The PRiME Framework: Application- & Platform-agnostic System Management

Graeme M. Bragg, Domenico Balsamo, Charles Leech and Geoff V. Merrett
School of Electronics and Computer Science, University of Southampton, UK. Email: gmb@ecs.soton.ac.uk

***Abstract*—Multi-core and heterogeneous processors in modern embedded platforms have increased in complexity to provide both high-performance and energy-efficient execution of applications. As a result, the runtime management and control of these platforms has become a non-trivial process with many different approaches being reported in the literature. In addition, applications have become increasingly dynamic to exploit these processors runtime adjustable parameters that can be tuned to optimise and influence their behaviour. These two challenges motivate the need for a consistent approach to runtime management that is cross-platform and generic in the support of applications. This abstract presents the PRiME Framework, a cross-layer framework that enables application- and platform-agnostic runtime management by separating a system into three distinct layers connected by an API and cross-layer constructs called knobs and monitors. The motivation for the framework's underlying concepts are discussed and its use is demonstrated with a range of platforms and applications.**

An open-source implementation of this framework has been released. [1]

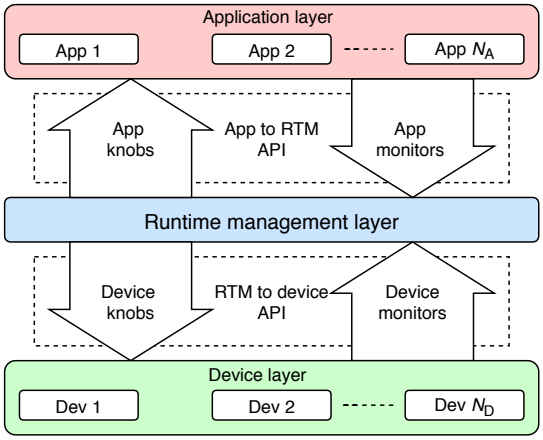

Fig. 1. Cross-layer framework and APIs enabling communication between the application, runtime management and device layers using knobs and monitors.

I. Introduction and Related Work

Runtime adaptation and control are crucial to the efficient execution of applications with varying performance requirements on modern embedded heterogeneous platforms. Users demand ever-greater energy efficiency and throughput from these systems, therefore proactive optimisation of their performance, energy and reliability are key research challenges. Runtime management represents an essential paradigm in tackling these challenges by enabling optimisation and tradeoffs between computational quality, application throughput, system reliability and energy efficiency.

System behaviour can be optimised through the use of various techniques whilst satisfying application requirements. These include dynamic voltage and frequency scaling (DVFS) [1], per-core power gating [2], concurrency throttling [3], dynamic task mapping and thread migration [4]. Furthermore, many different approaches to runtime management exist, from control based approaches that rely on off-line profiling for task mapping [4] to on-line learning based algorithms [5]. While RTMs are typically designed to address general challenges, such as energy efficiency or thermal management, they are largely implemented on specific platforms or with specific classes of application, *e.g.* multimedia [6] or image processing [7].

This work was supported by the PRiME programme grant EP/K034448/1 (http://www.prime-project.org) and EPSRC grant EP/L000563/1.

[1]Available at: https://github.com/PRiME-project/PRiME-Framework

Runtime management can be enhanced by introducing abstraction between the different parts of a system with an interface that is consistent across different types of platform and application. Fig. 1 shows how this can be achieved by separating the system into three distinct layers, *i.e.* application, runtime management and device, that connected through an API with cross-layer constructs called knobs and monitors. Specifically, knobs allow the tuning of device and application parameters by the runtime management layer, while monitors enable the measurement of device properties and the observation of application behaviour.

