NPU-Accelerated Imitation Learning for Thermal- and QoS-Aware Optimization of Heterogeneous Multi-Cores

Martin Rapp, Nikita Krohmer, Heba Khdr, Jörg Henkel
Chair for Embedded Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
martin.rapp@kit.edu, nikita-krohmer@web.de, heba.khdr@kit.edu, henkel@kit.edu

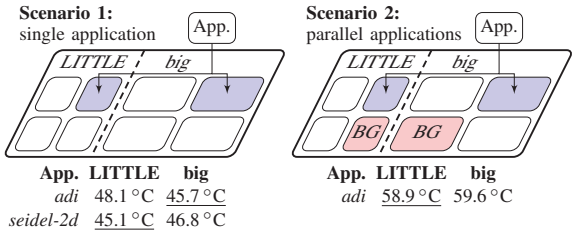
***Abstract*—Task migration and dynamic voltage and frequency scaling (DVFS) are indispensable means in thermal optimization of a heterogeneous clustered multi-core processor under user-defined quality of service (QoS) targets. However, selecting the core to execute each application and the voltage/frequency (V/f) levels of each cluster is a complex problem because 1) the diverse characteristics and QoS targets of applications require different optimizations, and 2) V/f levels are often shared between cores on a cluster, which requires a global optimization considering all running applications. State-of-the-art techniques for power or temperature minimization either rely on measurements that are often not available (such as power) or fail to consider all the dimensions of the problem (e.g., by using simplified analytical models). Imitation learning (IL) enables to use the optimality of an oracle policy, yet at low run-time overhead, by training a model from oracle demonstrations. We are the first to employ IL for temperature minimization under QoS targets. We tackle the complexity by using a neural network (NN) model and accelerate the NN inference using a neural processing unit (NPU). While such NN accelerators are becoming increasingly widespread on end devices, they are so far only used to accelerate user applications. In contrast, we use an accelerator on a real platform to accelerate NN-based resource management. Our evaluation on a *HiKey970* board with an *Arm big.LITTLE* CPU and an NPU shows significant temperature reductions at a negligible overhead while satisfying QoS targets.**

***Index Terms*—Imitation Learning, Neural Networks, Neural Processing Unit, Temperature Minimization, Quality of Service**

I. Introduction

Elevated on-chip temperature accelerates aging mechanisms in processors, and thereby degrades the system reliability [1]. Moreover, in mobile devices, it may adversely affect the user experience since it leads to increased skin temperature [2]. That makes temperature minimization of paramount importance. The two main knobs to reduce the temperature are task migration, to dynamically change the mapping, and DVFS. Using these knobs without considering the application characteristics misses significant optimization opportunities and may degrade the QoS of the applications, also degrading the user experience [3]. The reason is that the impact on performance and power when migrating an application between clusters differs from one application to another. Similarly, their performance and power sensitivities to DVFS also vary. Hence, the possibilities of QoS-aware thermal optimization vary from one application to another as demonstrated in the following motivational example.

This work was partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 146371743 – TRR 89 Invasive Computing. Due to the 4-page space constraint, not all of the originally submitted content is included in this IP paper.



App.	LITTLE	big
adi	48.1 °C	45.7 °C
seidel-2d	45.1 °C	46.8 °C

App.	LITTLE	big
adi	58.9 °C	59.6 °C

Fig. 1. The optimal mapping of applications with QoS targets varies between applications, and with other parallel applications (BG). The QoS target of *adi* / *seidel-2d* is 30 % of their respective IPS at max. V/f level on the big cluster.

