Learning-based Run-time Power and Energy Management of Multi/Many-core Systems: Current and Future Trends

Amit Kumar Singh*, Charles Leech, Basireddy Karunakar Reddy, Bashir M. Al-Hashimi, Geoff V. Merrett

School of Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK;
Emails: {a.k.singh, cl19g10, krb1g15, bmah, gvm}@ecs.soton.ac.uk

*corresponding author: Amit Kumar Singh

Address:
University of Southampton
School of Electronics and Computer Science
University Road
Southampton, Hampshire, SO17 1BJ, UK

Office : (+44) 23 8059 3119
Fax : (+44) 23 8059 2901
Email : a.k.singh@soton.ac.uk

Date of Receiving: **to be completed by the Editor**

Date of Acceptance: **to be completed by the Editor**

Learning-based Run-time Power and Energy Management of Multi/Many-core Systems: Current and Future Trends

Amit Kumar Singh, Charles Leech, Basireddy Karunakar Reddy, Bashir M. Al-Hashimi, Geoff V. Merrett

Abstract– Multi/Many-core systems are prevalent in several application domains targeting different scales of computing such as embedded and cloud computing. These systems are able to fulfil the ever-increasing performance requirements by exploiting their parallel processing capabilities. However, effective power/energy management is required during system operations due to several reasons such as to increase the operational time of battery operated systems, reduce the energy cost of datacenters, and improve thermal efficiency and reliability. This article provides an extensive survey of learning-based run-time power/energy management approaches. The survey includes a taxonomy of the learning-based approaches. These approaches perform design-time and/or run-time power/energy management by employing some learning principles such as reinforcement learning. The survey also highlights the trends followed by the learning-based run-time power management approaches, their upcoming trends and open research challenges.

Keywords– Multi/many-core systems, power/energy optimization, run-time, machine learning

1 INTRODUCTION

Multi/many-core systems are becoming prevalent for various domains such as embedded and high performance computing (HPC). These systems provide increased parallelism by performing parallel execution of tasks on various cores [1]. This leads to high performance and thus helps to achieve increased performance requirements of modern computing systems. Chip manufacturers have developed several multi/many-core processors, e.g., Samsung's Exynos 8-core processor [2], Intel's Teraflop 80-core processor [3], AMD's Opteron 16-core processor [4], Tilera's TILE-Gx family 100-core processor [5], and Kalray's MPPA 256-core processor [6]. Depending upon the number of cores in the chip, they are connected by a shared bus or on-chip interconnection network [7–9]. These many-core processors are being exploited in various application domains to realize efficient many-core systems. It is also expected that higher number of cores will be integrated within a chip with technological advancements [10].

For these systems, usually, the applications need to be partitioned (parallelized) into multiple tasks that can be executed concurrently on different cores. Such partitioning is referred to as functional partitioning and can be furnished with the help of state-of-the-art application parallelization tools, e.g., MPSoC Application Programming Studio (MAPS) [11] and MNEMEE project tool-chain [12], and/or manual analysis. This procedure requires detailed application knowledge and involves finding the tasks, adding synchronization and inter-task communication in the tasks, management of the memory hierarchy communication and checking of the parallelized code (tasks) to ensure for correct functionality [13]. In case the multi/many-core system is heterogeneous, i.e. contains different types of cores, a task *binding* process that specifies the core types on them the task can be allocated along with the cost of allocation is required [14]. To compute the allocation cost, the binding process analyses the implementation costs (e.g., performance, power and resource utilization) of each task on different supported core types such as general purpose processor (GPP), digital signal processor (DSP) and coarse grain re-configurable hardware.

Power and energy efficient execution of applications on multi/many-core systems is desired in order to enhance operational time of battery-powered systems or energy cost of HPC datacenters. From several decades, enormous efforts have been put to optimize energy at circuit, architecture and system levels. The optimization of energy during application execution concerns to system level efforts. These efforts have tried to optimize energy by employing three essential ingredients: *mapping* [15–17], *dynamic voltage and frequency scaling (DVFS)* [18–21], and *dynamic power management (DPM)* [22–26]. The mapping defines assignment and ordering of the tasks and their communications onto resources of multi/many-core system in view of some optimization criteria such as compute performance and energy consumption. In DVFS, the voltage/frequency of the cores is adjusted dynamically to save energy consumption while meeting certain level of performance. The DPM process shuts down the cores when they are inactive. Several principles have been followed for shutting the cores down, for example, greedy approach where a core enters into sleep mode as soon as processing on the core is finished and timeout approach that enters the core into sleep mode after certain time of idleness if no request is received within that time. Out of mapping, DVFS and DPM, they have been applied individually and in combinations as well, e.g., mapping in [15, 16] and both mapping and DVFS in [27, 28].

While optimizing power and energy consumption during execution, the timing requirements of applications need to be satisfied. Different types of timing requirements are imposed depending upon the kind of target system, e.g., hard real-time and soft real-time systems [29]. Examples of hard real-time systems are time critical systems such as automotive engine and flight control software. In soft real-time systems such as video processing and HPC datacenters, the deadline violations can be tolerated. There has also been energy optimization efforts of mixed criticality systems, where part of the system has hard real-time requirement and rest has soft real-time requirement. Example of such a system is aeroplane, where cockpit system part has hard real-time requirement due to its safety critical issues and the parts in the passenger area such as lighting and television screen have soft real-time requirement.

Machine learning based power and energy optimization during multi/many-core system operation (i.e. at run-time) has gain significant attention over the last decade, although it exists since 1959. It provides learning ability to systems without being explicitly programmed. With respect to power and energy optimization, the

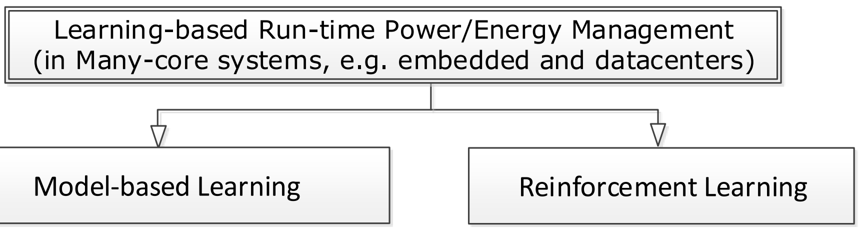

Learning-based Run-time Power/Energy Management
(in Many-core systems, e.g. embedded and datacenters)
Model-based Learning
Reinforcement Learning

Figure 1: A taxonomy of learning based run-time power and energy management approaches.

learned information over time is used to predict appropriate mapping, DVFS or DPM policy to be applied for future execution. Such learning is helpful to make data driven predictions/decisions, where prediction models can be derived from sample inputs. Despite the fact that several articles have been published and significant progress has been made for machine learning based power and energy optimization of multi/many-core systems, there still remains many open questions and research challenges.

1.1 Learning-based Run-time Power and Energy Management Challenges

It is well known that the design space to map tasks on cores increases exponentially with the number of tasks and cores. Therefore, for large size problems, the learning based approach has the challenge to predict the most energy efficient *mapping* during application execution and adapt to the predicted mapping. Further, since modern cores possess DVFS capability, the prediction of appropriate voltage/frequency level is required during application execution in the view of optimizing energy consumption and satisfying timing requirement. However, finding the best energy efficient voltage/frequency level at run-time involves the challenge to perform accurate predictions within a short amount of time. Similar challenges exist to apply DPM. Additionally, while considering mapping and DVFS together, the prediction step needs to identify both the mappings and DVFS levels during execution, which adds further complexity as two aspects need to considered jointly.

1.2 Classification of Learning-based Run-time Power and Energy Management Strategies

Learning-based run-time power and energy management strategies can be classified with a number of taxonomies, e.g., criticality (hard or soft real-time), optimization ingredient (mapping, DVFS, and DPM), employed learning principles, etc. Broadly, the classification can be done based on learning principle and other taxonomies can be included at some hierarchy in the learning principle based classification. For example, model-based (supervised/regression) learning can be used to find appropriate run-time mapping or DVFS level while trying to satisfy soft real-time requirements. Figure 1 shows classification of learning based power/energy management strategies based on the employed principle. The model-based learning approaches perform offline analysis to derive the system behaviour for all the possible inputs and use the appropriate bahaviour at run-time depending upon the input. In reinforcement learning, the system behaviour is learnt at run-time during execution and predictions are made based on the current system status.

Paper Organization: Section 2 and Section 3 cover analysis and elaboration of model-based and reinforcement learning approaches, respectively. A comparative study of strategies falling into different categories has been performed into Section 4. Section 5 provides the upcoming trends that could be followed as the future research and open research challenges for learning based run-time power/energy management approaches. Finally, Section 6 provides some concluding remarks.

2 SUPERVISED MODEL-BASED LEARNING

The aim of model-based learning is to make predictions about future responses based on evidence in the presence of uncertainty. Supervised machine learning is a collective terms for a group of algorithms that perform predictive modeling to establish a relationship between a set of predictor (independent) variables and a target (dependent) variable. In this section, each of the algorithms is introduced and discussed in the context of learning-based run-time power or energy management strategies. Following this, the modeling approaches are grouped by the optimization and control methods that they employ, those of mapping, DVFS and DPM, which are introduced in section 1.

2.1 Model-based learning Approaches

Most of the supervised machine learning algorithms can be engineered to operate as either classification or regression techniques:

- Classification techniques predict discrete responses - for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring.
- Regression techniques predict continuous responses - for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.

