Machine Learning for Power, Energy, and Thermal Management on Multicore Processors: A Survey

Santiago Pagani, *Student Member, IEEE*, P. D. Sai Manoj, *Member, IEEE*, Axel Jantsch, *Member, IEEE*, and Jörg Henkel, *Fellow, IEEE*

***Abstract*—Due to the high integration density and roadblock of voltage scaling, modern multicore processors experience higher power densities than previous technology scaling nodes. When unattended, this issue might lead to temperature hot spots, that in turn may cause nonuniform aging, accelerate chip failure, impair reliability, and reduce the performance of the system. This paper presents an overview of several research efforts that propose to use machine learning (ML) techniques for power and thermal management on single-core and multicore processors. Traditional power and thermal management techniques rely on a certain *a-priori* knowledge of the chip's thermal model, as well as information of the workloads/applications to be executed (e.g., transient and average power consumption). Nevertheless, these *a-priori* information is not always available, and even if it is, it cannot reflect the spatial and temporal uncertainties and variations that come from the environment, the hardware, or from the workloads/applications. Contrarily, techniques based on ML can potentially adapt to varying system conditions and workloads, learning from past events in order to improve themselves as the environment changes, resulting in improved management decisions.**

***Index Terms*—Energy efficiency, machine learning (ML), multicore systems, neural networks, on-chip resource management, power management, reinforcement learning, thermal management.**

I. Introduction

HIGH power densities and temperatures on many-core systems are the result of ever-increasing transistor integration coupled with the observed limits on voltage scaling for next-generation technology nodes, which are the causes behind the emerging *dark silicon problem* [1]. Particularly, if we wish to keep cooling costs constant (by using a common cooling solution for several scaling generations) without violating the chip's thermal constraints, it is no longer possible to simultaneously activate all the cores on a chip at the nominal operating levels [1]–[4]. When unattended, this issue might lead to temperature hot spots, that in turn may cause nonuniform aging, accelerate chip failure, impair reliability, and reduce the performance of the system. Therefore, given that this effect can slow down the current performance gain trends between scaling generations, it challenges the viability of further cost-effective technology scaling [5]. Due of these reasons, effective power/thermal management techniques are now more relevant, especially for performance optimization, as they intend to avoid chips from possible overheating while not incurring in high cooling costs, thus promising to maintain technology scaling trends feasible. Moreover, to prolong battery lifetime of embedded systems or to cut power bills of servers, energy management for minimizing overall energy consumption while satisfying performance (or real-time) constraints is another relevant (almost dual) problem.

In addition to the traditional prediction techniques, machine learning (ML)-based predictors, such as Bayesian learning [6]–[8], neural networks [9], reinforcement learning [10]–[16], and regression analysis [17]–[21] are also widely employed for prediction and to perform dynamic voltage and frequency scaling (DVFS). The ML-based power/performance/temperature management adds the benefit of learning the characteristics or trends of workloads in order to take control decisions.

In this paper, we present an overview of several research efforts that propose to use ML techniques for power (or energy) and thermal management on single-core or multicore processors. Traditional power and thermal management techniques generally rely on a certain *a-priori* knowledge of the chip's thermal model, as well as information of the workloads/applications to be executed (e.g., transient and average power consumption). Nevertheless, these *a-priori* information is not always available, and even if it is, it cannot reflect the spatial and temporal uncertainties and variations that come from the environment, the hardware, the input data, or from the workloads/applications. The system level power/thermal manager should consider such uncertainty and variability, while resource and power management techniques that are statically optimized are unlikely to achieve the best performance when these characteristics are changing [22], [23]. Contrarily, techniques based on ML can potentially adapt to varying system conditions and workloads, learning from past events in order to improve themselves as the environment changes,

Manuscript received December 25, 2017; revised March 23, 2018 and June 18, 2018; accepted July 29, 2018. Date of publication October 25, 2018; date of current version December 23, 2019. This work was supported in part by the German Research Foundation and in part by the Transregional Collaborative Research Centre *Invasive Computing* under Grant SFB/TR 89. This paper was recommended by Associate Editor J. Xu. *(Corresponding author: Santiago Pagani.)*

S. Pagani was with Embedded Systems, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany. He is now with the Graphics and Multimedia Division, ARM Ltd., Cambridge CB1 9NJ, U.K. (e-mail: santiago.pagani@arm.com).

P. D. Sai Manoj is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030 USA (e-mail: saimanoj.p.2013@ieee.org).

A. Jantsch is with the System-on-Chip Group, Institute of Computer Technology, TU Wien, 1040 Vienna, Austria (e-mail: axel.jantsch@tuwien.ac.at).

J. Henkel is with the Embedded Systems, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany (e-mail: henkel@kit.edu).

Digital Object Identifier 10.1109/TCAD.2018.2878168

TABLE I
SUMMARY OF STATE-OF-THE-ART WORKS FOCUSING ON POWER, ENERGY, AND THERMAL MANAGEMENT USING ML TECHNIQUES. WORKS ARE ORDERED CHRONOLOGICALLY (FROM NEWEST TO OLDEST) AND CATEGORIZED ACCORDING TO THEIR OPTIMIZATION GOAL (MARKED AS "G"), CONSTRAINT (MARKED AS "C"), OPTIMIZATION KNOBS, TARGET ARCHITECTURE, AND THE USED ML TECHNIQUE

Year	Work -	Optimize (G) / Constraint (C)				Optimization Knobs			Architecture			Machine Learning Technique		
		Performance	Power	Energy	Temperature	Task Allocation	DPM	DVFS	Single Core	Homogeneous Multicore	Heterogeneous Multicore	Supervised Learning	Unsupervised Learning	Reinforcement Learning
2016	[48]	C		G			✓			✓	✓			TD(λ)-learning
2015	[15]	G	C					✓		✓				Q-learning
2015	[14]				G	✓				✓				Q-learning
2015	[16]	G	C		G/C	✓		✓		✓	✓			Q-Learning
2015	[36]	C		G				✓		✓			Clustering	
2015	[24]	C		G		✓			✓					TD(λ)-learning
2015	[18]	C		G/C				✓		✓		Rigid linear regression		
2015	[37]	C		G/C				✓		✓				Q-learning
2014	[38]	C			G	✓		✓		✓				Q-Learning
2014	[9]	C		G	C	✓	✓			✓		Neural Network		Q-learning
2014	[25]	C		G			✓		✓					TD(λ)-learning
2013	[26]	C		G	C		✓	✓	✓					Q-learning
2013	[6]	C	G				✓		✓			Bayes classifier		TD(λ)-learning
2013 2011	[39] [40]	C		G	C			✓		✓		Least squares regression		
2012	[7]	G		C			✓		✓					Q-learning
2012	[41]				G	✓				✓		Genetic algorithm	k-means clustering	
2012	[27]	G/C		G/C	G/C			✓	✓					Q-learning
2011	[28]	G/C	G/C				✓		✓			Bayes classifier		TD(λ)-learning
2011	[29]	G			C			✓	✓				k-means clustering	Q-learning
2011	[30]	C		G					✓	✓		Least squares regression		
2011	[42]	G/C			G/C	✓		✓		✓				ad hoc
2010	[43]	G			C		✓	✓		✓		Least squares regression	k-means clustering	
2010	[8]	C		G				✓		✓		Bayes classifier		
2010	[44]			C	G/C			✓		✓		Least squares regression		
2010	[45]	C/G				✓		✓		✓				Observe-decide act
2010	[31]	C	G					✓	✓			Least mean square linear predictor		
2009	[32]	C	G				✓		✓					Q-learning
2009	[33]	G		G			✓	✓	✓					ad hoc
2008	[46]				G/C			✓		✓		LWPR		
2008	[47]	G			G/C	✓	✓	✓		✓				ad hoc
2005	[21]	C		G				✓	✓			Least squares regression		
2002	[34]	G/C	G				✓		✓					Markov Decision Process
1999	[35]			G			✓		✓			Adaptive learning tree		

resulting in improved management decisions. In other words, good power/thermal management controllers should be able to observe, learn, and adapt to different hardware and working environments [23]. Despite ML being widely employed for different purposes, some of the primary challenges in deploying ML for power management can be outlined as follows: the amount of data to be learned and processed can be of higher dimension, i.e., processing overheads, high accurate prediction is needed in some cases, the hardware footprint (if deployed in hardware layer), and the machine learner should have a smaller response time. Furthermore, in some of the techniques, such as reinforcement learning (say Q-learning), convergence concerns are nontrivial to be addressed.

Table I summarizes the state-of-the-art works discussed in this paper. The works in [6], [7], [21], and [23]–[35] focus on single-core systems. The works in [8], [9], [14], [15], [18], and [36]–[47] focus on homogeneous multicore systems. Finally, focus on heterogeneous multicore systems.

II. BACKGROUND

A. Optimization Goals and Constraints

1) Computational Performance: Computational performance refers to how quickly a system can execute an application or a given set of applications. An application's

performance can be measured using multiple metrics, such as execution time, throughput, instructions per cycle (IPC), instructions per second (IPS), speed-up factor (normalized to a known reference), and so on. *Overall system performance* is the term commonly used for an application set, which is a generic term that can, e.g., refer to minimizing the longest execution time among all the applications (i.e., the *makespan*) maximizing the summation of the weighted throughput of all applications (weights add some sort of priority to the applications), and so on.

An application's resulting performance will depend on how the application is executed, e.g., the types of cores which the application is executed on, the execution frequency of the cores, in how many threads the application is parallelized, the simultaneous usage of the shared resources by other applications [such as caches, the network-on-chip (NoC)], etc. An application's individual characteristics [e.g., their level of instruction-level parallelism (ILP) or thread-level parallelism (TLP)] also play a main role in its final performance and directly impact how this performance scales in regards to execution frequency and the number of parallel threads. Applications with a high ILP generally scale well with increasing execution frequencies, while applications with high a TLP generally scale well when they are parallelized in many threads.

Overall system performance maximization is normally the most commonly pursued optimization goal. Nonetheless, for real-time applications, meeting their hard deadlines can be formulated as satisfying performance requirements, and hence performance is considered as a constraint in such cases.

2) Power and Energy Consumption: Power consumption produces heat, and every core doing some computation consumes power (unit *Watt* [*W*]). Power consumption is an instantaneous metric which changes over time. Specifically, a certain application thread running on a specific core will consume different amounts of power at different points in time. Furthermore, the power consumption measured on a core at a certain time also depends on several parameters, e.g., the underlying architecture of the core, the technology node, the voltage/frequency settings, the power mode of the core (e.g., active, idle, sleep), the temperature of the core at that point in time, the current application phase being executed.

Contrarily, the integration of power over time is energy consumption (units *Joule* [*J*] or *Watt second* [*Ws*]). This means that, if plotted, the energy consumed during a time interval (e.g., an application instance) is equal to the area below the power curve during that interval. Thus, when the power consumption during a time interval remains constant, the energy consumed during that interval can simply be computed by multiplying the power by the duration of the time interval.

