Approximation-Aware Coordinated Power/Performance Management for Heterogeneous Multi-cores

Anil Kanduri [†], Antonio Miele[‡], Amir M. Rahmani[§¶], Pasi Liljeberg[†], Cristiana Bolchini[‡], Nikil Dutt[§]
[†] University of Turku, Finland [‡] Politechnico de Milano, Italy [§] University of California, Irvine, USA [¶] TU Wien Austria
{spakan@utu.fi, antonio.miele@polimi.it, amirr1@uci.edu, pakrli@utu.fi, cristiana.bolchini@polimi.it, duttdutt@ics.uci.edu}

ABSTRACT

Run-time resource management of heterogeneous multi-core systems is challenging due to i) dynamic workloads, that often result in ii) conflicting knob actuation decisions, which potentially iii) compromise on performance for thermal safety. We present a run-time resource management strategy for performance guarantees under power constraints using functionally approximate kernels that exploit accuracy-performance trade-offs within error resilient applications. Our controller integrates approximation with power knobs - DVFS, CPU quota, task migration - in coordinated manner to make performance-aware decisions on power management under variable workloads. Experimental results on Odroid XU3 show the effectiveness of this strategy in meeting performance requirements without power violations compared to existing solutions.

CCS CONCEPTS

• **Hardware → On-chip resource management**; • **Computer systems organization** → *Multicore architectures*;

KEYWORDS

On-chip resource management, approximate computing

1 INTRODUCTION

Heterogeneous multi-core systems that are resource constrained exacerbate the run-time management challenge with i) diverse performance requirements of applications ii) fixed power budgets iii) dynamic workload characteristics iv) core-level heterogeneity. Existing resource management techniques use Dynamic Voltage and Frequency Scaling (DVFS) [2], task migration [15], power gating and CPU quota scaling [18, 19] etc., for power optimization. Such techniques propose joint actuation of power knobs [2, 14, 15, 18, 25] and application characteristics exploitation [7, 9] to maximize performance within the fixed power budgets. These strategies minimize the effect of power actuation on performance, yet inevitably compromise on performance for thermal safety. This specifically affects a class of streaming applications from domains such as machine learning, artificial intelligence, multimedia processing, computer vision and Internet-of-Things [12, 21]. These applications may include user interaction and serve as intermediate results to other computational kernels (e.g., computer vision in self-driving vehicles), making them latency-sensitive. Typical power actuation techniques cannot provide real-time performance guarantees required to ensure user satisfaction and responsiveness. At the same time, such applications are relatively error resilient (i.e., approximable) due to their algorithmic tolerance nature, redundant input data and perceptive end results. Approximate computing [4] has become a popular choice for energy and performance gains exploiting the error resilience of certain application domains. Functionally approximate kernels present a Pareto space of accuracy-performance trade-offs that can be leveraged at run-time to achieve better performance within lower power and/or energy resources, for an acceptable loss in accuracy [1, 22]. In this work, we propose a novel run-time resource management strategy for heterogeneous multi-core systems running dynamic and unknown workloads, exploiting approximation together with traditional power knobs. Our strategy jointly coordinates with power knobs and approximation knob to scale accuracy of the applications as per available resources, when power/performance requirements cannot be met with traditional knobs. Subject to system dynamics and applications' requirements, we switch application's mode of execution from accurate to approximate to meet i) power constraints - by lowering resources to approximate kernels, and/or ii) performance requirements - by opportunistically reducing workloads. Coordinated usage of approximation knob complements the traditional power knobs to overcome their limitations and conflicting decisions. The major contributions of our work are:

- Coordinated actuation of approximation knob with other power knobs to satisfy both power and performance constraints subject to system dynamics,
- Resource allocation policy for heterogeneous multi-core systems running unknown/dynamic workloads using the coordinated actuation strategy through DVFS, CPU quota assignment, task migration and approximation,
- Evaluation of our strategy on a real test-bed of Odroid XU3 8-core ARM big.LITTLE platform, over a set of error resilient machine learning applications.

Organization: Motivation for coordinated actuation of approximation with other power knobs and related work are presented in Sections 2 and 3, highlighting the novelty of our solution. Section 4 discusses the background on system architecture. The proposed resource management strategy for power-performance actuation is described in Section 5. Section 6 presents an evaluation of our approach with selective workloads on Odroid XU3 HMP platform. Finally, Section 7 concludes the paper with future work.

