HAT-DRL: Hotspot-Aware Task Mapping for Lifetime Improvement of Multicore System using Deep Reinforcement Learning*

Jinwei Zhang, Sheriff Sadiqbatcha, Yuanqi Gao, Michael O'Dea, Nanpeng Yu and Sheldon X.-D. Tan
Department of Electrical and Computer Engineering, University of California, Riverside, CA 92521 USA
{jzhan319,ssadi003,ygao024,mode001,nanpeng.yu,sheldon.tan}@ucr.edu

ABSTRACT

In this work, we propose a novel learning-based task to core mapping technique to improve lifetime and reliability based on advanced deep reinforcement learning. The new method, called *HAT-DRL*, is based on the observation that on-chip temperature sensors may not capture the true hotspots of the chip, which can lead to sub-optimal control decisions. In the new method, we first perform data-driven learning to model the hotspot activation indicator with respect to the resource utilization of different workloads. On top of this, we propose to employ deep reinforcement learning algorithm to improve the long-term reliability and minimize the performance degradation from NBTI/HCI effects. It penalizes continuously stressing the same hotspots and encourages even stressing of cores. The proposed algorithm is validated with an Intel i7-8650U four-core CPU platform executing CPU benchmarks for various hotspot activation profiles. Results show that HAT-DRL balances the stress between all cores and hotspots, and achieves 50% and 160% longer lifetime compared to non-hotspot-aware and Linux default scheduling respectively. The proposed method can also reduce the average temperature by exploiting the true-hotspot information.

CCS CONCEPTS

• **Hardware → Operations scheduling**; **Aging of circuits and systems**.

KEYWORDS

task mapping; multicore; lifetime; reinforcement learning

ACM Reference Format:
Jinwei Zhang, Sheriff Sadiqbatcha, Yuanqi Gao, Michael O'Dea, Nanpeng Yu and Sheldon X.-D. Tan. 2020. HAT-DRL: Hotspot-Aware Task Mapping for Lifetime Improvement of Multicore System using Deep Reinforcement Learning. In *2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20), November 16–20, 2020, Virtual Event, Iceland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3380446.3430623

*This work is supported in part by NSF grants under No. CCF-1816361, in part by NSF grant under No. CCF-2007135 and No. OISE-1854276.

MLCAD '20, November 16–20, 2020, Virtual Event, Iceland

ACM ISBN 978-1-4503-7519-1/20/11.
https://doi.org/10.1145/3380446.3430623

1 INTRODUCTION

Power density keeps increasing with technology scaling, causing severe thermal and reliability problems in high performance multicore systems [14]. Temperature has a profound impact on all the major long-term reliability effects such as electro-migration (EM) for interconnects, and negative bias-temperature-instability (BTI) and hot-carrier-injection (HCI) for CMOS devices [1]. In order to find efficient methods to solve the high temperature issue and improve both system performance and reliability, researchers have proposed many dynamic thermal/reliability management (DTM/DRM) methods, such as task mapping strategies [6, 7, 9].

Recently deep learning based approaches have been explored for DTM/DRM [9]. Das *et al* [2] proposed to take advantage of the thermal profile within (intra) and across (inter) applications based on Q-learning, which learns the relationship between the task allocation, voltage/frequency and device aging/mean-time-to-failure (MTTF). Lu *et al* [8] presents a task allocation method based on core and router temperatures and predicts near-future temperature that assists the DTM. Rathore *et al* [10] proposed a heuristic approach to manage the dimensions of state action space and a task mapping technique across heterogeneous cores through Q-learning. Recently, Kim *et al* [6] proposed a DRM technique by considering both long-term reliability (hard errors) and soft errors. All of those methods, however, have several drawbacks. First, they mainly depend on the temperature information from sensors and do not consider the true hotspots, which we will show in this paper, can be quite different than the sensor temperature. Recent study [11] shows that one can identify the true hotspots of multicore processors with advanced characterization. Second, existing approaches mainly use simple table-based Q-learning or its variations, which is not very robust and does not scale well for large control problems for many-core processors.

