Modular Reinforcement Learning for Self-Adaptive Energy Efficiency Optimization in Multicore System

Zhe Wang, Zhongyuan Tian, Jiang Xu, Rafael K. V. Maeda, Haoran Li, Peng Yang, Zhehui Wang,
Luan H. K. Duong, Zhifei Wang, Xuanqi Chen
The Hong Kong University of Science and Technology
Email: {zhe.wang, jiang.xu}@ust.hk

***Abstract*—Energy-efficiency is becoming increasingly important to modern computing systems with multi-/many-core architectures. Dynamic Voltage and Frequency Scaling (DVFS), as an effective low-power technique, has been widely applied to improve energy-efficiency in commercial multi-core systems. However, due to the large number of cores and growing complexity of emerging applications, it is difficult to efficiently find a globally optimized voltage/frequency assignment at runtime. In order to improve the energy-efficiency for the overall multicore system, we propose an online DVFS control strategy based on core-level Modular Reinforcement Learning (MRL) to adaptively select appropriate operating frequencies for each individual core. Instead of focusing solely on the local core conditions, MRL is able to make comprehensive decisions by considering the running-states of multiple cores without incurring exponential memory cost which is necessary in traditional Monolithic Reinforcement Learning (RL). Experimental results on various realistic applications and different system scales show that the proposed approach improves up to 28% energy-efficiency compared to the recent individual-RL approach.**

I. Introduction

As Dennard scaling fails, shrinking device feature size can no long bring proportional benefit, and the system performance is strictly constrained by the high power consumption. By integrating multiple light-weight cores with lower voltage and clock frequency in a single chip, multi-core architecture has been proved to be able to effectively reduce power consumption without sacrificing system performance. To fully take advantage of technology scaling and further improve system performance, tens and hundreds of cores are integrated in one chip, and quickly as a result, the power wall is reached again. For this reason, a lot of low-power techniques are applied in modern multi-core processors.

Dynamic Voltage and Frequency Scaling, as one of the most effective technique, has been widely available in commercial multi-core processors. It is able to change the working voltage/frequency (V/F) of a core at runtime so that the system can make trade-offs between performance and energy-consumption. The basic principle of the DVFS-based power management is to adaptively assign appropriate operating points (V/F levels) for each core according to runtime workload and system conditions. However, this is a non-trivial job due to the complexity of hardware system and varying runtime conditions. Even provided with the detailed workload characteristics, finding the optimal V/F assignment for each core to achieve the maximum global energy-efficiency is proved to be an NP-hard problem which can hardly be solved in an on-line approach. A lot of works have proposed to use formal polynomial-time algorithms and heuristics to quickly search for acceptable sub-optimal solutions [1]–[3]. Recently, learning-based techniques [4]–[12] has been proposed to effectively improve the system energy-efficiency thanks to its advanced capability of adapting to different runtime conditions and variabilities.

Most of the aforementioned existing works adopt the Reinforcement Learning (RL) [13] technique and mainly target single core system or systems with small number of cores (less than 8). In order to find a globally optimized V/F assignment for multi-/many-core systems, the traditional monolithic RL methods have to jointly consider the states of all cores. In this way, the state-space is exponentially enlarged with respect to the core count, and the computation and memory overhead also increases accordingly. As more and more cores being integrated in a system, the state-space easily explodes, which renders the traditional RL method impractical for modern multi-core systems. To solve this problem, some works [14]–[16] presented improved learning-based techniques by ignoring the relationships among different cores and applying individual per-core RL locally. And these new techniques show better scalability than previous methods. However, emerging multi-task/thread applications usually have complicated execution causality and task-level dependences, and the execution states of one core would have profound impact on other cores. In order to globally optimize the policy of V/F assignment to improve energy efficiency, it is important to explicitly consider the relationships among different cores.

Motivated by the above reason, in this work, we propose a novel *Modular Reinforcement Learning (MRL)* [17], [18] based online DVFS control strategy for multi-core systems to improve system energy-efficiency. Instead of making decisions (V/F-selection) only based on the state of the local core, MRL integrates multiple modules, which monitor the states of other cores, to jointly guide the V/F-selection process. We distributively apply the modular Q-learning (MQL), one of the most commonly used MRL algorithm, to each core to comprehensively learn the system behaviors and select appropriate operating points for them to maximize the global energy-efficiency. Since MRL allows to decompose the huge state-space into much smaller parts, the storage and computation overhead can be significantly reduced. In this work,

This work is partially supported by GRF16235616, IRS17EG48, and FSGRF15EG04.