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00

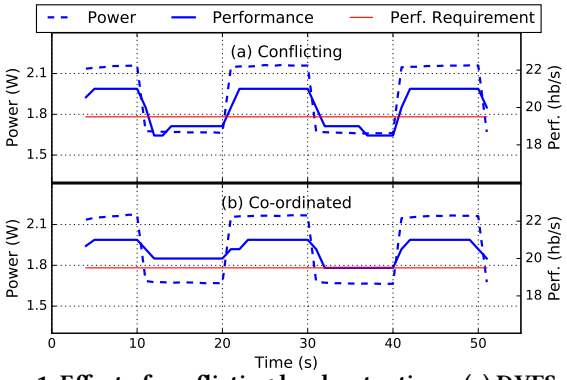

Figure 1: Effect of conflicting knob actuations. (a) DVFS and APX are triggered without co-ordination [13]. (b) DVFS and APX are tuned with co-ordination.

2 MOTIVATION

We demonstrate the effect of combining approximation (APX) knob with power actuation through an example, using the control strategy presented in [13]. For simplicity, we use only DVFS as the power knob, and loop perforation [22] based approximation as the APX knob. As a test case, we use least squares curve fitting application over 24000 pairs of inputs, and 10% of the loops skipped to realize the approximate version. Figure 1(a) shows the power consumption and corresponding performance (measured in heartbeat/s, as described in Section 4) of the application. We assume $2W$ as the threshold for power actuation, 10 seconds (for presentation ease) as the control period for resource management decisions , along with a target performance requirement (the red line). At $t = 10s$, power exceeds the threshold, triggering DVFS to reduce power, while performance requirements are satisfied. Subsequently, power consumption is reduced during the control period, also reducing the performance below the requirement. At $t = 20s$, performance requirements are not met - triggering the APX knob, while power consumption is below the threshold - triggering DVFS (upscaling). With the uncoordinated actuation decisions performed by [13], performance is restored with some accuracy loss, while power consumption again exceeds the threshold during the following control period. At $t = 30s$, performance requirements are met, hence the application is switched to accurate execution; DVFS is triggered again to lower power consumption. A similar oscillation between accurate-approximate executions and high-low frequencies will continue in the subsequent control periods. Although the performance violation at $t = 10s$ could be recovered with approximation at $t = 20s$, the lack of coordination between DVFS and approximation either over-compensates or under-compensates the knobs' actuation. This occurs because of the *reactive nested loop structure of such control policies (as in [13]), where power and performance actuation happens independently, and is mutually agnostic.*

Figure 1(b) shows power and performance for the same scenario, with a coordination among DVFS and approximation. At $t = 10s$, DVFS is triggered to address power violation. At the same time, considering the performance loss with power actuation, approximation is triggered pro-actively by predicting the potential loss in performance due to DVFS (downscaling). As a result, performance requirements are met during the subsequent control period despite DVFS downscaling. At $t = 30s$, VF is upscaled with availability of power headroom and the same is indicated to the performance manager. This allows the performance manager to make informed decisions on performance actuation, which in turn restores the accurate mode of execution. As a conclusion, while combination of approximation and DVFS addresses the performance loss due to power actuation to an extent, tight coordination between power and performance actuation provides better results in i) meeting performance requirements often, and ii) minimizing the accuracy loss due to approximation. In a realistic scenario, the resource management policy has to consider 1) a wider set of power knobs viz., DVFS, CPU quota assignment, task migration and fine-grained levels of approximation, and 2) variable workload characteristics having applications entering and leaving the system with an unknown trend, and each one being characterized by a specific performance requirement. When considering these aspects, knob actuation for maximizing performance within minimal power consumption and accuracy loss becomes dramatically complex and challenging. Our goal is therefore to address this issue by enabling coordinated decision making on power knobs and approximation actuation.

3 RELATED WORK

Resource Management. We primarily focus on run-time resource management techniques proposed for heterogeneous multi-core systems, targeting big.LITTLE architecture. HMP is the last version of Linux scheduler for big.LITTLE architectures designed by Samsung [26]. It maps compute-intensive applications on the big cluster and I/O bounded ones on the LITTLE cluster. However, it neither considers performance requirements expressed by the user, nor acts on CPU quota and DVFS. *Race-to-idle* policy is not efficient in the case of asymmetric processors for neither power dissipation nor energy consumption, as shown in [11]. Therefore, advanced strategies are required especially when considering also performance requirements. Numerous works enhanced the HMP idea with advanced run-time resource management to meet performance requirements while concurrently optimizing energy and power consumption [3, 6, 10, 15]. The most representative one is [15] - using task mapping, DVFS, CPU quota and task migration, which eventually coordinate and converge towards meeting performance requirements within power limits.