Several framework exist that achieve partial abstraction; however, the PRiME framework is the first to provide a fully application- and platform-agnostic solution. Frameworks based on the Heartbeats API [8]–[12] provide abstraction between application and runtime management with monitors that use periodic signals to convey application throughput. Heartbeats-based frameworks cannot convey non-temporal metrics, such as accuracy or quality, which can be used to provide additional opportunities for optimisation by expanding the operating space. For example, exposing throughput and quality can enable tradeoffs between these metrics and power. ARGO [13] provides more comprehensive abstraction of application to runtime management interaction by providing both knobs and monitors; however, ARGO does not consider interaction between the runtime management and device layers.

The remainder of this abstract provides an overview of the PRiME Framework and demonstrates its operation with multiple applications and platforms.

II. Framework Concepts

This section summarises the PRiME Framework's underlying concepts.

Structure: Separation of the system into distinct layers reduces the design complexity of runtime management approaches and provides flexibility during operation. The application layer comprises any number of software processes, while the device layer includes the platform and its software drivers. The runtime management layer comprises a runtime manager (RTM) that is responsible for the control and monitoring of the other layers. This separation ensures portability and cross-compatibility; applications and device drivers only need to be written once for use with any implemented RTM.

Communication: Knobs and monitors facilitate communication between the layers. Bounds, attached to both knobs and monitors in the form of *minima* and *maxima*, convey targets and constraints to the runtime management layer. Knob bounds represent a range of *allowed* values while monitor bounds represent a range of *desired* values. An RTM's primary objective is to ensure that the monitor values of all applications and the device remain within bounds.

Weights: Applications may have multiple monitors that expose different performance objectives with differing priorities. In the PRiME Framework, priority is expressed as a numeric weight attached to each monitor. These weights instruct the RTM to expend proportional effort in optimising each monitor's value. In a similar manner, application priority is indicated through an attached weight such that a higher level of consistency can be ensured by foreground processes in a multi-tasking scenario.

Concurrency: Real-world systems commonly execute applications concurrently that compete for hardware resources. The RTM is required to manage resources so that each application meets its performance targets. The PRiME framework provides a mechanism to identify and manage concurrently executing applications. This allows knobs and monitors to be grouped and traded-off between applications by the RTM.

Types: Knobs and monitors each have a *type* selectable from a discrete set of options. This gives hints to the RTM, which simplifies the process of determining the function of knobs and the properties represented by monitors, and represents a compromise between complete agnosticism and the full provision of information.

Spaces: Knobs and monitors are expressed in unit-less formats to maintain application and device agnosticism. The PRiME Framework allows discrete- and continuous-valued versions of each knob and monitor so that the appropriate optimisation and control process can be used by the RTM. For example, a boolean choice (*i.e.* $\{0, 1\}$) does not require iterative convergence. These spaces enable the translation of application-specific information into agnostic sets.

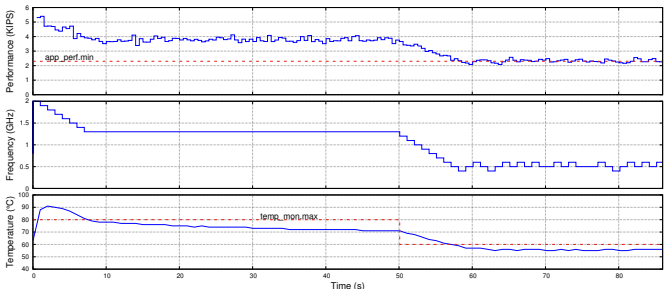

Fig. 2. Device temperature optimisation under application performance constraints using the controller RTM, including dynamic adjustment of the temperature threshold from 80 to 60 °C. Reproduced from [16].

Adaptability: All bounds and weights are adjustable at runtime and no restrictions are placed on when update to these can occur. Most commonly, applications create their knobs and monitors before being executed, however no limitation is imposed on such events occurring partway through application execution. Applications are allowed to be attached to and detached from the framework at any point during runtime. This capability is in contrast to existing frameworks, most of which assume a constant application set, contrary to the typical use of many contemporary systems.