A. Motivational Example

In Scenario 1 in Fig. 1, we execute one application, *adi* or *seidel-2d* from *Polybench* [4], on an Arm big.LITTLE CPU. The QoS target is selected as 30 % of the performance, measured in IPS, when executed on the big cluster at the highest V/f level. The clusters are operated at the lowest V/f level that satisfies the QoS target. Intuitively, executing the applications on the LITTLE cluster should minimize the temperature. However, this is not always the case. For *adi*, mapping it to the big cluster minimizes the temperature. The reason is that *adi* requires 1.8 GHz on the LITTLE cluster to achieve its QoS target, but only 0.7 GHz on the big cluster. In contrast, *seidel-2d* reaches its QoS target already at 1.2 GHz on the LITTLE cluster, and requires 1.0 GHz on the big cluster, which results in a similar temperature on both clusters, with a small advantage of the LITTLE cluster. The reason for the different V/f level requirements is that the applications benefit differently from the out-of-order execution on the big cluster. Consequently, such different application characteristics render different mappings optimal. *An optimal management technique must consider application characteristics and QoS targets.*

Scenario 2 studies *adi* with the same QoS target as in Scenario 1 but now, additional applications with high QoS targets run on both clusters. Intuitively, mapping *adi* to the big cluster as in Scenario 1 should still minimize the temperature. However, the additional applications require to operate both clusters at the highest V/f level. Since our platform has per-cluster DVFS, *adi* is also executed at the highest V/f level. In this case, mapping *adi* to the LITTLE or big cluster has almost the same temperature, unlike in Scenario 1. Per-cluster

DVFS changes the optimal mapping when several applications run in parallel. *An optimal management technique must perform global optimization considering the characteristics of all running applications.*

B. Challenges and Contributions

There are several challenges in temperature minimization on heterogeneous multi-cores under QoS targets. Firstly, there is high complexity in all involved aspects of the platform. For instance, the power and performance of applications depend on the instruction sequence, CPU microarchitecture, and V/f level, while temperature depends on the power density, floorplan, and cooling. Secondly, per-cluster DVFS runs all applications on the same cluster at the same V/f level, requiring global optimization. Thirdly, there is limited access to measurements. For instance, most platforms, such as the one studied in this work, have no power sensors and only few temperature sensors.

Many works perform optimization with models for individual aspects such as power, performance, or temperature. These models can be built analytically [5] or by machine learning (ML) [6], [7]. However, building the models requires fine-grained access to measurements like power, which may not be available. To solve this, end-to-end learning of management actions based on the available measurements can be employed, i.e., reinforcement learning (RL) or IL. In both cases, NN learning can be used to cope with the high complexity.

RL suffers from several problems. It requires to combine objective and constraints in a single scalar reward, which does not reflect their different properties and may lead to sub-optimal actions. Moreover, RL trains at run time, which is computationally expensive, preventing a low-overhead implementation, and may result in instability such as catastrophic forgetting. However, run-time thermal minimization while satisfying QoS targets requires a lightweight, yet near-optimal optimization to improve user experience, and a stable policy to avoid abrupt QoS violations. *IL is the only method that provides all of these capabilities.* In particular, it enables using the optimality of an oracle policy, which explicitly considers objectives and constraints, yet at low run-time overhead, by design-time training from oracle demonstrations. Design-time training until convergence also provides stability.

Motivated by these advantages, researchers have started to apply IL in resource management [8]–[10], but they all target power/energy. This significantly differs from temperature optimization due to spatial (heat transfer) and temporal (heat capacity) effects that do not exist in power/energy. We are the first to employ IL for temperature optimization.

To accelerate ML-based resource management, few works have proposed their own specific ML accelerators [11], [12]. However, they incur additional area overhead to the used platform. Recently, generic NN accelerators, e.g., NPUs or DSPs, became common in end devices such as smartphones [13]. These accelerators are intended to increase the performance and energy-efficiency of user applications that rely on NN inference. Despite their increasing spread and benefits, these existing accelerators have never been used to speed up NN-based resource management, and we are the first to do that.

We make the following novel contributions in this work:

- We employ NN-based IL for temperature optimization under QoS targets, as it enables near-optimal decisions at low run-time overhead. Our solution, *TOP-IL*, employs task migration and DVFS on heterogeneous multi-cores.
- We accelerate *TOP-IL* using an existing generic NN accelerator (NPU) on a real platform.

