Dynamic Power Management in Large Manycore Systems: A Learning-to-Search Framework

GAURAV NARANG and ARYAN DESHWAL, Washington State University, Pullman, WA
RAID AYOUB and MICHAEL KISHINEVSKY, Intel Research Labs, Hillsboro, OR
JANARDHAN RAO DOPPA and PARTHA PRATIM PANDE, Washington State University, Pullman, WA

The complexity of manycore System-on-chips (SoCs) is growing faster than our ability to manage them to reduce the overall energy consumption. Further, as SoC design moves toward three-dimensional (3D) architectures, the core's power density increases leading to unacceptable high peak chip temperatures. In this article, we consider the optimization problem of dynamic power management (DPM) in manycore SoCs for an allowable performance penalty (say, 5%) and admissible peak chip temperature. We employ a machine learning– (ML) based DPM policy, which selects the voltage/frequency levels for different cluster of cores as a function of the application workload features such as core computation and inter-core traffic, and so on. We propose a novel learning-to-search (L2S) framework to automatically identify an optimized sequence of DPM decisions from a large combinatorial space for joint energy-thermal optimization for one or more given applications. The optimized DPM decisions are given to a supervised learning algorithm to train a DPM policy, which mimics the corresponding decision-making behavior. Our experiments on two different manycore architectures designed using wireless interconnect and monolithic 3D demonstrate that principles behind the L2S framework are applicable for more than one configuration. Moreover, L2S-based DPM policies achieve up to 30% energy-delay product savings and reduce the peak chip temperature by up to 17 °C compared to the state-of-the-art ML methods for an allowable performance overhead of only 5%.

CCS Concepts: • **Hardware → On-chip resource management** • **Computing methodologies → Machine learning algorithms**;

Additional Key Words and Phrases: Dynamic power management, large manycore systems, voltage frequency island, machine learning, thermal-aware

ACM Reference format:
Gaurav Narang, Aryan Deshwal, Raid Ayoub, Michael Kishinevsky, Janardhan Rao Doppa, and Partha Pratim Pande. 2023. Dynamic Power Management in Large Manycore Systems: A Learning-to-Search Framework. *ACM Trans. Des. Autom. Electron. Syst.* 28, 5, Article 84 (September 2023), 21 pages.
https://doi.org/10.1145/3603501

This work was supported in part by the National Science Foundation's grants CNS-1955353, OAC-1910213, and IIS-1845922 and in part by the Semiconductor Research Corporation's AI Hardware program task 3014.001.
Authors' addresses: G. Narang, A. Deshwal, J. Rao Doppa, and P. P. Pande, Washington State University, Pullman, WA; emails: gaurav.narang@wsu.edu, aryan.deshwal@wsu.edu, jana.doppa@wsu.edu, pande@wsu.edu; R. Ayoub and M. Kishinevsky, Intel Research Labs, Hillsboro, OR; email: raid.ayoub@intel.com, michael.kishinevsky@intel.com.

1084-4309/2023/09-ART84
https://doi.org/10.1145/3603501

1 INTRODUCTION

Large-scale manycore systems are essential for executing compute and data-intensive applications [1]. However, the design of high-performance manycore chips is dominated by power and thermal constraints. Higher on-chip temperature accelerates aging, thereby degrading the system reliability [2]. Hence, we need to establish suitable power-performance-thermal tradeoffs while designing a manycore system. In this regard, **Voltage-Frequency Island (VFI)** is an enabling methodology to create energy-efficient and thermally optimized manycore architectures [3]. VFI works on the premise that each core's computation and communication patterns vary during the execution of the application and similar cores and associated routers/links should be clustered together. The **voltage/frequency (V/F)** of each cluster can be regulated dynamically depending on the workload. VFI is a scalable **dynamic power management (DPM)** strategy for manycore chips. Developing a DPM strategy to control V/F knobs of VFI clusters in a manycore chip poses two key challenges. First, the search space of DPM decisions is exponential in the number of VFIs, V/F levels, and decision epochs (i.e., number of decision intervals while executing given application workloads). Second, optimal DPM decisions change depending on the desired tradeoff among target objectives (e.g., optimize power subject to $p\%$ performance penalty and the thermal budget). In addition to power-performance optimization, controlling temperature is important for three key reasons. First, thermal effect (unlike power/energy) is both spatial (heat transfer) and temporal (heat capacity) [4]. As a result, on-chip temperature can have non-trivial impact on lifetime and reliability of the chip, and higher on-chip temperatures can even lead to permanent chip failures [5]. Second, even low but persistent power consumption can lead to hotspots (temporal thermal effects). Third, power and performance depend on the instruction sequence, CPU microarchitecture, and V/F levels, whereas chip temperature depends on physical aspects of the chip as well such as power density, floorplan, and cooling [6]. This article considers the general problem of creating DPM policies to make optimal decisions for any pre-specified power-performance-thermal tradeoff.

Prior work has demonstrated the effectiveness of **machine learning (ML)** - based methods to implement VFI control policies. Both **reinforcement learning (RL)** and **imitation learning (IL)** have been extensively studied for VFI control and other DPM policies for manycore chips [7–9]. Moreover, IL has been shown to outperform RL for implementing VFI control in manycore systems [6, 10]. Unlike RL, which relies on exploratory learning guided by a hand-designed reward function, IL relies on supervised learning guided by an expert policy. However, the effectiveness of IL critically depends on the accuracy of the expert policy. Prior work on IL for DPM has used hand-designed expert policies, which are based on performing heuristic search in the combinatorial space of DPM decision sequence guided by power and performance models. These expert policies can be sub-optimal for different target design objectives and tradeoffs [11]. Moreover, configuring RL and IL appropriately is even more challenging when considering more than two objectives as we demonstrate by considering power-performance-thermal tradeoffs.

In this article, we propose a novel **learning-to-search (L2S)** framework to automatically construct high-quality expert DPM policies for any desired power-performance-thermal tradeoff. The key and significant advantage of L2S over prior ML methods including RL and IL is that it provides a design automation view for DPM: The designer specifies the available control knobs and the target tradeoffs for a set of design objectives (what part), and L2S automatically creates DPM policies to achieve the specified tradeoffs (how part). L2S employs parameterized DPM policies, which consider workload-aware features of the system state as input and obtain power management decisions in each epoch (DPM policy is executed at 1-ms interval). The key idea behind L2S is to formulate an explicit search in the continuous space of DPM policy parameters and solve this search problem using the principles of Bayesian optimization [12]. Specifically, the search is guided by learned statistical models from the training data in the form of parameters (input) and the

corresponding power, performance, and thermal evaluations (output). In each iteration, L2S selects one candidate policy parameter and evaluates the corresponding power, performance, and peak temperature by executing the application workload on the target manycore platform with the goal of quickly finding highly optimized DPM decisions. L2S employs an information-theoretic principle to select the parameters for DPM policy evaluation, one that maximizes information gain of the constrained optimal pareto front. The constraint, in this article, corresponds to p% performance penalty and a user-specified thermal budget. The pareto front refers to the feasible solution in the power, performance, and the peak chip temperature space.

The search space of DPM decisions is exponential in the number of VFIs, V/F levels, and decision epochs (e.g., $(8^4)^{1000}$ candidate policies for a four VFI system with eight V/F levels running an application with 1,000 epochs). Hence, the above search problem is extremely challenging, because there will be a handful of policies that satisfy the p% performance constraint for various workloads considered here. To overcome this challenge, we propose a refined policy evaluation approach, i.e., we prune such DPM decisions that do not satisfy the p% performance penalty constraint and select the highest scoring DPM decision from the promising ones at each decision epoch. This modified policy evaluation scheme allows L2S to uncover high-quality DPM decisions that optimize power and meets the joint performance-thermal constraints in a small number of iterations. To the best of our knowledge, L2S is the first ML framework that allows designers to automate the construction of ML-based joint performance-thermal constrained DPM policies without significant input from the system designer (RL requires good hand-engineered reward function, and IL requires hand-designed expert policy).

Contributions. The key contribution of this article is the development and evaluation of an L2S framework with refined policy evaluation to create VFI-based DPM policies in manycore systems to achieve target power-performance-thermal tradeoffs. Specific contributions include the following:

- We propose a scalable, automated L2S framework for constructing high-quality expert DPM policies as a search process in the continuous space of policy parameters. Subsequently, we iteratively improve the accuracy of this search process guided by the predictions and uncertainty of statistical models created from past policy evaluations.
- We demonstrate the effectiveness of L2S framework via evaluation on two manycore architectures designed using emerging technologies such as wireless interconnect and **monolithic 3D (M3D)**.
- Experimental results show that the DPM policies from the L2S framework reduce **Energy-Delay Product (EDP)** by up to 26% and 30% and reduce the peak temperature by 13 °C and 17 °C when compared to IL and RL, respectively.