2.1.1 Generalized Linear Models

The most common group of predictive algorithms used for learning-based run-time management are Generalized Linear Models (GLM). This term encompasses the majority of empirically and analytically derived models of performance or power commonly derived for prediction in management systems. Ordinary Least Squares (OLS) is an example of a GLM algorithm and is based on a linear combination of the predictor variables in the form of a hypothesis function and is defined as:

$$\hat{y}(\omega, x) = \omega_0 + \sum_{i=1}^{n} \omega_i x_i \tag{1}$$

The coefficients ω_i are established using a series of j training samples $(x_{i,j}, y_j)_{i=1\ldots n, j=1\ldots m}$ with i predictor variables. The OLS process minimizes the mean-squared prediction error $E(\omega)$ of the model, expressed as:

$$E(\omega) = \sum_{j=1}^{m} (f_\omega(x_j) - y_j)^2 \tag{2}$$

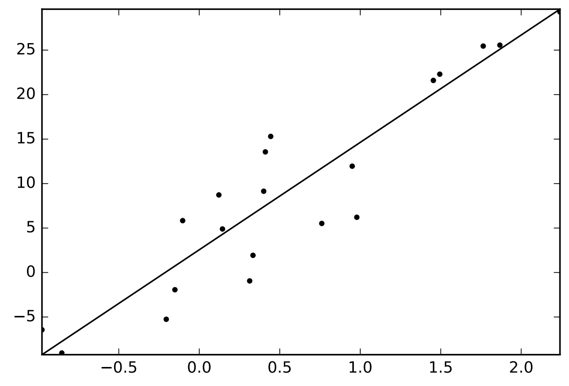
Future target values $\hat{y}$ are predicted given new data x_{n+1}. Figure 2a illustrates how a OLS regression function is calculated from the training data points such that the mean-squared error is minimized.

GLM models can be built using many different algorithms besides OLS. Ridge regression addresses some of the problems of OLS by imposing a penalty on the size of coefficients and therefore the ridge coefficients minimize a penalized residual sum of squares. Further alternatives include Lasso, least angle and Bayesian regression, Orthogonal Matching Pursuit (OMP), the perceptron, passive aggressive algorithms and polynomial regression. Logistic regression is a GLM algorithm used for classification rather than regression. Also known in literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier, in this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

2.1.2 Support Vector Machines

Support vector machines (SVMs) are a set of supervised learning methods that construct a hyperplane or set of hyperplanes in a high-dimensional space, which can be used for classification, regression or other tasks such as outlier detection.

In Support Vector Classification (SVC), the model is a representation of the examples points in space, mapped so that the separate categories are divided by a clear gap that is as wide as possible. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (the


(a) Regression Example

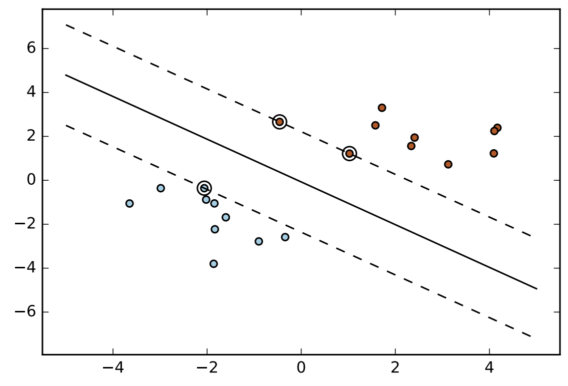

(b) Classification Example

Figure 2: Graphical examples of model-based learning approaches for (a) linear regression and (b) support vector machine classification.

functional margin). The larger the margin the lower the generalization error of the classifier. New data points are mapped into that same space and predicted to belong to a category based on which side of the gap they fall. The model is trained using a dataset of n points in the form $(\vec{x}_1, y_1)$, ..., $(\vec{x}_n, y_n)$ where y_i are either 1 or -1, each indicating the class to which the point $\vec{x}_i$ belongs. Each $\vec{x}_i$ is a p-dimensional vector. The maximum-margin hyperplane that divides the group of points $\vec{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, is defined so that the distance between the hyperplane and the nearest point $\vec{x}_i$ from either group is maximized. Any hyperplane can be written as the set of points $\vec{x}$ satisfying $\vec{w} \cdot \vec{x} - b = 0$ where $\vec{w}$ is the normal vector to the hyperplane. Figure 2b illustrates the output of the SVC process with the hyperplane (solid line) and the functional margins (dashed lines) shown. The training data points are shown with the bounding points highlighted.

Support Vector Regression (SVR) uses the same principles as the SVC, but in this case the intention is to develop a function and hyperplane that minimizes deviation of y_i for all training data [30]. As with classification, the algorithm takes input vectors X and y, but in this case y is expected to have floating point values instead of integer values. The model produced depends only on a subset of the training data, because the cost function for building the model ignores any training data close to the model prediction.

SVMs have several advantages. They are effective in high dimensional spaces, even in cases where the number of dimensions is greater than the number of samples. They use a subset of training points in the decision function (called support vectors), so are memory efficient. They are versatile because different kernel functions can be specified for the decision function in order to classify data more appropriately. However, if the number of features is much greater than the number of samples, the method is likely to give poor performances. Also, SVMs do not directly provide probability estimates, these must be calculated using expensive cross-validation methods.

2.1.3 Naive Bayes

Naive Bayes is a simple technique for performing classification and can build models that assign class labels to instances of features where the class labels are identified from some finite set. Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with the "naive" assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1 through x_n, Bayes' theorem states the following relationship:

$$P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \dots, x_n)} \tag{3}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{4}$$

As a result, the classification task is essentially the assignment of the maximum a posteriori (MAP) class given the vector x_i and the prior of class assignments to y_i [31]. An advantage of naive Bayes is that it only requires a small number of training data to estimate the parameters necessary for classification. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem.

2.1.4 Neural Networks

Neural networks (NNs) build a computational model based on a large collection of connected units called artificial neurons, analogous to axons in a biological brain. Connections between neurons carry an activation signal which can also be weighted to affect the strength of connections and the likelihood of activation. Neural networks must be trained from examples, rather than explicitly programmed, and excel in areas where the solution or feature detection is difficult to express in a traditional computer program. With sufficient training, NN can expose complex and hidden relationships that are difficult to characterize using rule-based programming. Typically, neurons are connected in layers, and signals travel from the input, to the output layer. Back propagation is the use of forward stimulation to modify connection weights, and is done to train the network using training data with known correct outputs. Increasing the number of hidden units in an NN leads to better representational power and the ability to model more complex functions, but increases the amount of training data and time required to arrive at accurate models.

If trained for too long, NNs can become overfitted and the model will include characteristics of outliers from the sample data, yielding an approximation with excellent accuracy on training examples, yet poor performance on further data from the same distribution. Overfitting can be prevented by reserving part of the data as a test set to such that unbiased estimate of the NN's accuracy. However, the model accuracy can be degraded as a result of holding aside training data for error estimation as it reduces the number of samples used for training which may be required. Cross validation is a mechanism to overcome this whereby the data set is divided into N equal-sized folds with N NN model is built instead of just a single. Each NN is trained on N1 folds and tested on the remaining fold, therefore the test fold for each NN differs from the other models [32].

2.2 Model-based learning for Run-time Management

In the context of run-time learning and management, modeling enables prediction of the current and future states of a system. This can include physical quantities, such as power, temperature and energy, or the specific properties of applications, such as performance, latency and accuracy. When applying specific requirements or constraints to these properties, models can be used to determine the system configuration that will minimize power consumption and maximize performance before execution, including control over parallelism, DVFS and DPM settings. The model-based learning approaches that have been used in literature are discussed, divided into three main control methods; task mapping (including parallelism/multi-threading control), DVFS and DPM. Categorization of existing literature is shown in Table 1. Many model-based approaches use multiple control methods in conjunction to achieve power/energy and performance optimization, with mapping applied first and then DVFS or DPM refinements made afterwards.

2.2.1 Task Mapping and Parallelism

Model-based learning is commonly employed to build power and performance models of applications executing on platforms to predict the optimal mapping of an application's tasks to processors and determine the level of parallelism that should be created.

Table 1: Classification of existing model-based run-time learning techniques for power and energy management

Domain	Reference	DPM	DVFS	Mapping
Embedded/Desktop	[32,33]		✓	
	[34–36]			✓
	[31,37]	✓	✓	
	[38]	✓		✓
	[39–42]		✓	✓
Datacenters/HPC	[39,43]		✓	✓
	[44]	✓	✓	
	[45–47]	✓		✓

The first approaches to determine the required level of parallelism and task mapping using GLMs were empirically derived using Amdahl's Law [48]. However, this fails to capture inter-thread communication, data dependence synchronizations and hardware contentions that lead to sub-optimal scaling. Modeling approaches have been developed to characterize these penalties with feedback-driven threading for homogeneous architectures [49] and Scale-Up/Scale-Out for heterogeneous architectures [38]. In the latter, *scale-out* refers to thread-level parallelism [50] and *scale-up* to the adaptive thread-core mapping enabled by heterogeneous cores. These processes are characterized by two orthogonal functions, which are derived from a combination of empirical modeling and fitting of additional coefficients via linear regression [38]. Similarly, composite models can be constructed, with a combination of an empirically-derived component and a GLM component. These are designed to characterize the system whilst also mitigate the modeling error that arises from unknown factors. In [34], a holistic and resource-agnostic scalability model is developed in order to determine the degree of parallelism to assign to each task. The model is based on predicting speed-up from Amdahl's Law, with consideration given to the aforementioned parallelism penalties, however in addition it employs linear regression to analyze the speed-up properties of particular task in order to assign the correct level of parallelism.

Coarse-grain mapping of applications to computational resources can be driven by regression-based learning [42]. The approach uses OLS approaches to build a model of the energy/performance trade-offs between using different computing resources in a heterogeneous system for a particular task. The task is mapped on a computing resource at run-time based on the minimum energy consumption for a given application performance requirement. Parallelism within each resource is not considered because of the particular platform.