III. Framework Demonstration

Experiemnts have been carried out to validate the framework across a range of platforms, including multi-core heterogeneous and many-core homogeneous systems, and a range of applications from the numerical, multimedia and iterative domains. The PRiME Framework has been used for exploring reliability tradeoffs [14], demonstrating the benefit of exposing application monitors and controls [15] and to identify pareto-optimal points of one application operating on different heterogeneous platforms [16]. This section presents a selection of these experiments to demonstrate the framework's capabilities.

A. Agnostic Runtime Management

Fig. 2 demonstrates the basic operation of the framework and dynamic nature of knobs and monitors with a simple illustrative runtime controller that attempts to keep device temperature below a specified maximum, `temp_mon.max`, whist maximising application throughput above `app_perf.min`. This experiment was carried out on the CPU cores of an Odroid XU3, a heterogeneous multi-core system with two quad-core CPU clusters and a GPU, and a Whetstone numerical benchmark application. The application exposes a throughput monitor and the device exposes a temperature monitor and frequency knob. Initially, the controller set the device frequency to maximum and observed the device temperature until it exceeded the limit specified by `temp_mon.max` (80°C). The controller then reduced the device frequency until the temperature bound was met. At 50 seconds, the device reduced the value of `temp_mon.max` to 60°C and the controller responded by reducing the device frequency further.

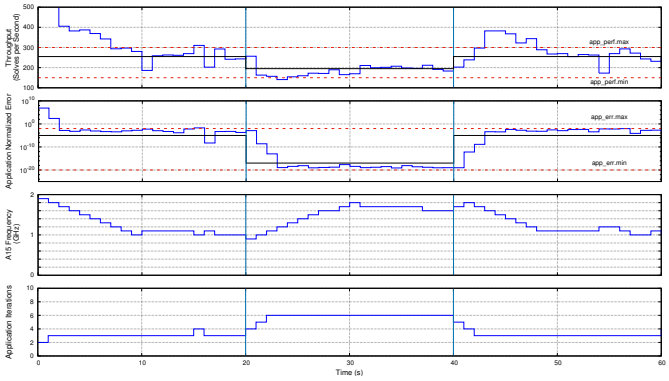

Fig. 3. Demonstration of application monitor weights for performance and error being used to influence runtime management of the Jacobi application. Vertical lines show when the monitor weights are updated by the application and black lines indicate the new target values. Reproduced from [15]

The controller used for this experimenent contains no device or application specific code. As such, it is application- and platform-agnostic and could operate with any application and platform that exposes the correct knobs and monitors.

B. Monitor Weighting and Tradeoffs

Fig. 3 shows how monitor weighting can be used to priotitise the optimisation of one application metric over another and how application and device knobs enable this. The controller in this experiment uses the application monitor weighting to set proportional targets for each monitor within their bounds and then adjusts device and application knobs to meet these targets. The Jacobi iterative method, an algorithm that is commonly embedded in real-world applications as a computational kernel for the purpose of generating an approximation to a physical process, was used as the application for this experiment. Jacobi exposes throughput and accuracy monitors along with an iterations knob. Again, the Odroid XU3 was used as the experimental platform.

Initially, the throughput monitor (top graph in Fig. 3) had a higher weighting than the accuracy monitor (second graph in Fig. 3). As such, the controller determined proportional targets and adjusted the frequency and iterations knobs to prioritise performance over accuracy. At 20 seconds, the weights of the two monitors were swapped and the controller set new target values that prioritise accuracy over performance. To meet these updated targets, the controller increased the iterations knob to improve accuracy. This reduced the throughput below the target so the operating frequency was increased. At 30 seconds, the weights were swapped again and the controller responded.

C. Overheads

As with any form of abstraction, the framework introduces overheads. Energy overheads were evaluated by implementing two recently reported runtime management approaches within the framework and comparing to their behaviour without the framework. Latency overheads were evaluated by timing the operation of API interactions. The framework was found to introduce a modest energy overhead of 4.23% to 5.48% and a latency overhead of 80–200 μs per API call [16].