978-1-5090-1558-0/17/$31.00 ©2017 IEEE

task-dependence information of applications are used to help efficiently create and associate the most useful modules to each core so that the modular structure incurs only linear cost with respect to the core count. Experimental results on realistic benchmarks and four system scales show that the proposed method can improve energy-efficiency by up to 28% compared to the state-of-the-art individual-learning approach.

The rest of the paper is organized as follows. Section II briefly reviews the related studies and motivates our work. Section III introduces the basics of RL and general problem formulation. In Section IV, we present the MQL-based framework. Section V analyzes the experimental results, and Section VI concludes the work.

II. Related work and motivation

A. Related work

Learning techniques have been studied and applied in both Dynamic Power Management (DPM) and DVFS management. RL as a self-evolving model-free method requiring no prior information, recently shows great potential for online power management. Tan *et al.* [6] applied model-free RL to DPM with a partially observable environment and constrained cost function. Liu *et al.* [8] proposed an enhanced Q-Learning (QL) algorithm that exploits the sub-structure of the cost function in power management to lead the exploration for better solutions. Shen *et al.* [9] further improved the RL algorithm in power management by adopting a second layer controller to help decide the parameters in its cost functions. Shafik *et al* [11] proposed a learning transfer mechanism to tackle the intra-/inter-application variabilities. Wang *et al.* [12] presented to use the Temporal Difference RL learning algorithm to cope with the semi-Markov decision process, and proposed a Bayesian Classifier to improve the state-prediction accuracy. These works effectively improved system energy-efficiency for single core systems or systems with a small number of cores.

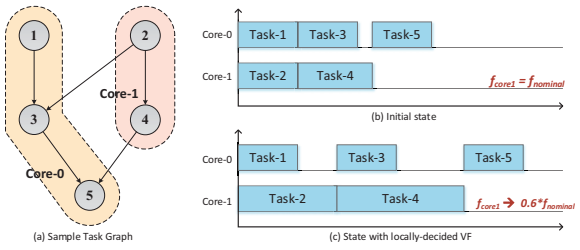

Fig. 1. Motivation example: (a) task graph of a sample application; (b) initial task execution scenario (c) execution scenario with locally-decided V/F level of core-1.

In order to continue exploiting the good features of RL while reducing its implementation overhead on multi-core systems, some works proposed improvements to the traditional RL framework. Ye *et al.* [14] modified the original Q-learning by using a Neural Network to approximate the Q-value function instead of saving them in a table. Otoom *et al.* [16] and Chen *et al.* [15] both proposed a hierarchical structure with core-level RL to select the locally best action and a chip-level power-budget allocator for global coordination. These works either assume the task execution on different cores are independent or did not explicitly consider the impact of task-dependencies.

B. Motivation

A simple motivative example is shown in Fig. 1 to highlight the importance of inter-core relationship. As shown in Fig. 1(a), a synthetic application composed of five tasks are running in a multi-core system, and these tasks are allocated to two cores as shown in the figure. In real systems task-to-core mapping can be decided by the operating system (OS) at runtime or by users at static-time, and in this work we focus on static task-mapping scenarios. For the simplicity of illustration, only the operating point of core-1 is adjusted and the V/F level for core-0 is fixed, and we assume the controller tries to minimize the energy-consumption. If a local DVFS control strategy is applied, as shown in (c), only the state of core-1 itself will be considered, core-1 will tend to select a low-performance V/F-level in order to reduce energy. However, this decision increase the execution time and energy consumption of core-0 due to task dependences. For applications with more tasks and systems with more cores, this inter-core impact will accumulate and makes the situation exponentially sophisticated. If the DVFS controller makes decisions only based on local information, the induced negative impact on other cores will even be possible to offset the gain on this core.

When applying RL technique for multi-core power management, only learning local core conditions will lower the learning-rate and make the policy-selection process easy to stuck in a local-optimal point. On the other hand, although monolithic RL can provide globally optimized solutions, it is known to be non-scalable as core count increases. In order to applying RL efficiently in a multi-core system without ignoring the inter-core relationship, we propose a Modular Reinforcement Learning based DVFS control strategy to improve system energy efficiency with incurring only linear memory cost to the number of cores.

III. Problem formulation

In this section, we describe the general problem formulation of RL based power management for a computing system.