On the other hand, approaches that target HPC and datacenter systems consider task-level parallelism an essential component of their predictive models. These approaches considered modeling the system in order to perform Dynamic Concurrency Throttling (DCT) [39] and thread packing [40], processes which we include as part of mapping. Curtis-Maury et al. [39] consider an IPC-based linear regression solution trained from samples of the power-performance adaptation search space collected from real workloads. They derive a performance prediction model which dynamically adjusts DCT, DVFS, and thread placement at the granularity of program phases.

In order to more accurately and generically predict performance improvements for changes in mapping, GLMs can leverage performance monitoring counters (PMCs) which are built into the hardware architecture [32,35]. This approach is portable across many applications as it only relies on information from the hardware. Furthermore, metrics such as instructions-per-cycle (IPC) and processor utilization can be used to predict performance and build linear models across many platforms [39]. Pack & Cap is an example of a model-based approach to control mapping which relies on PMC data [40]. Furthermore, it is different to other approaches in that it employs a multinomial logistic regression (MLR) classifier to make optimal DVFS and thread packing control decisions in order to maximize performance within a power budget. The addition of thread packing to DVFS as a control knob increases the range of feasible power constraints by an average of

21% when compared to DVFS alone and reduces workload energy consumption by an average of 51.6%.

SVC models can also be found in run-time management scheme and used to classify tasks or programs as suitable for particular functional units in a heterogeneous architecture. Wen et al. [36] develop an OpenCL task scheduling scheme to map kernels from multiple programs on CPU/GPU heterogeneous platforms. At run-time, it determines which kernels are likely to best utilize a device from a performance model that predicts a kernel's speedup based on its static code structure. Naive Bayes has also been used in power management to build power-performance model and perform classification [31]. In the context of this work, the goal is to devise a power management policy for issuing DVFS commands on a CMP system that minimize the total energy dissipation based on the load conditions and workload characteristics [37]. The motivation for utilizing a Bayesian classifier is to reduce the overhead of the power management activities which are performed regularly to determine and assign DVFS settings for each processor core in the system.

The use of Artificial Neural Networks (ANN) for modeling and prediction in run-time management system is not common, given the extensive training time and large volume of data that is required to achieve an accurate model, as discussed in 2.1.4. However, ANNs can have a role to play in the static components of a hierarchical system such as modeling the behavior of applications as in [32]. A resource allocation framework is created composed of per-application ANN performance models and a global resource manager. Shared system resources are periodically redistributed between applications at fixed decision-making intervals. Each application model's its performance as a function of its allocated resources and recent behavior, using an ensemble of ANNs to learn an approximation of this function. Past program behavior and allocated resource amounts are presented at the input units, and performance predictions are obtained from the output units [32]. The drawback of these model is the training of the NN weights which can only be done by performing successive passes over training examples.

2.2.2 DVFS

DVFS is used to control the performance/throughput of tasks by adjusting the operating frequency of the processor. Dynamic power dissipation is reduced as a result and energy can be saved if further voltage scaling occurs. The power and performance relationship is often modeled to enable prediction of the optimal DVFS settings. A basic model can be built from understanding of the underlying physical characteristics of the static and dynamic power dissipation of components and how these are affected by frequency and voltage or empirically from experimentation using training samples. The later may be done with the aid of hardware performance statistics such as IPC [39] or PMCs [40, 41]. The former uses multivariate linear regression to estimate specific coefficients for the hardware event rates of a particular configuration in order to determine the required DCT and DVFS settings. The *Pack & Cap* [40] approach makes use of L_1-regularization to select the most relevant PMC metrics automatically and a multinomial logistic regression (MLR) classifier to determine the DCT and DVFS settings that maximize performance under a power constraint by selecting the output with the highest probability.

Yang et al. [42] use hypotheses about the affect of frequency and voltage on the current and latency for each resource in a heterogeneous platform as the basis for their power/performance model which is used to determine the most energy efficient resource to execute on and the DVFS settings to apply given a performance requirement. This model is trained using a OLS linear regression technique at the beginning of application execution. Although it can be done a run-time, it does not change over the remainder of the application so it cannot adapt to changes in application behavior. In a similar way, Juan et al. [33] use constrained-polynomial (positive polynomial) functions to learn the relationship between performance and power and build a model based on frequency and utilization. Additional energy reduction is achieved through additional DPM techniques including turbo-mode and near-threshold operation in what they call extended-range DVFS.

A Bayesian classification approach to DVFS setting is used by Jung et al. [31, 37] in the prediction of power and performance. The predicted state is used to look up the optimal power management action from a pre-computed policy lookup table. The motivation for using this form of model is the reduced overhead of prediction in the *power manager (PM)* and as a result can provide energy savings for even rapidly and widely varying workloads.

2.2.3 DPM

DPM process is often driven by predictions from models in conjunction with mapping or DVFS actions. DPM can be achieved through migration in heterogeneous to resources with different power/performance operating points [38, 42] or by power gating CPU cores in homogeneous multi/many-core systems [32, 40]. These two processes are captured as Scale-Up and Scale-Out by Ma et al. [38] who perform DPM and mapping on a heterogeneous architecture based on prediction from GLM performance and power models with additional heuristic scheduling. Cochran et al. [40] propose a similar approach for performance optimization under a power budget with PARSEC benchmarks on a homogeneous CMP through a combination of thread packing to control parallelism and DVFS setting to reduce power.

2.3 Model-based Management in Datacenters

Model-based approaches, including supervised machine learning, have been used to increase energy efficiency in server [51] and datacenter [43] platforms. Mapping, DVFS and DPM techniques have all been employed such as workload consolidation, which aims to reduce the number of active processing resources and as result reduce the power consumption and heat dissipation [45]. Modeling the effect of these management processes is even more important as the hardware in datacenters becomes increasingly heterogeneous [43]. In this situation, power and performance must be modeled for all the settings of each resource as well as the partitioning of workloads across multiple resources. Wu et al. use linear regression to build a model of a heterogeneous CPU and FPGA platform, with support for workload partitioning [43], from both compile-time and run-time profiling, and use it to for run-time average power estimation. Lama et al. [46] apply machine learning to build fuzzy power and performance models to capture non-linear system behavior and drive a model predictive control framework. This self-adaptive modeling allows them to capture time-varying relationships between application performance and allocation of resources for dynamic and bursty workloads. Distributed controllers coordinate with each other to allocate resources and meet the service level agreements of applications.

3 Reinforcement Learning

3.1 Introduction to Reinforcement Learning

Reinforcement learning (RL) is a machine intelligence approach that has been applied in many different areas. It mimics one of the most common learning styles in natural life. The machine learns to achieve a goal by trial-and-error interaction with a dynamic environment. RL algorithms are developed to find the optimal solution to sequential decision problem, and have been proven effective in a variety of problems from different areas [21]. RL is inspired by the trial-and-error method humans used for making decisions for millions of years. In RL, the agent interacts with the system (Fig. 3).

The general learning model consists of:

- An agent
- A finite state space S
- A set of available actions A for the agent
- A reward function R: S X A $\rightarrow$ R

The goal of RL is to find the best actions under different states such that by following those best actions, the agent can optimize the long-term reward. It is achieved by learning a policy, i.e. a mapping between the states and the actions.

Q-learning is one of the most popular algorithms that perform reinforcement learning. At each step of interaction with the environment, the agent observes the environment and issues an action based on the system state. By performing the action, the system moves from one state to another. The new state gives the agent

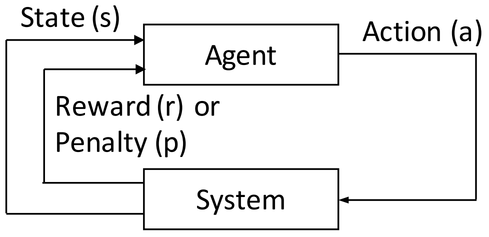


Figure 3: Agent-system interaction in RL. The system is characterized by state s. The action a is taken by an agent to change the state, with resulting outcomes *r* or *p* and new state *s'*

a reward (a real or natural number) or punishment (a negative reward) which indicates the value of the state transition. The agent keeps a value function for each state-action pair, which represents the expected long-term reward if the system starts from state s, taking action a, and thereafter following a policy. Based on this value function, the agent decides which action should be taken in current state to achieve the maximum long-term rewards. The core of the Q-learning algorithm is a value iteration update of the value function. The Q-value for each state-action pair is initially chosen by the designer and later, it is updated each time an action is issued and a reward is received, based on the following expression.

$$Q(s_i, a_i) \leftarrow Q(s_i, a_i) + \alpha \left(r_i + \gamma * \max_{a'} Q(s', a') - Q(s_i, a_i)\right) \tag{5}$$

In the above expression, r_i is the reward given at time i, and $0 \leq \alpha \leq 1$ is the learning rates which may be the same value for all pairs. The discount factor γ is a value between 0 and 1, and s' and a' are the next state and action after taking a_i, respectively. The next time when state s is visited again, the action with the maximum Q-value will be chosen. As a model-free learning algorithm, it is not necessary for the Q-learning agent to have any prior information about the system, such as the transition probability from one state to another. Thus, it is a highly adaptive and flexible algorithm.

Q-learning is originally designed to find the policy for a Markov Decision Process (MDP). It is proved that the Q-learning is able to find the optimal policy when the learning rate is reduced to 0 at an appropriate rate, given the condition that the environment is MDP. However, it is important to point out that a computing system for power/energy management is typically non-Markovian. Therefore, it is not guaranteed that Q-learning will find the optimal policy in case of such problems.