II. Related Work

The state-of-the-practice Android/Linux resource management [14] comprises scheduling and DVFS. Global Task Scheduling (GTS) aims at increasing the energy efficiency of heterogeneous processors by migrating mostly-idle applications to the LITTLE cluster. DVFS is performed by different governors, such as *powersave* for power minimization or *ondemand* for a power-performance trade-off. However, these techniques do not consider application characteristics nor their QoS targets, and only indirectly affect the temperature (via power/energy).

ML provides powerful algorithms to support system-level optimization [15]. Supervised learning can build models that predict system properties like performance or power. Rule-based power/thermal management can predict the impact of a decision with such models, and thereby achieve proactive management. However, training the models requires access to measurements like power, which are often not available [16].

Several works have employed RL for power/thermal optimization. The works in [12], [17], [18] use RL for power management via DVFS. For thermal optimization, an RL-based technique [19] is proposed to make decisions of both task migration and DVFS. However, this work does not cope with parallel applications. In contrast, the work in [20] considers parallel applications and employs RL for task migration with the goal of peak temperature minimization.

Several recent works have started to employ IL for system-level optimization. In [8], an IL technique is proposed for DVFS to minimize the energy under a performance constraint. The work in [9] uses IL to select the types, number, and V/f levels of the active cores, to optimize power. A hierarchical IL technique is proposed in [10] to select the number of active cores and the V/f level in each cluster to maximize the energy-efficiency of a heterogeneous multi-core processor under performance constraints. All of these IL-based techniques target power/energy minimization. Power/energy minimization is related to temperature minimization, but these solutions are not applicable to our problem and platform. Temperature is subject to both spatial (heat transfer) and temporal (heat capacity) effects that do not exist in power/energy. As a result, the block/frame/slice-based solutions do not apply. In addition, power sensors required for the oracle are often not available.

None of these works has employed IL for thermal optimization despite its unique capabilities to combine the optimality of an oracle policy with a low run-time overhead.

III. Problem Formulation and Technique Overview

We target a heterogeneous multi-core processor with per-cluster DVFS, where $\mathcal{F}_x$ is the list of frequencies of cluster x and f_x is its current frequency. There are two clusters in

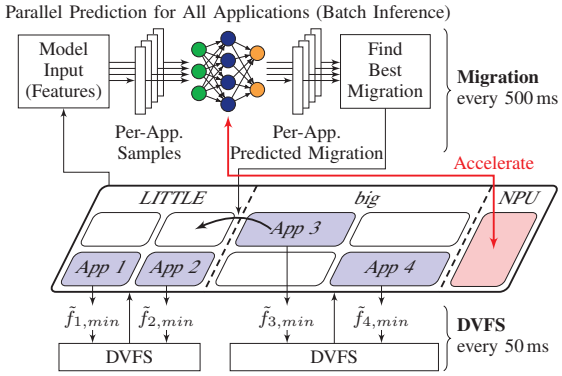


Fig. 2. Illustration of *TOP-IL* at run time. Task migration uses the NPU to predict the best migration per each application in parallel (batch inference).

our platform, LITTLE and big: $x \in \{l, b\}$, but our solution is compatible with any number of clusters. The processor executes parallel applications, each with its own QoS target Q_k and current QoS q_k, which are expressed in terms of the IPS.

Objective minimize the on-chip temperature
Constraint maintain QoS of applications (IPS)
Knob app.-to-core mapping (migration), per-cluster DVFS

We split the problem into two parts: 1) application-to-core mapping (via migration), and 2) per-cluster DVFS. Decisions on task migration are made with NN-based IL. We accelerate the run-time inference with an NPU.