IV. Conclusions and Future Work

This abstract has introduced the PRiME Framework and shown how it enables application- and platform- agnostic runtime management for contemporary many-core, multi-core and heterogeneous systems with acceptable overheads. Work is ongoing to provide further validation of the framework and investigations have begun into its application to future many-core systems. While the framework enables fully agnostic runtime management of systems, existing approaches still rely on having specific knobs and monitors available to optimise. Work is required to investigate the feasability of a fully general runtime management approach that can operate with any application and platform combination.

References

[1] A. Das *et al.*, "Reinforcement Learning-based Inter- and Intra-application Thermal Optimization for Lifetime Improvement of Multicore Systems," in *ACM/EDAC/IEEE Design Automation Conf.*, 2014.

[2] A. M. Rahmani *et al.*, "Reliability-aware runtime power management for many-core systems in the dark silicon era," *IEEE Trans. on VLSI Syst.*, vol. 25, no. 2, 2017.

[3] R. Cochran *et al.*, "Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-44. New York, NY, USA: ACM, 2011, pp. 175–185.

[4] B. K. Reddy *et al.*, "Inter-cluster Thread-to-core Mapping and DVFS on Heterogeneous Multi-cores," *IEEE Trans. on Multi-Scale Comput. Syst.*, vol. PP, no. 99, pp. 1–1, 2017.

[5] L. A. Maeda-Nunez *et al.*, "PoGo: An Application-specific Adaptive Energy Minimisation Approach for Embedded Systems ," in *HiPEAC Workshop on Energy Efficiency with Heterogenous Comput.*, 2015.

[6] Y. G. Kim *et al.*, "Enhancing Energy Efficiency of Multimedia Applications in Heterogeneous Mobile Multi-core Processors," *IEEE Trans. Comput.*, vol. 66, no. 11, 2017.

[7] S. Yang *et al.*, "Adaptive Energy Minimization of Embedded Heterogeneous Systems using Regression-based Learning," in *Int'l Workshop on Power and Timing Modeling, Optim. and Sim.*, 2015.

[8] H. Hoffmann *et al.*, "Application Heartbeats: A Generic Interface for Specifying Program Performance and Goals in Autonomous Computing Environments," in *Int'l Conf. on Autonomic Comput.*, 2010.

[9] H. Hoffmann *et al.*, "A Generalized Software Framework for Accurate and Efficient Management of Performance Goals," in *Int'l Conf. on Embedded Software*, 2013.

[10] E. Paone *et al.*, "Evaluating Orthogonality between Application Auto-tuning and Run-time Resource Management for Adaptive OpenCL Applications," in *IEEE Int'l Conf. on Appl.-specific Syst., Arch. and Proc.*, 2014.

[11] F. Gaspar *et al.*, "A Framework for Application-guided Task Management on Heterogeneous Embedded Systems," *ACM Trans. on Arch. and Code Optim.*, vol. 12, no. 4, 2015.

[12] A. Baldassari *et al.*, "A Dynamic Reliability Management Framework for Heterogeneous Multicore Systems," in *IEEE Int'l Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Syst.*, 2017.

[13] D. Gadioli *et al.*, "Application Autotuning to Support Runtime Adaptivity in Multicore Architectures," in *Int'l Conf. on Embedded Comput. Syst.: Architectures, Modeling, and Simulation*, 2015.

[14] V. Tenentes *et al.*, "Hardware and software innovations in energy-efficient system-reliability monitoring," in *2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2017.

[15] C. Leech *et al.*, "Application Control and Monitoring in Heterogeneous Multiprocessor Systems," in *13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, Submitted.

[16] G. M. Bragg *et al.*, "An Application- and Platform-agnostic Control and Monitoring Framework for Multicore Systems," in *3rd International Conference on Pervasive and Embedded Computing (PEC)*, Submitted.