2 RELATED WORK

VFI-based power management has become a mainstream solution to minimize the energy consumption of mobile and manycore systems [9, 13, 14]. Classical (i.e., non-ML) proactive [15–18] and reactive [19, 20] approaches have been proposed to either predict operating frequency such that temperature constraint is not violated in subsequent intervals (proactive) or throttle the cores if certain threshold temperature is reached (reactive) to manage the temperature and power consumption. Reactive methods are not efficient due to the delay between the action (V/F tuning) and response (temperature violation). Proactive methods such as in Reference [15] use core utilization and temperature to predict the operating frequency in the next decision epoch. However, simple average core utilization may not capture the information required to accommodate every core and router within a VFI with large intra-VFI workload variance. The **dynamic thermal and power**

management (DTPM) algorithm [16, 17] regulates temperature with minimal performance impacts using **gradient search algorithm (GSA)**. A power budget is computed using predicted temperature for a future decision epoch, and the number of active cores and their maximum frequencies are computed using GSA to avoid temperature violations. However, GSA can become intractable for large decision spaces such as in the case of manycore systems. Furthermore, these methods do not formulate DPM as a multi-objective optimization problem (i.e., optimize all three objectives collectively: temperature, energy, and performance).

ML methods such as RL and IL have successfully been deployed in various architectures starting from manycore systems to mobile platforms [7, 4, 13] and shown to be better than classical non-ML methods [9, 10]. Boosting metric is proposed in Reference [21], which is based on V/F sensitivities of performance, power, and temperature, and it maximizes performance under temperature constraint. Their method utilizes a **neural network (NN)** - based model to estimate the sensitivity of power and performance from applications' performance counters at runtime. Their principal focus was on minimizing the temperature violations while boosting the performance for mobile SoCs. Recent work used a Q-learning-based RL approach to solve the VFI control problem [9]. However, the hardware overhead to store the VFI control policies increases for large state spaces and can become very high without a function approximator. To address this challenge, a Q-function can be approximated as a linear combination of a series of radial basis functions [22] or a deep Q-learning method is used in which a NN acts as a function approximator [23, 24]. However, none of these RL-based methods address the challenge of designing a good reward function, which is critical for an effective RL-based DPM policy. Prior RL-based work used a single objective reward function comprisingonly thermal objective, and it was oblivious to energy minimization [22, 24]. Even the state model also comprisedonly the temperature values [22]. To better model the system variance, the combination of each core's frequency and utilization is used to model the environmental state but most RL-based investigations either optimize temperature or save energy [24]. However, a joint power-performance-thermal optimization in manycore system is necessary. Generally, such DPM policy involves establishing suitable tradeoffs between multiple objectives. A scalar parameter λ is associated with each objective in the reward function and tuning λ to achieve the desired tradeoff also makes RL a computationally expensive method. In a recent work [25], both energy and thermal savings are combined in a single reward function using a **proximal policy approximation (PPO)**-based RL technique, but the work is limited to objective of optimal task scheduling. Moreover, to avoid convergence and complexity arising from RL, feasible action space is limited to four V/F levels or less [5].

IL has addressed the challenges of RL and demonstrated its superiority over RL for VFI-based power management in large-scale manycore systems [9, 10, 26]. Besides the goal of power/energy minimization, temperature optimization with performance constraint has been studied for mobile platforms where IL has been shown to be a lightweight and optimal solution compared to RL [6]. IL requires the construction of a high-quality expert policy for providing supervised training. Prior work has employed hand-designed heuristic search procedures over a large combinatorial space of DPM decision sequences guided by power and performance models to construct the expert policies [4, 9]. Expert policy is constructed by dividing the application into phases, and the best configuration for energy minimization with p% performance penalty is searched for each phase. This applies well to the power and performance metric, since these models are phase independent and depend on V/F configuration. However, such application window splitting cannot be accurately applied to temperature models due to their spatial/temporal effects i.e., temperature in one phase depends on the temperature values in all previous phases. As a result, expert policy is constructed for the task migration objective while per-cluster V/F tuning is not done using IL (but using a simpler feedback control loop) [6]. Thus, creating expert policy for joint energy–performance–thermal

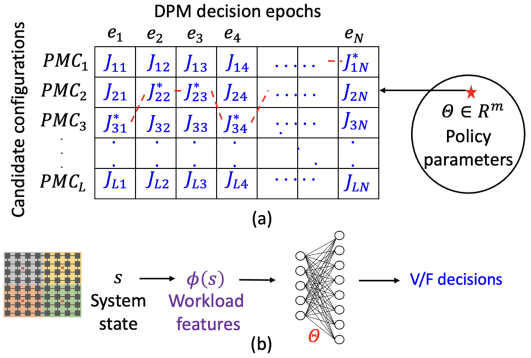

Fig. 1. (a) Mapping from continuous DPM policy parameter space to discrete DPM configuration space. (b) DPM policy maps the system state s to produce a power-management decision.

optimization is challenging. The generated expert policy can be sub-optimal for such complex target design objectives and tradeoffs, which means IL-based DPM policies can be sub-optimal.

In contrast, the proposed L2S framework is aimed at automating the construction of high-quality expert policies. L2S formulates a search process in a relatively small continuous space of policy parameters and employs ML to automatically improve the accuracy of this search process. Furthermore, L2S does not need to break the application into phases or windows, and it can very effectively consider temporal effects of temperature leading to more accurate search process. L2S learns statistical models from training data in the form of evaluation of policy parameters to make DPM decisions and employs an information-theoretic principle to select the sequence of candidate policy parameters for evaluation to quickly uncover high-quality expert DPM policies.

3 PROBLEM SETUP

We consider a manycore system with C cores (e.g., systems with 64 cores) divided into m VFIs. We are given a set of target application workloads, which will be executed on the manycore system. Our target is to create a runtime power management policy to optimize power consumption subject to an allowable performance penalty and admissible peak chip temperature. The DPM policy takes the current system state (e.g., key performance indicators, temperature information and workload features) and produces a decision vector $(d_1, d_2, \ldots, d_m)$, where each decision variable allocates the V/F for a single VFI. The system state is represented by the workload features such as each VFI's average and peak inter-VFI communication (or traffic), VFI's average and peak core computation (measured by instructions per cycle or IPC), and VFI's previous epoch V/F level. These features capture the average computation and communication patterns of the VFI and variance of the computation and communication patterns within the VFI and use the contextual knowledge of the previous prediction.

In this work, we consider DPM policies represented as functions of the system state with continuous parameters $\Theta \in R^m$, where Θ represents the weights of a **multi-layer perceptron (MLP)**. Prior IL work [9] demonstrated that simple linear functions are very effective and regression tree based non-linear models provide small benefits. Therefore, we consider a simple MLP as a non-linear function without adding too many weight parameters beyond the linear model. For example, the system state s is represented as input features $\Phi(s)$, and the DPM policy $\pi(\Phi(s), \Theta)$ maps the system state s to produce a power-management decision vector as shown in Figure 1(b). Suppose $E(\Theta)$, $T(\Theta)$, $Q(\Theta)$ denote the energy consumption, execution time, and peak temperature using the policy π with parameters Θ over N decision epochs respectively, i.e., the cumulative

sum of energy and execution time in each decision epoch with respect to the corresponding V/F allocation decisions and the peak chip temperature when running the target application workload. In other words, every candidate Θ corresponds to one candidate DPM sequence (trajectory) J_{ij} (red-colored path) as shown in Figure 1(a), where the **power management configuration (PMC)** sequence space is a discrete space of size L^N (PMC_1, $PMC_2, \ldots, PMC_L$ are the L candidate configurations). Total possible PMC in each epoch, i.e., L, is equal to $(number\ of\ V/F\ levels)^m$, where m is the number of VFIs. The effectiveness of the DPM policy π critically depends on the parameters Θ. Given a manycore architecture, application workload APP, and a maximum allowed performance penalty (p%), our goal is to find the optimal parameters Θ^* such that $E(\Theta)$ is minimized with respect to the p% performance penalty constraint and peak temperature ($Q_{\max}$) for a given distribution of initial states,

$$\begin{aligned} &\Theta^* = argmin_{\Theta} E(\Theta) \\ &subject\ to\ :\ T(\Theta)/T\left(\pi_{nominal}\right) \leq p \\ &and\ Q(\Theta) \leq\ Q_{\max}, \end{aligned} \tag{1}$$

where $\pi_{nominal}$ is the policy that selects the highest V/F (i.e., nominal V/F) for all decision variables. To aid in this process, we assume the availability of power and performance models that can be used to estimate the energy and execution time for any given sequence of power management decision vectors. L2S performs search in the continuous space of parameters $\Theta \in R^m$ to iteratively improve the quality of power management configuration sequence. Finally, we perform supervised learning to identify the parameters $\hat{\Theta}$ that mimic the behavior of the best uncovered power management configuration sequence. We demonstrate that the principles behind the L2S framework are applicable for more than one configuration by experimenting with manycore platforms integrated via two different emerging **network-on-chip (NoC)** architectures, viz., wireless NoC and M3D NoC.

Wireless NoC: The achievable performance of VFI-based manycore platforms depends on the overall communication backbone, which relies predominantly on NoCs. Traditionally mesh-based NoCs have been used in VFI-based systems. However, mesh-based NoCs have large latency and energy overheads due to their inherently long multihop paths. In a wireless NoC, where the long-range shortcuts are implemented through mm-wave wireless links operating in the 10- to 100-GHz range, is shown to improve the energy dissipation profile and latency characteristics of manycore chips [27]. In a VFI-based system the wireless links are mainly used for inter-VFI data exchange [28]. It has been shown to improve the energy dissipation profile and latency characteristics compared to mesh NoC-enabled VFI systems [29].