To capitalize on the recent advances in true hotspot identification and DRL, we propose a novel DRM method, called *HAT-DRL*, for performing more efficient task mapping to improve the lifetime and reliability based on an advanced deep reinforcement learning (DRL) technique. We first develop a data-driven approach based on deep neural network (DNN) to model the hotspot activation profile/indicator with respect to the resource utilization of workloads. Then, we propose to employ a recently proposed highly robust, sample-efficient DRL technique, called *soft-actor-critic* or *SAC* method. SAC is a model-free off-policy DRL algorithm that provides sample-efficient learning while retaining the benefits of stability. The algorithm has been successfully applied in many engineering domains such as the smart grid [15]. In this paper, it learns optimal policy to improve long-term reliability from NBTI/HCI

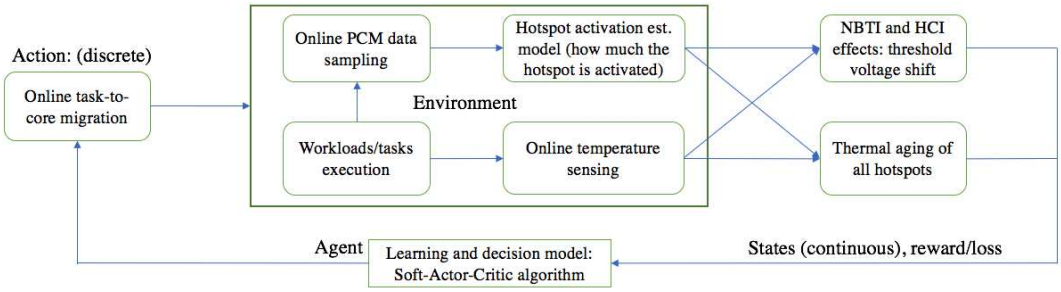

Figure 1: The overall HAT-DRL framework and algorithm workflow

effects and minimize performance degradation. The proposed approach is experimented with an Intel i7-8650U quad-core CPU platform executing CPU benchmarks for various hotspot activation profiles. Experimental results show that HAT-DRL balances the stress between all cores and hotspots, and achieves 50% and 160% longer lifetime than non-hotspot-aware and Linux default scheduling, respectively. The proposed method can also reduce the average temperature by exploiting the true-hotspot information.

2 THE OVERALL FRAMEWORK AND ALGORITHM FLOW

The whole algorithm flow, shown in Fig. 1, contains two stages. The first stage estimates how much percentage a hotspot area is activated (intensive power) or idle (low power). For the Intel i7 8650U quad-core processors, we find that each core has two primary hotspot areas. The second stage performs the task mapping based on an advanced DRL method. The new scheduler will take action of assigning or migrating tasks from current core(s) to desired core(s) every Δt interval.

The framework consists of three modules: the processor related dynamic environment, the state and reward functions based on NBTI/HCI thermal effects and aging, and the agent that interacts with as well as learns from the environment, as shown in Fig. 1.

In the environment, aging and thermal effect values of all variables of hotspots will form the states of the environment, hence the states are continuous. The actions are discrete, deciding which one of the cores to move to. Rewards are evaluated based on how well the aging and thermal effects are controlled, especially in terms of the worst hotspot. The policy is learned by using the recently proposed SAC algorithm [3] consisting of the state value network, state-action value network as critics and the actor network.

3 MOTIVATION EXAMPLE

3.1 Thermal sensors v.s. true hotspots

Direct sensor readings may miss the true hotspots of the cores, which can lead to sub-optimal task migration decisions. Fig. 2(a) illustrates an interesting case when temperature of all cores are the same at 90°C from a four-core system. However, the power distribution of the each core can be different due to different workloads, as shown in Fig. 2(b). As shown in the sequel, those deep knowledge of the true hotspots of a core can lead to different results for both temperature and reliability of the whole multicore processor.

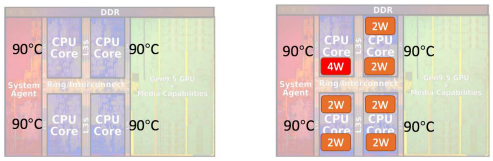

Figure 2: Thermal measurements without and with power distribution information on each core.

3.2 Motivation example for hotspot-aware v.s. non-hotspot-aware

In essence, the motivated idea is to avoid stressing the same hotspot continuously for long period of time, thus minimizing the weakest point of the entire processor. Basically, there are three pieces of required information. First, the number of primary hotspots in each core. Second, the hotspot stress conditions of all cores. Third, the hotspot activation percentage of the task(s). Before we dive into the DRL based control algorithm and the complete workflow, we first prove that the hotspot-aware idea indeed has positive difference over the non-hotspot-aware scheduling.