Overall energy consumption minimization under performance/timing constraints is a common optimization goal for mobile systems that wish to prolong their battery lifetime. On the other hand, power consumption itself is rarely optimized, and thus power is generally viewed as a constraint, e.g., to run the system under a certain power budget.

3) Temperature: As mentioned above, heat is generated when a chip consumes power. Nevertheless, although for practical purposes power consumption changes can be considered as instantaneous, temperature changes do not occur instantaneously, as there are thermal capacitances associated to all elements in a chip. If enough time elapses without changes in power (not something usually observable in computer systems), the temperatures throughout the chip would eventually reach its *steady-states*. The intermediate temperatures observable until these steady-state temperatures are achieved are called *transient temperatures*.

Keeping the chip's temperature under a certain thermal threshold (or critical value) is of paramount importance, as otherwise high temperatures may cause permanent failures in a chip's transistors. In order for this power and temperature to be dissipated, chips are provided with a cooling solution (e.g., the coupling of the thermal paste, heat spreader, heat sink, cooling fan, etc.). To aid designing the cooling solution for a certain chip, the common industry practice is to provide the thermal design power (TDP) of a particular chip, defined as "the highest expected sustainable power while running known power intensive real applications" [49]. Hence, given that the system should be able to safely consume TDP power, manufacturers normally recommend to design the cooling solution to dissipate TDP, avoiding the cooling solution from being over-dimensioned. However, since TDP is not the maximum power that can be consumed, chips are also generally provided with some sort of dynamic thermal management (DTM) technique. These DTM techniques are very commonly reactive (i.e., triggered once the critical temperature is exceeded) and can power-down cores, reduce their supply voltages and execution frequencies, gate their clocks, boost-up the fan speed, etc. In other words, if the chip heats up above a critical value (identified using thermal sensors distributed across the chip), then DTM is triggered to reduce the temperature.

Thermal constraints are the biggest limiting factors for maximizing performance, particularly in modern chips that have very high power densities due to the dark silicon problem.

B. Optimization Knobs

1) Task-to-Core Allocation: Due to the characteristics of different types of cores, the core type in which a thread is executed affects its resulting execution time and power/energy consumption. Nonetheless, the physical location in which a thread is executed is also a nontrivial issue, as this impacts the resulting temperature distribution throughout the chip (due to the heat transfer among cores, which can potentially create or avoid hot spots), as well as the execution time (due to communication latencies among cores, simultaneous utilization of shared resources, potential NoC link congestions, etc.). Thus, task-to-core allocation refers to the decision making process that determines to which specific core a thread is mapped to, both in terms of type and physical location of the core.

2) Dynamic Power Management: Dynamic power management (DPM) refers to the dynamic selection of the power states of individual cores. Namely, cores can be individually set to execution/active mode, or they can be individually set to some low-power mode (e.g., idle/clock-gated, sleep, power-gated, etc.). The specific low-power modes that are available depend on the chip, and every different low-power mode has an associated power consumption, as well as different latencies for transitioning across power modes.

For DPM, an undesirable situation is when a core is put to sleep only to be awakened immediately (e.g., when new service requests arrive earlier than expected), such that the system pays for the extra energy and latency of waking up the core to process the new requests. Contrarily, if a core remains idle and no service requests arrive in that period, then the core wastes energy by not entering a low-power mode.

There are mainly three classes of DPM policies proposed in [35]. *Fixed time-out* policies shut down cores after they have being idle longer than the fixed time-out. *Adaptive time-out* policies are more efficient than their fixed counterpart as they adapt the time-out value according to past history. In contrast, *predictive* techniques do not wait for a time-out to expire, and rather shut down cores as soon as they become idle if the idle time is predicted to be long enough to amortize the cost of shutting down and waking back up.

3) Dynamic Voltage and Frequency Scaling: As its name states, DVFS refers to the ability of dynamically scaling the voltage and/or frequency of cores. The maximum frequency at which a core can be stably executed is limited by its supply voltage, where higher frequencies can be stably achieved at higher voltages. As shown in [50], the relationship between the supply voltage of a core and the maximum frequency for stable execution can be modeled according to

$$f_{\text{stable}} = k \cdot \frac{(V_{\text{dd}} - V_{\text{th}})^2}{V_{\text{dd}}} \tag{1}$$

where f_{stable} is the maximum stable frequency for the core, V_{dd} is the supply voltage, V_{th} is the threshold voltage for the given technology, and k is an architecture-dependent fitting factor. For a given V_{dd}, running at frequencies lower than f_{stable} is power/energy inefficient, and therefore generally avoided.

The granularity of voltage scaling and frequency scaling varies across different chips [51]. For example, we could have the following.

1) *Global Voltage and Frequency Scaling:* All cores in the chip share a common voltage and frequency.
2) *Global Voltage Scaling:* All cores in the chip share a common voltage, but individual cores can change their frequencies independently.
3) *Voltage and Frequency Islands/Clusters:* Different groups of cores share a common voltage and frequency.
4) *Voltage Islands/Clusters:* Different groups of cores share a common voltage, but individual cores can change their frequencies independently.
5) *Per-Core Voltage and Frequency Scaling:* The voltage and frequency of all cores is selected individually.

Because of the quadratic relationship between power consumption and voltage, having a system with a single global supply voltage for all cores can be power/energy/thermal inefficient. The cause of this inefficiency is that the voltage of all the cores is then determined by the core that requires the highest frequency, while other cores that might require lower frequencies are also forced to run using a high voltage, hence consuming more power/energy and producing more heat than needed. Contrarily, to have an individual voltage for each core is very power/energy/thermal efficient, as each core could be supplied with the lowest stable voltage for its required execution frequency. Nevertheless, it has been suggested by VLSI circuit simulations [52] that it may be costly for implementation as it can suffer from complicated design problems. Thus, for multicore and many-core systems, cluster-based architectures with multiple voltage islands are a promising compromise, as different clusters/islands can run at different voltages at any point in time. For example, Intel's single chip cloud computer [53] clusters cores into groups of eight cores that share a common voltage, while frequency can be selected for every tile of two cores, resulting in up to four different frequencies inside every cluster of eight cores. Contrarily, in the Exynos 5 Octa (5422) processor [54], all the cores inside every cluster share both their voltage and frequency, and hence only frequency settings are exposed to the operating system, while the voltage is automatically selected according to the selected frequency.

C. Workloads

As different works employ different benchmarks depending on the application or the need, we summarize the most widely used applications for evaluating power/performance/temperature management here. Depending on the necessity the workloads are chosen, i.e., to evaluate data center level power management big-data benchmarks are chosen, for evaluating commercial microprocessors general purpose benchmarks are a better choice. Some of the widely employed benchmarks are listed below: general purpose benchmarks (memory intensive, I/O intensive, and CPU intensive) are: SPLASH-2 [55], PARSEC [56], SPEC CPU-2006 [57], SPEC CPU-2017 [58] big data benchmarks, such as HiBench [59], and BigdataBench [60].

D. Machine Learning

1) Supervised Learning: Supervised learning is the ML operation of extrapolating a function from labeled training data consisting on a set of training examples [61]. In supervised learning, every example is a pair composed by an input object (typically a vector of inputs) and a desired output value (also called the supervisory signal). A supervised learning algorithm will analyze the training data and produce an extrapolated function, which can be then used for mapping new examples. For a given set of N training samples, i.e., the labeled data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where x_i represents the feature vector for ith input sample correspondingly labeled as y_i (i.e., ith class), the supervised learner derives a model F, such that $F(X) = Y$ or $F : X \rightarrow Y$ based on the input X with corresponding labels as Y. In an optimal scenario, the algorithm would be able to correctly determine the class labels for unseen instances, which would require that the learning algorithm generalizes from the training data to unseen situations in a *reasonable* manner, i.e., for a new input x^*, the supervised learner uses the model F to determine the corresponding output label y^* ($F(x^*) = y^*$).

Some of the commonly used supervised learning techniques for power, energy, and thermal management are, for example, online naïve Bayesian classifier (used in [6], [8], and [28]), neural networks (used in [9] and [26]), least squares regression (used in [39], [40], and [43]), genetic algorithms (used in [41]),

locally weighted projection regression (LWPR) (used in [46]), and adaptive learning trees (used in [35]).

2) Unsupervised Learning: The main idea of unsupervised learning is to extrapolate a function that describes a hidden structure from *unlabeled* data (i.e., a categorization which is not included in the list of observations because it cannot be measured). Given that the examples which are given to the learning algorithm are unlabeled, as opposed to supervised learning, the accuracy of the learning process cannot be estimated in any way. For a given unlabeled data $\{x_1, x_2, \ldots, x_N\}$ comprising of N samples, the unsupervised learner attempts to exploit the relationship among the inputs X so as to derive association rules or define the boundaries/manifolds that differentiates one set of data from other. Unsupervised learning is generally used in clustering and association applications. Despite the benefit of requiring no labels for data clustering with unsupervised learning, the associated complexity of unsupervised learning and the accuracy hampers the usage in many real-world applications.

The most commonly used unsupervised learning technique for power, energy, and thermal management is the k-means clustering method (used in [29], [41], and [43]).

3) Reinforcement Learning: Reinforcement learning is a type of ML technique that mimics one of the most common learning styles in natural life, which is to learn to achieve a goal by trial-and-error interaction with a dynamic/uncertain environment [23], [32]. The interactions between the learning agent and the environment are generally modeled using a finite state space $\mathbf{S}$ (corresponding to environment inputs), a set of available actions $\mathbf{A}$ (corresponding to control/optimization knobs used by the agent), and a reward function $\mathbf{R} : \mathbf{S} \times \mathbf{A} \rightarrow \mathbf{R}$ (used to decide which action to take for a given state). The ultimate goal of reinforcement learning is to figure out a policy $\pi(s) = a$, which chooses action $a \in \mathbf{A}$ in each state $s \in \mathbf{S}$ (i.e., a mapping between the states and the actions), to optimize a reward function (i.e., to maximize the cumulative rewards over a potentially infinite time span).

Decision epochs are a sequence of points in time $\{t_0, t_1, t_2, \ldots, t_k, \ldots\}$ at which an action is chosen and a state transition may appear. At time t_k, when the system just transitioned to state $s_k \in \mathbf{S}$, the agent selects an action $a_k \in \mathbf{A}$. This action will lead to an instant reward rate $r_{(s_k, a_k)}(t)$ in regards to state-action pair (s_k, a_k). In the next decision epoch (i.e., at time t_{k+1}), the system switches to state s_{k+1}.