Monolithic 3D NoC: Emergence of M3D integration has opened the possibility of designing the ultra-low-power and high-performance circuits and systems. The smaller dimensions of **monolithic inter-tier vias (MIV)** offer high-density integration, the flexibility of partitioning logic blocks across multiple tiers, and significantly reduced total wire-length [30]. However, NoC is an enabling solution for integrating large numbers of embedded cores in a single die. M3D NoC architectures combine the benefits of these two paradigms (M3D IC and NoC) to offer an unprecedented performance gain even beyond the Moore's law regime. By exploiting the MIV-based vertical connections in M3D, the multi-hop long-range planar links can be placed along the shorter Z dimension, and hence, overall system performance is improved significantly [31, 32].

4 LEARNING TO SEARCH FRAMEWORK

In this section, we first provide a high-level overview of the proposed L2S framework to create optimized dynamic power management policies to achieve a target power–performance–thermal tradeoff. Subsequently, we describe the details of the key elements of the L2S framework.

ALGORITHM 1: L2S for power management in Manycore systems

Input: $ARCH$ = target manycore system with C cores,
APP = target applications,
$\Phi(s)$= features of the system state s,
$\pi(\Phi(s), \Theta)$= neural network (NN) based DPM policy with parameters Θ,
p = maximum allowable performance penalty
Q_{max} = Peak temperature constraint
Output: $\hat{\Theta}$, parameters of the optimized DPM policy

1: **Initialize**: D_0 = small number of training examples in the form of Θ and corresponding $E(\Theta), T(\Theta), Q(\Theta)$; and $t = 0$
2: Expert policy = sequence of nominal V/F pairs (empty if none identified to meet the p% performance penalty)
3: **Repeat:**
4: Learn statistical models $E(\Theta), T(\Theta), Q(\Theta)$ from training data D_t (Section 4.1)
5: Select Θ_{t+1} to maximize the information gain about optimal energy with performance constraint (see Equation 3)
6: Evaluate policy $\pi(\Phi(s), \Theta_{t+1})$ over samples of initial states:
$$E(\Theta_{t+1}), T(\Theta_{t+1}), Q(\Theta_{t+1}) = Evaluate(ARCH, APP, \pi(\Phi(s), \Theta_{t+1}))$$
7: Update training data:
$$D_{t+1} = D_t \; U \; \{\Theta_{t+1}, E(\Theta_{t+1}), T(\Theta_{t+1}), Q(\Theta_{t+1})\}$$
8: **If** $T(\Theta_{t+1})$ meets the p% performance constraint and $Q(\Theta_{t+1})$ meets Q_{max} peak temperature constraint, update the Expert policy (lowest energy DPM configuration sequence) depending on the value of $E(\Theta_{t+1})$ (Section 4.2)
9: $t = t + 1$
10: **Until convergence or maximum iterations**
11: Perform supervised learning using Expert policy to estimate $\hat{\theta}$ (Section 4.4)
12: **return** $\hat{\theta}$, the parameters of the learned DPM policy

Overview of L2S framework. The L2S approach has two key steps. First, L2S conducts search in the continuous space of policy parameters Θ to identify the optimized sequence of power management decision vectors using the principles of Bayesian optimization [12]. Bayesian optimization algorithms intelligently search the given input space to find optimized solutions using a small number of iterations or expensive-to-evaluate objective function calls (e.g., energy and execution time by running the target applications on the given manycore platform). These methods employ a statistical model to guide the search process. The main advantage of this model-guided search is that it helps in reducing the number of expensive objective function evaluations to solve optimization problems over large search spaces. A key distinguishing feature of this step compared to IL is that L2S uses an information-theoretic reasoning principle to automatically guide the search toward high-quality power management decisions. This contrasts with the existing sub-optimal approach of executing a hand-designed heuristic search procedure directly over the combinatorial space of power management decision sequences [9, 10]. Second, L2S performs supervised learning to estimate $\hat{\Theta}$ to mimic the best power management decision-making behavior uncovered by the search process in the first step.

To identify the best sequence of power management decisions, we learn statistical models for energy, execution time, and temperature over the parameter space Θ using the training data in the form of policy evaluations $E(\Theta)$, $T(\Theta)$, and $Q(\Theta)$ and use them to guide our search. These surrogate statistical models allow L2S to make predictions with quantified uncertainty about energy, execution time, and peak temperature for policy parameters that are not yet evaluated. We perform the following steps in each iteration: (1) We reason using the current statistical models to select the next candidate policy parameters Θ that maximizes the information gain about optimal energy with the performance constraint and peak temperature constraint. (2) We evaluate the power management policy $\pi(\Phi(s), \Theta)$ by executing it on the manycore system running the target applications APP to measure energy $E(\Theta)$, execution-time $T(\Theta)$, and on-chip peak temperature $Q(\Theta)$. The next step is to use power/performance models to prune "bad" power management decisions before selecting the highest-scoring power management decisions by the DPM policy $\pi(\Phi(s), \Theta)$. This pruning step allows us to identify power management decision sequences, which satisfy the

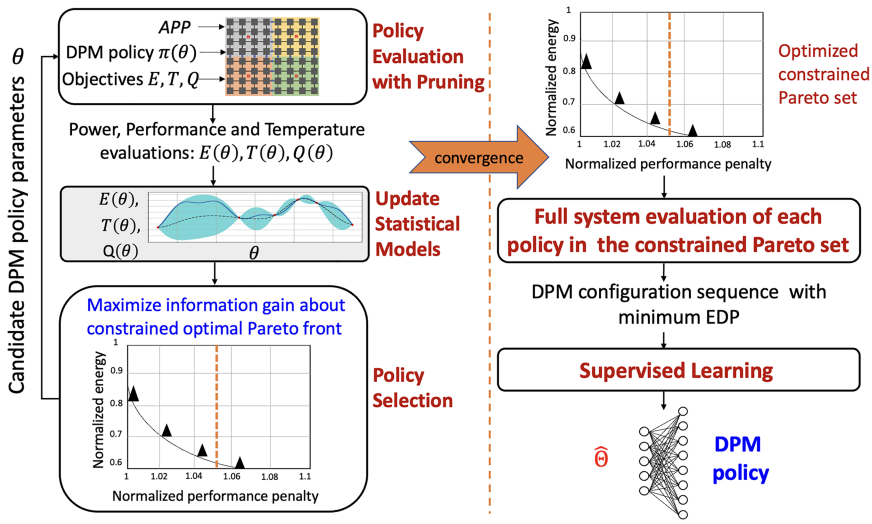

Fig. 2. High-level overview of the L2S framework, which is executed *offline* and the trained DPM policy is deployed for execution.

performance penalty constraint. The fraction of feasible DPM decision sequences is significantly smaller than all possible decision sequences, which makes it a particularly challenging problem. This step provides critical training data for the statistical models. We determine the peak temperature by executing the application workload and prune such policies that do not satisfy the peak temperature constraint. (3) We use the training data in the form of (input) policy parameters Θ and (output) policy evaluations $E(\Theta)$, $T(\Theta)$, and $Q(\Theta)$ to update the statistical models. After maximum iterations or convergence, we use the best uncovered sequence of power management decisions (minimum energy and meets the performance/peak temperature constraint) to perform supervised learning to estimate the corresponding policy parameters $\hat{\Theta}$. Algorithm 1 provides the pseudo-code, and Figure 2 shows an overview of the L2S framework for power–performance–thermal design objectives. Please note that the L2S framework is general for any user-defined design objectives.

4.1 Training Data and Learning Statistical Models

In each iteration of the L2S framework, we collect one training example by evaluating a candidate policy parameter Θ (input variables) to get the corresponding energy $E(\Theta)$, execution time $T(\Theta)$, and temperature $Q(\Theta)$ (output variables) when running the given application workload APP on the target manycore platform. At the end of t iterations, the aggregate training dataset consists of t training examples of input–output pairs.

We want to learn statistical models from the aggregate training dataset after each iteration. Figure 3 depicts the evolution of the statistical model for an objective over t iterations. These statistical models are used for two purposes. First, to make fast predictions about the energy and execution time of (unknown) policy parameters Θ that are not evaluated yet, i.e., outside the training data. Second, to quantify uncertainty of predictions, which is a critical component that allows us to reason about which candidate policy parameters to evaluate next to quickly uncover the optimal constrained pareto front. Note that power and performance models estimate the power and performance in each decision epoch, whereas statistical models map the policy parameters to cumulative power, performance, and peak temperature over N decision epochs that is critical

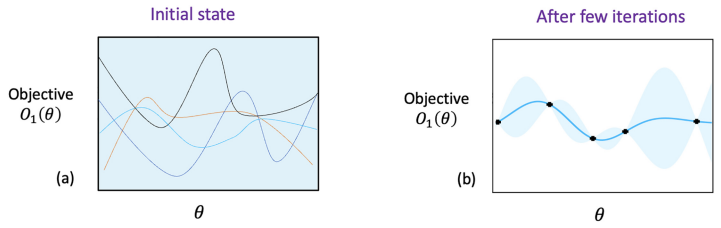

Fig. 3. Statistical model of the objective function $\boldsymbol{O}_1(\boldsymbol{\Theta})$. For example, $\boldsymbol{E}(\Theta)$ is a random GP model and looks like (a) prior to any optimization iteration (at $\boldsymbol{t} = 0$). Over the iterations, training examples (shown as datapoints in (b)) give us more information about the objective function $\boldsymbol{O}_1(\boldsymbol{\Theta})$ and guide the search toward promising candidate parameters for solving the optimization problem. Uncertainty is low for policy parameters $\boldsymbol{\Theta}$ close to those in the training data and vice-versa as shown by the shaded region in (b). The thick line corresponds to the mean of the GP model and the shaded region corresponds to the variance in the prediction.