3.2 Reinforcement Learning for Run-time Management

System level power/energy management must consider the uncertainty and variability that comes from the environment, application and hardware. Statically optimized resource and power management are not likely to achieve the best performance when the input characteristics are changing. As a result reinforcement learning has been used for DPM [22–26], DVFS [18–21,52], or combination of DPM, DVFS and mapping [28,53,54] in embedded, desktop and datacenter domains. A detailed classification of existing RL based approaches for power/energy management is given Table 2.

3.3 Mapping with DPM or DVFS

Ye et al. [54] claims that existing learning based DPM is not dedicated to power reduction in multi-core processors. Further, they address this issue by including task allocation into their Q-learning based DPM framework. As the multi-core processors offer flexibility in assigning tasks to any processor core, it helps in partially controlling idle periods for power savings. This technique judiciously allocates the tasks to cores which offer a better trade-off between power consumption and system performance. To further achieve better power savings, core temperatures are integrated into DPM framework as the temperature has profound impact on leakage power consumption. The simulation-based approach is used for evaluating the effectiveness of their approach.

Table 2: Classification of existing reinforcement learning based techniques for power/energy management

Domain	Ref.	DPM	DVFS	Mapping
Embedded/Desktop	[22–26]	✓		
	[18–21,52]		✓	
	[53]	✓	✓	
	[54]	✓		✓
	[28]		✓	✓
Datacenters	[55–58]			✓
	[59,60]	✓		✓

To improve the energy efficiency and thermal aspects (average and peak temperature, and thermal cycling) in multi-core systems, a Q-learning based approach for DVFS and POSIX thread allocation is proposed in [28]. It simultaneously optimizes the temperature and energy consumption with reduced learning space by employing a two-stage hierarchical approach: a heuristic-based thread allocation at a longer time interval to improve thermal cycling, followed by a learning-based DVFS at a much finer interval to improve average temperature, peak temperature and energy consumption. This approach is evaluated on Nvidias Tegra SoC, featuring four ARM Cortex-A15 cores, running Linux with set of applications from MiBench [61], PARSEC [62] and SPLASH-2 [63].

3.4 DVFS

Reinforcement learning based approaches have been widely used for energy minimization through DVFS [18–21, 52]. Shen et al. [18] proposed an approach for simultaneous temperature, performance and energy (TPE) management. It dynamically selects the processor's voltage and frequency to achieve the required balance among energy, performance and temperature of the processor. This approach models the processor's energy as convex function, which first decreases and then increases with the frequency. Further, performance and temperature are modelled as concave and convex functions increasing with the normalized frequency. The relation among temperature, performance and energy based on the above observations determines the possible trade-off space of the TPE management. Their environment state is a vector of four components, (f, T, IPS, μ) representing the clock frequency, the temperature, the instructions per second (IPS) and the CPU intensiveness, respectively. The TPE controller offers two modes: $free$ and $constrained$. $Free$ mode allows user to explore the trade-off by tuning the weight coefficients in the penalty function and $constrained$ mode lets the user to set constraints to two out of the three parameters in T, P and E, while optimizing the third one. The approach is validated on Dell Precision T3400 workstation with Intel Core 2 Duo E8400 Processor running Linux.

Juan et al. [19] present a semi-supervised reinforcement learning (RL) based approach for performing dynamic voltage and frequency scaling (DVFS) to efficiently utilize the available on-chip power budget while maximizing the performance. Further, an evaluation and comparison among core-only, uncore-only and cooperative core/uncore DVFS control for performance boosting is discussed for the first time along with a "$reverse$" DVFS technique for maximizing performance under power constraints. The semi-supervised RL separates the centralized agent into several "distributed" agents, and arrange a supervisor to coordinate the actions among agents to best exploit the power budget for the performance increase. This helps in reducing the exponential growth of state complexity and maintains a linear complexity with the number of cores. The targeted architecture is a symmetric, NoC-based CMP containing 16 tiles, and each tile has a Pentium4 core, a private L1 cache, a shared L2 cache and an on-chip router. The evaluation with a wide spectrum of parallel, multi-threaded applications shows an average 10.9% improvement in program execution time while satisfying given power constraints.

The emerging multi-task/thread applications generally have complicated execution causality and task-level dependencies, and the execution states of one core would affect other cores in a multi-core system. To globally optimize the policy of voltage/frequency selection for improving energy efficiency in such scenarios, a core-level modular RL (MLR) based online DVFS, which explicitly considers the relationship between different cores, is proposed in [21]. This approach distributively applies modular Q-learning (MQL), one of the most commonly used MRL algorithm, to each core to learn the system behaviour. MQL decomposes the state-space into several much smaller modules and integrates multiple modules in an agent (DVFS controller of each core) to select actions based on more than its own state. At each learning-epoch, DVFS controller collects necessary information required by every module from the corresponding cores in the system and passes it into associated modules to learn and update their Q-table. As each module has its own Q-table and updates independently, a mediator is used in an agent to arbitrate the solutions reported by each module. Furthermore, task-dependency information of applications, collected from static-time workload characterization or dynamic profiling, is used to help efficiently create and associate most useful modules to each core which incurs only linear cost in core count. Experiments conducted on a homogeneous system with 4, 16, 32 and 64-cores (implemented in JADE full-system simulator [64]) running realistic applications from COSMIC benchmark suite [65], show up to 28% energy savings compared to individual-learning method.

Considering the future many-core systems, an On-line Distributed RL (OD-RL) based DVFS control algorithm to improve performance under power constraints is discussed in [52]. At the finer grain, a per-core RL method is used to learn the optimal control policy of the voltage/frequency levels that maximizes the performance under the budget. At the coarser grain, an efficient global power budget reallocation algorithm is used to maximize the overall performance. Spatially, distributed RL works on each core locally and independently, while the budget re-allocator reallocates the budget among all the cores. Temporally, distributed RL operates at every control epoch, while the budget re-allocator executes every M epochs. This approach is validated using Sniper simulator with PARSEC and SPLASH-2 multi-threaded benchmark suites. Experimental results show up to 98% less budget overshoot, up to 44.3x better throughput per over-the-budget energy, up to 23% higher energy efficiency and two orders of magnitude speedup over existing techniques [66, 67].

3.5 DPM

Learning-based DPM techniques have been proved to be effective in reducing power consumption [22–26]. An on-line learning algorithm in [25] dynamically selects the best DPM policies from a set of candidate policies called experts. Each expert has a weight factor, the value of which indicates the benefit gained if the correspondent expert was chosen during the last idle period. The one with the highest value will control the device for the next idle period. Prabha et al. [26] propose a similar approach using a different learning algorithm. The expert-based machine learning algorithm is able to find an appropriate DPM policy in short time without any prior workload information. However, it cannot explore the power-performance trade-offs effectively.

The traditional Q-learning algorithm provides a model-free solution for the MDP with a provable convergence to the optimal solution. However, in a non-markovian environment or a partially observable Markovian environment [68, 69], Q-learning is capable of achieving the same performance as other reference learning algorithms at the cost of slower convergence (number of epochs for the Q-values becoming stable). To address this issue, Liu et al. [22] propose a system level power management based on enhanced Q-learning to improve the convergence speed. It adopts the method in [70] and enhances the performance of traditional Q-learning by exploiting the sub-modularity and monotonic structure in the cost function of a power management system. The enhancement restricts the search space of Q-learning algorithm to policies whose action is non-decreasing to the number of waiting requests, which helped in improving power consumption and latency by 40% and 90%, respectively, compared to traditional Q-learning. Further, it can adapt to changing performance constraint during run-time and converge to a policy that delivers just enough performance with minimum power consumption within 50 updates. Their experimental results show up to 30% and 60% reduction in power consumption for synthetic and real workloads, respectively, when compared against existing expert-based power management technique [71].

The reinforcement learning technique is applied to multi-cores in [54] and shown superior than the distributed power managers using [72]. All the core states and actions are encoded and the learning function is approximated using the back-propagation neural network (BPNN) [54]. Tasks are assumed independent, thus their policy is able to not only determine the power mode, but also assign each task to a specific core. However, task assignment (context scheduling) has other considerations, such as data locality, that greatly affect the performance in real environments. Besides, scalability is still a problem for its time complexity O(n^2).

A policy called Multi-level Reinforcement Learning (MLRL) that is much more scalable in efficiency and effectiveness for multi-cores is proposed in [23]. The multilevel paradigm, having $coarsening$ and $uncoarsening$ phases, was first proposed for circuit partitioning and then applied to other VLSI problems. The coarsening phase does recursive clustering of elements until the problem size is small enough to be solved. Then, in the $uncoarsening$ phase, the coarsened elements are de-clustered by applying the refinement algorithm. A coarse solution is produced between the two phases. Coarsening phase generates a good approximation of the original problems, so better initial solutions can be obtained and makes refinement algorithms more effective due to local problems with smaller sizes in the uncoarsening phase.

The multilevel paradigm is exploited to compress the searching space, speed-up the convergence rate, and result in O($n$log$n$) time complexity for n cores. Target architecture is a multiprocessor system, ARM Cortex-A9 MPCore, containing n homogeneous cores and m threads per core, which can simultaneously execute $n \times m$ contexts. The workload characteristics are assumed unknown in advance; it may comprise several single- or multi-threaded programs. Simulation results, using Multi2Sim [73] and the power simulator McPAT [74], show that their policy runs 53% faster and outperforms the state-of-the-art work [54] with 13.6% energy savings and 2.7% latency penalty on average for the SPLASH-2 benchmarks while the performance penalty is close to zero.