Run-time Approximation. [24] has proposed a static scheduling algorithm considering various approximate versions of tasks in an application to maximize performance within a given error bound, while honoring power constraints. It is assumed that approximate versions of tasks (based on loop perforation) are provided before hand and the scheduler relies on an off-line heuristic to determine scheduling decisions of the approximate task. [13] proposes the use of approximation as a performance knob in covering up for the performance lost during power capping in many-core architectures. They assume two approximation levels to be provided for each task and switch the execution mode from accurate to approximate and vice-versa based on performance surges. No tight coordination between approximation and other power knobs is the limit of this work, as shown in Figure 1. Run-time approximation for thermal management of a single video application through content-aware error resilience characterization has been proposed in [16]. It defines four approximate levels for reducing power and temperature opportunistically over specific cases of video applications, however

Table 1: Qualitative comparison of proposed solution w.r.t. existing approaches.

Technique	Goal	Multi-Programmed	Unknown Workload	Knobs				APX Levels	Coord.
				DVFS	CPU quota	Task Mig.	APX		
HierCtrl [15]	Performance within power cap	✓	✓	✓	✓	✓	✘	✘	+/-
ApxSched [24]	Energy efficiency and QoS	✓	✘	✓	✓	✓	✓	5	✘
ApxTherm [16]	Thermal optimization	✘	✘	✓	✘	✘	✓	4	✘
ApxKnob [13]	Power capping	✓	✓	✓	✓	✘	✓	2	+/-
Our Appr.	**Performance within power cap**	✓	✓	✓	✓	✓	✓	**Fine-grained**	✓

it does not use approximation from a resource management perspective, nor it deals with multi-programmed workloads. Finally, [23] proposes a proactive control of knobs for approximating a given program, satisfying a set of system parameters for minimum loss in accuracy. This technique is based on an off-line learning, where the approximation knob takes into account, for a given set of inputs, the introduced error and the achieved performance improvement; moreover, no resource management is considered.

Table 1 reports and compares the most representative among the mentioned existing solutions, by listing their supported features. The last line in the table shows our proposed approach, which considers a more comprehensive scenario compared to the others, addressing the resource management and application approximation in a more fine-grained and tightly coordinated way.

4 SYSTEM ARCHITECTURE

Figure 2 presents our system overview comprising hardware, software and interface. The specific processing platform deploys the well-known ARM big.LITTLE asymmetric multi-core model, composed of a cluster of high-performance power-hungry *big* cores, and an alternative cluster of low-power *LITTLE* cores. The processor is constrained by a Thermal Design Power (TDP) [5] - the conservative maximum power specified for the chip's thermal safety, and is provided with per-cluster DVFS and power sensors to support power management. The loaded Operating System (OS) utilities provide control over resource allocations through: task migration between different clusters, task mapping on a specific core within a cluster, and CPU quota assignment to set the percentage of time/resources the core will devote per application. The overall system we propose is similar to the ones used in [15, 17, 24] - the key distinctive aspect of our approach is the APX knob set by the controller to determine the accuracy levels of the applications at run-time.

In this work, we target applications with a computationally intensive loop, where the main kernel is continuously repeated. Applications may enter and leave the system with an unknown trend, thus causing a highly variable workload. The applications are assumed to be enhanced with the HeartBeat API [8] for a throughput measure (in terms of loop iterations or *heartbeats*, performed per unit of time, hb/s, which is proportional to the amount of processed data per unit of time). This also allows expression of user-level performance requirements, in terms of the minimum throughput to be guaranteed. Finally, we assume the compute kernel to be possibly implemented with an approximation strategy, e.g., the loop perforation [22], that can be tuned at run-time.

The last component in Figure 2 is a run-time controller (similar to [15]), capable of accessing the described system-level sensors and actuation knobs through the OS interfaces, and the application-level ones through the HeartBeats API. The proposed run-time

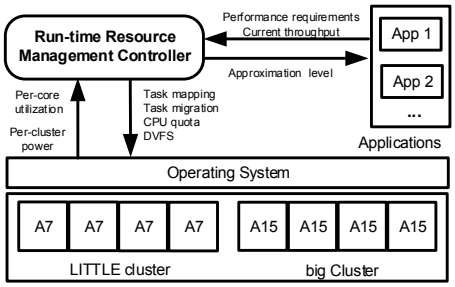

Figure 2: System Architecture.

resource management policy implemented as the controller module is presented in the following section.