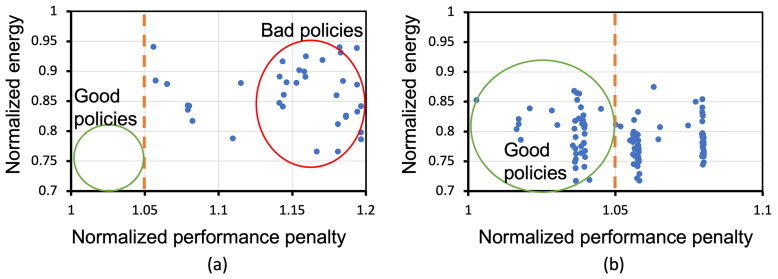

Fig. 4. Effectiveness of policy-refinement algorithm (a) no refinement (b) refined policy evaluation.

to solve Equation (1). We employ **Gaussian processes (GPs)** [33] as our statistical models due to their ability to approximate arbitrarily complex functions and principled uncertainty quantification due to Bayesian interpretation. Intuitively, uncertainty will be low for policy parameters Θ close to those in the training data and vice versa. We learn three GP models M_1, M_2, and M_3 for execution time $T(\Theta)$, energy $E(\Theta)$, and peak chip temperature $Q(\Theta)$ objectives, respectively. Note that GPs are typically used in the small training data settings, and since we are performing active learning to automatically select new training examples, our approach is naturally designed to be efficient in terms of the number of training examples required to solve the given optimization problem. Importantly, the role of statistical models is not to mimic the true energy, execution time, and peak temperature functions uniformly over the entire policy parameter space but to only guide the search toward efficiently solving the optimization problem at hand.

4.2 Policy Evaluation via Power and Performance Models

The straightforward approach to perform policy evaluation is to select the highest scoring DPM configuration from all the candidates using the policy parameters Θ at each decision epoch. However, the space of candidate policy parameters is very large and only a tiny fraction of the policy parameters will meet the $p\%$ performance penalty constraint. Hence, the goal of L2S is to uncover the optimal pareto set from this tiny set of feasible policy parameters.

Each candidate policy Θ can be mapped to K-dimensional output space for the given K target objectives. Figure 4 shows such policies evaluated and mapped to two-dimensional space for two objectives, i.e., energy and performance when L2S is run for 1,000 iterations. Figure 4(a) illustrates

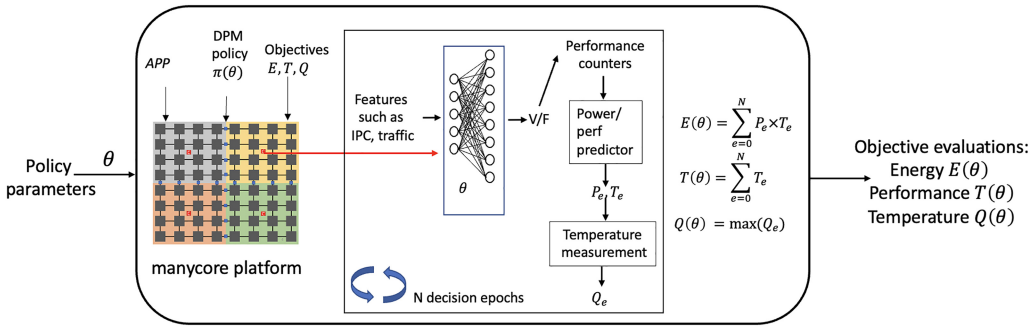

Fig. 5. Detailed description and illustration of policy evaluation step (Section 4.2) within the L2S framework. Policy parameters Θ are input to this block and corresponding objective evaluations are output. Target application(s) ***APP*** is run on the given manycore platform for $\boldsymbol{N}$ decision epochs. In each decision epoch, each VFI controller predicts V/F level by taking ***APP*** features into consideration. Pruning is done in this step to predict V/F that satisfies the given performance penalty constraint. Performance counters corresponding to pruned V/F level are given as input to power/performance models to predict power consumption and execution time in each decision epoch. Next, temperature corresponding to this V/F trajectory is calculated using power and physical floorplan details of the manycore system using a temperature measurement tool. Finally, cumulative Energy $\boldsymbol{E}(\Theta)$, execution time $\boldsymbol{T}(\Theta)$, and temperature $\boldsymbol{Q}(\Theta)$ are then calculated at the end of $\boldsymbol{N}$ decision epochs.

Table 1. List of performance counters for power/performance predictors

Performance counters recorded			
IPC	Branch instructions	Instruction fetch access	Branch mispredictions
Instructions retired	Floating point instructions	Memory access latency	L2 cache requests
Num cycles	Number of load/stores	L2 cache miss	Data cache access

that the naïve L2S approach is not able to uncover even a single policy in the desired "good" policies space (5% is the user-defined performance penalty constraint). From the sequence of DPM configurations, we observed that there are many DPM configurations, which do not satisfy the allowable performance penalty. One reason is that we do not have any training data about the feasible DPM policies yet, and we rely on exploration using the available (possibly incorrect) knowledge in the form of statistical models. This observation motivated us to refine the policy evaluation by avoiding such obvious bad DPM configurations. The key idea is to use power/performance models [34] and the definition of DPM problem to prune undesired DPM configurations at each decision epoch and have the power management policy select the best scoring DPM configuration among the promising DPM configurations after pruning. As shown in Figure 4(b), our refined policy evaluation with pruning allows us to quickly uncover policy parameters that meet the p% (e.g., 5%) performance constraint: The initial ones will serve as high-quality training examples for statistical models, and the learned statistical models will allow us to further accelerate the search for feasible DPM policies and the optimal constrained pareto set of policy parameters. Figure 5 provides the detailed description and illustration of policy evaluation step within the L2S framework.

To perform pruning, we use power/performance models that are parametric functions of the performance counters listed in Table 1. These models are trained using a non-linear NN regressor [35], and Table 2 shows the MLP configuration of the NN regressor used. Training these models requires characterization of the applications while running at different configurations. Specifically, we sweep the V/F levels from 0.65 to 1.0 V in steps of 0.05 V (and corresponding frequency levels

Table 2. MLP Regressor Configuration for Power/performance Models (Section 4.2)

Model Hyperparameters	No. of hidden layers	2
	No. of Neurons	20 in each layer
	Activation	ReLU
	Optimizer	Adam
	Learning Rate	0.001
	Loss function	Cross entropy
Training parameters	Batch size	200
	Epochs	500

mentioned in Section 5.1). Next, we divide the aggregate set of training data into ten folds. We separate out three randomly selected folds for validation and use the remaining seven folds for training. **Mean absolute percentage error (MAPE)**, as shown in Equation (2), is typically used as an error metric for the regression-based approach. Here Y_e is the real value, and $\hat{Y}$ is the predicted value. MAPE error of the NN regressor remains within 3% to 4% for power/performance models across all applications. It may be noted that any other regression model such as support vector regression, regression tree, and their ensemble variants can be employed to form the model and analysis of different regression models lie outside the scope of the article.

$$Error = \frac{\sum_{e=0}^{N}(Y_e - \hat{Y})/\ Y_e}{N} \times 100. \tag{2}$$

At each decision epoch, we prune the bad DPM configurations to identify the promising set using the power/performance models as follows. First, we check whether the predicted V/F levels by the power management policy with parameters Θ satisfies the p% performance constraint using the performance model. If not, then we iterate over the next best V/F configurations from the policy until we find a V/F configuration that meets the p% performance constraint. In other words, we improve the performance of mth VFI with highest performance penalty by iteratively increasing the V/F level. Second, we iterate over the next best V/F configurations from the policy to find the V/F configuration that maximizes energy savings (via power model) while satisfying the p% performance constraint (via performance model). In other words, we reduce the power by iteratively lowering the V/F level of mth VFI such that the p% performance constraint holds true for the entire system. These two steps are repeated for each VFI. Once we identify the performance constrained pareto frontier policies, we further constrain the DPM policy space by examining $Q(\Theta)$ trajectory and pruning the policies that do not satisfy the peak temperature constraint $Q_{\max}$. Pruning is done in the order performance > energy > temperature to reduce the algorithmic computation and, hence, L2S framework's runtime to find an optimal policy. Performance pruning is executed before energy and temperature due to an observation that majority V/F configurations do not satisfy the performance constraint.

4.3 Selecting Policy Parameters

The effectiveness of the L2S framework also depends on the reasoning procedure to select the candidate policy parameters Θ for evaluation in each iteration. Our goal is to use the predictions and uncertainty estimates from the learned statistical models to quickly approximate the constrained optimal pareto front (i.e., the part of the optimal pareto front that meets the p% performance penalty constraint) in a small number of iterations. For the sake of completeness, we also define the notion of optimal pareto set and pareto front. The set of policy parameters $\mathcal{X}^*$ such

that no other policy parameters Θ' $\notin \mathcal{X}^*$ pareto-dominates a policy Θ in $\mathcal{X}^*$ is called the optimal pareto set of policies and the corresponding objective values ($\vec{Y}$, execution time and energy for each policy Θ in $\mathcal{X}^*$) is called the optimal pareto front $\mathcal{Y}^*$. Generally, $\vec{Y}$ is an output vector over K design objectives.

