic future systems.

REFERENCES

[1] E. S. Alcorta, P. Brisk, and A. Gerstlauer. *ML for System-Level Modeling*, pages 545–579. Springer International Publishing, Cham, 2022.

[2] E. S. Alcorta and A. Gerstlauer. Learning-based workload phase classification and prediction using performance monitoring counters. In *ACM/IEEE 3rd Workshop on Machine Learning for CAD (MLCAD)*, 2021.

[3] E. S. Alcorta and A. Gerstlauer. Learning-based phase-aware multi-core cpu workload forecasting. *ACM Transactions on Design Automation and Electronic Systems (TODAES)*, 2022. Just Accepted.

[4] A. Annamalai, R. Rodrigues, I. Koren, and S. Kundu. Dynamic thread scheduling in asymmetric multicores to maximize performance-per-watt. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum*, pages 964–971, 2012.

[5] Z. Chen, H. T. Blair, and J. Cong. Energy-efficient lstm inference accelerator for real-time causal prediction. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 27(5), jun 2022.

[6] A. Edun, R. Vazquez, A. Gordon-Ross, and G. Stitt. Dynamic scheduling on heterogeneous multicores. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1685–1690, 2019.

[7] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pages 359–370, 2006.

[8] T. B. Johnson and C. Guestrin. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31, 2018.

[9] A. Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

[10] R. Kumar, D. Tullsen, P. Ranganathan, N. Jouppi, and K. Farkas. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pages 64–75, 2004.

[11] C. V. Li, V. Petrucci, and D. Mossé. Exploring machine learning for thread characterization on heterogeneous multiprocessors. *SIGOPS Oper. Syst. Rev.*, 51(1):113–123, sep 2017.

[12] A. Prodromou, A. Venkat, and D. M. Tullsen. Platform-agnostic learning-based scheduling. In D. N. Pnevmatikatos, M. Pelcat, and M. Jung, editors, *Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 142–154, Cham, 2019. Springer International Publishing.

[13] J. Roeder, B. Rouxel, S. Altmeyer, and C. Grelck. Energy-aware scheduling of multi-version tasks on heterogeneous real-time systems. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing (SAC)*, 2021.

[14] L. Sawalha, S. Wolff, M. P. Tull, and R. D. Barnes. Phase-guided scheduling on single-isa heterogeneous multicore processors. In *2011 14th Euromicro Conference on Digital System Design*, pages 736–745, 2011.

[15] A. Sembrant, D. Black-Schaffer, and E. Hagersten. Phase behavior in serial and parallel applications. In *IEEE International Symposium on Workload Characterization (IISWC)*, pages 47–58, 2012.

[16] A. Sembrant, D. Eklov, and E. Hagersten. Efficient software-based online phase classification. 2011.

[17] T. Sondag, V. Krishnamurthy, and H. Rajan. Predictive thread-to-core assignment on a heterogeneous multi-core processor. In *Proceedings of the 4th Workshop on Programming Languages and Operating Systems*, PLOS '07, New York, NY, USA, 2007. Association for Computing Machinery.

[18] K. Taht, J. Greensky, and R. Balasubramonian. The pop detector: A lightweight online program phase detection framework. 2019.

[19] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[20] W. Zhang, J. Li, Y. Li, and H. Chen. Multilevel phase analysis. *ACM Trans. Embed. Comput. Syst.*, 14(2), mar 2015.

[21] X. Zheng, L. K. John, and A. Gerstlauer. Accurate phase-level cross-platform power and performance estimation. In *Proceedings of the 53rd Annual Design Automation Conference (DAC)*, 2016.

[22] X. Zheng, L. K. John, and A. Gerstlauer. LACross: Learning-Based Analytical Cross-Platform Performance and Power Prediction. *International Journal of Parallel Programming*, 45(6):1488–1514, Dec. 2017.