IV. Run-Time Temperature and QoS Management with Imitation Learning

Employing IL for task migration requires to select features, create oracle demonstrations, and train the model that is used at run time. Creating the training data, i.e., oracle demonstrations, is simplified by restricting the management to migrate one application at a time. We train the NN to predict the best migration w.r.t. the temperature and QoS constraints for one *application of interest (AoI)*, while fixing all other applications to their respective cores. The details of training the model with IL have been omitted due to space constraints. At run-time, *TOP-IL* integrates the IL-based task migration with a control loop for per-cluster DVFS as shown in Fig. 2. While it would be intuitive to train a single NN for both migration and DVFS, performing only migration with the model reduces its complexity (create training data, topology, inference overhead).

A. Task Migration with NPU-Accelerated IL

If K applications run in parallel, each should be migrated to its optimal core w.r.t. temperature and QoS. However, migrating several applications at once results in a high number of potential combinations, i.e., large action space, and the impact of several migrations at once is difficult to predict. We solve this by migrating only one application at a time, but we find in each iteration the best migration among all applications. Our NN model has been trained for one AoI, which is migrated, and several other background applications, which are not migrated. We perform parallel inference, where each application is used as the AoI once. We select the overall best migration among all best migrations for all AoIs.

To reduce the overhead of the NN inference, we employ the already existing NPU of the *HiKey 970* board. The available parallelism in the NPU allows performing the parallel inference for all applications simultaneously in a single batch. The NPU is accessible via the *HiAI DDK*, which originally is designed to speed up user apps. We develop a C++ binary that runs in user space, employs the NPU for inference via the *HiAI DDK* (non-blocking call), and uses the Linux *affinity* feature for migration.

B. Control Loop for Per-Cluster DVFS

IL-based migration is integrated with a DVFS control loop to select the per-cluster V/f-levels. It first estimates the minimum V/f level for each running application k that is required to satisfy its QoS target Q_k using linear scaling from the current V/f level f_{x_k} of its cluster x_k:

$$\tilde{f}_{k,min} = \min\{f \in \mathcal{F}_{x_k} : q_k \cdot f / f_{x_k} \geq Q_k\} \quad (1)$$

It then determines per cluster x the minimum required V/f level to satisfy the QoS target of all applications running on it:

$$\tilde{f}_x = \max\{\tilde{f}_{k,min} : \text{application } k \text{ mapped to cluster } x\} \quad (2)$$

Since the estimates of $\tilde{f}_{k,min}$ are based on linear scaling, they are only accurate for small V/f level changes. Therefore, we adjust the current V/f level f_x by only one step towards f'_x and instead call this control loop more frequently than migration, i.e., every 50 ms. We skip iterations when the task migration is executed and directly after a migration to account for transient effects of cold caches that result in spurious QoS violations. Idle clusters are set at the lowest V/f level. We use the Linux *userspace* governor to set per-cluster V/f levels.

The combination of IL-based task migration and DVFS control loop enables us to achieve the goal of temperature optimization under QoS, as evaluated in the next section.

V. Experimental Evaluation

We perform experiments on a *HiKey970* [16] board with a HiSilicon Kirin 970 smartphone SoC. It implements the common Arm big.LITTLE architecture with four Arm Cortex-A53 and four Arm Cortex-A73 cores, operating with per-cluster DVFS. Furthermore, it comes with an NPU to accelerate the inference of NNs. We place the board in an A/C room to maintain a constant ambient temperature. The on-chip temperature is monitored with the on-board thermal sensor.

TOP-IL is compared with Linux GTS, paired with either *ondemand* or *powersave* governors. GTS assigns processes to a cluster depending on the computational requirements, i.e., mostly-idle and performance-hungry processes are migrated to the LITTLE and big cluster, respectively. *Ondemand* aims at providing a high performance but saving power when low performance is required. It achieves this by scaling the V/f-levels according to the CPU utilization. *Powersave* minimizes the power consumption by always operating at the lowest V/f levels, irrespective of the associated performance losses. All these policies are not aware of application characteristics or QoS targets. *GTS/ondemand* is the default configuration that is shipped with Android 8.0 on *HiKey970*.