An important issue in reinforcement learning is exploration versus exploitation. A reinforcement learning agent must exploit the best action known so far in order to gain rewards, while exploring all possible actions such that it can find a potentially better choice. The risk is thus always choosing the action with the temporary highest reward, as this can lead to reaching a local maximum and getting stuck in a suboptimal solution.

The most commonly used reinforcement learning techniques for power, energy, and thermal management are Q-learning (used in [7], [9]–[16], [22], [26], [27], [29], [32], [38], [62], and [63]) and TD(λ)-learning (used in [6], [25], [28], and [48]).

4) Q-Learning: Q-learning is one of the most popular algorithms used to perform reinforcement learning [23], [32], [64]. In Q-learning, a Q-value is associated to every state-action pair (s, a), denoted as $Q(s, a)$. The value of $Q(s, a)$ approximates the expected long-term cumulative reward of taking action a starting from state s [7]. In this way, the agent decides which action should be taken in current state in order to achieve the maximum long-term rewards based on this value function $Q(s, a)$. Namely, at decision epoch t_k when the system has just transitioned to state $s_k \in \mathbf{S}$, the action a_k with the highest Q-value will be chosen. Furthermore, given that it is a model-free learning algorithm, it is not necessary for the Q-learning agent to have any prior system information, such as the transition probability from one state to another. Therefore, it is a highly adaptive and flexible algorithm.

The fundamental aspect of the Q-learning algorithm is a value iteration update of the Q-value function. Particularly, the Q-value for each state-action pair is initially chosen by the designer. However, these values are updated every time an action is issued and a reward is received. That is, at decision epoch t_{k+1}, the Q-value $Q(s_k, a_k)$ is updated according to the received reward as shown in the following expression:

$$Q(s_k, a_k) \leftarrow \underbrace{Q(s_k, a_k)}_{\text{old value}} + \underbrace{\beta_k(s_k, a_k)}_{\text{learning rate}} \cdot \left[\overbrace{\underbrace{r_{k+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_{a \in \mathbf{A}} Q(s_{k+1}, a)}_{\text{max future value}}}^{\text{expected discounted reward}} - \overbrace{Q(s_k, a_k)}^{\text{old value}} \right] \tag{2}$$

where r_{k+1} is the reward measured at time t_{k+1} for having taken action a_k at time t_k, value $\gamma \in (0, 1)$ is the discount factor, and $\beta_k(s_k, a_k) \in (0, 1)$ is the learning rate at time t_k for state-action pair (s_k, a_k) (which may or may not be equal for all pairs, and which may be constant or variable in time). The next time state s is visited, the action with the maximum Q-value will be chosen, i.e., $\pi(s) = \max_{a \in \mathbf{A}} Q(s, a)$, and given that the Q-value was updated, it might be a different action than the one taken the last time state s was visited.

5) TD(λ)-Learning: In some of the realtime resource and power management problems, the system may not have predefined policy or knowledge regarding the state transitions. In such cases, the system has to learn the policy as well as make the decision in parallel. The TD(λ)-learning methods can be applied to learn the policy and perform the decision making.

For every state s_k visited at epoch t_k, the TD(λ) algorithm chooses an action either with a maximum Q-value, i.e., $\max_{a \in \mathbf{A}} Q(s_k, a)$ for different possible actions a, or by using the semi-greedy policies given in [65]. The estimated Q-value is updated in the next epoch based on the action chosen a_k and the next state s_{k+1}. The Q-value update is similar to that of traditional Q-learning algorithm, but with different estimated Q-value and error terms, particularly

$$\forall (s, a) \in \mathbf{S} \times \mathbf{A} : Q(s, a) \leftarrow Q(s, a) + \beta \cdot \varepsilon_k(s, a) \times \left[\frac{1 - e^{-\gamma \tau_k}}{\gamma} r(s_k, a_k) + e^{-\gamma \tau_k} \max_{a' \in \mathbf{A}} Q(s_{k+1}, a') - Q(s_k, a_k) \right]$$

where the amount of time that system remains in state s_k is given by $\tau_k = t_{k+1} - t_k$, $\beta \in (0, 1)$ is the learning rate, $([1 - e^{-\gamma \tau_k}]/\gamma) r(s_k, a_k)$ is the sample discounted reward received in τ_k time units, and $Q(s_{k+1}, a')$ is the estimated value of the state-action pair (s_{k+1}, a') with s_{k+1} being the next state. The term $\varepsilon_k(s, a)$ represents the eligibility for each state-action pair, updated as

$$\varepsilon_k(s, a) = \lambda \cdot e^{-\gamma \tau_{k-1}} \cdot \varepsilon_{k-1}(s, a) + \delta((s, a), (s_k, a_k))$$

where $\delta((s, a), (s_k, a_k))$ is the delta-Kronecker function.

6) Discussion: The complexity, and processing overheads of different ML techniques not just depends on the class but also on the underlying principle of the ML techniques. In general terms of complexity at high-level, the order with increasing complexity is supervised, reinforcement, and unsupervised learning. However, there can be some techniques, such as neural networks in supervised learning which have higher complexity compared to unsupervised learning, such as k-means clustering, and so on. As such, depending on the size of the input features, and the control knobs the learning technique has to be chosen. The supervised ML approaches are widely employed when there exists a labeled data, i.e., input–output mapping is available. This is of great help when the applications or data that will be executed are known in-advance or specified, such as accelerators or application-specific processing units. This has lower complexity compared to unsupervised learning. However, the unsupervised learning can be deployed for resource management when there exists no explicit input–output mapping. In case of reinforcement learning, this has a complexity in between supervised and unsupervised, and highly suitable when the system has to learn from its experience or deployed in unseen environments. However, the convergence issues might arise in reinforcement learning, which has to be addressed as performed in some of the works, such as [23], [26], and [32].

Till now a glimpse of different ML techniques, optimization goals, and constraints are presented and an overview is outlined in Table I. Before presenting different works, we outline the general framework details as follows: most of the works that focus on DVFS/DPM consider the workload characteristics, and the resource consumption (power, energy, temperature, or idle time of the processor) as the main inputs and the temperature, throughput or other relevant metrics as the constraints and goals. In most of the works, the input data is captured via application or OS layers and the DVFS/DPM control is performed at OS (such as using the task-to-core allocation, scheduling) or at hardware layers (VF changes or low power mode). Furthermore, the employed ML techniques are widely deployed in application or OS layer and the control (action) is either performed at OS or hardware layers.

III. Techniques for Single-Core Systems

A. Energy/Power Minimization Under Performance Constraints

Chung *et al.* [35] proposed a DPM technique for an arbitrary number of sleep states that shuts down idle components based on idle period clustering and adaptive learning trees. Specifically, authors propose to use a learning tree to accurately predict the duration of future idle periods by observing idle periods in the recent past. The proposed approach has some analogy with advanced branch prediction schemes that are widely used in computer architectures to reduce the penalty of mispredicted branches. Moreover, based on the expected duration of the next idle period and while accounting for the transition time between sleep states, authors derive a function that selects the optimal sleep state in which to set the core in order to reduce the energy consumption. For example, if the future idle period is very short, then the core remains idle, as going to a sleep state would not be energy/time efficient. Contrarily, if the future sleep state is very long, then the core can enter the deepest sleep state.

Liu *et al.* [23] and Tan *et al.* [32] proposed a DPM technique for an arbitrary number of sleep states that minimizes the average power consumption under a given performance constraint which can change during runtime. The technique is based on enhancing the traditional Q-learning algorithm. Specifically, authors propose a modified cost function by defining a Lagrangian cost when taking action a from state s, that accounts for the power consumption and latency caused by the action. Furthermore, in order to improve the search speed of the Q-learning algorithm, the technique only searches policies whose actions are not decreasing with respect to the number of waiting requests. Finally, authors propose a linear adaption of the Lagrangian multiplier to search for the policy that minimizes the power consumption while delivering the exact required performance. Namely, increasing the Lagrangian multiplier will monotonically increase the performance and power consumption, but the exact relationship between the Lagrangian and the performance cannot be quantified in advance. Thus, for a given performance constraint, authors present an iterative algorithm that adapts the Lagrangian multiplier to the most suitable value.

Wang *et al.* [6] presented an online hierarchical DPM framework with application-level scheduling for an embedded system composed of a core and a connected I/O device. The main goal is to minimize the average power consumption and find a good power-latency tradeoff for the connected I/O device, where the tradeoff of each type of application is controlled by a user-defined parameter. In order to address the complexity converns, the proposed technique consists of two layers: 1) a reinforcement learning-based component-level local power manager (LPM) and 2) a system-level global power manager (GPM). The LPM employs a TD(λ)-learning algorithm enhanced from the technique presented in [28] (discussed in Section III-C) by improving the state-action spaces and making improvements for handling multiple types of user applications. Specifically, a more general power management framework is used, in which each type of application may have its own performance degradation constraint, defined as the constraint on the average latency per request of that specific type of application. Therefore, unlike [28], this DPM framework can minimize the average component power consumption for each type of application, while satisfying the application's performance degradation constraint. The cost rate in the TD(λ)-learning algorithm is a linearly weighted combination of power consumption and the number of buffered

requests in the service queue. Also as done Wang *et al.* [28] incorporated workload prediction based on an online naïve Bayesian classifier in order to provide partial information about the service request state for the LPM. In regards to the second layer, the GPM acts as a central controller that tries to meet a performance constraint for the component while minimizing the component power consumption. Particularly, the GPM interacts with the CPU scheduler to perform application-level scheduling, thus enabling the LPM to achieve more component power optimizations. The fairness issue related to distributing execution times among different applications is also handled by the GPM.

The work in [24] presents a technique with the same architecture as in [6]. The LPM policy is prespecified and fixed, and it is provided with the timeout policies to perform state transitions. When a task or workload is idle for more than the timeout period, it is moved into idle or sleep state. Based on the present state, the reward and actions are estimated. The GPM learns based on the temporal differences on semi-Markov decision process (SMDP). The GPM, although at a higher level, cannot overwrite the decisions of the LPM. To perform overwriting, a service flow control (SFC) that commands the scheduler is embedded into the architecture. The GPM monitors the service requests in the queue, employs TD(λ)-learning, and guides the SFC. In order to perform the policy learning, authors employ a cost function based on the considered action in TD(λ)-learning, the power consumption, and the delay in servicing the requests.