5 THE PROPOSED POLICY

The run-time resource management policy functions in *idle*, *decide* and *refine* phases, implemented as a finite state machine shown in Figure 3, and described as follows.

5.1 Policy phases

Idle. This phase corresponds to the unloaded system or to a thoroughly balanced one, satisfying power and performance requirements, eventually. When in this phase, the controller monitors relevant events causing significant variations in system dynamics. Such an event typically occurs when a new application enters the system (requesting system resources), or a currently running application leaves it (freeing them). In such events resource allocation decisions are to made, by transitioning into the *decide* phase.

Decide. Resource allocation decisions on core selection, DVFS, CPU quota, task migration and approximation are made in this phase. When a new application enters the system, it is conservatively mapped on a LITTLE core. Each time an application enters/exits, the running workload on each cluster is re-mapped to make utilization uniform on the various cores. Actuation decisions are based on power and throughput measurements from the OS-level and application-level monitoring interfaces. Knob settings are determined to be proportional to applications' performance requirement and power budget available. Failing to meet an application's performance requirement or violating the upper bound on power budget are two instances when decision making becomes critical. Upon such events, the controller decides between combinations of power knobs alone (DVFS, CPU quota and/or task migration) and/or power knobs combined with approximation, subject to whichever yields better performance within acceptable power constraints and accuracy loss (the logical flow of this strategy is presented in a more detailed way in the next subsection). We use estimation models (detailed in Section 5.3) to predict near-optimal knob settings during the *decide* phase. They minimize the knob setting space to be pruned, allowing a faster arrival at a near-optimal

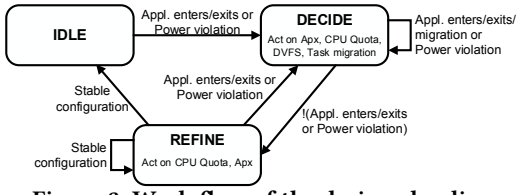

Figure 3: Work flow of the designed policy.

solution. Estimating performance loss/gain over every power actuation decision allows to establish coordination among conflicting power-approximation knobs settings. At the end, selected resource allocation decisions are enforced. If task migration has been performed, this significantly alters application's performance thus the *decide* phase is repeated; otherwise the subsequent *refine* phase will take place.

Refine. The *refine* phase monitors power and performance metrics and fine-tunes any coarse-gained knob settings enforced during the *decide* phase. This provides precise control over resource allocation, primarily using CPU quota and APX knobs. Further, any possible aberrations, or over/under compensation decisions made during the *decide* phase, will be converged towards required knob settings in the subsequent *refine* phase so that a stable resource allocation is eventually reached. The *refine* phase will last as long as needed to ensure the performance target, for all running applications, is met within the power budget.

5.2 The decision strategy

The algorithmic flow of the *decide* phase is shown in Algorithm 1. Actuation decisions are invoked in two cases - a power violation, i.e., measured power consumption (P) exceeding TDP (Lines 1–13) and power constraints honored (Lines 14–45), detailed next.

Power Violation. Recovery actions have to be taken by acting mainly on the big cluster since it is the responsible of relevant power consumption. Application(s) that have a minimal performance penalty (set to < 10%) upon migrating to the LITTLE cluster are chosen, if any (Line 3), and each selected application is migrated from the big cluster to the LITTLE one. Approximation is invoked simultaneously, by setting an accuracy level that is proportional to the estimated loss in performance with task migration (Lines 3–5). This ensures that a potential performance loss with migrating to the low-power LITTLE cluster is pro-actively addressed with reducing the workload through approximation. If no application with tolerable performance loss upon task migration is available, DVFS is used for reducing power consumption (Lines 7–13). Target voltage and frequency (VF) levels are determined by the ratio of current power consumption and budget P/TDP, using the estimation model in Equation 1 (presented next – Line 7). For each application, the possible performance loss with lowered VF levels is estimated with the model in Equation 4 (Line 10). Approximation is triggered over such applications that could suffer performance loss with the new VF levels. The level of approximation is set proportional to the estimated performance loss (Lines 12–13). In this scenario, DVFS and APX are triggered coordinately such that any possible performance loss incurred in power actuation is pro-actively and proportionally covered up through approximation, using the estimation models.