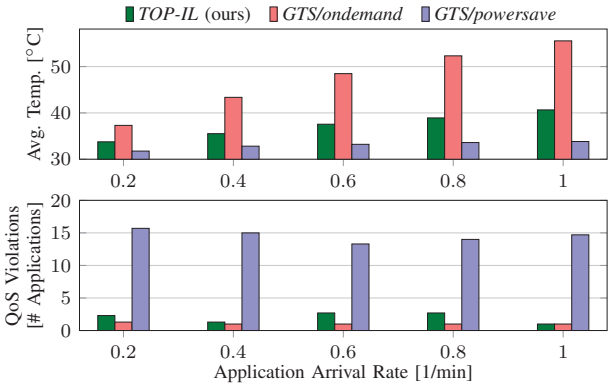


Fig. 3. *TOP-IL* significantly reduces the temperature, while achieving low QoS misses.

A. *Temperature Reduction with* TOP-IL

We first evaluate the capabilities of all techniques to reduce the temperature under QoS targets. We create a mixed workload of 20 randomly selected applications from *blackscholes*, *bodytrack*, *canneal*, *dedup*, *facesim*, *ferret*, *fluidanimate*, and *swaptions* from PARSEC [21], and *adi*, *fdtd-2d*, *floyd-warshall*, *gramschmidt*, *heat-3d*, *jacobi-2d*, *seidel-2d*, and *syr2k* from *Polybench* [4]. Only the *Polybench* applications (except *jacobi-2d*) are used for training *TOP-IL*. All other applications are unseen. We select a random QoS target for each application. The arrival times are distributed by a *Poisson* distribution with varying arrival rate to test different system load scenarios. With *TOP-IL*, the average / peak system utilizations vary from 13 % / 38 % to 37 % / 75 %. We let the board cool down for 10 min between experiments. This also reduces the impact of run-time variability due to the Android OS and workload fluctuations.

As shown in Fig. 3, *TOP-IL* reduces the average temperature by up to 15 °C compared to *GTS/ondemand* at only slightly more QoS violations. *GTS/powersave* achieves the lowest temperature but at the cost of the majority of applications missing their QoS target. In summary, *TOP-IL* is the only technique to achieve temperature minimization at low QoS violations.

B. *Run-Time Overhead*

The results in Fig. 3 already inherently contain the run-time overhead (additional CPU load, induced temperature) of *TOP-IL* as it is running in parallel to the workload. We report in this section the overhead with the maximum number of eight concurrent applications, i.e., the average value would be lower. IL-based task migration has an overhead of 3.4 ms. This includes the NN batch inference, which is only 580 µs due to employing the NPU. The majority of the time is required to prepare the input features. The relative overhead of task migration is 0.67 % of the epoch of 500 ms. The DVFS control loop runs for 540 µs on average. Its relative overhead is 0.87 % (8 invocations per 500 ms). The total (single-threaded) run-time overhead of *TOP-IL* is $\leq$1.54 %, and therefore negligible.

VI. Conclusion

Temperature minimization under QoS targets using task migration and DVFS requires jointly considering the diverse characteristics and QoS targets of all running applications, and, hence, is a complex problem. We tackle the complexity with NN-based IL, which enables us to combine the optimality of the oracle policy with a low run-time overhead. We employ the NPU of a smartphone SoC to accelerate the run-time inference. We accelerate the run-time inference with a smartphone NPU.

References

[1] H. Khdr, H. Amrouch, and J. Henkel, "Aging-Constrained Performance Optimization for Multi Cores," in *Design Automation Conference (DAC)*. IEEE, 2018.

[2] B. Egilmez, G. Memik, S. Ogrenci-Memik, and O. Ergin, "User-Specific Skin Temperature-Aware DVFS for Smartphones," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015.

[3] A. Pathania, H. Khdr, M. Shafique, T. Mitra, and J. Henkel, "QoS-aware Stochastic Power Management for Many-Cores," in *Design Automation Conference (DAC)*, 2018.

[4] T. Yuki and L.-N. Pouchet, "Polybench 4.0," 2015.