The above approaches, including the enhanced Q-learning which is a model-free RL requiring no knowledge of state transition probability functions, needs knowledge of state action spaces and reward function. The enhanced Q-learning based DPM learns a policy online by identifying which action is the best for a certain system state, based on the reward or penalty (cost) received. In this way, the approach does not depend on any pre-designed experts, and can achieve a much wider range of power-latency trade-offs. However, as this is based on a discrete-time model of the stochastic process, it suffers from the following limitations: (i) the discrete-time controller has relatively high overhead to make frequent and regular decisions, and (ii) discrete-time controller may not make timely decisions for fast state changes.

As a solution to the above problems, Wang et al. [24] presented an online adaptive DPM technique based on the model-free RL method, which requires no prior knowledge of the state transition probability function and the reward function. Furthermore, this approach can perform policy learning and power management in a continuous-time and event-driven manner to learn a desirable timeout policy (optimal DPM policy when the service requests inter-arrival times are stationary but non-exponentially distributed [75], which derives optimal timeout value using MDP methods). It also utilizes the enhanced temporal difference (TD(λ)) learning algorithm for semi-MDP (SMDP) [76] in order to accelerate convergence and alleviate the reliance on the Markovian property. TD(λ) is more robust in non-Markovian cases as it seamlessly combines the simple one-step TD algorithm and the Monte Carlo method and has fast learning rate. Workload prediction is incorporated in this work to provide partial information about service requester (SR) state for the RL algorithm. Specifically, an online Bayesian classifier [28] is chosen as the workload predictor because of its relatively high prediction accuracy, low implementation cost, and the fact that the information it provides comes with a certain degree of certainty due to the use of posterior probability [28]. Further, it uses only two actions "keep sleep" and "wake-up", instead of time-out policy to reduce state-action pairs of RL and improves the convergence speed using multiple-update initialization, dynamic action sets, and locally randomized action selection techniques. The experiments are performed considering two different devices: a hard disk drive (HDD) and a wireless adapter card (WLAN card) on both synthesized and real workloads traces. Without sacrificing any latency, it achieves a maximum 18.6% saving in power dissipation compared to the reference expert-based approach proposed in [71]. Alternatively, the maximum latency saving without any power consumption increase is 73% compared to the existing best-of-bread DPM techniques. This method works only for single-core systems having single device without DVFS capability.

3.6 DVFS with DPM

As discussed above, DPM and DVFS has proven to be effective techniques for reducing power/energy consumption. Both DPM and DVFS provide a set of control knobs for run-time power management. From this perspective, both are fundamentally the same. While the DVFS is usually found as the power control knob for CMOS digital ICs, such as micro-controllers or microprocessors, during the active time; the DPM is usually for the peripheral devices, such as the hard disk drives and network interface, or for microprocessors running interactive applications accompanied with long idle intervals. Shen et al. [53] consider both DPM and DVFS as power management and propose a novel approach for system level power management using enhanced Q-learning. This technique considers the peripheral device (an interactive system that processes the I/O requests generated by software applications) and the microprocessor. Experiments for power management of peripheral devices are done using a simulated HDD with synthetic and real workloads. Furthermore, microprocessor power management approach, Q-learning based DVFS controller, is evaluated on a Dell Precision T3400 workstation with Intel Core 2 Duo E8400 Processor with applications from MiBench [61] and MediaBench [77]. The result showed that, compared to existing machine learning approaches, their power management technique is more flexible in adapting to different workload and hardware, and provides a wider range of power-performance trade-off.

3.7 Reinforcement Learning in Datacenters

Reinforcement learning based power management techniques have been proved to be effective to reduce energy costs in datacenters [56–60]. These learning based approaches are used for resource/workload consolidation, which involves turning off under-utilized/sleep servers to save power, and resource allocation. Fundamentally, resource/workload consolidation can be seen as DPM with mapping.

A localized version of online RL for resource allocation is presented in [56–58]. The RL module observes the application's local state, the local number of servers allocated by the arbiter, and the reward specified by the local Service Level Agreement (SLA). Considering the scalability of look-up table based Q-learning, severe approximations have been made by representing the application state solely by current mean arrival rate of page requests, and ignoring other sensor readings (e.g. mean response times, queue lengths, number of customers, etc.). Further, to improve the initial performance of the arbiter's policy, a heuristic initialization is employed. However, these approaches work well for simple systems running simple applications with less state variables and need certain amount of exploration of actions believed to be suboptimal. To address this issue, Tesauro et al. [58] proposed a hybrid method combining the advantages of both explicit model-based methods and model-free RL. This approach involves offline training of RL module, instead of online training, on data collected while an externally supplied initial policy (e.g., based on an approximate queuing model) makes management decisions in the system.

Lin et al. [59] discussed a power management technique which does both the job dispatching and resource consolidation. The job dispatch is performed at a finer interval than the resource consolidation based on the job arrival time and the server resource availability, while the resource consolidation decision on a fixed-length time slot basis. The proposed approach was verified by simulations using real Google cluster data traces. Further, Farahnakian et al. [60] proposed a Q-learning based dynamic Virtual Machine (VM) consolidation method to optimize the number of active hosts according to the current resources utilization. The method intelligently decides on when to switch a host into the active or sleep power mode through a learning agent which learns host power mode detection policy.

4 COMPARATIVE STUDY

This section presents a comparative study of various machine learning based approaches

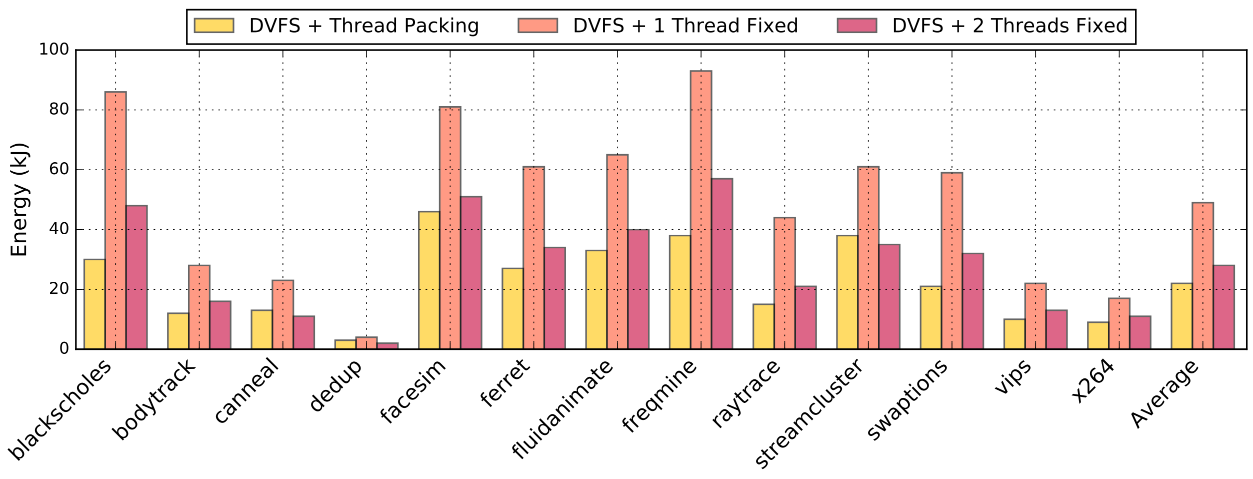


Figure 4: Comparison of energy consumption for benchmarks with and without thread-packing on a homogeneous CMP, reprinted from [40].

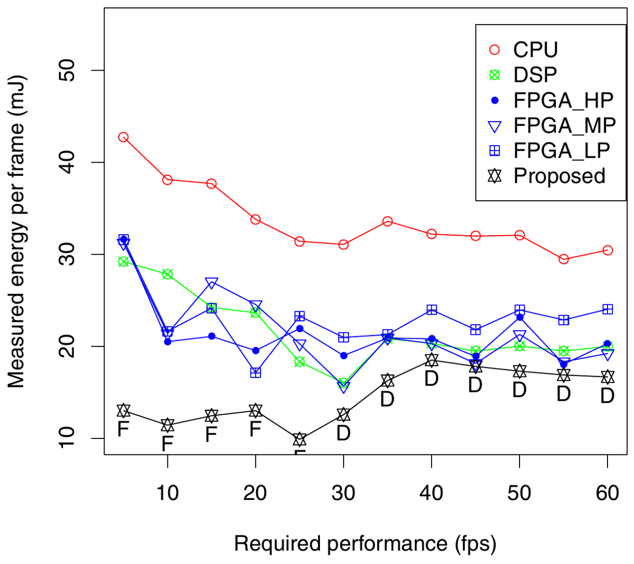


Figure 5: Energy consumption of an edge detection filter executing on the different processing elements of a heterogeneous multiprocessor over a range of performance constraints [42].

4.1 Comparison of Model-based learning Approaches

Figure 4 shows that energy savings can be achieved by combining DVFS and adaptive thread mapping on a homogeneous CMP with the aid of an MLR classifier [40]. The experiments are conducted across a series of PARSEC benchmarks. In the first experiment (DVFS + Thread Packing), the thread packing technique is used to dynamically adjust the number of threads to meet a changing power budget, with DVFS settings applied on top. The second (DVFS + 1 Thread Fixed) and third (DVFS + 2 Threads Fixed) experiments use a fixed one and two threads respectively and so must rely on predicted DVFS settings alone. For the majority of cases, thread packing increases the range of achievable power constraints over DVFS alone. Given that DPM techniques to shutdown cores has not been applied, the one thread fixed case consumes a lot more energy due to the static power consumption of the idle cores. The thread packing experiment is able to utilize all four cores when the power budget allows and the performance gain from this outweighs the increase in dynamic power.