Power Budget Honored. We analyze each cluster separately (Line 15). We start on the LITTLE and we first identify possible applications that do not meet target performance at the highest resources. If any, such applications are migrated on the big cluster and the current decision phase ends and it will be restarted on the subsequent control period to have an updated measure of the throughput in the new configuration (Lines 18–22).

After that, on each cluster the application having the highest assigned CPU quota is chosen as the most demanding (Line 25). The VF level is then determined such that performance requirements of this selected application are met at maximum CPU quota; indeed as shown in [15, 17] selecting the minimum necessary VF level at maximum CPU quota minimizes power consumption among all configurations offering the same level of performance (Lines 26–27). Performance of other applications is estimated with the newly determined VF level (Line 29). The applications which have met the performance requirements and were approximated in the previous decision phases are recovered back to accuracy, subjective and proportional to the availability of surplus power budget and current performance levels (Lines 30–32). Among the other applications, CPU quota of applications that have already met the performance

Algorithm 1 Decide phase workflow.

Inputs: $Apps$: Applications currently running
Global Variables: U : Overall utilization of all the cores
V_b, F_b, V_l, F_l : VFs of big and LITTLE clusters
P: Measured power consumption
Global Constants: TDP: Power band limit
$migPenalty$: Performance penalty threshold upon task migration

Body:

1: **if** $P \geq TDP$ **then**
2: **for** every app_i in $Apps \in big$ **do**
3: **if** $\frac{app_i.maxPerfOnLITTLE}{app_i.targetPerf} \geq 1 - migPenalty$ **then**
4: $TaskMigration(app_i, LITTLE)$ //it forces decide phase re-execution
5: $APX(app_i, migPenalty)$
6: **else**
7: $V_r, F_r \leftarrow powerEstimate(V_l, F_l, U, P/TDP)$
8: $setDVFS(V_r, F_r)$
9: **for** every app_i in $Apps$ **do**
10: $app_i.perfEstimate \leftarrow perfEstimate(V_r, F_r, app_i.quota)$
11: **if** $app.perfEstimate < 1$ **then**
12: $loss \leftarrow 1 - \frac{app_i.perfEstimate}{app_i.perfTarget}$
13: $APX(app_i, loss)$
14: **else**
15: **for** each cluster $c \in \{big, LITTLE\}$ **do**
16: $Apps_c \leftarrow get\_appls\_on\_cluster(c)$
17: $redecide \leftarrow false$
18: **if** $c = LITTLE$ **then**
19: **for** every app_i in $Apps_c$ **do**
20: **if** $app_i.perf \leq app_i.targetPerf$ **then**
21: $TaskMigration(app_i, big)$
22: $redecide \leftarrow true$
23: **if** $redecide = true$ **then**
24: exit() //it forces decide phase re-execution
25: $appH \leftarrow get\_appl\_with\_max\_CPUquota(Apps_c)$
26: $V_r, F_r \leftarrow powerEstimate(V_c, F_c, U, appH.perf)$
27: $setDVFS(V_r, F_r)$
28: **for** every app_i in $Apps_c$ **do**
29: $app_i.perfEstimate \leftarrow perfEstimate(V_r, F_r, app_i.quota)$
30: **if** $app.perfEstimate > 1$ **then**
31: **if** $app_i.apx > 0$ **then**
32: $APX(app_i, app_i.perf)$
33: **else**
34: $q_{old} \leftarrow app_i.quota$
35: $q \leftarrow perfEstimate(V_r, F_r, app_i.quota)$
36: $setQuota(app_i, q)$
37: $U \leftarrow U - q_{old} + q$
38: **else**
39: $q_{old} \leftarrow app_i.quota$
40: $q \leftarrow perfEstimate(V_r, F_r, app_i.quota)$
41: $setQuota(app_i, q)$
42: $U \leftarrow U - q_{old} + q$
43: $app_i.perfEstimate \leftarrow perfEstimate(V_r, F_r, u)$
44: **if** $app_i.perfEstimate < 1$ **then**
45: $APX(app_i, app_i.perf)$

requirements (if any) are lowered proportionally within their performance requirements (Lines 33–37). For applications that do not meet the performance requirements with new VF settings, CPU quota is increased proportional to the target throughput if possible (Lines 39–42). In case of persistent performance violation, approximation is invoked, setting the level of approximation proportional to the loss in performance, as estimated (Lines 43–45).