Shen *et al.* [26] extended the work in [27] (discussed in Section III-C) by adding a Q-learning DPM technique for peripheral devices (that processes the I/O requests generated by software applications), as the algorithm from [27] only focuses on the core and not the peripherals. Namely, two separate Q-learning algorithm are proposed, one used to select the DPM state of the peripheral devices (where its workload is captured by the distribution of idle intervals and the request generation rate), and another (the one from [27]) used to select the DVFS level of the core. Similar to [27] in *constrained mode*, the goal of this paper is to minimize the energy consumption under performance and temperature constraints. For the Q-learning algorithm used to manage the peripheral devices, in order to find the best tradeoff between power and performance, authors define a Lagrangian cost for each state-action pair (s, a) that combines power consumption and performance penalty (i.e., the number of waiting request of current state). Therefore, the Q-value of state-action pair (s, a) reflects the expected average power and request delay caused by the action a taken in state s. Moreover, in order to speed up convergence, authors propose to update more than one Q-value at every decision epoch. These state-action pairs are denoted as virtual state-action pairs, as we assume that the system virtually visits them and updates their Q-values accordingly, although these state-action pairs are not currently being visited. Furthermore, compared to the traditional Q-learning, authors further improve the convergence speed of the proposed Q-learning algorithm by adopting a variable learning rate for every state-action pair (s, a). Hence, the learning rate for state-action pair (s, a) is inversely proportional to the number of times that the state-action pair (s, a) has been visited. Finally, a 2-level control unit tunes the value of the Lagrangian in order to keep the system aligning to the given constraint. In the first level, a neural network-based coarse grained controller is used to set the upper and lower bounds of the Lagrangian based on the long term average workload. In the second level, there is a feedback controller that fine tunes the value of the Lagrangian based on the instantaneous workload variations.

For energy or power optimization under the constraints of the performance in single-core systems, the existing techniques ensure that the minimal performance criteria is met either by performing task scheduling, or DPM (setting the system to low-power or sleep-states) based on predicted idle time or the idle time window. The major advantage is that the design of power management unit is simpler. This kind of methodology suffers a drawback when the idle times are too small or the employed idle time predictions are inaccurate. Additionally, the task scheduling though is an effective methodology adds overhead at the middle-ware layers resulting in additional delays.

B. Performance Maximization Under Temperature or Energy Constraints

The work in [29] presents a DTM technique based on Q-learning in order to minimize the execution time of multimedia applications under a thermal constraint. Given that most multimedia applications are naturally arranged into frames and the computation load of processing each frame has high temporal correlation, the environment observation and policy adaptation is performed at a single frame granularity. The technique learns the temperature change and workload switching patterns by observing the thermal sensor and event counters of the processor, finding a management policy that provides good performance-thermal tradeoff during runtime. Particularly, we consider the DTM controller as a learning agent and model the rest of the system (e.g., the temperature, the hardware, and application status) as the environment. Through Q-learning, at every decision epoch the agent observes the current state of the environment and chooses the voltage/frequency level for the next frame according to the Q-values in the Q-table. After switching to a new frequency, the agent observes the environment and estimates the reward caused by this action, learning from this experience and attempting to improve its future action selection in order to maximize the reward. Since the temperature of the core is a continuous variable within a working range, authors discretize it to get a finite set of states. The region near the thermal constraint is divided in a fine granularity and the other region is divided in a coarse granularity, such that the agent can take better control at a finer resolution when the temperature is approaching the threshold. For the performance counters, in order to classify the space of the event counter readings, authors use the k-means clustering method starting from a small number and gradually increasing it until the classification error is less than 5%.

The work in [7] presents a model-free reinforcement learning technique for an adaptive DPM framework in systems with a hybrid power supply comprised of Li-ion batteries and supercapacitors. Here, the focus is to minimize the weighted average

of system delay and energy consumption, while the voltage of the supercapacitor remains within a given range. Given that the charges of both the battery and supercapacitor are sampled at regular intervals regardless of system events, discrete-time Q-learning is used for the power supply manager. Contrarily, since the device power manager operates in an event-driven manner with events coming at arbitrary times, continuous-time Q-learning is used for the device power manager. To deal with the fluctuating part of the load current demand, authors first focus on integrating supercapacitors with a battery pack such that they can reduce the battery energy loss caused by rate capacity effect. Second, authors formulate and solve a joint optimization problem for the hybrid power supply and load device. The supply power manager is provided with the load component state (i.e., the status of the battery bank, the status of the supercapacitor bank, and the load power demand), and it controls the power supply network by adjusting the output setups of the converters in an attempt to minimize the energy loss during a time interval. The energy loss is computed by subtracting the load energy consumption and the supercapacitor energy increase from the total energy drawn from the battery. The device power manager collects information from both the load component and the power supply, and it attempts to minimize the weighted average of system delay and energy consumption, where the energy-delay tradeoff is controlled by two user-defined parameters. The device power manager makes two types of decisions. First, every time the core transitions from active to idle state, it decides whether to go to sleep immediately or set a timeout. If a timeout is set and no requests arrive during this period, the core will subsequently be put to sleep. Second, while the core is sleeping, the device power manager decides whether or not to wake it up based on the number of waiting requests in the queue. The two power managers communicate with each other to exchange information: the device power manager provides the supply power manager with the current power consumption of the core, while the supply power manager provides the device power manager with the current status of the supercapacitor.

As the performance of system depends on how fast the application is executed, the performance maximization approaches enforce stringent constraints on the runtime, i.e., minimize the execution time or avoid missing the deadlines for the applications. In case of the single-core systems, as there exists only one core, performing DVFS or DTM is more of a centralized power or thermal management to ensure that no power or thermal budgets are crossed (for core power management).

C. Simultaneous Performance, Power/Energy, and Temperature Management

Dhiman and Rosing [33] proposed an online learning algorithm to perform DPM and DVFS. Based on a user-defined constant, their algorithm can minimize energy consumption, maximize the performance, or achieve a compromise between both. Both DPM and DVFS problems, a set of experts is maintained and the learning algorithm is responsible of selecting the expert that has the best chance to perform well based on the current workload characterization. For DPM, the characterization focuses on the duration of idle periods, while for DVFS, it focuses on the CPU/memory intensiveness of the executing task. Particularly, there is a weight vector associated to every task, where the value of every weight factor reflects the performance of an expert. At any point in time and according to the current workload, the best-performing expert has the highest probability factor (obtained by normalizing the weight vector) among all the experts. Hence, the controller simply selects the expert with the highest probability factor as the operational expert for the next control period. After the control period ends, the controller evaluates all the experts and updates the weight factors depending on how suitable an expert would have been for the previous control period.

The work in [28] presents an online adaptive DPM technique based on model-free reinforcement learning, where the tradeoff between power consumption and latency can be controlled by a user-defined parameter. The proposed approach can perform learning and power management in a continuous-time and event-driven manner. Specifically, the proposed DPM technique is based on an enhanced TD(λ)-learning algorithm for SMDP in order to accelerate convergence and alleviate the reliance on the Markovian property. When the core is in the idle state, the available actions are a finite set of timeout values, and the power manager has to learn the best timeout value by using the TD(λ)-learning algorithm. When the core is in the sleep state, authors propose to use the N-policy, such that the core is turned back to the active state in case the number of waiting requests in the service queue is more than a specified value denoted as N. Otherwise, the core remains in the sleep state new requests arrive. The action set can be denoted as a finite set of integers corresponding to the different specified N values. The power manager thus learns the best threshold N value of waiting request by using a TD(λ)-learning N-policy. In order to provide estimates of the request interarrival times (i.e., future idle periods) to the DPM agent, authors also present a workload predictor based on an online naïve Bayes classifier.

The work in [25] compares the technique presented in [28] (without using the workload classifier) with a traditional Q-learning approach, both in terms of performance and convergence speed. The experimental results suggest that TD(λ)-learning can achieve better power savings without sacrificing latency and that it has faster convergence speed compared to Q-learning.

Shen *et al.* [27] presented a Q-learning algorithm for simultaneous management of temperature, performance, and energy using DVFS. The technique has two working modes, *free mode* and *constrained mode*, where the difference between the two modes is in the calculation of the cost function of each state-action pair. In free mode, the controller is designed for exploring the tradeoffs among temperature, performance, and energy based on user-defined constants, and the Q-learning algorithm selects the voltage/frequency level to achieve the required balance. In constrained mode, the controller can set one or two parameters as constraints while optimizing the third one (e.g., minimize energy consumption under performance and temperature constraints), and the Q-learning algorithm will select the voltage/frequency level accordingly by searching

for policies that minimize the objective penalty. In regards to the state space of the Q-learning algorithm, temperature, IPS, and core utilization are discretized, such that the environment state space is a vector with four components: 1) frequency; 2) temperature; 3) IPS; and 4) core utilization. IPS and core utilization are jointly used to estimate power consumption. The temperature penalty is calculated based on temperature changes, not on absolute temperature values. Therefore, in case the temperature constraint is violated, the action leading to reducing the temperature will result in a negative penalty in order to reinforce the corresponding action.

Simultaneous power, temperature, and performance management adds relatively higher complexity in terms of implementation due to larger search and optimization spaces. This not only leads in additional area overhead, but as well processing overheads. As seen from the existing works, many of the works try to make one of the metrics constant and try to optimize the others. This aids in optimizing the search space and reduce the computational delays. Furthermore, the works suggest employing a controller which is an integration multiple objective solvers is a viable solution.

IV. Techniques for Homogeneous Multicores

A. Energy Minimization Under Performance and Temperature Constraints

The work in [8] presents a supervised learning-based power management framework for energy minimization on a multicore chip with per-core DVFS. Particularly, the power manager uses a probability-based learner (i.e., Bayesian classifier) to predict the performance state of the processor for each incoming task by inspecting some readily available input features (such as the occupancy state of a global service queue), and then uses this predicted state to look up the optimal power management action (i.e., select the voltage/frequency settings of the cores) from a precomputed policy table. The motivation for using a Bayesian classifier is to reduce the overhead of the power manager that has to repetitively select the voltage/frequency settings of every core. The proposed technique is composed of three parts: 1) extraction; 2) classification; and 3) policy generation. Essentially, authors aim to use supervised learning in order to enable the automatic discovery of the relations between input features and output measures, and to predict the power consumption of the cores and execution time of every task by using the classification. In more detail, the first step is the input features and output measures extraction phase, where system knowledge is required in order to produce well-prepared training sets. During the process of feature extraction, the power manager gathers input features, such as the type of tasks (e.g., high-priority or low-priority), the state of the service queue, and the arrival rate of tasks (which affect the performance of the cores). Furthermore, the manager observes the power consumption of the cores and the execution time of the tasks as the output measures. Having obtained the training set, the second step is the classification phase, that uses supervised learning to train a classifier. The goal here is to predict the most likely class label of the output features given the input features, where the class of each output measure is as a predefined range. The third step is the discriminative Bayesian classifier, whose key advantage is the ability to deal with missing information during the classification phase (i.e., missing input features that are relevant to the identification of output features). For example, cache miss statistics and branch misprediction rates, which affect the performance of the cores, are considered missing input features during practical implementation of the proposed technique. Finally, the main goal of the power manager is to derive a DVFS policy for selecting the voltage/frequency levels of the cores that minimize the overall energy consumption based on the load conditions and workload characteristics.