Recall that we formulate the problem of finding DPM policy as an optimization problem in the space of policy parameters Θ. Our goal is to find policy parameters in the search space with optimal energy consumption subject to performance and temperature constraints. Intuitively, L2S iteratively selects candidate policy parameters Θ for evaluation that will take us closer to the optimal solution in a small number of iterations. We formalize this through the notion of maximizing information gain about the constrained optimal pareto front $\mathcal{Y}_c^*$. We propose to apply an information-theoretic algorithm [36] that selects the next candidate policy parameters Θ, given the aggregate training data of policy evaluation D: pairs of input (policy parameters) and output (energy, execution time, and peak temperature evaluation) as explained in Section 4.1. Our utility function is given by the following mathematical expression:

$$\alpha(\Theta) = I\left(\{\theta, \vec{Y}\}, \mathcal{Y}_c^* \mid \mathcal{D}\right), \tag{3}$$

$$= H(\mathcal{Y}_c^* \mid \mathcal{D}) - E_{\mathbb{Y}}\left[H\left(\mathcal{Y}_c^* \mid \mathcal{D} \cup \{\Theta, \vec{Y}\}\right)\right], \tag{4}$$

$$= H(\vec{Y} \mid \mathcal{D}, \Theta) - E_{\mathbb{Y}_c^*}\left[H\left(\vec{Y} \mid D, \Theta, \mathcal{Y}_c^*\right)\right]. \tag{5}$$

Information gain $I(.)$ is defined as the expected reduction in entropy $H(.)$ of the posterior distribution $P(\mathcal{Y}_c^* \mid \mathcal{D})$ over the optimal constrained pareto front $\mathcal{Y}_c^*$ as given in Equations (4) and (5) resulting from the symmetric property of information gain. The first term in the right-hand side of Equation (5), i.e., the entropy of a factorizable K-dimensional Gaussian distribution $P(\vec{Y} \mid \mathcal{D}, \Theta)$ can be computed in closed form as shown in Equation (6):

$$H(\vec{Y} \mid \mathcal{D}, \Theta) = \frac{K(1 + \ln(2\pi))}{2} + \sum_{i=1}^{K} \ln(\sigma_i(\Theta)), \tag{6}$$

where $\sigma_i^2(\Theta)$ is the predictive variance of the ith GP model at input Θ. Intuitively, it says that the entropy is distributed over the K GP models by the sum of their log standard-deviations. The second term in the right-hand side of Equation (5) is an expectation over the optimal constrained pareto front $\mathcal{Y}_c^*$. We can approximately compute this term via Monte Carlo sampling as shown in Equation (7), where S is the number of samples and $\mathcal{Y}_{c_s}^*$ denotes a sample pareto front. The reader is referred to Reference [36] for complete details of the derivation,

$$E_{\mathbb{Y}_c^*}\left[H\left(\vec{Y} \mid \mathcal{D}, \theta, \mathcal{Y}^*\right)\right] \simeq \frac{1}{S}\sum_{s=1}^{S}\left[H\left(\vec{Y} \mid \mathcal{D}, \Theta, \mathcal{Y}_{c_s}^*\right)\right]. \tag{7}$$

4.4 Supervised Learning to Estimate $\hat{\Theta}$

Once we identify the constrained pareto front (or pareto set) from L2S, we perform full system simulations using a cycle-accurate simulator to measure the EDP associated with the sequence of DPM configurations selected by each candidate policy in the pareto set, i.e., policy evaluation with pruning. Next, we select the best policy Θ_{opt} with the lowest EDP and perform supervised learning using the sequence of DPM configurations obtained by policy Θ_{opt} after pruning for a sample of initial states to learn the parameters of the DPM policy function $\hat{\Theta}$. Recall that we get one single trajectory (sequence of DPM decisions for each epoch) for each initial state. Our goal is to learn the parameters of policy function $\hat{\Theta}$ to mimic the corresponding power management

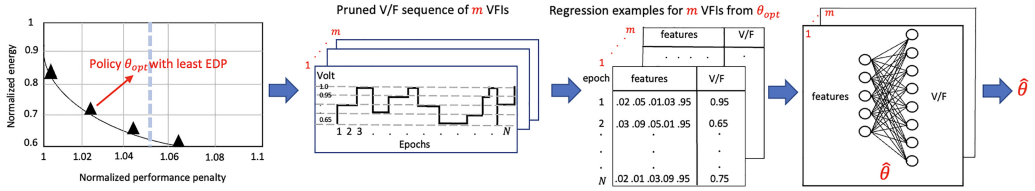

Fig. 6. Policy parameters Θ_{opt} with least EDP uncovers a V/F trajectory (selected configuration after pruning at each decision epoch). For each decision epoch, we add $\boldsymbol{m}$ regression/training examples (features such as IPC, inter-core traffic and the V/F of the VFI from the V/F trajectory), one for each VFI. Finally, DPM policy parameters $\hat{\Theta}$ are learned for $\boldsymbol{m}$ VFIs using supervised learning via a MLP classifier.

Table 3. MLP Classifier Configuration for Supervised Learning (Section 4.4)

Model Hyperparameters	No. of hidden layers	1
	No. of Neurons	5
	Activation	ReLU
	Optimizer	Adam
	Learning Rate	0.003
	Loss function	Cross entropy
Training parameters	Batch size	20
	Epochs	200

behavior without any pruning. In other words, if we use the policy function with parameters $\hat{\Theta}$ to make DPM decision at each epoch using the input features of the application workload, then these decisions should match with the trajectory. This is done by collecting classification examples at each decision epoch (features of the system as input and V/F level from the trajectory as output) and the aggregate set of classification training examples over (different) application workloads and initial states are used to estimate the parameters $\hat{\Theta}$ by minimizing the classification error as shown in Figure 6. A MLP classifier (parameters listed in Table 3) is used for supervised learning. We divide the aggregate set of classification training examples into ten folds. We separate out three randomly selected folds for validation and use the remaining seven folds for training. MAPE loss (Equation (2)) of the MLP regressor is within 5% on the validation set.

5 EXPERIMENTS AND RESULTS

5.1 Experimental Setup

Manycore platform and Benchmarks. We employ GEM5 [37], a full-system simulator, to obtain detailed processor and network-level information. In all the experiments, we consider a system with 64×86 cores running Linux within the GEM5 platform in full-system mode, noting that L2S principles are applicable to higher core count as well. Three SPLASH-2 [38] benchmarks (FFT, LU, and WATER) and four PARSEC [39] benchmarks: (CANNEAL, FLUIDANIMATE (FLUID), DEDUP, and VIPS) are considered for experimental evaluation noting that our findings are similar for the other benchmarks. These applications are selected, as they are representative of various characteristics as follows: FFT (high IPC and high traffic), CANNEAL (memory intensive but low IPC), WATER and LU (high IPC and low traffic), and FLUID (high off-chip bandwidth requirement). The performance counters generated by GEM5 simulations are given as input to McPAT [40] to determine the power values. Steady state on-chip temperature at each decision epoch is calculated by Hotspot [41] using the power traces as input.

Table 4. VFI Cluster Sizes for Various Benchmarks

Benchmark	VFI 1	VFI 2	VFI 3	VFI 4
CANNEAL	22	22	16	4
FFT	29	23	7	5
FLUID	40	16	4	4
LU	32	24	4	4
WATER	41	15	4	4
DEDUP	40	16	4	4
VIPS	30	26	4	4

VFI system. We consider four VFI clusters as shown in Table 4, while imposing a minimum VFI cluster size of four cores. By using the k-means algorithm, we cluster the cores to minimize each VFI's intra-cluster variation in the time-varying computation and traffic statistics [42]. It should be noted that the analysis of VFI clustering methods is beyond the scope of this article and any clustering approach could be used to similar effect.

Design Objectives. We consider three primary design objectives, namely, performance, energy, and peak chip temperature, to test the effectiveness of different DPM algorithms.

Decision space for DPM policies. We consider nominal range of operation in the 28-nm technology node. We use eight discrete V/F pairs for both the wireless- and the M3D NoC-enabled architectures. Due to the difference in the physical layer characteristics of wireless and M3D architectures, their V/F levels differ. The V/F levels for wireless architecture are (volts/GHz): 1.0/3.0, 0.95/2.75, 0.9/2.5, 0.85/2.23, 0.8/1.94, 0.75/1.64, 0.7/1.33, and 0.65/1.02 and the corresponding levels for the M3D architecture are (volts/GHz): 1.0/3.5, 0.95/3.2, 0.9/2.9, 0.85/2.58, 0.8/2.25, 0.75/1.9, 0.7/1.54, and 0.65/1.18. The DPM decision space is defined by the number of VFIs and their respective V/F values. As we have four VFIs and eight V/F pairs, there are 4,096 possible DPM decisions for each system state.

DPM policy representation. One function (e.g., MLP) is used for each VFI to predict V/F values at each decision epoch using the following input features: each VFI's average and peak traffic, average and peak computation, and previous epoch V/F level [9]. The MLP configuration used to represent each of the four VFI controllers is as follows: one input layer with the ReLU activation and an output layer with the softmax activation. The number of output layer neurons is equal to number of possible DPM decisions (e.g., eight for discrete V/F levels).