5.3 Power and Performance Estimates

Within the discussed Algorithm 1, we used performance and power estimation models adapted from [17, 20, 24] and experimentally verified on the considered board. We represent power consumption P_j of a cluster j running a set of N applications App_i as a function of the overall cluster utilization U_j and the related V_j, F_j. As presented in [17, 20, 24], a fast/almost-accurate estimation of power for each V_j, F_j pair can be obtained as a function of U_j:

$$P_{V_j,F_j} = a_{V_j,F_j} \cdot U_j + b_{V_j,F_j} \tag{1}$$

where a_{V_j,F_j} and b_{V_j,F_j} are empirically derived at each V_j/F_j pair. Utilization U_j of a cluster j indicates the amount of time in a given control period for which the included cores are busy:

$$U_j = \sum_{i=0}^{N} U_i \tag{2}$$

where N is the number of cores in the cluster. Moreover, U_i measure is directly provided by the OS, whereas the assigned CPU quota is directly proportional to the utilization. Then, power consumption P_t of the system is the sum of values of the two clusters. As shown in previous works (e.g.[17, 24]), performance ($Perf_i$) of an application i has an almost linear relationship with the CPU quota Q, VF setting of the cluster it is mapped on, expressed as:

$$Perf(i) = \alpha Q(App_i) F_i \tag{3}$$

where α is a variable parameter that depends on application characteristics and cluster where the application is running. Thus, we estimate the performance penalty with actuation of power knobs viz., CPU quota assignment, VF level and task migration from a configuration *old* to the *new* one as:

$$Penalty(App_i) = \frac{Perf_{new}}{Perf_{old}} = \frac{\alpha_{new} Q_{new}(App_i) F_{new}}{\alpha_{old} Q_{old}(App_i) F_{old}} \tag{4}$$

6 EXPERIMENTAL RESULTS

Experimental Setup. For the experimental evaluation, we considered an Odroid XU3 board running Linux Ubuntu 15.04. It features Samsung Exynos 5422 with 4 ARM A7 (LITTLE) and 4 ARM A15 (big) cores, operating within $200-1400$MHz and $200-2000$MHz frequencies respectively. TDP was set to $5W$, considering a restriction on-chip temperature to 80°C. The controller has been implemented in C++ and runs as a user space process. It uses Linux drivers to access on-board sensors for power measures and to apply DVFS settings. Application's execution and allocation of CPU quota are enabled through CGroups Linux library. Communication between applications and the controller is established through Linux shared memory mechanism. An in-house implementation of HeartBeats API uses this shared memory space to periodically 1) write the application throughput information - to enable the controller's throughput monitoring and 2) read approximation levels determined by the controller - to dynamically switch the application accuracy level. Approximation level is passed as parameter to the kernel function, dynamically creating a range of fine-grained approximate versions without any compilation and memory overhead. For experimentation, we used four applications from machine learning domain, listed in Table 2 and the type of software approximations used. Larger number of iterations over input data yields results that converge towards optimal result due to algorithmic nature of these applications. We only temporarily trade the accuracy to meet power/performance requirements, while accurate execution is enabled when enough resources are available. The epoch for the controller invocation is parametrizable and can be tuned as per platform and applications' characteristics. In our evaluation, we set the epoch to $1s$ to collect stable throughput measurements upon knob actuation, given the streaming nature of the chosen applications.

Table 2: Simulated workload.

Application	Input	Approximation
LeastSq	1 million pairs	loop perforation
KNN	25k points and 25 test cases	loop perforation and task skipping
kMeans	50k points into 3 clusters	relaxed convergence
LinReg	1 million pairs	relaxed convergence

Evaluation. To test our proposed solution, we created a dynamic workload scenario ranging from 1 to 4 concurrently running applications, emulating unknown workloads. We compare it against similar state-of-the-art approaches, namely ApxKnob [13] and HierCtrl [15]. These approaches exploit the uncoordinated usage of the APX knob with other power knobs, and the coordinated usage of power knobs without any approximation, respectively. All systems are fed with the same dynamic workload scenarios. Figure 4 shows a comparison of power consumption, and Figure 5 reports each application's performance against the set requirements over the workloads (with a $-1/+1hb$ tolerance band). Table 3 lists the average power consumption and percentage of power violations for each approach. ApxKnob violates power budget the most, at 35% of the overall execution period, mainly with conflicting power-performance decisions resulting in constant oscillation between high-low power-performance states. HierCtrl has a much lower power violation at 3.5%, as different controllers converge eventually towards stable configurations. Our solution has a negligible 0.5% power violation with pro-active and coordinated knob actuation, benefiting from the estimation models. Table 3 also reports for every approach, the performance guarantees in terms of the percentage of execution time during which the requested performance is achieved for each application. The loss in accuracy with approximation knob is presented as the weighted sum of execution time and approximation level set, also shown in Table 3. ApxKnob meets the performance requirements mostly by leveraging approximation, and largely due to uninhibited power usage. Although the loss in accuracy for performance guarantees is acceptable, this approach is limited by frequent thermal violations. HierCtrl relies on degrading throughput requirements to address power emergencies, resulting in dramatic performance loss, particularly under heavy workloads. Although HierCtrl achieves the lowest average power consumption