We deploy two tasks on just two cores of the processor where the two tasks have complementary hotspot activation areas, as a motivation experiment. We use two single-threaded benchmark tasks *hint* and *postmark* from the Phoronix Test Suite to do an experiment on an Intel i7-NUC mini PC (with i7-8650U quad-core CPU). The mini PC has four cores, from which we will use core-0 and core-3 to execute the two tasks, one on each core at the same time. We learned that each core contains two primary hotspots, marked as *A* and *B*. Besides that, we also learned that task *hint* (actually its subtask *DOUBLE*) almost always activates the hotspot *B* much greater than *A*, while *postmark* almost always activates *A* much greater than *B*, as shown in Fig. 5(b) and Fig. 5(c). We will elaborate on the method of how we learned the primary hotspots of the processor and the hotspot activation profile of the different tasks in Sec. 4.

There are two experiment settings, in the first setting, i.e. the non-hotspot-aware setting, each core will execute one task respectively and continuously for some time without task migration. In the second setting, i.e. the hotspot-aware setting, two cores will execute the two tasks interchangeably every 20 seconds (Fig. 3). Note that we run the second setting sufficiently long after the first setting run, waiting for the processor to cool down after the first run and making sure both runs have the same initial temperature.

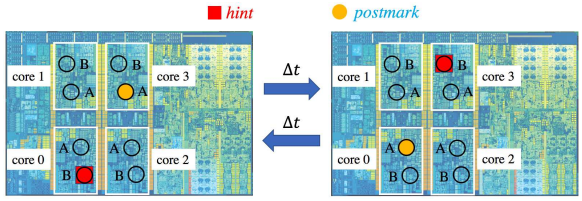

Figure 3: The hotspot-aware scheduling in the motivation experiment, where two tasks swap the cores every other interval.

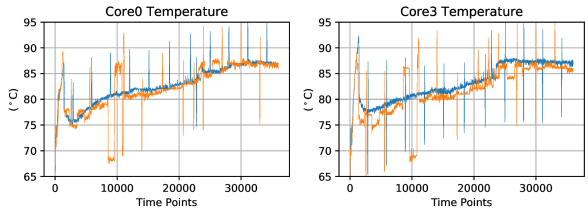

Figure 4: Temperature results of motivation experiment. Blue: non-hotspot-aware setting. Orange: hotspot-aware setting.

The background applications are minimized that their energy consumption can be ignored.

The temperature measured from sensors of core-0 and core-3 are recorded and plotted in Fig. 4. We conducted the same experiment repeatedly and found that the average temperature of the second setting always shows a 0.7 $\sim$ 1.1°C reduction. As for the execution time of the tasks, *postmark* are almost same on the two settings (average 52 seconds per execution). Actually *hint* shows a little bit faster execution on the second setting (average of 6 mins and 30 seconds per execution) over the first setting (average of 6 mins and 35 seconds per execution). Hence the performance of the system is not degraded at all.

We remark, in addition to the temperature reduction, more significant is to mitigate the long-term aging effect. As the migration of hot tasks (with hotspots) among different cores can lead to recovery effects of NBTI and HCI, which can lead to significant lifetime improvement as we show later. Hence, we have shown and proved that the hotspot-aware scheduling can indeed bring *brand new* opportunities to improve the aging and reliability of multicore systems.

4 DATA-DRIVEN MODEL FOR HOTSPOT ACTIVATION

One important aspect of the proposed method is to know which hotspots are active for a given workload. This can be achieved by using deep neural networks. As mentioned in the motivation example, for each core, we have two primary hotspot. In our work, we build a machine learning model to predict the hotspot activation indicator p based on the the on-chip workload utilization metrics. Here p is normalized to $[0, 1]$, where 1 means fully activated, and 0 not activated. We propose to build DNN based supervised learning method that takes use of the processor's IPCM [4] data to estimate the task's hotspot activation indicator. In our case, we select 24 most relevant PCM metrics as inputs for the DNN model, which is built by a multi-layer perception neural (MLP) network. Input vectors are obtained by the IPCM tool. IPCM tool is launched at the same time when the processor is under workload, data of the PCM vector is sampled at the frequency of 60 Hz. So the input matrix for the training data will have 24 columns where each column represents a time series of a PCM metric, and each row is one sample of the PCM vector.

As for the hotspots been activated, they are found indirectly through the use of thermal imaging system. We follow the similar hotspot characterization method in [11] in which hotspots are obtained by computing power maps from the measured thermal maps first. Then we take the power peaks that surpass a certain threshold as the activated hotspots. Note that thermal images must be synchronized with the PCM data sampling, so that each hotspot label will be connected to the correct PCM vector.