In addition, Figure 5 shows that learning the power/performance trade-offs for each processing resource of a heterogeneous system enables prediction of the optimal mapping and DVFS settings. A linear regression

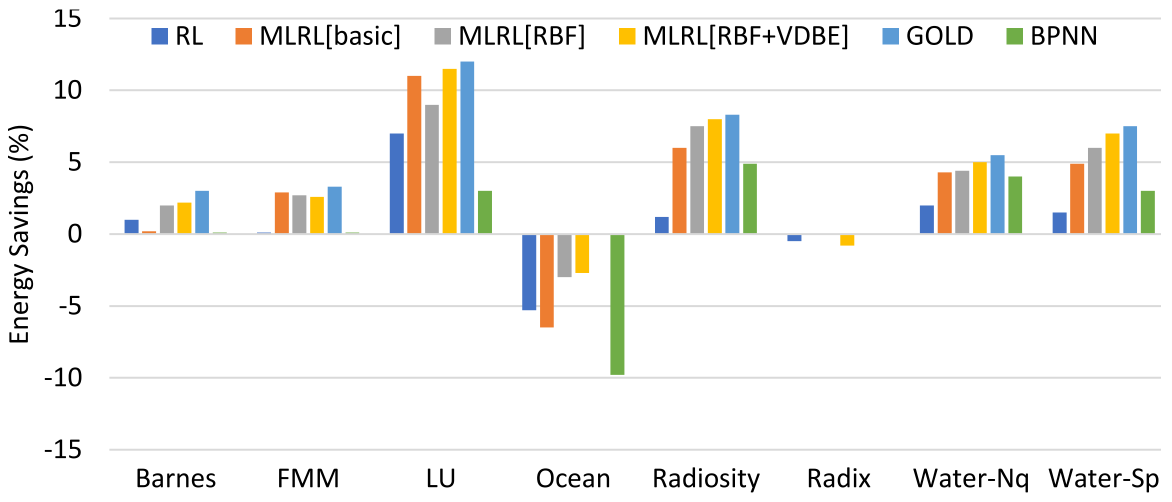


Figure 6: Comparison of energy savings achieved by various DPM techniques for SPLASH benchmark applications. Extracted from [23].

model is built from training data and used to predict the power and performance of an edge detection filter for particular mapping and DVFS settings on the heterogeneous CPU/DSP/FPGA platform [42]. The run-time management system implemented will switch the mapping of the filter between resources if the performance requirement changes to ensure optimal energy consumption per frame. The combination of mapping and adaptive DVFS setting (black line) produces a lower energy consumption for the entire range of performance levels when compared to static mapping of the filter to any of the individual resources on the platform. For the CPU and DSP experiments, the ondemand Linux governor selects the DVFS settings for each frame and the three FPGA experiments operate at fixed frequencies of 10, 50 and 100 MHz for FPGA_LP, FPGA_ MP and FPGA_HP respectively.

4.2 Comparison of Reinforcement learning Approaches

This section compares the results obtained by reinforcement learning approaches for DPM and DVFS on multi-core systems. Figure 6, extracted from [23], compares six DPM techniques: the original reinforcement learning policy (RL), MLRL without enhancement technique (MLRL[basic]), MLRL with GGAP-RBF [78] without VDBE-softmax [79] (MLRL[RBF]), MLRL with all the enhancement techniques (MLRL[RBF+VDBE]), the golden reference (GOLD), and the BPNN policy (BPNN). The RL policy has least energy saving due to its large state space (at least 2^n) as it directly encodes the power states. For MLRL[basic], the state space is still large on the root node; many samples are needed for n states and n actions, but the agent seldom tries other actions using ϵ-greedy. This leads to higher energy saving than BPNN (10.99% $>$ 5.71%), demonstrating that the proposed multilevel paradigm is more effective than directly applying function approximation with BPNN. The function approximation MLRL[RBF] achieves energy saving similar to MLRL[basic] (10.99% 11.31%) and close to the upper bound. Finally, combining VDBE-softmax scheme further saves more energy (11.31% to 13.58%). Furthermore, it has been reported that the performance penalty of MLRL[RBF+VDBE] is close to zero and energy savings are close to GOLD (14.66%).

Figure 7 shows comparison of energy consumption of four DVFS approaches with two different switching intervals: predictive [80], learning-based [53], ondemand [81], learning transfer-based [20]. The energy values are normalized with respect to learning transfer-based approach. Among all these approaches, learning transfer-based approach achieves better energy savings (up to 38 %) and is efficient to adapt to workload and performance variations within the application (intra) and across the applications (inter). Moreover, as per the observations made in [20], both predictive and learning-based approaches fail to meet application performance requirement despite their energy savings.

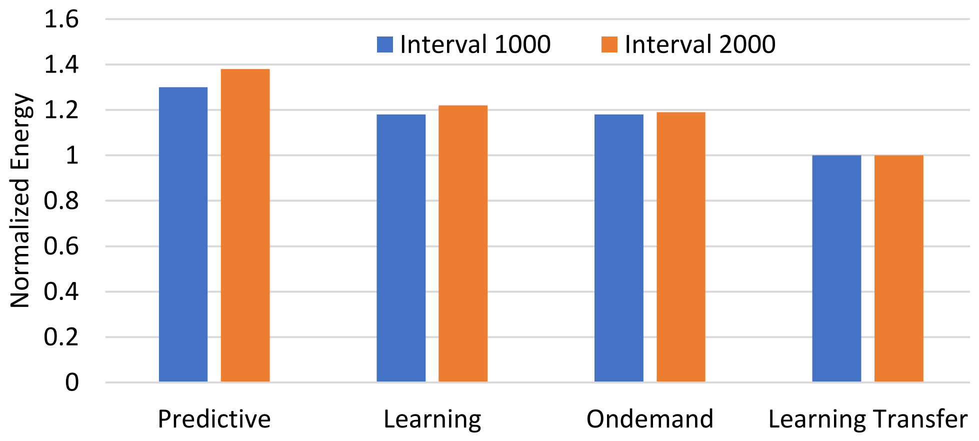


Figure 7: Comparison of normalized energy consumption of various DVFS approaches. Extracted from [20].

5 Upcoming Trends for Future Research and Open Challenges

This section highlights some of the upcoming trends and challenges to be faced to take the learning-based run-time power/energy management approaches for multi/many-cores into the next era.

5.1 Distributed Reinforcement Learning

The literature indicates that many reinforcement learning (RL) algorithms exist to find near-optimal solutions in polynomial time. These algorithms have been applied for single core systems or multi-core systems with small number of cores. Since the number of states increases exponentially with the number of cores, the RL method is known to be non-scalable. An explosion in the state space (number of states) takes place for large scale problems and thus they become very expensive to use at run-time. Therefore, efficient and scalable RL algorithms are desired for large scale many-core systems with hundreds of cores.

In order to address the scalability issues imposed by centralized resource management in many-core systems, distributed or hierarchical management has been employed [29, 82]. In distributed management, each core is checked independently to make management decisions, whereas the cores are grouped into multiple clusters in hierarchical management and each cluster is managed by a local manager. It is evident that hierarchical approach exploits feature of both the centralized and distributed approaches.

Similar to resource management, distributed reinforcement learning is desired to address the scalability issues [52]. In order to reduce the learning complexity, a hierarchical reinforcement learning can be explored, which can provide trade-off between solution quality and computational complexity. The parameters of the hierarchical approach (e.g. cluster size) can be adjusted to achieve several trade-off points, which can be used to optimize energy consumption.

5.2 Learning-based Multi-objective Optimization

We have shown that learning-based power/energy optimization of multi/many-core systems has been extensively focused. However, along with the power/energy, modern multi/many-core systems need to be optimized for several other metrics, e.g. temperature, reliability, fault-tolerance, and security. The temperature is optimized to improve reliability, reduce the cooling cost of many-core based HPC datacenters and mitigate its effect on performance and leakage power. Reliability is to be optimized to increase the mean time to failure of a system. However, in case a fault has happened, optimization need to be performed to achieve fault-tolerance. Optimization for security has become an important concern due to possible attack in the communication channels of connected devices and their interaction with untrusted devices. All these requirements indicate the need for multi-objective optimization.

It has also been observed that learning-based approaches have been used to optimize one metric, e.g. performance while satisfying power budget requirement [52]. In future, it is expected that the requirements to jointly optimize for several performance metrics will grow. Towards it, some progress has already been made, e.g., three metrics execution time, energy consumption and temperature are optimized in [83]. However, in [83], learning-based optimization is not employed. Since learning-based approaches have tremendous potential for optimizations, they should be explored to perform multi-objective optimization.

It is evident that optimization for multiple objectives will increase the design space and thus the learning time. Therefore, the design space needs to be efficiently pruned so that Pareto-fronts for several conflicting objectives can be derived quickly at run-time. In order to maintain a low complexity, the learning process can be bounded by an upper limit on the exploration time.

5.3 Learning-based Optimization for Diverse Application Domains

In addition to multi/many-core systems, learning-based optimization can be performed for several application domains. In fact, it has already be explored for some application domains. For example, a reinforcement learning approach for self optimizing memory controllers [84], learning-based energy management in a hybrid electric vehicles [85] and learning approaches for manufacturing [86]. Such diversity for application domains indicates that, machine learning is effective across a wide spectrum of application domains.

Based on the above, it is expected that machine learning is going to prevail for the design and optimization of systems for various application domains. Several companies have already started projecting the potential of machine learning, e.g., ARM's DynamIQ technology based processors will include dedicated processor instructions for machine learning and are expected to deliver up to a 50x boost in performance over the next 3-5 years relative to Cortex-A73-based systems today. These evidences indicate widespread adoption of machine learning approaches for future computing systems.