Table 3: Power, performance and approximation behavior.

Technique	Avg. Power (W)	Power Violation (%)	Perf. Violation (%)				Approximation (%)			
			LR	LeSq	KNN	kM	LR	LeSq	KNN	kM
ApxKnob [13]	4.6	35.8	1.6	15.3	3.5	4.5	0	2	5.3	2.6
HPM [15]	2.82	3.5	28.6	89.7	44.2	42.6	0	0	0	0
Our Appr.	3.34	0.58	0.62	3.04	1.85	1.3	0.1	5	6.4	3.9

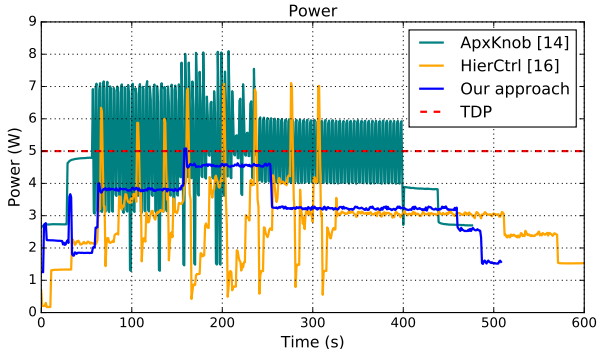

Figure 4: Power consumption measures.

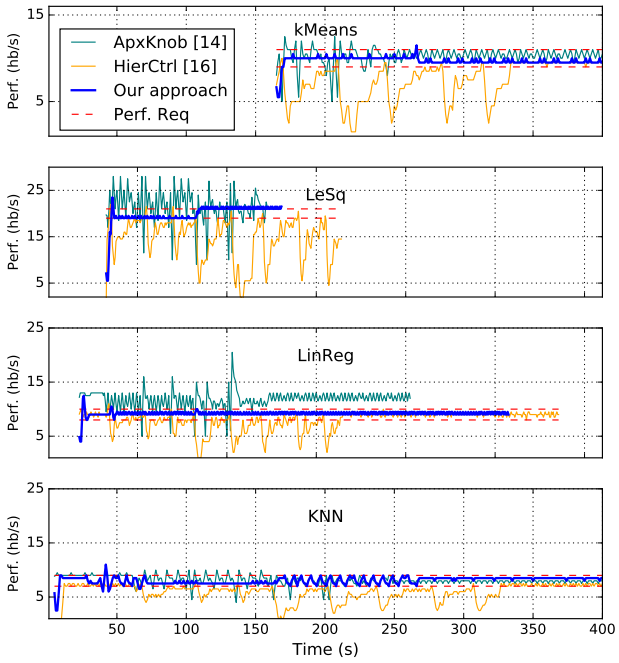

Figure 5: Performance measures for different applications.

and has a lower power violation rate, performance losses are significant. Our proposed approach, on the other hand, balances both performance requirements and power constraints, in particular, it is effective in delivering the requested performance, within minimal power consumption and an acceptable loss of accuracy. The effectiveness of approximation knob in power management although is trivial, requires disciplined tuning to achieve performance gains within power limits. Figure 5 shows that ApxKnob over compensates for performance (resulting in power budget violation), while HierCtrl violates performance requirements (with conservative power actuation). Our approach tunes approximation effectively to obtain the requested performance while minimizing the necessary power consumption. This can also be observed in Figure 5, where our approach consistently provides precisely the required performance, while ApxKnob and HierCtrl often over/under provision, resulting in their respective power and performance violations.