5 PROPOSED DRL-BASED APPROACH

In this section, we first formulate the task-to-core control problem as an MDP, then discuss the soft actor critic algorithm to solve the MDP problem.

5.1 Formulating the task-to-core control problem as an MDP

5.1.1 The basics of MDP. We will use the following notations for an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathrm{P}, r)$ [13]: $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively. $\mathrm{P}(s'|s,a)$ $\forall s', s \in \mathcal{S}, a \in \mathcal{A}$ is the environment state transition probability. $r(s,a)$ is the reward function. For each time step t, the RL agent takes an action a_t based on the environment's state s_t. Then the environment returns a reward $r_{t+1} = r(s_t, a_t)$ and moves to the next state s_{t+1} according to $\mathrm{P}(s_{t+1}|s_t, a_t)$. The goal of the RL agent is to learn a policy $\pi(a|s)$ that maximizes the expected long-term return starting from any state, which are captured by the action value function $Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{t=0} \gamma^t r_{t+1} | s_0 = s, a_0 = a]$ or the state value function $V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a)]$. $\gamma \in [0,1)$ is the discount factor. Next, we identify the state, action, and reward for the task mapping problem.

5.1.2 Formulation of states. As discussed in Section 3, an important factor that determines the lifetime and reliability of a core is their hotspots and related lifetimes.

The state is defined as $s_t = [dA_t, dV_t, P_t]$, where $dA_t = [p_t^1 dA_t^1, \ldots, p_t^N dA_t^N]$ denotes the increments of aging of the N hotspots at time t; similarly, $dV_t = [p_t^1 dV_t^1, \ldots, p_t^N dV_t^N]$ denotes the threshold voltage shift of the N hotspots. p_t^i denotes the power activation rate of the i-th hotspot. In this paper, the increment of aging and threshold voltage shift are assumed to be linear in the power activation. They are used to distinguish the aging and stress of different hotspots. Finally, $P_t \in \{P_t^1, \ldots, P_t^J\}$ where $P_t^j = [P_t^{j1}, \ldots, P_t^{jM}]$ stands for the power activation profile of the j workload thread, which contains the power profile of the task to be mapped or migrated. M is the number of primary hotspots per core.

5.1.3 Formulation of actions. The Action taken by the HAT-DRL agent is defined as the indexes of mapped cores corresponding to the tasks.

5.1.4 Formulation of rewards. The reward function reflects the aging and threshold voltage shift of the worst hotspot and is defined as follows

$$r(s_t, a_t) = c_1 \cdot \text{range}(A_t) + c_2 \cdot \text{range}(V_t) \tag{1}$$

where A_t and V_t are aging and voltage shift of all hotspots at time t; range(x) is the difference between the largest and the smallest elements of vector x; c_1 and c_2 are scaling constants. The detailed formulation of each of the terms in (1) is discussed below:

• **Thermal aging** A_t: The thermal aging related lifetime reliability of a hotspot is defined as $R(t) = e^{-A_t{}^{\beta} t}$ where A_t is the aging rate of the core per iteration of the application and is given by ([12])

$$A_t = \frac{1}{t_p} \sum_i \frac{\Delta t_i}{\alpha(T_i)} \tag{2}$$

where t_p is the period of the application graph and $\alpha(T_i)$ is the fault density (typically Weibull or Lognormal distribution). T_i is the average temperature in the time interval Δt_i. The MTTF of hotspot hs_j with reliability $R_j(t)$ is given by

$$MTTF_j = \int_0^{\infty} R_j(t)dt = \int_0^{\infty} e^{-t(A_j)^{\beta}} dt \tag{3}$$

We define the fault density as a Weibull distribution by

$$\alpha(T_i) = \frac{\beta}{\eta} \left(\frac{T_i}{\eta}\right)^{\beta-1} e^{-\left(\frac{T_i}{\eta}\right)^{\beta}} \tag{4}$$

where η is a positive constant and $\beta \in (0, 1]$, $\alpha(T_i)$ declines monotonically as the temperature T_i increases. Hence, to maximize the thermal aging related lifetime is equivalent to minimize the aging rate.