The works in [39] and [40] present a distributed thermal management solution based on model predictive control and self-calibration in order to minimize energy under performance and temperature constraints. Particularly, according to the incoming task workload characteristics, every core first takes as input the predicted CPI value of the running task for the following time interval and selects the minimum voltage/frequency level (that minimizes the power consumption of the core) while meeting the performance constraints. Second, when required, every core trims the previously selected voltage/frequency in order to ensure a safe working temperature. Individual cores jointly optimize global system operation by exchanging a limited amount of information (temperature in particular) at runtime with the neighboring cores. Third, the proposed technique addresses model uncertainty by self-calibration through offline supervised learning (e.g., by running the self-calibration routine during start-up phase and each time the model behavior differs from the measured), where every core learns the local thermal model by applying a set of training stimuli and monitoring the thermal response of the neighboring cores. Specifically, every core implements an auto-regressive exogenous model that has a single output (the temperature of the core) and multiple inputs (the power consumption on the core, the ambient temperature, and the temperatures of neighboring cores). The self-calibration routine first forces a pseudo-random binary sequence power input to every core, while probing the temperature of the cores. Then it derives the model's parameters by solving a least square problem that minimizes the error.

Ye and Xu [9] presented a DPM and task allocation framework based on Q-learning in order to achieve a good tradeoff between performance and power consumption, while also maintaining the system under a temperature constraint. The impact of temperature on leakage power consumption is considered by integrating the temperatures of the cores into the framework, thus achieving better energy savings. Particularly, in order to achieve a tradeoff between power and performance, similar to as done in [23], [32], and [26], the reward function used in the Q-learning model includes a Lagrange multiplier that allows the power consumption of state-action pair (s, a) to be added to the result of multiplying the Lagrange multiplier and the response time of pair (s, a). If the value of the Lagrange multiplier is changed, the weights of average power and response time in the reward function are adjusted to satisfy the system demand, where a larger Lagrange multiplier means that the response time is more important. Due to the large design space, which increases exponentially with respect to the number of cores, authors propose to use a back propagation

neural network to approximate the Q-function in order to improve the convergence speed of learning process. As a whole, this neural network describes a nonlinear mapping, where the state-action pairs are the inputs and the output is the Q-value corresponding to a given state-action pair. At every decision epoch, the parameters of the network are updated in a gradient manner with the help of the back-propagation algorithm, such that the errors propagate backwardly from the output nodes to the inner nodes in order to adjust the network's weights. Moreover, to avoid local maximums which can occur when the Q-learning algorithm always takes the action with the highest Q-value for a given state without exploring new actions, authors employ a ε-greedy method for action selection. This greedy method allows the learning agent to reinforce the evaluation of the actions known to be good, while also exploring unknown actions in an attempt at avoiding local maximums. In this way, authors give the action that has the highest Q-value a high selected probability of $(1 - \varepsilon)$, and all the other actions equally share the remaining probability ε. Hence, considering for example $\varepsilon = 10\%$, there is a 10% probability to select another action instead of the action with the highest Q-value, hopefully avoiding local maximums. Furthermore, in case that the current temperature of the core selected to execute a task is higher than the temperature constraint, this action is ignored and the algorithm tries to select another action from the remaining cores by again using the ε-greedy concept. Finally, when the number of cores in a chip is large, authors propose to classify cores with similar states (e.g., with similar power, queue utilization, and temperature) into groups, change the action of the neural network from task-to-core allocation to task-to-group allocation, and then finally choose a core in the selected group hierarchically.

One more reinforcement learning-based approach for performing power management is adopted in [37]. Here, during voltage and frequency scaling, the framework considers workload execution characteristics, processor configurations, and available DVFS techniques. To avoid the overheads of scaling, the controller first determines the hyper periods for different tasks, i.e., least common factor of periods among different tasks. This aids in finding the lower bound to run the set of applications. When an event occurs according to the scheduling policy, the system invokes the DVFS controller to determine the voltage and frequency settings. Based on the power consumption, state, and action pairs, the penalty is determined and the values are updated. In order to determine the lowest frequency, the parameters are provided to a Q-learning algorithm. If the hyper period exists, the controller calculates the energy consumed and the penalty for the utilized DVFS settings in the previous hyper period to choose the DVFS settings for the current hyper period. However, when there exists no hyper period, then the system determines the frequency simply based on the previously employed DVFS settings.

Similarly, a per-core power management with the aid of modular reinforcement learning is proposed in [62]. In this paper, reinforcement learning is employed by considering the inter- and intra-state of the adjacent cores for a better power management. Due to smaller dimensionality, the convergence concern can be reduced. Furthermore, this method achieves a lower computational complexity as the data considered is smaller in dimension. However, employing per-core power management can incur large overhead. In [22], Q-learning-based approach considering the performance variation along with the workload variations across different applications, i.e., inter- and intra-workload variations and performance variations are utilized to perform power management. In contrast to most of the works, where the workload variations are primarily considered, this paper addresses the challenge of nonconsideration of performance variations across different applications. In contrast to power management with on-chip power manager, a cloud-based power management for smart devices that are connected to Internet is proposed in [63]. The learning of data for multiple devices with same learning model are communicated to cloud via lightweight communication. This is highly attractive for IoT devices, but with increasing in heterogeneity and if the data to be transmitted increases, the communication overhead can be high. Furthermore, use of cloud resources might be costlier in some cases.

A rigid linear regression with learning-based modeling and DVFS execution is proposed in [18]. The model of the system is built during runtime based on the sensed values. First, the operating frequencies are varied and different measurements, such as latency and current are read from the sensors. These values are averaged over time and used further to test the hypothesis of the model. Once the model converges or the testing period is completed, the model is updated and stabilized. Here, the rigid linear regression is used to build and update the model. Primarily, output current and latency are the two variables whose models are built to perform DVFS. Once the models are built, the models are updated based on the detected variations in the workload characteristics. Lastly, a gradient-descent search in the optimization space is utilized to predict the desired voltage frequency settings. Based on the determined settings, the workloads are provided with the corresponding voltage and frequency settings. This paper is light weighted and has a linear computational complexity. However, the effectiveness of the methodology depends on the accuracy of the regression model. Furthermore, when the model does not converge the accuracy of the model is affected.

A simple yet efficient technique considering the hardware characteristics of the voltage regulators (VRs) is proposed in [36]. Contrary to other works, this paper considers the peak efficiency and energy saving of the VRs are considered as constraints to perform DVFS. Based on the demanded voltage levels of the cores, they are clustered. Based on the driving capability of the VRs and their peak efficiency characteristics, the VRs are assigned to the cluster. In this paper, the clustering is performed under the constraints of energy saving and peak efficiency of the VRs. The proposed technique is effective in terms of hardware implementation with less computational complexity. However, monitoring and adapting the characteristics of the VRs to reflect the real-time scenario is pivotal.

The energy minimization techniques in multicore systems have higher computational complexity due to larger feature space and more control knobs for energy optimization. The techniques, such as using hierarchical, and agent-based

techniques reduce the complexity are recommended, however, the number of agents required and high-level optimization might be complex. Furthermore, when using computationally expensive techniques like Q-learning one or other kind of optimizations are recommended to ensure that the convergence is achieved. Use of low complex and lightweight ML techniques are encouraged.

B. Performance Maximization Under Power or Temperature Constraints

The work in [47] presents an online thermal management technique to maximize performance and reduce thermal cycles and thermal gradients under a temperature constraint, based on *expert* and *specialist* policies selected by using a reinforcement learning algorithm. An expert policy applies a certain power management technique (e.g., DPM, DVFS, thread migration, etc.) in order to optimize the desired goal. A specialist is a higher level policy that selects a particular expert to run during the next control interval. For example, a specialist policy could simply decide to use one expert policy the entire time (e.g., a standard reactive DTM technique based on DVFS) or it may choose to use different experts at given time points according to certain conditions (e.g., we could have a core utilization-based specialist that uses thread migration when there is high utilization on a core, DVFS when there is medium utilization, and DPM when there is low utilization). In the proposed technique, only one of the expert policy is active at any given time. The decision to switch to another expert (or to continue using the current expert) is performed at every control interval by the specialist currently responsible for making decisions. After every time interval, the reinforcement learning algorithm computes a multivariate loss function that provides feedback on the temperature profile and the performance cost. Particularly, for the selection of a specialist policy, weight vectors are maintained for the specialists, and these vectors get updated at every time interval based on the observed loss, where the loss is a non-negative value representing how well a specialist performs in terms of the given objective. Moreover, in order to reduce complexity, at every iteration, only the weights of the specialists that are associated with the active ground expert are updated. For example, if the active expert policy for the last time interval was DPM, the learning algorithm only updates the weights of the specialists that would have selected DPM during that interval. Finally, at every decision epoch, the specialist with the highest weight factor is simply selected by the learning algorithm.

The work in [43] presents a thermal prediction methodology and a simple runtime DTM technique for performance maximization under a thermal constraint. The thermal prediction method is characterized entirely at design-time and the runtime overheads amount to table lookups and a few arithmetic operations. The method takes raw performance counter data (periodically measured for each core during workload operation) and thermal measurements for each core, and translates this data into a temperature projection for a future time interval by using the concept of global workload phases. A phase is a stage of execution in which a workload exhibits near identical power, temperature, and performance characteristics (i.e., a workload phase is not identified by individual applications). Furthermore, the methodology uses principal component analysis (PCA) to reduce the computational effort by transforming a number of correlated variables into a smaller number of uncorrelated variables (or principal components) while retaining most of the original information. Within the d-dimensional space defined by the principal component axes, the method uses k-means clustering to define the global phase locations that most closely approximate the n observations in the representative workload data (i.e., it finds a partition of the n observations into $k < n$ sets). The predictive thermal model is then learned using least squares regression on the phase designations calculated for representative workloads gathered at every operating frequency. The DTM technique uses the thermal prediction to estimate the future temperatures and set the voltage/frequency levels of the cores to the maximum values for which the maximum predicted temperature remains below the thermal constraint.

The work in [42] presents a hierarchical power budgeting trading technique that uses task allocation/migration and DVFS. The goal is to distribute the temperatures throughout the chip as evenly as possible in order to reduce the chip's peak temperature (such that it can be maintained below a thermal constraint), while maximizing the performance of best effort tasks and meeting the timing constraints of real-time tasks. The technique incorporates a back propagation economic learning algorithm to improve trading decisions and perform simple thermal management based on a classification of the system's thermal state. The technique is implemented through core agents (consumers with a fixed income used to buy their power budget) and market agents (one for every group of cores, receiving a power budget from the global power source and trading it among its core agents), such that it can be considered as a hybrid that aims to tradeoff the effectiveness of a central approach (with global knowledge of the chip) against the scalability of a fully distributed approach. The power budget assigned to a core is scaled according to its temperature. Cores with low temperatures are able to receive their entire assigned power budgets, while cores with high temperatures only receive a fraction of their assigned power budget, decreased linearly with the measured temperature. The scaling function that decides what percentage of the assigned power budget can be used by each core depending on its temperature is derived by using reinforcement learning. Specifically, decisions resulting in transitions to good states (low temperatures) are reinforced, while decisions resulting in transitions to bad states (high temperatures) are penalized.