A. Reinforcement Learning basics

Reinforcement Learning is an artificial intelligence technique that is akin to the common learning paradigm of natural lives. The entity performing an RL algorithm is called an agent, and the RL agent gradually learns the appropriate actions in an interactive dynamic environment so as to achieve a certain goal. The fundamental elements in an RL model consists of: a finite state space S; a set of candidate actions A; a state transition function $s_{t+1} = T(s_t, a_t)$ which defines the next state when performing a certain action in the current state; and a reward/penalty function $r_t = R(s_t, a_t)$ which returns a numeric value. So the goal of RL is to find the optimal policy π, which is defined as action-selecting rule at a certain

state ($S \rightarrow A$), so that the average long-term reward can be maximized. Q-Learning is one of the most commonly used RL algorithm. It is an incremental learning method that does not require the model of the environment (the state transition function). Every time the agent comes to a learning-epoch t, the agent observes the environment and select an action a_t to perform based on the current state s_t. The environment will move to a new state s_{t+1} and report a reward value r_t for the last state-action pair. Based on these information, an indicative value, called Q-value, is updated as per Eq. (1)

$$Q_{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (1)$$

In the above expression, α is the learning-rate coefficient, γ is the discount factor with value in $[0, 1]$ and it is used to discount the impact from future states. The updated Q-value associated with each state-action pair is stored in a look-up table (Q-table). A general rule for action-selection is to choose the action with the largest Q-value. However it is also necessary to have chances to explore other actions to avoid stuck in a sub-optimal policy. The most common strategy to balance exploration and exploitation is the ϵ-greedy, in which the agent selects the current best action with a probability $1-\epsilon$ and randomly select an action, otherwise.

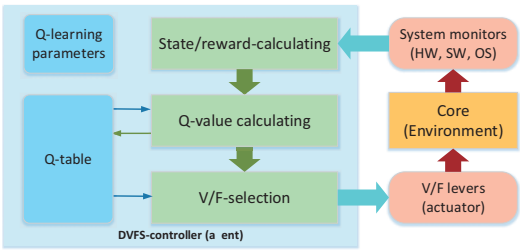

Fig. 2. Overview of Q-Learning based DVFS control.

B. Power management problem formulation

The QL is applied on each core for DVFS control. Note that, for the ease of illustration, we describe the general problem formulation in a single core environment, and the modular extension for multi-core system will be detailed in the next section. Fig. 2 illustrates the general structure and work-flow of QL-based DVFS control. As the figure shows, the DVFS controller is treated as the agent, and the core system, including both hardware and software, can be treated as the environment. The controller regularly acquires the system performance information from monitors which could be hardware sensors, OS performance counters and software-layer monitors. Based on these information, the controller determines the current state of the core and calculates the reward. After updating the Q-value, the controller selects an action and issue it to V/F levers to change the operating point of the core accordingly.

As aforementioned, there are three major elements in Q-Learning algorithm: a finite state-space, a set of available actions and a reward function. In this work, we define the state as a 2D tuple $s_t = (h_t, \mu_t)$, where h_t is the core throughput in terms of busy-cycle-count per unit time, and can be calculated by the following equation:

$$h_t = \frac{num\_busy\_cycles_t}{time\_elapsed_t}, \quad (2)$$

and μ_t is referred as *CPU intensiveness* [9] indicating the impact of cache-miss on performance and can be calculated as follows,

$$\mu_t = 1 - \frac{num\_cycles\_L1\$\_stalled_t}{num\_busy\_cycles_t}. \quad (3)$$

All the values used for the calculation of these two metrics are easy to access by reading the corresponding performance counters. Both h and μ are divided into five bins in our work. The bin-widths are adaptively determined according to their statistical distributions as performed in [15]. Regarding the action set, we intuitively define each V/F-level as an available action. The reward definition should reflect the the final objective. Since we aim to optimize the system energy-efficiency, the reward r_t is defined as throughput per energy-consumption, which is expressed as $r_t = \frac{h_t}{e_t}$. It is easy to see that r_t is inversely proportional to energy-delay-product (EDP), so the higher it is, the more energy-efficient a system is. For the parameters in Eq. (1), we set that the learning-rate α will decay with the visit times of the corresponding state-action ($\alpha_t = \alpha_0 / visit\_time(s_t, a_t)$). In order to make smart trade-offs between exploration and exploitation, we use an improved ϵ-greedy strategy by letting the ϵ decay with the visit times of the state.

IV. Modular Reinforcement Learning for Multicore Energy Efficiency Optimization

Traditional monolithic QL method formulates the whole multicore system as a fully joint state-space and action-space of all cores. Assuming there are N cores, the Q-table will have $|S|^N \cdot |A|^N$ entries. This explosive state/action-space will result in dramatically high memory overhead, and significantly lower the convergence speed and decision-quality because of the insufficient experiences in states. To tackle this problem, Modular Learning paradigm is proposed and widely used. The Modular Q-Learning algorithm [17], [18] is developed to address the problem of combinatorial explosion in multi-agent/goal reinforcement learning. The basic principle of MQL is to decompose the state-space into several much smaller modules and integrate multiple modules in an agent to help it select actions with consideration of more than its own states. The modular architecture can be interpreted as that each module is monitoring the impact of the local agent on the corresponding part of the environment. Since 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 based on some rules. MQL has been reported to be able to improve learning rate and the quality of learned policies. Additionally, it can also facilitate the learning of cooperative behaviors among multiple agents.