• **NBTI effect model**: NBTI effect is an increase in the absolute threshold voltage, a degradation of the mobility, drain current, and trans-conductance of p-channel MOSFETs. NBTI related threshold voltage shift of a PMOS in the stress phase is given by [5]

$$\begin{aligned} \Delta V_{th,st} &= \delta \cdot t^{0.25} \\ \delta &= B_{NBTI} \cdot t_{ox} \cdot \sqrt{C_{ox}(V_{dd} - V_{th})} \cdot e^{\left(\frac{V_{dd}-V_{th}}{t_{ox} \times E_0} - \frac{E_a}{k \times T}\right)} \end{aligned} \tag{5}$$

where t_{ox} is the oxide thickness, and C_{ox} the gate capacitance per unit area. The constants E_0 and E_a stand for device-dependent parameters, B_{NBTI} is a technology-dependent constant, and k the Boltzmann constant. T represents the temperature, and t the continuous stress time. As discussed, the threshold voltage shift of a PMOS transistor will be partially recovered if the transistor is placed in the recovery phase. Then the final undesired threshold voltage shift of PMOS transistors is expressed as following

$$\Delta V_{th,NBTI} = \Delta V_{th,st} \times \left(1 - \sqrt{\varepsilon \frac{t_{rec}}{t_{rec} + t_{st}}}\right) \tag{6}$$

where ε is equal to 0.35, and t_{st} and t_{rec} represent the stress and recovery time durations in a short term, respectively.

• **HCI effect model**: HCI is a phenomenon in solid-state electronic devices where electrons or holes have high kinetic energy to tunnel through the thin oxide gate to show up as gate current, or as substrate leakage current and eventually cause performance degradation. The equation given below evaluates the HCI-induced threshold voltage shift [5]

$$\begin{aligned} \Delta V_{th,HCI} &= \omega \cdot f \cdot t^{0.5} \\ \omega &= B_{HCI} \cdot \nu \cdot e^{\left(\frac{V_{dd}-V_{th}}{t_{ox} \times E_1}\right)} \end{aligned} \tag{7}$$

where t stands for time, ν for the activity factor and f for the core frequency, respectively. In addition, t_{ox} is the oxide thickness, and E_1 depends on the device specifications, temperature, and V_{dd}. Further, B_{HCI} is a technology-dependent constant. For the convenience of computation, we treat δ and ω as constants in this work.

5.2 Task-to-core control by SAC-based reinforcement learning

Traditional DRL methods suffer very high sample complexity and poor convergence due to high sensitivity of hyperparameters. Recently *SAC* method [3] was proposed to address the above mentioned challenges. It combines off-policy actor (policy function) and critic (value function) architecture with stochastic maximum entropy framework, which balances the exploitation and exploration nature in the DRL to achieve better performance and convergence.

5.2.1 Soft actor critic (SAC) algorithm in a nutshell. SAC is built on the maximum entropy RL framework, which regularizes the reward with the entropy of the policy. The value functions under entropy regularized reward are shown to satisfy the Bellman equations:

$$V_{\pi}^{\text{h}}(s) = \mathbb{E}_{a\sim\pi}\mathbb{E}_{s'\sim P}\left[r(s,a) + \gamma V_{\pi}^{\text{h}}(s')\right] + \alpha H(\pi(\cdot|s)) \tag{8}$$

$$Q_{\pi}^{\text{h}}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim P}\left[V_{\pi}^{\text{h}}(s')\right] \tag{9}$$

$$V_{\pi}^{\text{h}}(s) = \mathbb{E}_{a\sim\pi}[Q_{\pi}^{\text{h}}(s,a)] + \alpha H(\pi(\cdot|s)) \tag{10}$$

where H is the entropy function. α is a constant coefficient. Superscript h denotes the entropy regularization. (8)-(10) will be used for the training of the SAC agent.

To handle continuous state spaces, SAC uses function approximations for both the Q-value and the policy, and alternates between optimizing both networks with stochastic gradient descent using data from a memory buffer $D = \{(s_t, a_t, r_{t+1}, s_{t+1})\}$. Parameterized state value function $V_{\psi}(s_t)$, soft Q-value $Q_{\theta}(s_t, a_t)$, and a tractable policy $\pi_{\phi}(a_t|s_t)$ are considered, where the value networks are called critics and policy network is called the actor. The parameters of these neural networks are ψ, θ and ϕ.