[5] G. Bhat, G. Singla, A. K. Unver, and U. Ogras, "Algorithmic Optimization of Thermal and Power Management for Heterogeneous Mobile Platforms," *IEEE Tran. Very Large Scale Integration (VLSI) Systems*, 2017.

[6] K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi, and G. V. Merrett, "AdaMD: Adaptive Mapping and DVFS for Energy-Efficient Heterogeneous Multicores," *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 39, no. 10, pp. 2206–2217, 2019.

[7] M. Rapp, M. B. Sikal, H. Khdr, and J. Henkel, "SmartBoost: Lightweight ML-Driven Boosting for Thermally-Constrained Many-Core Processors," in *Design Automation Conference (DAC)*, 2021.

[8] R. G. Kim, W. Choi, Z. Chen, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, "Imitation Learning for Dynamic VFI Control in Large-Scale Manycore Systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2458–2471, 2017.

[9] S. K. Mandal, G. Bhat, C. A. Patil, J. R. Doppa, P. P. Pande, and U. Y. Ogras, "Dynamic Resource Management of Heterogeneous Mobile Platforms via Imitation Learning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2842–2854, 2019.

[10] A. L. Sartor, A. Krishnakumar, S. E. Arda, U. Y. Ogras, and R. Marculescu, "Hilite: Hierarchical and Lightweight Imitation Learning for Power Management of Embedded SoCs," *IEEE Computer Architecture Letters (CAL)*, vol. 19, no. 1, pp. 63–67, 2020.

[11] Q. Fettes, M. Clark, R. Bunescu, A. Karanth, and A. Louri, "Dynamic Voltage and Frequency Scaling in NoCs with Supervised and Reinforcement Learning Techniques," *IEEE Tran. on Computers (TC)*, 2019.

[12] E. Kwon, S. Han, Y. Park, J. Yoon, and S. Kang, "Reinforcement Learning-Based Power Management Policy for Mobile Device Systems," *IEEE Tran. on Circuits and Systems I: Regular Papers (TCAS-I)*, 2021.

[13] A. Ignatov, R. Timofte, W. Chou, K. Wang, M. Wu, T. Hartley, and L. Van Gool, "AI Benchmark: Running Deep Neural Networks on Android Smartphones," in *Euro. Conf. on Computer Vision (ECCV)*, 2018.

[14] B. Jeff, "big.LITTLE Technology Moves Towards Fully Heterogeneous Global Task Scheduling," *ARM white paper*, 2013.

[15] M. Rapp, H. Amrouch, Y. Lin, B. Yu, D. Pan, M. Wolf, and J. Henkel, "MLCAD: A Survey of Research in Machine Learning for CAD," *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2021.

[16] Linaro 96Boards, "Hikey970," https://96boards.org/product/hikey970/.

[17] B. Donyanavard, A. Sadighi, F. Maurer, T. Mück, A. Rahmani, A. Herkersdorf, and N. Dutt, "SOSA: Self-optimizing Learning with Self-adaptive Control for Hierarchical SoC Management," *Int. Symp. on Microarchitecture (MICRO)*, 2019.

[18] S. M. P. Dinakarrao, A. Joseph, A. Haridass, M. Shafique, J. Henkel, and H. Homayoun, "Application and Thermal-Reliability-Aware Reinforcement Learning based Multi-Core Power Management," *ACM Jrnl. on Emerging Tech. in Computing Systems (JETC)*, vol. 15, no. 4, 2019.

[19] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, "Reinforcement Learning-based Inter-and Intra-Application Thermal Optimization for Lifetime Improvement of Multicore Systems," in *Design Automation Conference (DAC)*, 2014.

[20] S. Lu, R. Tessier, and W. Burleson, "Reinforcement Learning for Thermal-Aware Many-Core Task Allocation," in *Great Lakes Symp. on VLSI (GLSVLI)*, 2015, pp. 379–384.

[21] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," in *Int. Conf. on Parallel Architectures and Compilation Techniques (PACT)*. ACM, 2008.