5.2 L2S Framework and Baseline DPM Algorithms

L2S method. We used initial 30 samples (consisting of policy parameter Θ and corresponding energy $E(\Theta)$, execution time $T(\Theta)$, and temperature $Q(\Theta)$) to bootstrap the statistical models. L2S is an iterative method and high-quality pareto-optimized policies are generated within 200 iterations across all the benchmarks. We select the policy from the pareto front that minimizes EDP subject to p% (set to 5% in experiments) performance penalty and $Q_{\max}$ (set to 85 °C) peak temperature constraint for runtime execution. We compare the performance of our L2S framework with the existing ML methods such as RL and IL.

Reinforcement Learning. We use the state-of-the-art RL method, namely, PPO, in our experiments. PPO is shown to achieve high accuracy for the learned policy [43, 44]. We employ the same policy representation as the L2S framework for actor–critic networks for PPO. Prior work [9, 11] has constructed a single reward function for two objectives, i.e., energy $E(s, a)$ and performance $T(s, a)$ in Equation (8), where (s, a) represents state–action pair. In this work, we extend the reward function to include the third objective, i.e., temperature $Q(s, a)$ for peak temperature

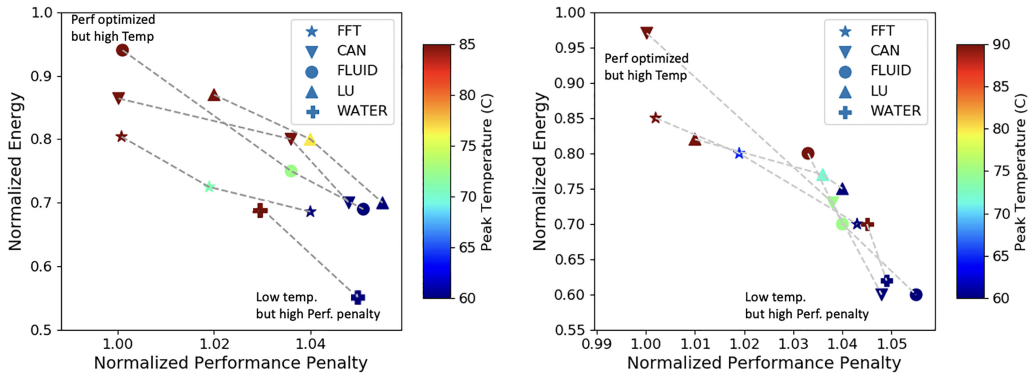

Fig. 7. Pareto-optimal policies uncovered by L2S method, illustrating energy-performance-thermal tradeoff for (a) wireless, and (b) M3D-NoC architecture for various applications.

constraint. The scalarization parameters λ_1, λ_2 are varied in the range {1, 10, 100, 1000} to achieve a desired tradeoff for each application workload. Although PPO is a sample-efficient method, over 1,000 iterations were needed for convergence,

$$R(s,a) = E(s,a) + \lambda_1 \cdot T(s,a) + \lambda_2 \cdot Q(s,a). \tag{8}$$

Imitation Learning. For VFI-enabled systems, an expert is defined as the policy that allocates the best V/F levels for each VFI to minimize EDP while satisfying the p% performance constraint and peak temperature constraint. In our experiments, we consider 5% performance penalty and 85 °C peak temperature constraint to construct the hand-designed expert DPM policy from prior work [9]. Exact IL algorithm with regression tree learning combined with data aggregation technique (to avoid error propagation) is employed to mimic the expert DPM policy.

DTPM. We adapt the algorithm proposed in Reference [16] to the VFI-enabled manycore platform for comparison with the proposed L2S framework. The DTPM algorithm starts from the temperature constraint $Q_{\max}$ (85 °C) and works backward to compute power budget and corresponding maximum V/F values of the VFIs to maintain the temperature below $Q_{\max}$. The goal of DTPM is to prevent temperature violations while maximizing the performance. If the predicted temperature of any VFI cluster exceeds the $Q_{\max}$ constraint, then power budget is computed, and V/F is reduced to the level that satisfies the power budget.

5.3 Energy–Performance–Thermal Tradeoff

One key advantage of L2S over IL- and RL-based methods is that pareto-optimized policies are available to the designer with minimal effort, i.e., pareto-optimal policies are available to the designer in one L2S run, whereas one needs to vary the scalarization parameters λ_1, λ_2 to uncover multiple policies in IL- and RL-based methods. Figure 7 shows the pareto-optimal L2S policies demonstrating energy–performance–thermal tradeoff for various applications for (a) wireless and (b) M3D NoC-enabled manycore systems. We show that L2S policies with 4% to 5% performance penalty dissipate less energy and have lower peak temperatures compared to policies with near-zero performance penalty. L2S uncovers DPM policies that reduce energy by up to 20% for various application workloads almost at zero performance penalty. These policies may appear attractive to the designer if thermal constraints were not considered. However, such policies may not satisfy peak temperature constraint and thereby reduce thermal reliability. For example, in M3D NoC-enabled manycore systems, several L2S policies with low (near-zero) performance penalty do not

satisfy the 85 °C peak temperature constraint and, hence, cannot be considered. We demonstrate that the joint performance-thermal-constrained pareto front is both workload and NoC architecture dependent, and L2S automates the search process enabling the designer to choose a DPM policy along the pareto front that has desired thermal margin and performance penalty.

Next, we compare the performance of the proposed L2S framework with respect to IL- and RL-based methods. All the results are normalized with respect to a system without VFI (NVFI). Since EDP is a metric that captures both energy and execution time in one parameter, we use it as the relevant measure to evaluate the quality of L2S, IL, and RL methods. Figure 8(a) and (b) shows the EDP and peak temperature comparison for L2S, IL, and RL for the wireless NoC-enabled manycore architecture. The optimized DPM policy uncovered by the proposed L2S reduces the EDP over IL and RL by up to 10% and 22%, respectively. Furthermore, these application-specific L2S policies reduce the peak temperature by up to 3 °C and 12 °C over IL- and RL-based methods, respectively. Similarly, Figure 8(c) and (d) show the EDP and peak temperature results for the M3D NoC-enabled architecture, where L2S policy reduces EDP (peak temperature) by up to 26% and 30% (5 °C and 17 °C) compared to IL and RL, respectively. L2S performs better compared to IL and RL due to its effective search process in the continuous space of parameters guided by learned statistical models, thereby reducing the EDP and peak temperature by highest margin. Figure 9 illustrates the V/F sequence and corresponding temperatures for L2S-, IL-, and RL-based DPM policies considering the FFT application on the wireless NoC-based system as an example. It is evident that throughout the application lifetime, the predicted V/F values are lower for L2S than both IL and RL policies, reducing the power consumption and peak temperature while satisfying the user-defined p% performance penalty constraint. For brevity, we do not show the V/F sequence for all other applications. However, similar results are observed across all the benchmarks on both wireless and M3D architectures. Overall, our results demonstrate that the L2S policy performs equally well irrespective of the physical layer of the NoC architecture. Hence, we can conclude that the proposed L2S methodology is effective for wireless- and M3D-NoC enabled architectures.

5.4 Comparison with DTPM

In this section, we compare the performance of the L2S policy with the DTPM (non-ML state-of-the-art) method in terms of the EDP and peak temperature. As shown in Figure 10(a), EDP achieved via DTPM policy is up to 45% higher than the L2S policy for the LU benchmark. Moreover, the temperature in case of DTPM method remains closer to the 85 °C peak temperature constraint in four of the benchmarks and is higher than L2S policies across all the benchmarks, as shown in Figure 10(b). There are two key reasons behind these sub-optimal results of DTPM policy. First, DTPM utilizes a default frequency governor, such as Ondemand [45], which takes only CPU utilization into consideration to predict frequency. Second, the goal of DTPM is to reduce frequencies only if the temperature is violated. In case temperature is less than the 85 °C peak temperature constraint, it tries to increase the frequency to the highest V/F level of the system without considering power consumption. In other words, it does not solve the multi-objective problem of optimizing power, performance, and temperature and is sub-optimal.