The loss functions are derived on (8)-(10). ψ is trained to minimize:

$$J_V(\psi) = \mathbb{E}_{s_t\sim D}\left[\frac{1}{2}\left(V_{\psi}(s_t) - \mathbb{E}_{a_t\sim\pi_{\phi}}[Q_{\theta}(s_t,a_t) - \alpha \log \pi_{\phi}(a_t|s_t)]\right)^2\right] \tag{11}$$

which is a sample-approximated version of (10). Similarly, θ is trained to minimize the sample-approximated version of (9):

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_{\theta}(s_t,a_t) - \hat{Q}(s_t,a_t)\right)^2\right] \tag{12}$$

with $\hat{Q}(s_t, a_t) = r(s_t, a_t) + \gamma V_{\bar{\psi}}(s_{t+1})$, where $V_{\bar{\psi}}$ is the target value network and $\bar{\psi}$ is an exponentially moving average of the value

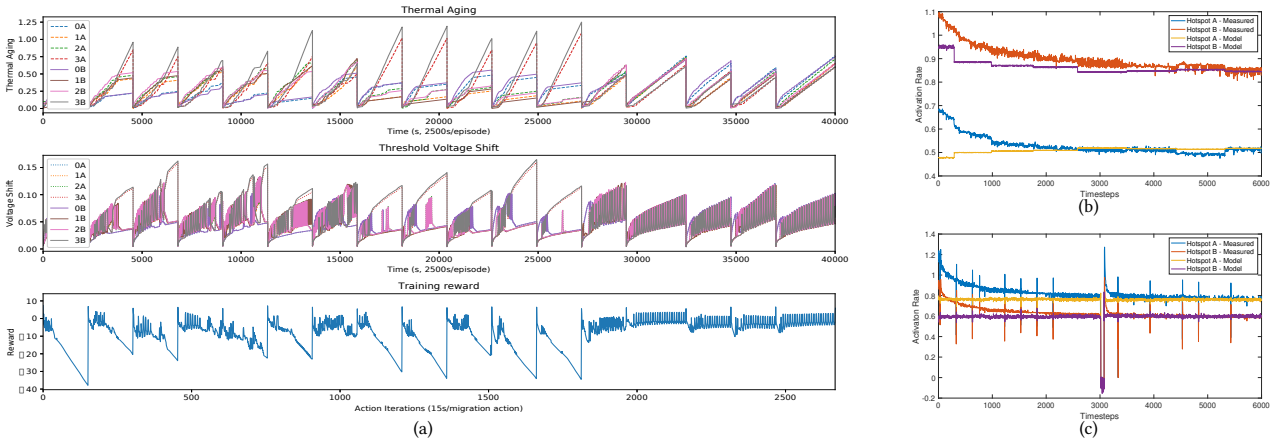

Figure 5: Model training visualization. (a) SAC-based model training, 2500 seconds per episode and 15 seconds per iteration interval. (b) Hotspot activation profile of *hint* and (c) *postmark* computed by the DNN model.

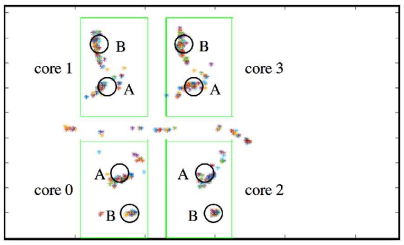

Figure 6: Scattered hotspots activated by various workloads.

network weights updated by $\bar{\psi} \leftarrow \rho\bar{\psi} + (1-\rho)\psi$. Finally, the policy parameter ϕ is learned by minimizing the objective function:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim D}\left[\mathbb{E}_{a_t \sim \pi_\phi(\cdot|s_t)}\left(\log \pi_\phi(a_t|s_t) - Q_\theta(s_t, a_t)\right)\right] \quad (13)$$

5.2.2 SAC-based task-to-core control. In our work, we implement the actor network, Q-critic and V-critic networks as three-layer fully connected DNNs, respectively. Once the action is determined by the agent, the scheduler of the operating system will be overridden by setting the CPU mask of the corresponding tasks dynamically in the run-time.

6 EXPERIMENT RESULTS AND DISCUSSIONS

6.1 Results for hotspot activation indicator modeling

We experimented with 9 benchmark tasks from Phoronix Test Suite and for each task we captured over 10 thousand IPCM data samples and corresponding thermal images. The power peaks are scattered in Fig. 6. Hotspots are mainly dropped into two clusters in each core. Hence there are two primary hotspots (A and B) can be activated. Hotspot activation profiles are recorded as the label values. For example, if a task activates hotspot A of core-0 at 50% and B of core-1 at 95% in a four-core processor, then the output label of the DNN is formed as [0.5, 0, 0, 0.95, 0, 0, 0, 0].

The results of hotspot activation indicator model are reasonably accurate. We used the profiling result of task *hint* as an example to demonstrate its logic and accuracy. As illustrated in Fig. 5(b) and Fig. 5(c), this model is able to correctly characterize the hotspot activation rates for workloads respectively when the processor is under workloads.