Otoom *et al.* [15] presented a hierarchical power management technique that attempts to deliver maximum the energy efficiency (particularly, to maximize the total executed instructions per joule) under a fixed per-chip power budget (e.g., TDP). Particularly, this paper proposes to apply Q-learning at the core level in order to learn the voltage/frequency level that maximizes the performance of the core for the executed application. The power needed by a core to run at the desired voltage/frequency is then requested from a chip-level heuristic controller that dynamically distributes the fixed

power budget of the chip among all cores, while attempting to maximize the overall chip performance. The chip-level controller first orders all power requests decreasingly according to the expected reward of running every core at the voltage/frequency level learned through Q-learning. Then, without exceeding the available per-chip power budget, the chop-level controller verifies if it can grant the requested power to the core with the highest expected reward while providing the baseline power (i.e., the power consumption for running a core at the minimum voltage/frequency level) to all other cores. If this is possible, the requested power is granted, and otherwise the baseline power is granted. The process is then greedily repeated until all cores have an assigned power budget.

Different cores in multicore system can achieve different performances depending on the applications they are executing. In order to achieve high-performance under temperature and energy/power budget constraints task allocation or migration and use of distributed management techniques can be beneficial. This results in an efficient resource utilization.

C. Temperature Minimization

In [46], in order to maintain the chip under a temperature constraint, authors present an application-oriented learning-based DTM technique that pro-actively applies DVFS using a future temperature predictor trained offline through supervised learning. Particularly, by using LWPR, authors learn a model to predict the future temperature of an application based on repetitive executions of the application. LWPR is an incremental learning algorithm for nonlinear function approximation in high dimensional spaces, where the key concept is to approximate the underlying function by local linear models and to compute the final prediction value as the weighted mean of all linear models. In regards to the DTM technique, when the average temperature of more than 30% of cores rises above the thermal constraint, global DVFS is applied. Otherwise, per-core DVFS is applied to the cores with a temperature higher than the thermal constraint. The number of voltage/frequency steps to decrease is computed according to the current voltage/frequency, the core's utilization, and a user-defined constant.

Khanna *et al.* [41] presented a phase-aware workload placement scheme for data centers that helps in maintaining a uniform thermal profile (i.e., reduce the thermal variance across the rack) in a cluster of compute nodes, thus optimizing the cooling requirement and reducing the energy consumption of the data center. Particularly, the technique is based on a phase-aware ML approach that results in a node level thermal model used to forecast the server thermal trends/profile based on input variables like current temperature and utilization. Micro-architectural event counters are used to derive the system utilization. The scheme consist of three main step: 1) model reduction; 2) workload phase identification; and 3) the thermal prediction model. During *model reduction*, similar to [43], PCA is used to transform a set of correlated variables into a smaller number of uncorrelated variables, thus reducing the dimensionality of data without the loss of information. The goal of model reduction is to retain the most significant datasets that are sufficient to identify the phases of workload operations that demonstrate time-varying behavior. For the *workload phase identification* step, just like in [43], a phase is a stage of execution in which a workload demonstrates similar power, temperature, and performance characteristics. Therefore, the training block performs unsupervised learning in order to build data structures that partition workloads into homogeneous clusters, such that similar workload intervals with similar characteristics are classified within the same class (i.e., phase). As done, Cochran and Reda [43] and Ge and Qiu [29] used the k-means clustering algorithm that partitions the d-dimensional principal observations into k clusters, such that each observation is grouped in the cluster with the nearest mean. Finally, the *thermal prediction model* is mainly a genetic algorithm block that performs dynamic training in order to construct a linear model that associates principal components and cluster assignments to predict the future thermal variations in real-time. The prediction period is equivalent to one sampling period that can vary based on the workload type and end-user requirements. This thermal model is used at a rack granularity in order to aggregate the predicted temperature and construct a variance index to identify thermal hotspots and emergencies. The complete data center is then modeled as a collection of racks, each rack composed of several stacked server nodes. The predictive model is integrated into simple linux utility for resource monitoring running on a representative cluster. Before distributing jobs to nodes, the forecasting model is used to analyze the job's impact on performance for the perspective nodes using statistical simulation. In this way, thermal prediction allows for the evaluation of a sample job assignment before the job is committed through workload placement or migration, thus resulting in an improved thermal balance.

The work in [38] presents a DTM technique that adapts to thermal variations within (intra) and across (inter) applications, with the goal of improving lifetime reliability (i.e., to maximize the mean time to failure) by reducing average temperature and thermal cycling under performance constraints. In this technique, a runtime system interfaces with the on-board thermal sensors and performance counters in order to compute the thermal stress, aging, and to monitor performance. Q-learning is then used to learn the relationship between the mapping of tasks to cores, the voltage/frequency of a core, and its temperature, where the stress and aging values form the states of the Q-table. To incorporate intra- and inter-application workload variations, moving averages of the stress and the aging are determined at the start of every decision epoch.

The work in [14] presents a task allocation/migration scheme based on Q-learning, which considers the temperature of the cores and NoC routers, aiming at reducing the peak temperature of the chip. When doing task allocation, the scheme uses the current chip temperature information and stochastic predictions about how each possible task assignment will affect peak temperature of the chip in the future. In every decision epoch, the accuracy of the previous prediction is checked and the model used to make the prediction is updated. The set of possible states is conformed by taking all thermal sensor readings from all cores in the chip, i.e., it corresponds to all possible combinations of temperatures on the chip. Similarly,

the possible actions correspond to the core index in which to allocate/migrate a task. Due to the large design space, authors propose a continuous approximated function to map state-action pairs to Q-values. This approximated function is a linear combination of a series of basis functions. The reward function is computed as the difference between the thermal constraint and the current peak temperature on the chip, such that a higher reward means that there is a larger thermal headroom.

Reinforcement learning, though effective in many ways, adds complexity to the system. For resource constrained and very light weighted systems, relatively low complex techniques, such as regression could be of aid. Linear modeling-based temperature prediction to perform temperature-aware DVFS is proposed in [44]. Here, the voltage values and the temperature values are sensed for different applications that run on the system. Based on the observed characteristics, a linear model to determine the temperature based on the DVFS settings and workload is determined. Under the constraints of temperature, the DVFS settings are tuned. This paper can also be used using *soft sensors*, i.e., using performance counters and regression techniques to augment temperature sensors. Using soft sensors might be area effective, but adds computational overhead.

Temperature is often a bottleneck, especially in multicore systems. However, as the temperature variations are not as spontaneous as power or energy, use of simple predictors, and application or task migration which incur significant delays can also be afforded.

V. Techniques for Heterogeneous Multicores

A. Energy Minimization Under Performance Constraints

The work in [48] presents a reinforcement learning-based DPM framework for data centers which attempts to minimize the overall energy consumption of a server pool while maintaining a reasonable average job response time. Particularly, a TD(λ)-learning algorithm is used to periodically choose how many servers from the server pool to turn on or turn off. The overall profit of the server pool is computed as the total revenue of processing all the incoming jobs minus the total energy cost. Moreover, the income made by processing a job is considered to be inversely proportional to the response time of that job (including both waiting time in the service queue and processing time in the server). Therefore, the reward function for taking action a while in state s is computed as the negative of the weighted sum of the average job response time and the average power consumption of the server pool during that time slot, including a Lagrangian multiplier to decide the relative weight of the power consumption to the response time (as already done in [9], [23], [26], and [32]). In the TD(λ)-learning algorithm, Q-value $Q(s, a)$ for state-action pair (s, a) approximates the expected discounted cumulative return of taking action a at state s.

Compared to homogeneous multicore energy minimization, as different cores consume different energy/power, the cost function or the system power estimator requires large number of knobs and have larger complexity. As such use of game-theory kind of methods can be beneficial as well as the weighted ML solutions.

B. Performance Maximization Under Power and Temperature Constraints

Iranfar *et al.* [16] proposed an ML-based power/thermal management technique that uses a heuristic to limit the learning space by assigning a specific set of available actions to each existing state. The objective of this paper is to increase the performance while reducing the thermal stresses (including spatial and temporal temperature gradients and thermal cycles) under power and thermal constraints, by migrating tasks between cores and by selecting the voltage/frequency levels of every core. Particularly, a heuristic algorithm assigns one of seven predefined states to each core. For every state, another heuristic algorithm selects an action set, such that there is a limited number of actions that can be taken by a core in a certain state (the idea is that only certain actions can be potentially beneficial when a core is in a certain state, e.g., if the temperature of a core is above the thermal threshold, there is no point in evaluating an action that would increase its frequency). There are four action types which the heuristic can include in an action set: increasing the frequency of a core, decreasing the frequency of a core, migrating a task from a core, and migrating a task to a core. Q-learning is then used at a per-core granularity in order to select the most appropriate action for the corresponding state of each core which leads to the highest reward value.

In order to perform performance maximization, the number of possibilities can be numerous especially when the system is heterogeneous. This leads to a bigger optimization space. As such, employing some heuristics or performing high-levels tasks, such as thread migration, DVFS per core level can be effective in order to alleviate overheads.

VI. Uncore Power Management

In addition to the cores in a multicore or a single-core system, there exists other components, such as interconnects (bus, NoC, I/O peripherals), memory, and cache which consume power. The two major components that account for uncore power are memory and interconnects. Considering the brevity and the focus of this paper, we provide an overview of power management techniques for them below.

A. Memory Power Management

Memory controllers are widely employed for optimizing or managing the power of external memory components, such as DRAM [66], [67]. By predicting the idle duration for the memory, the power saving is achieved. Hardware-based approaches, such as scheduling or reordering the instructions to reduce the idle time and improve the performance are provided in [68]. Throttling-based methods in which the power for issued commands are estimated and power is scaled accordingly, and also the CPU instruction flow is also administered for enhanced DRAM power savings [69]. A temperature-based power management for memory by throttling the memory traffic is proposed in [70], especially targeting server class systems. The approaches using either heuristics or regression models [71]–[73] to predict the power-down timeout for the memory (amount of time spent idle before transitioning to a low-power state) is limited and mostly static [71].

A comprehensive power management for memory architectures is outlined in [74]. As seen, a general trend for uncore power management is either to predict the workload or memory-access request (or power) and scale down the voltage-frequency or reorder the requests in order to maximize the performance and energy efficiency.