5.5 Application-agnostic Policy

In this section, we discuss that an application-agnostic L2S policy can show similar performance as an application-specific L2S policy. To design an application-agnostic L2S-based DPM policy (termed as AVG), we consider a set of training applications and learn policy parameters from the aggregate training data from expert DPM policies for those applications. For each of the W applications, we create a different AVG policy using the set of remaining $W - 1$ applications (leave-one-out). Each AVG configuration executes the aggregate DPM policy (trained using the aggregate

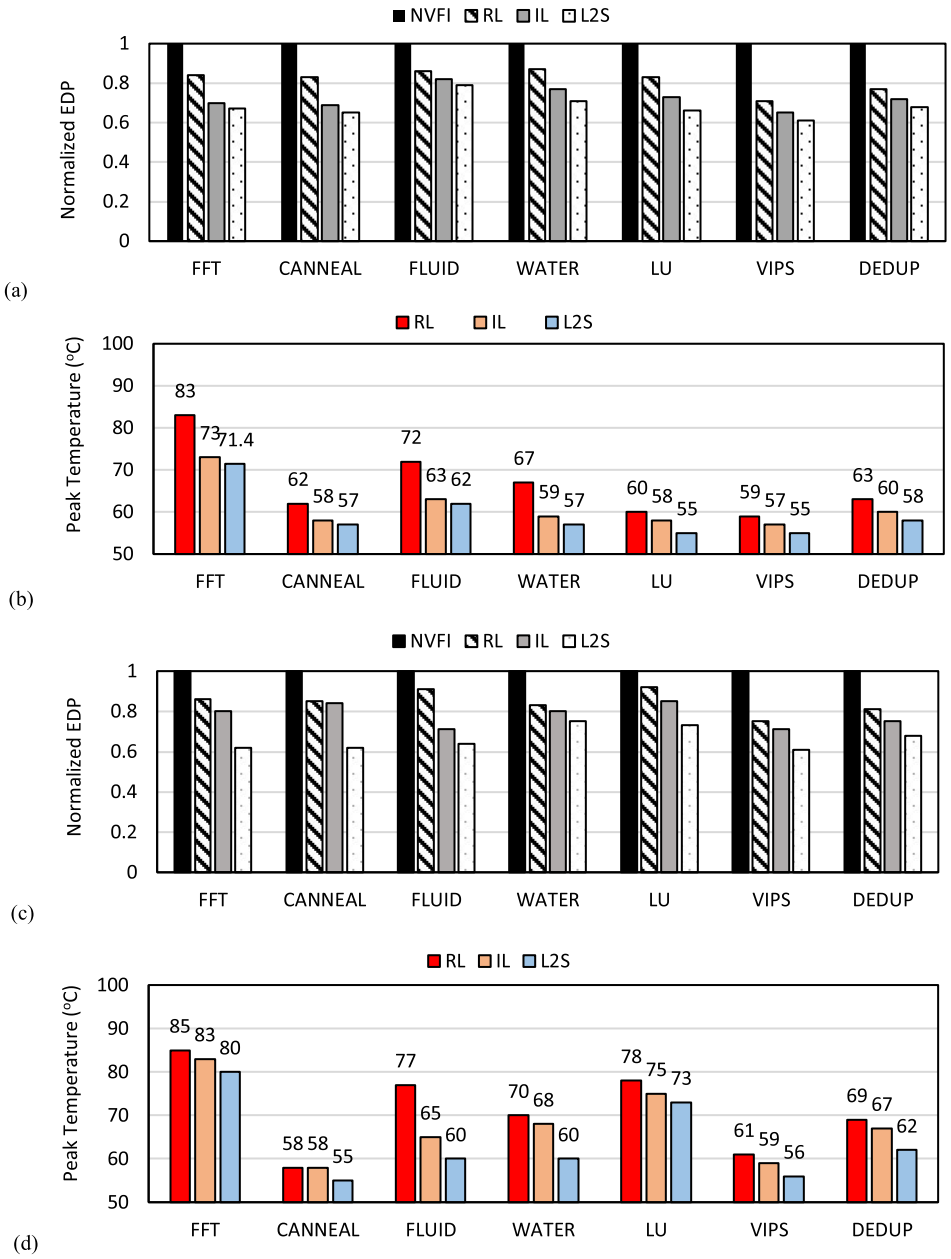

Fig. 8. EDP (normalized with respect to NVFI) and peak temperature comparison of RL, IL, and L2S policies for wireless (a and b), and M3D (c and d) NoC architectures.

supervised data from expert policy for each of the $W - 1$ applications) on the application that was left out during policy optimization (otherwise unknown to the optimization). As an example, we learned an AVG policy using CANNEAL, WATER, LU, and FLUID and left out FFT. Next, we execute FFT using this AVG policy. Figure 11 shows the normalized EDP and temperature of AVG policy on the wireless NoC-based architecture for all the applications under consideration. From

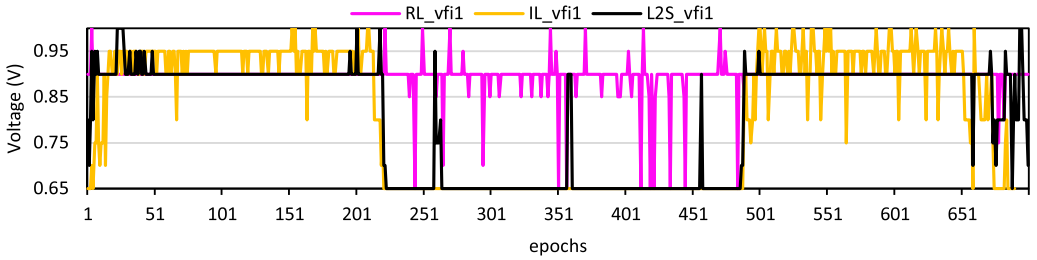

Fig. 9. VFI 1's predicted voltage for RL, IL and L2S policy running FFT on the wireless NoC architecture.

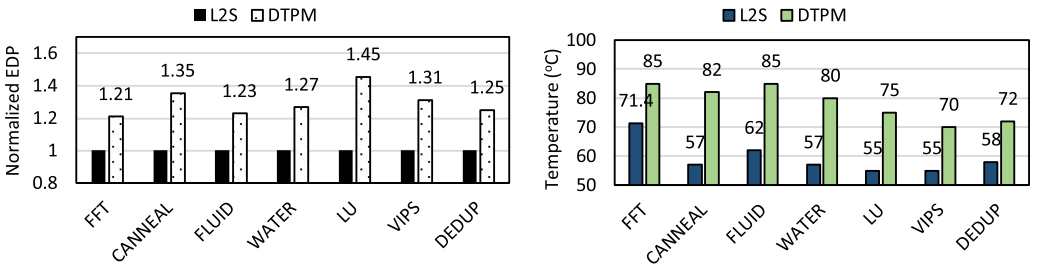

Fig. 10. (a) EDP (normalized with respect to L2S) and (b) peak temperature comparison of L2S and DTPM (state-of-the-art non-ML) policies on the wireless NoC architecture.

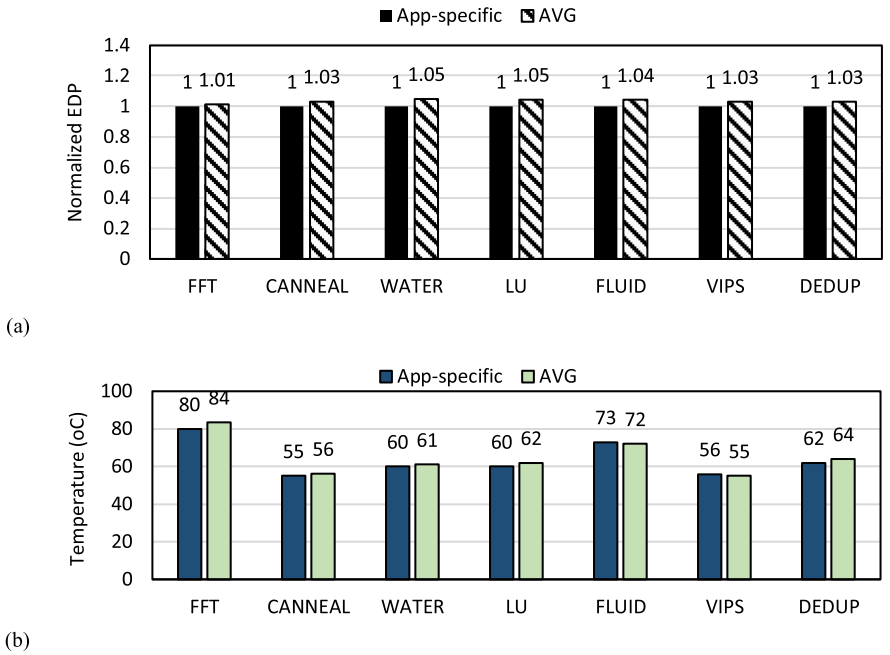

Fig. 11. (a) EDP comparison and (b) temperature of application-specific policies vs AVG policy.

Figure 11(a), we note that, on average, only 3.6% degradation in EDP is observed for all applications when compared to application-specific policies with worst case reaching only up to 5%. We see the same trend for the M3D-based architecture, too. Similarly, temperature of AVG policy has variance of 1.7 °C on an average with respect to application-specific policies, as shown in Figure 11(b). By learning from aggregate training data of multiple applications, application-agnostic policy can better generalize to the unseen application. Therefore, an application-agnostic policy optimized for a subset of applications can be reused for a new application of the suite without significant penalty in EDP.

5.6 Implementation Overhead

The VFI controller is represented by the same MLP function for all three ML-based methods (L2S, IL, and RL). Hence, the storage cost and decision-making time for each method is the same. The memory required to store the DPM policy is 4 Kb, which is negligible. Area overhead of the VFI controller is 0.03% for a 20×20 mm^2 die. Energy consumed per decision is 62.4 pJ, and per-decision execution of a DPM policy takes 0.24% of the decision epoch interval. Note that all ML-based methods (L2S, IL, and RL) are executed offline to create DPM policies that are executed at runtime (i.e., no training at runtime). Therefore, we do not report training overhead details noting that L2S has relatively less overhead.

6 CONCLUSION

DPM is a common strategy to reduce energy consumption of a manycore system without introducing unnecessary performance overhead. We proposed a L2S framework for creating optimized DPM policies for manycore systems, where the search is intelligently guided by learned statistical models. We considered a VFI-based DPM to show the effectiveness of L2S with respect to existing machine learning-based methods. Our experiments demonstrate that DPM policy uncovered by the proposed L2S framework reduces energy–delay–product (peak temperature) overIL and RL policies by up to 26% and 30% (13 °C and 17 °C), respectively, for two qualitatively different manycore architectures. Furthermore, we demonstrate the application-agnostic nature of the L2S policy. An application-agnostic policy achieves equally good performance as the application-specific counterpart.