6.2 Experiment setup for DRL-based control model

The study of the proposed method is conducted on a target test device and a simulator. As aforementioned, the hotspot activation model is trained based on the thermal measurements of the test device (Intel i7-8650U). This model is trained offline efficiently.

The SAC-based DRL model is trained from scratch as we train the agent with fresh memory while it interacts with the environment. Training process on real multi-core processors can be very time consuming as we set 10-20 seconds for task migration action interval (Δt) for real processors. It may cause undesired migration overhead it the interval is too short.

To facilitate the model training process, we built a simulator of environment to accelerates the training with only sub-millisecond per action. The simulator essentially simulates two functionalities of the physical system. One of the functionalities is the pseudo IPCM process that simulates the IPCM based on the hotspot activation and thermal profile, and the current core mapping of the tasks running in the system. The second is the OS scheduler simulated by a pseudo scheduler which manages a task mapping table. This is how pseudo IPCM works, when a task is mapped to a core, the pseudo IPCM creates the sequences of frequency, hotspot activation rate, temperature, and time for that core being mapped according to the task's profiles that we studied. The pseudo IPCM generates such sequences for every core. We assume there will be no multiple tasks mapped onto the same core at the same time, and that the tasks in this work are all single-threaded. To include the random behavior of the tasks, we added ±5% uniform random noise to the mean of hotspot activation rates, ±5°C to the temperature and ±0.3GHz to the frequency.

6.3 Results and comparison for the SAC-based DRL model

The proposed DRL based method demonstrates robust behavior during the control for multiple task to core mapping.

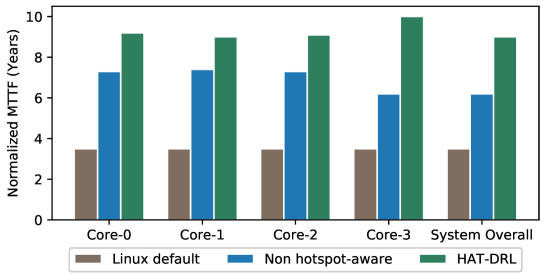

Figure 7: Normalized MTTF results.

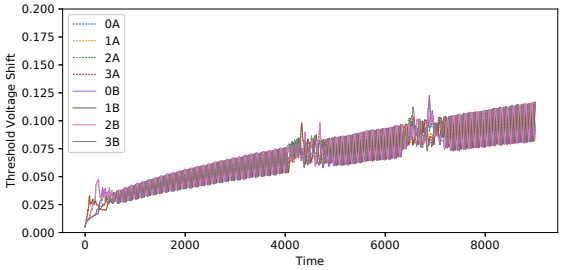

Figure 8: Threshold voltage shift traces of all 8 hotspots for the test device under HAT-DRL mapping.

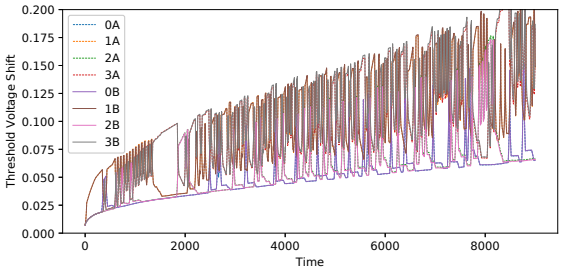

Figure 9: Threshold voltage shift traces of all 8 hotspots for the test device under non-hotspot-aware mapping.

First of all, Fig. 7 demonstrates the MTTF of the system under three different kinds of scheduling policies - Linux default scheduling, non-hotspot-aware and hotspot-aware (HAT-DRL). MTTF is normalized with the longest core to be 10 years. The MTTF of each core is presented by the worst hotspot on that core. And the system overall lifetime is limited by the shortest MTTF of all its cores. Non-hotspot-aware policy is a comparison with the hotspot-aware such that its input state vector does not distinguish the elements between the different types of hotspots. In our case, we replaced the elements of agings and voltage shifts of hotspot A and B with their mean value of each core in the state vector, hence the agent network cannot distinguish the hotspot stresses in the input state. It shows that HAT-DRL balances the stress between all cores and hotspots, which achieves about 50% and 160% longer lifetime than non-hotspot-aware and Linux default scheduling.