6 Conclusion

This paper provides a survey of learning-based run-time power and energy management strategies for multi/many-core systems. Specially, model-based and reinforcement learning approaches optimizing for energy consumption is embedded systems, desktop systems and HPC datacenters are surveyed and compared. Based on the analysis of the surveyed and compared learning-based strategies, upcoming trends and open challenges are identified. The research directions highlighted in this survey are expected to advance in future due to potential of learning-based approaches, which will also help to address the open challenges. These advances will need to explore efficient machine learning strategies to take the learning-based approaches for multi/many-core systems into the next era of computing.

Acknowledgements

This work was supported in parts by the EPSRC Grant EP/L000563/1 and the PRiME Programme Grant EP/K034448/1 (www.prime-project.org). Experimental data used in this paper can be found at DOI: http://doi.org/10.5258/SOTON/D0109.

REFERENCES

[1] A. Jerraya, H. Tenhunen, and W. Wolf, "Guest Editors' Introduction: Multiprocessor Systems-on-Chips," *Computer*, no. 7, pp. 36–40, 2005.

[2] "Exynos 5 Octa (5422)." www.samsung.com/exynos/, 2016.

[3] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS," in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 98–589, 2007.

[4] AMD, "AMD Opteron 6000 series processors," 2011. http://www.amd.com/en-us/products/server/opteron/6000 (Last visited: 12 February, 2016).

[5] TILE-Gx, "First 100-core Processor with the New TILE-Gx Family," 2009. http://www.tilera.com/ (Last visited: 12 February, 2016).

[6] B. D. De Dinechin, D. Van Amstel, M. Poulhiès, and G. Lager, "Time-critical computing on a single-chip massively parallel processor," in *Proceedings of IEEE Conference on Design, Automation and Test in Europe (DATE)*, pp. 1–6, 2014.

[7] L. Benini and G. De Micheli, "Networks on chips: a new SoC paradigm," *Computer*, no. 1, pp. 70–78, 2002.

[8] F. Worm, P. Ienne, P. Thiran, and G. De Micheli, "An adaptive low-power transmission scheme for on-chip networks," in *Proceedings of IEEE/ACM/IFIP Conference on Hardware/Software Codesign and System Synthesis (ISSS+CODES)*, pp. 92–100, 2002.

[9] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of Network-on-chip," *ACM Comput. Surv.*, no. 1, 2006.

[10] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of ACM Design Automation Conference (DAC)*, pp. 746–749, 2007.

[11] J. Ceng, J. Castrillon, W. Sheng, H. Scharwächter, R. Leupers, G. Ascheid, H. Meyr, T. Isshiki, and H. Kunieda, "MAPS: an integrated framework for MPSoC application parallelization," in *Proceedings of ACM Design Automation Conference (DAC)*, pp. 754–759, 2008.

[12] A. Mallik *et al.*, "MNEMEE - An Automated Toolflow for Parallelization and Memory Management in MPSoC Platforms," in *Proceedings of ACM Design Automation Conference (DAC)*, 2011.

[13] G. Martin, "Overview of the mpsoc design challenge," in *Proceedings of ACM Design Automation Conference (DAC)*, pp. 274 –279, 2006.

[14] L. Smit, G. Smit, J. Hurink, H. Broersma, D. Paulusma, and P. Wolkotte, "Run-time mapping of applications to a heterogeneous reconfigurable tiled system on chip architecture," in *Proceedings of IEEE International Conference on Field-Programmable Technology (FPT)*, pp. 421–424, 2004.

[15] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proceedings of IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 233–239, 2003.

[16] A. K. Singh, T. Srikanthan, A. Kumar, and W. Jigang, "Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms," *J. Syst. Archit.*, pp. 242–255, 2010.

[17] S. Kaushik, A. K. Singh, W. Jigang, and T. Srikanthan, "Run-time computation and communication aware mapping heuristic for noc-based heterogeneous mpsoc platforms," in *IEEE International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pp. 203–207, 2011.

[18] H. Shen, J. Lu, and Q. Qiu, "Learning based dvfs for simultaneous temperature, performance and energy management," in *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, pp. 747–754, IEEE, 2012.

[19] D.-C. Juan and D. Marculescu, "Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pp. 97–102, ACM, 2012.

[20] R. A. Shafik, S. Yang, A. Das, L. A. Maeda-Nunez, G. V. Merrett, and B. M. Al-Hashimi, "Learning transfer-based adaptive energy minimization in embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 6, pp. 877–890, 2016.

[21] Z. Wang, Z. Tian, J. Xu, R. K. Maeda, H. Li, P. Yang, Z. Wang, L. H. Duong, Z. Wang, and X. Chen, "Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system," in *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pp. 684–689, IEEE, 2017.

[22] W. Liu, Y. Tan, and Q. Qiu, "Enhanced q-learning algorithm for dynamic power management with performance constraint," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 602–605, European Design and Automation Association, 2010.

[23] G.-Y. Pan, J.-Y. Jou, and B.-C. Lai, "Scalable power management using multilevel reinforcement learning for multiprocessors," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 19, no. 4, p. 33, 2014.

[24] Y. Wang and M. Pedram, "Model-free reinforcement learning and bayesian classification in system-level power management," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3713–3726, 2016.

[25] G. Dhiman and T. S. Rosing, "Dynamic power management using machine learning," in *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, pp. 747–754, ACM, 2006.

[26] V. L. Prabha and E. C. Monie, "Hardware architecture of reinforcement learning scheme for dynamic power management in embedded systems," *EURASIP Journal on Embedded Systems*, vol. 2007, no. 1, pp. 1–1, 2007.

[27] A. K. Singh, A. Das, and A. Kumar, "Energy Optimization by Exploiting Execution Slacks in Streaming Applications on Multiprocessor Systems," in *Proceedings of ACM Design Automation Conference (DAC)*, pp. 115:1–115:7, 2013.

[28] A. Das, B. M. Al-Hashimi, and G. V. Merrett, "Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 2, p. 24, 2016.

[29] A. K. Singh, P. Dziurzanski, H. R. Mendis, and L. S. Indrusiak, "A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, p. 24, 2017.

[30] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199–222, Aug. 2004.

[31] H. Jung and M. Pedram, "Improving the efficiency of power management techniques by using bayesian classification," in *9th International Symposium on Quality Electronic Design (isqed 2008)*, pp. 178–183, March 2008.

[32] R. Bitirgen, E. Ipek, and J. F. Martinez, "Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach," in *Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 41, (Washington, DC, USA), pp. 318–329, IEEE Computer Society, 2008.

[33] D.-C. Juan, S. Garg, J. Park, and D. Marculescu, "Learning the optimal operating point for many-core systems with extended range voltage/frequency scaling," in *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, CODES+ISSS '13, (Piscataway, NJ, USA), pp. 8:1–8:10, IEEE Press, 2013.

[34] S. Sridharan, G. Gupta, and G. S. Sohi, "Adaptive, efficient, parallel execution of parallel programs," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, (New York, NY, USA), pp. 169–180, ACM, 2014.

[35] K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in *Proceedings of the 39th Annual International Symposium on Computer Architecture*, ISCA '12, (Washington, DC, USA), pp. 213–224, IEEE Computer Society, 2012.

[36] Y. Wen, Z. Wang, and M. F. P. O'Boyle, "Smart multi-task scheduling for opencl programs on cpu/gpu heterogeneous platforms," in *2014 21st International Conference on High Performance Computing (HiPC)*, pp. 1–10, Dec 2014.

[37] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, pp. 1395–1408, Sept 2010.

[38] J. Ma, G. Yan, Y. Han, and X. Li, "An analytical framework for estimating scale-out and scale-up power efficiency of heterogeneous manycores," *IEEE Transactions on Computers*, vol. 65, pp. 367–381, Feb 2016.

[39] M. Curtis-Maury, A. Shah, F. Blagojevic, D. S. Nikolopoulos, B. R. de Supinski, and M. Schulz, "Prediction models for multi-dimensional power-performance optimization on many cores," in *Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques*, pp. 250–259, 2008.

[40] R. Cochran, C. Hankendi, A. K. Coskun, and S. Reda, "Pack & cap: Adaptive dvfs and thread packing under power caps," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, (New York, NY, USA), pp. 175–185, ACM, 2011.

[41] H. Sasaki, S. Imamura, and K. Inoue, "Coordinated power-performance optimization in manycores," in *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, pp. 51–61, Sept 2013.

[42] S. Yang, R. A. Shafik, G. V. Merrett, E. Stott, J. M. Levine, J. Davis, and B. M. Al-Hashimi, "Adaptive energy minimization of embedded heterogeneous systems using regression-based learning," in *2015 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pp. 103–110, Sept 2015.

[43] Y. Wu, D. S. Nikolopoulos, and R. Woods, "Runtime support for adaptive power capping on heterogeneous socs," in *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pp. 71–78, July 2016.

[44] H. Shen and Q. Qiu, "Contention aware frequency scaling on cmps with guaranteed quality of service," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[45] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power" struggles: Coordinated multi-level power management for the data center," in *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIII, (New York, NY, USA), pp. 48–59, ACM, 2008.

[46] P. Lama, Y. Guo, C. Jiang, and X. Zhou, "Autonomic performance and power control for co-located web applications in virtualized datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 1289–1302, May 2016.

[47] M. A. H. Monil, R. Qasim, and R. M. Rahman, "Energy-aware vm consolidation approach using combination of heuristics and migration control," in *Ninth International Conference on Digital Information Management (ICDIM 2014)*, pp. 74–79, Sept 2014.

[48] D. H. Woo and H.-H. S. Lee, "Extending amdahl's law for energy-efficient computing in the many-core era," *Computer*, vol. 41, pp. 24–31, Dec. 2008.

[49] M. A. Suleman, M. K. Qureshi, and Y. N. Patt, "Feedback-driven threading: Power-efficient and high-performance execution of multi-threaded workloads on cmps," in *Proceedings of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XIII, (New York, NY, USA), pp. 277–286, ACM, 2008.