7 CONCLUSIONS

We presented a run-time resource management policy for asymmetric multi-cores that integrates functional approximation with other power knobs viz., DVFS, CPU quota assignment and task migration in a disciplined manner, to make performance-aware decisions on power management. The designed solution is able to minimize the penalty of power actuation decisions leading to performance loss with coordinated invocation of approximation together with other power knobs. Experimental results performed on an Odroid XU3 board demonstrates the effectiveness of the proposed solution w.r.t. the most representative existing approaches. Future work will focus on considering the management of multi-thread applications possibly targeting also the GPU resource.

ACKNOWLEDGMENT

We acknowledge financial support by the Marie Curie Actions of the European Union's H2020 Programme.

REFERENCES

[1] W. Baek et al. Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation. In *PLDI*, 2010.
[2] R. Cochran et al. Pack & cap: adaptive dvfs and thread packing under power caps. In *MICRO*, 2011.
[3] B. Donyanavard et al. SPARTA: Runtime Task Allocation for Energy Efficient Heterogeneous Many-cores. In *Proc. of CODES+ISSS*, pages 27:1–27:10, 2016.
[4] H. Esmaeilzadeh et al. Neural Acceleration for General-Purpose Approximate Programs. In *MICRO*, 2012.
[5] S. Pagani et al. TSP: Thermal Safe Power: Efficient Power Budgeting for many-core systems in dark silicon era. In *CODES+ISSS*, 2014.
[6] F. Gaspar et al. Performance-Aware Task Management and Frequency Scaling in Embedded Systems. In *In Proc. of SBAC-PAD*, pages 65–72, 2014.
[7] F Gaspar et al. A framework for application-guided task management on heterogeneous embedded systems. *ACM TACO*, 12(4):42, 2016.
[8] H. Hoffmann et al. Application heartbeats: a generic interface for specifying program performance and goals in autonomous computing environments. In *Proc. Int. Conf. on Autonomic computing*, pages 79–88. ACM, 2010.
[9] H. Hoffmann et al. Dynamic knobs for responsive power-aware computing. *ACM SIGPLAN Notices*, 2012.
[10] S. Holmback et al. Performance Monitor Based Power Management for big.LITTLE Platforms. In *Proc. HIPEAC Workshop on Energy Efficiency with Heterogeneous Computing*, pages 1–6, 2015.
[11] C. Imes and H. Hoffmann. Minimizing Energy Under Performance Constraints on Embedded Platforms: Resource Allocation Heuristics for Homogeneous and single-ISA Heterogeneous Multi-cores. *SIGBED Rev.*, 11(4):49–54, January 2015.
[12] Norman P Jouppi et al. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12, June 2017.
[13] A. Kanduri et al. Approximation knob: Power capping meets energy efficiency. In *In Proc. of ICCAD*, pages 1–8. IEEE, 2016.
[14] K. Ma and X. Wang. PGCapping: Exploiting power gating for power capping and core lifetime balancing in CMPs. *PACT*, 2012.
[15] T.S. Muthukaruppan et al. Hierarchical power management for asymmetric multi-core in dark silicon era. In *Proc. of DAC*, pages 1–9, 2013.
[16] D. Palomino et al. Thermal optimization using adaptive approximate computing for video coding. In *In Proc. of DATE*, pages 1207–1212, 2016.
[17] A. Pathania et al. Integrated CPU-GPU power management for 3D mobile games. In *Proc. of DAC*, pages 1–6, 2014.
[18] A. Rahmani et al. Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach. In *ISLPED*, 2015.
[19] A.M. Rahmani, P. Liljeberg, A. Hemani, A. Jantsch, and H. Tenhunen. *The Dark Side of Silicon.* Springer, 1st edition edition, 2016.
[20] H. Rexha et al. Core Level Utilization for Achieving Energy Efficiency in Heterogeneous Systems. In *Proc. of PDP*, pages 401–407, 2017.
[21] P. Schulz et al. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, 2017.
[22] S. Sidiroglou et al. Managing performance vs. accuracy trade-offs with loop perforation. In *FSE*, 2011.
[23] X. Sui et al. Proactive control of approximate programs. In *Proc. of ASPLOS*, pages 607–621. ACM, 2016.
[24] C. Tan et al. Approximation-aware scheduling on heterogeneous multi-core architectures. In *Proc. of ASP-DAC*, pages 618–623, 2015.
[25] A. Vega et al. Crank it up or dial it down: Coordinated multiprocessor frequency and folding control. In *MICRO*, pages 210–221, 2013.
[26] K. Yu et al. Power-aware task scheduling for big.LITTLE mobile processor. In *Proc. Int. SoC Design Conf.*, pages 208–212, 2013.