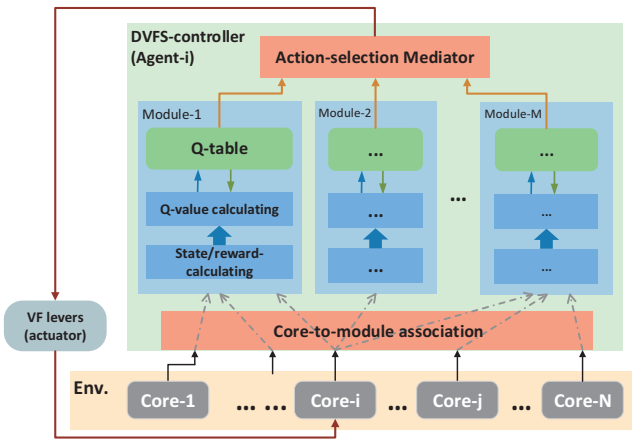

Fig. 3. Overview of Modular Q-Learning based adaptive DVFS control scheme.

Dynamic energy-efficiency optimization in multi-core systems can be easily modeled and solved as a multi-agent RL problem, in which the DVFS-controller of each core can be treated as an agent. An overview of the proposed MQL-based control framework is shown in Fig. 3. The overall multi-core system is the environment and each core has a individual DVFS-controller (agent) comprising of several modules. At each learning-epoch, agent-i collects necessary information required by every module from the corresponding cores in the system. Then the information is transferred into the associated modules, and each of them will learn to update its Q-table as traditional QL does. Since each module has a dedicated Q-table whose values are independent from each other, the mediator will go through the Q-table of each module and arbitrate a global selection. There are two important issues when applying MQL in a multi-core system. One is the modular structure, including module-composing and module-to-agent assignment, and the other is the mediation strategy for action-selection.

A. Modular structuring

There is no commonly best modular structure for all problems, and it requires specific knowledge to help create an effective learning architecture. Building a modular structure mainly includes two types of mappings: agent-spaces to module, and modules to agent. In MQL, one module can either consider the state-space of only one core or multiple (yet much smaller than the total number of cores) cores jointly. As shown in Fig. 3, module-2 only consists of the state space of core-i, while module-1 considers three cores jointly. In the traditional MQL structure, each module covers the state of one core, and each agent includes all different modules. In this case, the Q-table size for one agent is $N \cdot |S| \cdot |A|$. However, based on empirical observations, we find that usually the behavior of a core only affects limited number of cores, and the relevance degrees to different cores are not identical either. So the traditional modular structure that fairly considers every other core is unsuitable due to the imbalanced core-relationship, and it will also induce extra overhead by integrating unnecessary modules.

Algorithm 1 Proposed Modular Q-Learning Based DVFS control algorithm.

Require: Q-table $QT^{ij}(s,a)$,
Q-Learning parameters α, γ, ϵ,
Input: μ_t, h_t, e_t

// main learning loop.
when reaching a learning-epoch t **do**
  for all agent-i in parallel **do**
    for all module-j in agent-i **do**
      $s_t = calculate\_state\_id(\mu_t^i, h_t^i)$
      $r_t = calculate\_reward(h_t^i, e_t^i)$
      $QT^{ij}(s_{last}, a_{last}) =$
        $update\_q\_value(QT^{ij}(s_{last}, a_{last}), s_t, r_t, \alpha, \gamma)$
    end for
    $rand\_action = \epsilon\text{-}greedy(\epsilon)$
    if $rand\_action$ is $true$ **then**
      $a_t^i = random\_action(A)$
    else
      $a_t^i = great\_mass(QT^{ij}(s,a))$
    end if
    $issue\_action(a_t^i)$
  end for
end when

We use the knowledge of application task-dependences to help properly construct modular structure for each agent. The knowledge of task/thread-level dependences can be acquired through either static-time workload characterization or dynamic profiling [19]. By combining this information with the static task assignment, we can quantitatively access the relevance-degree between two cores in terms of the number of pair-wise tasks which are inter-related and assigned on these two cores respectively. For instance, in the example shown in Fig. 1, the relevance-degree of core-1 to core-0 is 2 and core-0 to core-1 is 0. In each agent, we integrate two modules including one focusing on the local core and another jointly considers the local core and the core with the highest relevance to the local core. So the Q-table size for each core is $(|S| + |S|^2)|A|$, which is constant for any system scale, and the overall Q-table size for the system is linear to the core count. Although our fixed modular structure in this work has already shown remarkable energy-efficiency improvement, more advanced and sophisticated modular-structuring strategies can also be explored in future works.