Training process and rewards are presented in Fig. 5(a). We implement episodes in the training which remarkably makes the training efficiently. Every 2500 seconds (15s between 2 iteration actions) the environment resets and the aging grows from start. As we see the aging and threshold voltage shift of eight hotspots grow very unevenly in the first few episodes, then they gradually grow evenly, meaning the worst stressed hotspot is dynamically mitigated.

Further, Fig. 8 shows the threshold voltage shift history of hots pots under *hint* and *postmark* controlled by HAT-DRL. The curves of hotspots under the same tasks with non-hotspot-aware scheduling is shown in Fig. 9. It is obvious that HAT-DRL minimizes the variation and maximum of threshold voltage shift quite well.

7 CONCLUSION

We propose a novel learning-based task to core mapping techniques to improve the lifetime and reliability based on advanced deep reinforcement learning technique. The new method, called *HAT-DRL*, is based on the observation that on-chip temperature sensors may not capture the true hotspots of the chip, which can lead to sub-optimal control decision. The new method is able to reduce the temperature by 1°C tested on Intel i7-8650U CPU. It also improves the lifetime by 50% over the limitation by temperature measurements and 160% over the Linux default scheduling. Furthermore, the threshold voltage shift is significantly mitigated for the multi-core system.

REFERENCES

[1] H. Amrouch, V. M. van Santen, T. Ebi, V. Wenzel, and J. Henkel. 2014. Towards interdependencies of aging mechanisms. In *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 478–485. https://doi.org/10.1109/ICCAD.2014.7001394

[2] Anup Das, Rishad A Shafik, Geoff V Merrett, Bashir M Al-Hashimi, Akash Kumar, and Bharadwaj Veeravalli. 2014. Reinforcement learning-based inter-and intra-application thermal optimization for lifetime improvement of multicore systems. In *Proceedings of the 51st Annual Design Automation Conference*. ACM, 1–6.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).

[4] Intel. [n.d.]. Intel Performance Counter Monitor (PCM). https://software.intel.com/en-us/articles/intel-performance-counter-monitor.

[5] Naghmeh Karimi, Thorben Moos, and Amir Moradi. 2019. Exploring the effect of device aging on static power analysis attacks. *UMBC Faculty Collection* (2019).

[6] Taeyoung Kim, Zeyu Sun, Hai-Bao Chen, Hai Wang, and Sheldon X.-D. Tan. 2017. Energy and Lifetime Optimizations for Dark Silicon Manycore Microprocessor Considering Both Hard and Soft Errors. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* 25, 9 (2017), 2561–2574.

[7] Z. Liu, S. X.-D. Tan, X. Huang, and H. Wang. 2015. Task migrations for distributed thermal management considering transient effects. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* 23, 2 (2015), 397–401.

[8] Shiting Lu, Russell Tessier, and Wayne Burleson. 2015. Reinforcement learning for thermal-aware many-core task allocation. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*. 379–384.

[9] S. Pagani, P. D. S. Manoj, A. Jantsch, and J. Henkel. 2020. Machine Learning for Power, Energy, and Thermal Management on Multicore Processors: A Survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 1 (2020), 101–116.

[10] Vijeta Rathore, Vivek Chaturvedi, Amit K Singh, Thambipillai Srikanthan, and Muhammad Shafique. 2019. LifeGuard: A Reinforcement Learning-Based Task Mapping Strategy for Performance-Centric Aging Management. In *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 179.

[11] Sheriff Sadiqbatcha, J. Zhang, H. Zhao, H. Amrouch, J. Hankel, and Sheldon X.-D. Tan. 2020. Post-silicon heat-Source identification and machine-learning-based thermal modeling using infrared thermal imaging. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* (2020). https://doi.org/10.1109/TCAD.2020.3007541

[12] Jayanth Srinivasan, Sarita V Adve, Pradip Bose, and Jude A Rivers. 2004. The case for lifetime reliability-aware microprocessors. *ACM SIGARCH Computer Architecture News* 32, 2 (2004), 276.

[13] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[14] Sheldon X.-D. Tan, Mehdi Tahoori, Taeyoung Kim, Shengcheng Wang, Zeyu Sun, and Saman Kiamehr. 2019. *VLSI Systems Long-Term Reliability – Modeling, Simulation and Optimization*. Springer Publishing.

[15] Wei Wang, Nanpeng Yu, Jie Shi, and Yuanqi Gao. 2019. Volt-VAR Control in Power Distribution Systems with Deep Reinforcement Learning. In *2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 1–7. https://doi.org/10.1109/SmartGridComm.2019.8909741