REFERENCES

[1] A. Aalsaud, R. Shafik, A. Rafiev, F. Xia, S. Yang, and A. Yakovlev. 2016. Power-aware performance adaptation of concurrent applications in heterogeneous many-core systems. In *Proceedings of the International Symposium on Low Power Electronics and Design*. 368–373.

[2] S. M. P. Dinakarrao, A. Joseph, A. Haridass, M. Shafique, J. Henkel, and H. Homayoun. 2019. Application and thermal-reliability-aware reinforcement learning based multi-core power management. *ACM J. Emerg. Technol. Comput. Syst.* 15, 4 (2019), 1–19.

[3] U. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung. 2019. Design and management of voltage-frequency island partitioned networks-on-chip. *IEEE Trans. VLSI Syst.* 17, 3 (2009), 330–341.

[4] A. Sartor, N. Krohmer, H. Khdr, and J. Henkel. 2020. HiLITE: Hierarchical and lightweight imitation learning for power management of embedded SoCs. *IEEE Comput. Arch. Lett.* 19, 1 (2020), 63–67.

[5] M. Shafique, S. Garg, J. Henkel, and D. Marculescu. 2014. The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives. In *Proceedings of the 51st Annual Design Automation Conference*. 1–6.

[6] M. Rapp, N. Krohmer, H. Khdr, and J. Henkel. 2022. NPU-Accelerated imitation learning for thermal- and QoS-Aware optimization of heterogeneous multi-cores. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'22)*. 584–587.

[7] H. Li, Z. Tian, R. K. Maeda, X. Chen, J. Feng, and J. Xu. 2018. Co-Manage power delivery and consumption for manycore systems using reinforcement learning. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'18)*. 1–8.

[8] Z. Chen and D. Marculescu. 2015. Distributed reinforcement learning for power limited many-core system performance optimization. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'15)*. 1521–1526.
[9] R. G. Kim, W. Choi, Z. Chen, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu. 2017. Imitation learning for dynamic VFI control in large-scale manycore systems. *IEEE Trans. VLSI Syst.* 25, 9 (2017), 2458–2471.
[10] S. K. Mandal, G. Bhat, C. A. Patil, J. R. Doppa, P. P. Pande, and U. Y. Ogras. 2019. Dynamic resource management of heterogeneous mobile platforms via imitation learning. *IEEE Trans. VLSI Syst.* 27, 12 (2019), 2842–2854.
[11] A. Deshwal, S. Belakaria, G. Bhat, J. R. Doppa, and P. P. Pande. 2021. Learning pareto-frontier resource management policies for heterogeneous SoCs: An information-theoretic approach. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC'21)*. 607–612.
[12] S. Bobak, K. Swersky, Z. Wang, R. P. Adams, and N. D. Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.
[13] S. Pagani, P. S. Manoj, A. Jantsch, and J. Henkel. 2018. Machine learning for power, energy, and thermal management on multicore processors: A survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 39, 1 (2018), 101–116.
[14] A. K. Singh, S. Dey, K. McDonald-Maier, K. R. Basireddy, G. V. Merrett, and B. M. Al-Hashimi. 2020. Dynamic energy and thermal management of multi-core mobile platforms: A survey. *IEEE Des. Test* 37, 5 (2020), 25–33.
[15] A. Prakash, H. Amrouch, M. Shafique, T. Mitra, and J. Henkel. 2016. Improving mobile gaming performance through cooperative CPU-GPU thermal management. In *Proceedings of the 53rd Annual Design Automation Conference*. 1–6.
[16] G. Singla, G. Kaur, A. Unver, and U. Ogras. 2015. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'15)*. 960–965.
[17] G. Bhat, G. Singla, A. K. Unver, and U. Y. Ogras. 2017. Algorithmic optimization of thermal and power management for heterogeneous mobile platforms. *IEEE Trans. VLSI Syst.* 26, 3 (2017), 544–557.
[18] E. W. Wächter, C. D. Bellefroid, K. R. Basireddy, A. K. Singh, B. M. Al-Hashimi, and G. Merrett. 2019. Predictive thermal management for energy-efficient execution of concurrent applications on heterogeneous multicores. *IEEE Trans. VLSI Syst.* 27, 6 (2019), 1404–1415.
[19] G. Bhat, S. Gumussoy, and U. Y. Ogras. 2017. Power temperature stability and safety analysis for multiprocessor systems. *ACM Trans. Embed. Comput. Syst.* 16, 5 (2017), 1–19.
[20] S. Isuwa, S. Dey, A. K. Singh, and K. McDonald-Maier. 2019. TEEM: Online thermal- and energyefficiency management on CPU-GPU MPSoCs. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'19)*. 438–443.
[21] M. Rapp, M. B. Sikal, H. Khdr, and J. Henkel. 2021. SmartBoost: Lightweight ML-driven boosting for thermally-constrained many-core processors. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC'21)*. 265–270.
[22] S. Lu, R. Tessier, and W. 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.
[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, and S. Petersen. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[24] S.-G. Yang, Y.-Y. Wang, D. Liu, X. Jiang, H. Fang, Y. Yang, and M. Zhao. 2019. ReLeTA: Reinforcement learning for thermal-aware task allocation on multicore. arXiv:1912.00189. Retrieved from https://arxiv.org/abs/1912.00189.
[25] S. Mandal, K. Gaurkar, P. Dasgupta, and A. Hazra. 2021. An RL based approach for thermal-aware energy optimized task scheduling in Multi-core processors. In *Proceedings of the 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID'21)*. 181–186.
[26] S. K. Mandal, G. Bhat, J. R. Doppa, P. P. Pande, and U. Y. Ogras. 2020. An energy-aware online learning framework for resource management in heterogeneous platforms. *ACM Trans. Des. Autom. Electr. Syst.* 25, 3 (2020), 1–26.
[27] S. Deb, K. Chang, X. Yu, S. P. Sah, M. Cosic, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo. 2012. Design of an energy-efficient CMOS-compatible NoC architecture with millimeter-wave wireless interconnects. *IEEE Trans. Comput.* 62, 12 (2012), 2382–2396.
[28] R. G. Kim, W. Choi, Z. Chen, P. P. Pande, D. Marculescu, and R. Marculescu. 2016. Wireless NoC and dynamic VFI codesign: Energy efficiency without performance penalty. *IEEE Trans.VLSI Syst.* 24, 7 (2016), 2488–2501.
[29] J. Murray, R. Kim, P. Wettin, P. P. Pande, and B. Shirazi. 2014. Performance evaluation of congestion-aware routing with dvfs on a millimeter-wave small-world wireless noc. *ACM J. Emerg. Technol. Comput. Syst.* 11, 2 (2014), 1–22.
[30] Y.-J. Lee and S. K. Lim. 2013. Ultrahigh density logic designs using monolithic 3-D integration. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 32, 12 (2013), 1892–1905.

[31] A. Chatterjee, S. Musavvir, R. G. Kim, J. R. Doppa, and P. P. Pande. 2021. Power management of monolithic 3D manycore chips with inter-tier process variations. *ACM J. Emerg. Technol. Comput. Syst.* 17, 2 (2021), 1–19.
[32] C. Liu and S. Lim. 2012. A design tradeoff study with monolithic 3D integration. In *Proceedings of the 13th International Symposium on Quality Electronic Design (ISQED'12)*. 529–536.
[33] C. Rasmussen. 2003. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, Springer, Berlin, 63–71.
[34] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang. 2014. PPEP: Online performance, power, and energy prediction framework and DVFS space exploration. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*. 445–457.
[35] P. Fabian et al. 2011. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, Vol. 12, (2011), 2825–30.
[36] Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. 2021. Output space entropy search framework for multi-objective Bayesian optimization. *Journal of Artificial Intelligence Research* 72, (2021), 667–715.
[37] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, and J. Hestness, et al. 2011. The gem5 simulator. *ACM SIGARCH Comput. Arch. News* 39, 2 (2011), 1–7.
[38] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. 1995. The SPLASH-2 programs: Characterization and methodological considerations. *ACM SIGARCH Comput. Arch. News* 23, 2 (1995), 24–36.
[39] C. Bienia, S. Kumar, J. P. Singh, and K. Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*. 72–81.
[40] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*. 469–480.
[41] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan. 2006. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Trans.VLSI Syst.* 14, 5 (2006), 501–513.
[42] S. Hajiamini, B. Shirazi, and H. Dong. 2022. A fast heuristic for improving the energy efficiency of asymmetric VFI-Based manycore systems. *IEEE Trans. Sust. Comput.* 7, 2 (2022), 358–370.
[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. Retrieved from https://arxiv.org/abs/1707.06347.
[44] Y. Meng, S. Kuppannagari, and V. Prasanna. 2020. Accelerating proximal policy optimization on CPU-FPGA heterogeneous platforms. In *Proceedings of the IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM'20)*. 19–27.
[45] V. Pallipadi and A. Starikovskiy. 2006. The ondemand governor. In *Proceedings of Linux Symposium* 2, 00216 (2006), 215–230.

Received 16 January 2023; revised 18 April 2023; accepted 18 May 2023