[50] P. Lotfi-Kamran, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, S. Idgunji, E. Ozer, and B. Falsafi, "Scale-out processors," in *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pp. 500–511, June 2012.

[51] J. Ng, X. Wang, A. K. Singh, and T. Mak, "Defragmentation for efficient runtime resource management in noc-based many-core systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 11, pp. 3359–3372, 2016.

[52] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1521–1526, EDA Consortium, 2015.

[53] H. Shen, Y. Tan, J. Lu, Q. Wu, and Q. Qiu, "Achieving autonomous power management using reinforcement learning," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 2, p. 24, 2013.

[54] R. Ye and Q. Xu, "Learning-based power management for multicore processors via idle period manipulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 7, pp. 1043–1055, 2014.

[55] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A hybrid reinforcement learning approach to autonomic resource allocation," in *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on*, pp. 65–73, IEEE, 2006.

[56] J. L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in *Proceedings of the 1st International Conference on energy-Efficient Computing and Networking*, pp. 215–224, ACM, 2010.

[57] G. Tesauro *et al.*, "Online resource allocation using decompositional reinforcement learning," in *AAAI*, vol. 5, pp. 886–891, 2005.

[58] G. Tesauro, R. Das, W. E. Walsh, and J. O. Kephart, "Utility-function-driven resource allocation in autonomic systems," in *Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, pp. 342–343, IEEE, 2005.

[59] X. Lin, Y. Wang, and M. Pedram, “A reinforcement learning-based power management framework for green computing data centers,” in *Cloud Engineering (IC2E), 2016 IEEE International Conference on*, pp. 135–138, IEEE, 2016.

[60] F. Farahnakian, P. Liljeberg, and J. Plosila, “Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning,” in *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pp. 500–507, IEEE, 2014.

[61] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” in *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pp. 3–14, IEEE, 2001.

[62] C. Bienia and K. Li, “Parsec 2.0: A new benchmark suite for chip-multiprocessors,” in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, vol. 2011, 2009.

[63] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, “The SPLASH-2 programs: Characterization and methodological considerations,” in *ACM SIGARCH Computer Architecture News*, vol. 23, pp. 24–36, ACM, 1995.

[64] R. K. Maeda, P. Yang, X. Wu, Z. Wang, J. Xu, Z. Wang, H. Li, L. H. Duong, and Z. Wang, “Jade: a heterogeneous multiprocessor system simulation platform using recorded and statistical application models,” in *Proceedings of the 1st International Workshop on Advanced Interconnect Solutions and Technologies for Emerging Computing Systems*, p. 8, ACM, 2016.

[65] Z. Wang, W. Liu, J. Xu, B. Li, R. Iyer, R. Illikkal, X. Wu, W. H. Mow, and W. Ye, “A case study on the communication and computation behaviors of real applications in noc-based mpsocs,” in *VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on*, pp. 480–485, IEEE, 2014.

[66] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, “An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget,” in *Proceedings of the 39th annual IEEE/ACM international symposium on microarchitecture*, pp. 347–358, IEEE Computer Society, 2006.

[67] J. A. Winter, D. H. Albonesi, and C. A. Shoemaker, “Scalable thread scheduling and global power management for heterogeneous many-core architectures,” in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pp. 29–40, ACM, 2010.

[68] M. D. Pendrith and C. Sammut, *On reinforcement learning of control actions in noisy and non-Markovian domains*. Citeseer, 1994.

[69] K. Sikorski and T. Balch, “Model-based and model-free learning in markovian and non-markovian environments,” in *Proceedings of Agents-2001 Workshop on Learning Agents*, 2001.

[70] D. V. Djonin and V. Krishnamurthy, “V-blast power and rate control under delay constraints in markovian fading channels-optimality of monotonic policies,” in *Information Theory, 2006 IEEE International Symposium on*, pp. 2099–2103, IEEE, 2006.

[71] G. Dhiman and T. S. Rosing, “Dynamic power management using machine learning,” in *Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided Design*, ICCAD ’06, (New York, NY, USA), pp. 747–754, ACM, 2006.

[72] Y. Tan, W. Liu, and Q. Qiu, “Adaptive power management using reinforcement learning,” in *Computer-Aided Design-Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pp. 461–467, IEEE, 2009.

[73] R. Ubal, J. Sahuquillo, S. Petit, and P. López, “Multi2sim: A simulation framework to evaluate multicore-multithread processors,” in *IEEE 19th International Symposium on Computer Architecture and High Performance computing, page (s)*, pp. 62–68, Citeseer, 2007.

[74] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, “Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures,” in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 469–480, IEEE, 2009.

[75] T. Simunic, L. Benini, P. Glynn, and G. De Micheli, “Event-driven power management,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 7, pp. 840–857, 2001.

[76] S. J. Bradtke and M. O. Duff, “Reinforcement learning methods for continuous-time markov decision problems,” *Advances in neural information processing systems*, pp. 393–400, 1995.

[77] “MediaBench.” http://euler.slu.edu/?fritts/mediabench/.

[78] G.-B. Huang, P. Saratchandran, and N. Sundararajan, “A generalized growing and pruning rbf (ggap-rbf) neural network for function approximation,” *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57–67, 2005.

[79] M. Tokic and G. Palm, “Value-difference based exploration: Adaptive control between epsilon-greedy and softmax.,” in *KI*, pp. 335–346, Springer, 2011.

[80] K. Choi, R. Soma, and M. Pedram, “Dynamic voltage and frequency scaling based on workload decomposition,” in *Proceedings of the 2004 international symposium on Low power electronics and design*, pp. 174–179, ACM, 2004.

[81] V. Pallipadi and A. Starikovskiy, “The ondemand governor,” in *Proceedings of the Linux Symposium*, vol. 2, pp. 215–230, sn, 2006.

[82] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, “Mapping on Multi/Many-core Systems: Survey of Current and Emerging Trends,” in *Proceedings of ACM Design Automation Conference (DAC)*, pp. 1:1–1:10, 2013.

[83] H. F. Sheikh and I. Ahmad, “Efficient heuristics for joint optimization of performance, energy, and temperature in allocating tasks to multi-core processors,” in *IEEE International Green Computing Conference (IGCC)*, pp. 1–8, 2014.

[84] E. Ipek, O. Mutlu, J. F. Martínez, and R. Caruana, “Self-optimizing memory controllers: A reinforcement learning approach,” in *Computer Architecture, 2008. ISCA’08. 35th International Symposium on*, pp. 39–50, IEEE, 2008.

[85] X. Lin, P. Bogdan, N. Chang, and M. Pedram, “Machine learning-based energy management in a hybrid electric vehicle to minimize total operating cost,” in *Computer-Aided Design (ICCAD), 2015 IEEE/ACM International Conference on*, pp. 627–634, IEEE, 2015.

[86] L. Monostori, A. Márkus, H. Van Brussel, and E. Westkämpfer, “Machine learning approaches to manufacturing,” *CIRP Annals-Manufacturing Technology*, vol. 45, no. 2, 1996.

7 BIOGRAPHIES

Amit Kumar Singh received the B.Tech. degree in Electronics Engineering from Indian Institute of Technology (Indian School of Mines), Dhanbad, India, in 2006, and the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University (NTU), Singapore, in 2013. He was with HCL Technologies, India for year and half before starting his PhD at NTU, Singapore, in 2008. He worked as a post-doctoral researcher at National University of Singapore (NUS) from 2012 to 2014 and at University of York, UK from 2014 to 2016. Currently, he is working as senior research fellow at University of Southampton, UK. His current research interests include system level design-time and run-time optimizations of 2D and 3D multi-core systems with focus on performance, energy, temperature, and reliability. He has published over 45 papers in the above areas in leading international journals/conferences. Dr. Singh was the receipt of ISORC 2016 Best Paper Award, PDP 2015 Best Paper Award, HiPEAC Paper Award, and GLSVLSI 2014 Best Paper Candidate. He has served on the TPC of IEEE/ACM conferences like ISED, MES, NoCArc and ESTIMedia.

Charles Leech is currently a Senior Research Assistant working for the PRiME Project in the Department of Electronics and Computer Science at the University of Southampton where he is completing is PhD and received a BEng Hons degree in Electronic Engineering. His focus is on the development of a cross-layer framework for run-time management and system-level approximate computing on embedded platforms. His work also includes power and performance optimisation of next-generation applications, including computer vision, on heterogeneous many-core systems.

Basireddy Karunakar Reddy received his M.Tech. degree in Microelectronics and VLSI from Indian Institute of Technology (IIT), Hyderabad, India in 2015. He is a Ph.D. student in Electronic and Electrical Engineering at the University of Southampton. His research interests include design-time and run-time optimization of performance and energy in many-core heterogeneous systems.

Bashir M. Al-Hashimi is an ARM Professor of Computer Engineering, Dean of the Faculty of Physical Sciences and Engineering, and the Co-Director of the ARM-ECS Research Centre, University of Southampton, Southampton, U.K. He has published over 380 technical papers. His current research interests include methods, algorithms, and design automation tools for low-power design and test of embedded computing systems. He has authored or co-authored five books and has graduated 35 Ph.D. students.

Geoff V. Merrett received the B.Eng. degree (Hons.) in electronic engineering and the Ph.D. degree from the University of Southampton, Southampton, U.K., in 2004 and 2009, respectively. He is currently an Associate Professor in electronic systems with the University of Southampton. His current research interests include low-power and energy harvesting aspects of embedded & mobile systems. He has published over 100 articles in journals/conferences in the above areas. Dr. Merrett was the General Chair of the Energy Neutral Sensing Systems Workshop from 2013 to 2015. He is a fellow of the The Higher Education Academy.