B. Interconnect Power Management

Interconnects are another class of uncore component with significant power consumption. A DVFS controller to keep the network load at its saturation point is proposed for NoC in [75]. Based on the network load, the voltage/frequency levels are altered here. This paper considers the sum of entire workload to perform DVFS, rather than considering at finer granularity. A per-link DVFS in NoC is proposed in [76]. In addition to links, DVFS for routers are also proposed, where the main parameter to perform DVFS are the metrics, such as input queue occupancy for downstream data [77]. A reinforcement learning-based communication power management is proposed for memory-logic communication is proposed in [10]–[13]. As observed, the interconnect power management is performed considering either the network traffic (communication) and the available bandwidth of the network. As there exists large number of links or interconnects in multicore systems lightweight controllers are recommended.

VII. Discussion and Analysis

Based on the above discussed power, performance, and temperature management methodologies for core, and uncore components and other referred works, we present the following analysis. The stochastic or heuristics-based methods are lightweight, low complex and are efficient when high-precision predictions or modeling is not the major concern for performing the optimization. Even some lightweight machine learners can be employed for such scenarios. These kinds of techniques are more suitable for online control or low-latency decision making scenario. However, the ML-based approaches provide two main benefits when employed for resource management.

1) The control for resource management is more accurate/precise and aids to perform resource management even under uncertainties, such as sporadic variations in the power demands, or memory-access and so on.
2) The ML techniques, such as reinforcement learning, or heavy weighted classifiers, such as neural networks are efficient to handle scenarios, where the system is not trained for, despite using these techniques are resource consuming.

In terms of power or performance optimization using ML in multicore systems, the posed challenges are the response time, captured/observed data accuracy to perform decision making (DVFS, task scheduling, and so on), and most importantly the involved complexity, computational power, and required hardware footprint are the main challenges. As such depending on the affordable resources and performance loss the ML solutions can be opted in future systems.

VIII. Conclusion

In this paper, we have presented an overview of several research efforts that propose to use ML techniques for power and thermal management on single-core and multicore processors. Techniques based on ML can potentially adapt to varying system conditions and workloads, learning from past events in order to improve themselves as the environment changes, resulting in improved management decisions. In addition, the uncore power management methodologies are outlined.

References

[1] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The EDA challenges in the dark silicon era: Temperature, reliability, and variability perspectives," in *Proc. 51st ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2014, pp. 1–185.

[2] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. 38th Int. Symp. Comput. Archit. (ISCA)*, 2011, pp. 365–376.

[3] M. B. Taylor, "Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse," in *Proc. 49th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2012, pp. 1131–1136.

[4] P. D. S. Manoj *et al.*, "A scalable network-on-chip microprocessor with 2.5D integrated memory and accelerator," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 6, pp. 1432–1443, Jun. 2017.

[5] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2015, pp. 1–119.

[6] Y. Wang, M. Triki, X. Lin, A. C. Ammari, and M. Pedram, "Hierarchical dynamic power management using model-free reinforcement learning," in *Proc. Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2013, pp. 170–177.

[7] S. Yue, D. Zhu, Y. Wang, and M. Pedram, "Reinforcement learning based dynamic power management with a hybrid power supply," in *Proc. 30th IEEE Int. Conf. Comput. Design (ICCD)*, Sep. 2012, pp. 81–86.

[8] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 9, pp. 1395–1408, Sep. 2010.

[9] R. Ye and Q. Xu, "Learning-based power management for multicore processors via idle period manipulation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 7, pp. 1043–1055, Jul. 2014.

[10] D. Xu, N. Yu, H. Huang, P. D. S. Manoj, and H. Yu, "Q-learning based voltage-swing tuning and compensation for 2.5-D memory-logic integration," *IEEE Des. Test*, vol. 35, no. 2, pp. 91–99, Apr. 2018.

[11] H. Hantao, P. D. S. Manoj, D. Xu, H. Yu, and Z. Hao, "Reinforcement learning based self-adaptive voltage-swing adjustment of 2.5D I/Os for many-core microprocessor and memory communication," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2014, pp. 224–229.

[12] D. Xu, P. D. S. Manoj, H. Huang, N. Yu, and H. Yu, "An energy-efficient 2.5D through-silicon interposer I/O with self-adaptive adjustment of output-voltage swing," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, 2014, pp. 93–98.

[13] P. D. S. Manoj, H. Yu, H. Huang, and D. Xu, "A Q-learning based self-adaptive I/O communication for 2.5D integrated many-core microprocessor and memory," *IEEE Trans. Comput.*, vol. 65, no. 4, pp. 1185–1196, Apr. 2016.

[14] S. Lu, R. Tessier, and W. Burleson, "Reinforcement learning for thermal-aware many-core task allocation," in *Proc. 25th Great Lakes Symp. VLSI (GLSVLSI)*, 2015, pp. 379–384.

[15] M. Otoom, P. Trancoso, H. Almasaeid, and M. Alzubaidi, "Scalable and dynamic global power management for multicore chips," in *Proc. 6th Workshop Parallel Program. Run Time Manag. Techn. Many Core Archit. (PARMA-DITAM)*, 2015, pp. 25–30.

[16] A. Iranfar, S. N. Shahsavani, M. Kamal, and A. Afzali-Kusha, "A heuristic machine learning-based algorithm for power and thermal management of heterogeneous MPSoCs," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2015, pp. 291–296.

[17] H. Sayadi, N. Patel, A. Sasan, and H. Homayoun, "Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, 2017, pp. 129–136.

[18] S. Yang *et al.*, "Adaptive energy minimization of embedded heterogeneous systems using regression-based learning," in *Proc. Int. Workshop Power Timing Model. Optim. Simulat. (PATMOS)*, Sep. 2015, pp. 103–110.

[19] P. D. S. Manoj, H. Yu, and K. Wang, "3D many-core microprocessor power management by space-time multiplexing based demand-supply matching," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3022–3036, Nov. 2015.

[20] P. D. S. Manoj, K. Wang, and H. Yu, "Peak power reduction and workload balancing by space-time multiplexing based demand-supply matching for 3D thousand-core microprocessor," in *Proc. ACM/EDAC/IEEE Design Autom. Conf.*, Austin, TX, USA, 2013, pp. 1–6.

[21] K. Choi, R. Soma, and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance tradeoff based on the ratio of off-chip access to on-chip computation times," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 1, pp. 18–28, Jan. 2005.

[22] R. A. Shafik *et al.*, "Learning transfer-based adaptive energy minimization in embedded systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 6, pp. 877–890, Jun. 2016.

[23] W. Liu, Y. Tan, and Q. Qiu, "Enhanced Q-learning algorithm for dynamic power management with performance constraint," in *Proc. Design Autom. Test Europe (DATE)*, Mar. 2010, pp. 602–605.

[24] M. Triki, Y. Wang, A. C. Ammari, and M. Pedram, "Hierarchical power management of a system with autonomously power-managed components using reinforcement learning," *Integr. VLSI J.*, vol. 48, pp. 10–20, Jan. 2015.

[25] M. Triki, A. C. Ammari, Y. Wang, and M. Pedram, "Reinforcement learning algorithms for dynamic power management," in *Proc. World Symp. Comput. Appl. Res. (WSCAR)*, Jan. 2014, pp. 1–6.

[26] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 2, pp. 1–24, Apr. 2013.

[27] H. Shen, J. Lu, and Q. Qiu, "Learning based DVFS for simultaneous temperature, performance and energy management," in *Proc. 13th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2012, pp. 747–754.

[28] Y. Wang, Q. Xie, A. Ammari, and M. Pedram, "Deriving a near-optimal power management policy using model-free reinforcement learning and Bayesian classification," in *Proc. 48th Design Autom. Conf. (DAC)*, New York, NY, USA, 2011, pp. 41–46.

[29] Y. Ge and Q. Qiu, "Dynamic thermal management for multimedia applications using machine learning," in *Proc. 48th Design Autom. Conf. (DAC)*, New York, NY, USA, 2011, pp. 95–100.

[30] M. E. Salehi *et al.*, "Dynamic voltage and frequency scheduling for embedded processors considering power/performance tradeoffs," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 10, pp. 1931–1935, Oct. 2011.

[31] B. Dietrich, S. Nunna, D. Goswami, S. Chakraborty, and M. Gries, "LMS-based low-complexity game workload prediction for DVFS," in *Proc. IEEE Int. Conf. Comput. Design*, Oct. 2010, pp. 417–424.

[32] Y. Tan, W. Liu, and Q. Qiu, "Adaptive power management using reinforcement learning," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2009, pp. 461–467.

[33] G. Dhiman and T. Rosing, "System-level power management using online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 5, pp. 676–689, May 2009.

[34] E.-Y. Chung, L. Benini, A. Bogliolo, Y.-H. Lu, and G. D. Micheli, "Dynamic power management for nonstationary service requests," *IEEE Trans. Comput.*, vol. 51, no. 11, pp. 1345–1361, Nov. 2002.

[35] E.-Y. Chung, L. Benini, and G. De Micheli, "Dynamic power management using adaptive learning tree," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 1999, pp. 274–279.

[36] W. Lee, Y. Wang, and M. Pedram, "Optimizing a reconfigurable power distribution network in a multicore platform," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1110–1123, Jul. 2015.

[37] F. M. M. U. Islam and M. Lin, "A framework for learning based DVFS technique selection and frequency scaling for multi-core real-time systems," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun.*, Aug. 2015, pp. 721–726.

[38] A. Das *et al.*, "Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems," in *Proc. 51st Design Autom. Conf. (DAC)*, 2014, pp. 1–170.

[39] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, "Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 170–183, Jan. 2013.

[40] A. Bartolini, M. Cacciari, A. Tilli, and L. Benini, "A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores," in *Proc. Design Automation Test Europe (DATE)*, Mar. 2011, pp. 1–6.

[41] R. Khanna, J. John, and T. Rangarajan, "Phase-aware predictive thermal modeling for proactive load-balancing of compute clusters," in *Proc. Int. Conf. Energy Aware Comput. (ICEAC)*, Dec. 2012, pp. 1–6.

[42] T. Ebi, D. Kramer, W. Karl, and J. Henkel, "Economic learning for thermal-aware power budgeting in many-core architectures," in *Proc. 7th Int. Conf. Hardw. Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2011, pp. 189–196.

[43] R. Cochran and S. Reda, "Consistent runtime thermal prediction and control through workload phase detection," in *Proc. 47th Design Autom. Conf. (DAC)*, 2010, pp. 62–67.

[44] J. S. Lee, K. Skadron, and S. W. Chung, "Predictive temperature-aware DVFS," *IEEE Trans. Comput.*, vol. 59, no. 1, pp. 127–133, Jan. 2010.

[45] F. Sironi, M. Triverio, H. Hoffmann, M. Maggio, and M. D. Santambrogio, "Self-aware adaptation in FPGA-based systems," in *Proc. Int. Conf. Field Program. Logic Appl.*, Aug 2010, pp. 187–192.

[46] W.-J. Kim, J.-W. Song, and K.-S. Chung, "On-line learning based dynamic thermal management for multicore systems," in *Proc. Int. SoC Design Conf. (ISOCC)*, vol. 1, Nov. 2008, pp. I-391–I-394.

[47] A. K. Coskun, T. S. Rosing, and K. C. Gross, "Temperature management in multiprocessor SoCs using online learning," in *Proc. 45th Design Autom. Conf. (DAC)*, 2008, pp. 890–893.

[48] X. Lin, Y. Wang, and M. Pedram, "A reinforcement learning-based power management framework for green computing data centers," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Apr. 2016, pp. 135–138.

[49] *Dual-Core Intel Xeon Processor 5100 Series Datasheet, Revision 003*, Intel, Santa Clara, CA, USA, Aug. 2007.

[50] N. Pinckney *et al.*, "Assessing the performance limits of parallelized near-threshold computing," in *Proc. 49th Design Autom. Conf. (DAC)*, 2012, pp. 1147–1152.

[51] S. Pagani, "Power, energy, and thermal management for clustered many-cores," Ph.D. dissertation, Chair Embedded Syst., Dept. Comput. Sci., Karlsruhe Inst. Technol., Karlsruhe, Germany, Nov. 2016.

[52] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 2007, pp. 38–43.

[53] *SCC External Architecture Specification (EAS), Revision 0.98*, Intel Corporat., Santa Clara, CA, USA, Jul. 2010.

[54] *Exynos 5 Octa (5422)*, Samsung Electron., Seoul, South Korea, 2014. [Online]. Available: www.samsung.com/exynos

[55] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," *SIGARCH Comput. Archit. News*, vol. 23, no. 2, pp. 24–36, May 1995.

[56] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," Dept. Comput. Sci., Princeton Univ., Princeton, NJ, USA, Rep. TR-811-08, Jan. 2008.

[57] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[58] *Spec CPU 2017*, SPEC, Richmond, VA, USA, 2017. [Online]. Available: https://www.spec.org/cpu2017/

[59] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The HiBench benchmark suite: Characterization of the MapReduce-based data analysis," in *Proc. IEEE Int. Conf. Data Eng. Workshops*, 2010, pp. 41–51.

[60] Z. Jia *et al.*, "Understanding big data analytics workloads on modern processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 6, pp. 1797–1810, Jun. 2017.

[61] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. Cambridge, MA, USA: MIT Press, 2012.

[62] Z. Wang *et al.*, "Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system," in *Proc. Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2017, pp. 684–689.

[63] G.-Y. Pan, B.-C. C. Lai, S.-Y. Chen, and J.-Y. Jou, "A learning-on-cloud power management policy for smart devices," in *Proc. IEEE ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, 2014, pp. 376–381.

[64] P. D. S. Manoj, "3D I/O designs for energy-efficient memory-logic integration towards thousand-core on-chip," Ph.D. dissertation, School Elect. Electron. Eng., Nanyang Technol. Univ., Singapore, Nov. 2015.

[65] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.
[66] I. Hur and C. Lin, "A comprehensive approach to DRAM power management," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2008, pp. 305–316.
[67] V. Delaluz, M. Kandemir, N. Vijaykrishnan, A. Sivasubramaniam, and M. J. Irwin, "DRAM energy management using software and hardware directed power mode control," in *Proc. Int. Symp. High Perform. Comput. Archit.*, 2001, pp. 159–169.
[68] B. K. Mathew, S. A. McKee, J. B. Carter, and A. Davis, "Design of a parallel vector access unit for SDRAM memory systems," in *Proc. Int. Symp. High Perform. Comput. Archit.*, 2000, pp. 39–48.
[69] W. Felter, K. Rajamani, T. Keller, and C. Rusu, "A performance-conserving approach for reducing peak power consumption in server systems," in *Proc. Int. Conf. Supercomput.*, 2005, pp. 293–302.
[70] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang, "Thermal modeling and management of DRAM memory systems," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 312–322, Jun. 2007.
[71] Y. Lu, D. Wu, B. He, X. Tang, J. Xu, and M. Guo, "Rank-aware dynamic migrations and adaptive demotions for DRAM power management," *IEEE Trans. Comput.*, vol. 65, no. 1, pp. 187–202, Jan. 2016.
[72] H. Huang, K. G. Shin, C. Lefurgy, and T. Keller, "Improving energy efficiency by making DRAM less randomly accessed," in *Proc. Int. Symp. Low Power Electron. Design*, 2005, pp. 393–398.
[73] H. Huang, P. Pillai, and K. G. Shin, "Design and implementation of power-aware virtual memory," in *Proc. USENIX Annu. Tech. Conf.*, 2003, p. 5.
[74] Y. Lu, B. He, X. Tang, and M. Guo, "Synergy of dynamic frequency scaling and demotion on DRAM power management: Models and optimizations," *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2367–2381, Aug. 2015.
[75] G. Liang and A. Jantsch, "Adaptive power management for the on-chip communication network," in *Proc. EUROMICRO Conf. Digit. Syst. Design*, 2006, pp. 649–656.
[76] L. Shang, L. Peh, and N. K. Jha, "Power-efficient interconnection networks: Dynamic voltage scaling with links," *IEEE Comput. Archit. Lett.*, vol. 1, no. 1, p. 6, Jan./Dec. 2002.
[77] A. K. Mishra *et al.*, "A case for dynamic frequency tuning in on-chip networks," in *Proc. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA, 2009, pp. 292–303.

Santiago Pagani (S'13) received the Diploma degree in electronics engineering from the Department of Electronics, National Technological University, Buenos Aires, Argentina, in 2010, and the Ph.D. degree (*summa cum laude*) in computer science from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2016.

He is currently a Firmware Development Team Lead, ARM Ltd., Cambridge, U.K., where he runs an Agile firmware development team researching on key components for the next generation Mali GPU products. From 2003 to 2012, he researched as a Hardware and Software Developer in the industry sector for several companies in Argentina, including two years as Technical Group Leader. From 2012 to 2017, he was a Research Scientist (a Doctoral Researcher and later a Post-Doctoral Researcher) as part of the research staff with KIT. His current research interests include embedded systems, real-time systems, energy-efficient scheduling, power-aware designs, and temperature-aware scheduling.

Dr. Pagani was a recipient of two Best Paper Awards (IEEE RTCSA in 2013 and IEEE/ACM CODES+ISSS in 2014), one Feature Paper of the Month (IEEE TRANSACTIONS ON COMPUTERS in 2017), three HiPEAC Paper Awards, and the 2017 ACM SIGBED Paul Caspi Memorial Dissertation Award in recognition of an outstanding Ph.D. dissertation.

P. D. Sai Manoj (S'13–M'15) received the master's degree in information technology from the International Institute of Information Technology Bangalore, Bengaluru, India, in 2012 and the Ph.D. degree in electrical and electronics engineering from Nanyang Technological University, Singapore, in 2015,

He was a Post-Doctoral Research Scientist with the System-on-Chip Group, Institute of Computer Technology, TU Wien, Vienna, Austria. He is a Post-Doctoral Fellow and a Research Group Leader with Accelerated, Secure, and Energy-Efficient Computing Laboratory, George Mason University, Fairfax, VA, USA. His current research interests include self-aware SoC design, machine learning for image processing and time-series analysis, emerging memory devices and integration techniques, and wireless communications.

Dr. Sai Manoj was a recipient of the Xilinx Open Hardware Contest in 2017 (student category) and the A. Richard Newton Young Research Fellow Award in Design Automation Conference, in 2013. His paper is nominated for Best Paper Award in Design Automation and Test in Europe, in 2018.

Axel Jantsch (M'95) received the Ph.D. degree from TU Wien, Vienna, Austria, in 1992.

He has been a Professor of Systems on Chips with the Institute of Computer Technology, TU Wien, since 2014. From 1997 to 2002, he was an Associate Professor with Karlsruhe Institute of Technology (KTH), Royal Institute of Technology, Stockholm, Sweden, where he was a Full Professor of electronic systems design from 2002 to 2014. There he has built up the research groups for formal systems design and for networks on chips, which are currently successfully headed by his former Ph.D. students. From 2009 to 2014, he was the Head of the Department of Electronic Systems with six professors, seven senior researchers, and over 40 Ph.D. students. He came to Sweden with an Alfred Schrödinger Scholarship in 1992 before he moved for 2 years to Siemens AG in Vienna. His current research interests include systems on chips, networks on chips, and embedded and cyber-physical systems. He has edited five books and authored one book, over 250 peer reviewed contributions in journals, books, and conference proceeding, and he has given over 100 invited presentations at conferences, universities, and companies.

Jörg Henkel (M'95–SM'01–F'15) received the Ph.D. degree (*summa cum laude*) from Braunschweig University, Braunschweig, Germany.

He was a Senior Research Staff Member with NEC Laboratories, Princeton, NJ, USA. He is currently with the Karlsruhe Institute of Technology, Karlsruhe, Germany, where he is directing the Chair for Embedded Systems CES. He holds ten U.S. patents.

Dr. Henkel was a recipient of the 2008 DATE Best Paper Award, the 2009 IEEE/ACM William J. Mc Calla ICCAD Best Paper Award, the CODES+ISSS in 2015, in 2014, and in 2011 Best Paper Awards, and the MaXentric Technologies AHS 2011 Best Paper Award, the DATE 2013 Best IP Award, and the DAC 2014 Designer Track Best Poster Award. He has/is organizing various embedded systems and low power ACM/IEEE conferences/symposia as the general chair and the program chair and was a Guest Editor on these topics in various journals like *IEEE Computer Magazine*. He was the Program Chair of CODES'01, RSP'02, ISLPED'06, SIPS'08, CASES'09, Estimedia'11, VLSI Design'12, ICCAD'12, PATMOS'13, and NOCS'14 and served as the General Chair for CODES'02, ISLPED'09, Estimedia'12, ICCAD'13, and ESWeek'16. He is/has been a steering committee member of major conferences in the embedded systems field like at ICCAD, ESWeek, ISLPED, CODES+ISSS, CASES and is/has been an Editorial Board Member of various journals like the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON MULTISCALE COMPUTING SYSTEMS, *ACM Transactions on Cyber-Physical Systems*, and the *Journal of Low Power Electronics*. He has been the Editor-in-Chief of the *IEEE Design and Test Magazine* since 2016. He is the Chairman of the IEEE Computer Society, Germany Section, and was the Editor-in-Chief of the *ACM Transactions on Embedded Computing Systems* for two consecutive terms. He is an initiator and the coordinator of the German Research Foundation's (DFG) Program on Dependable Embedded Systems (SPP 1500). He is the Site Coordinator (Karlsruhe site) of the Three-University Collaborative Research Center on Invasive Computing (DFG TR89).