B. Mediation strategy

The mediation strategy plays a critical role in action exploration and exploitation. Usually the mediator uses a simple heuristic to determine the final action based on objectives of the learning problem. There are two widely used mediation techniques: Greatest-Mass (GM) and Top-Value. The GM algorithm performs selection in a majority-voting manner, and the Top-Value is a winner-take-all method. In our method, we

applied the GM technique to select the action that benefits all modules the most. The selecting function is expressed as follows:

$$a_t^i = \underset{a^i \in A^i}{\operatorname{argmin}} \sum_{j=1}^{M} Q^{ij}(s_t^{ij}, a^i), \tag{4}$$

where Q^{ij} is the Q-value of the state-action pair in the $j-th$ module of agent-i, and a^i indicates the action candidate of agent-i. GM orders actions by their sum over each module and select the one with the greatest sum (mass). The intuition behind the GM strategy is to make the agent maximize the overall reward of multiple cores. And with GM, the agent may forgo a local best action to tradeoff for better global improvement. This algorithm is light-weight with linear complexity to the number of modules, so that is suitable for runtime implementation.

C. Overall flow

The overall work-flow of the proposed MQL-based DVFS control algorithm is described in Alg. 1. The control routine is invoked regularly at runtime. When the system reaches a learning/control-epoch, the control agent for each core will first update the Q-table in its inner modules (Line 8-12), then decide whether it should exploit the best action or explore a random action (Line 13). If it is going to exploit, the GM method will be applied to find the jointly optimized action.

V. Experimental Results

A. Experiment setup

We apply the proposed method to 4, 16, 32 and 64-core homogeneous systems to show its efficiency and effectiveness. The whole system is implemented in JADE full-system simulator [20] for cycle-level simulation. For all the systems, each processor core has a private L1-cache and shared L2-cache, and these modules are interconnected via mesh-based NoC architecture. Five operating VF levels are available for cores: 0.55V/1.4GHz, 0.5V/1.2GHz, 0.45V/1GHz, 0.4V/800MHz and 0.35V/600MHz, and the nominal operating frequency is set to 1GHz. Power and energy consumptions are estimated by McPAT tool [21]. We use five realistic applications from the COSMIC benchmark suite [22], which are RS-dec, RS-enc FFT, US and LDPC. The applications are executed continuously and the proposed DVFS-control process is invoked regularly at each epoch. The epoch length varies across different applications in terms of finishing specific unit input data-block, such as processing one frame for the US (Ultrasound-image diagnostic) application. In order to show the advantage in improving energy-efficiency, we compare the proposed MQL-based approach with the individual local RL method (referred as individual-learning), which is adopted in recent state-of-the-art learning based works [15], [16]. Since the original work is for power-budget overshooting minimization, for fair comparison, we change its reward function to optimize energy-efficiency as our approach does.

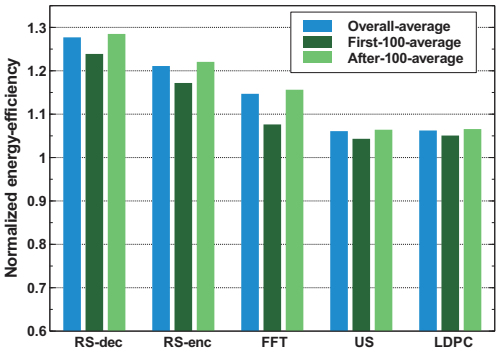

Fig. 4. Energy-efficiency improvement in different execution intervals on 32-core system.

B. Results and analysis

In this section, we use the metric $\frac{h}{e}$ ($\propto \frac{1}{EDP}$) to evaluate the energy-efficiency improvement of the proposed MQL technique. Fig. 4 shows the normalized average energy-efficiency of the proposed MQL-based approach with respect to individual-learning on 32-core system. The blue bar shows the overall average (over 700 epochs) results, and it is easy to see that MQL-based method achieves better energy-efficiency than individual-learning for all benchmark applications. For RS-dec, the improvement can be around 28%. As we discussed earlier, the MQL technique can help increase the learning-rate and improve the quality of the learned policy, which means that the system can reach a better state in shorter time. In order to illustrate these effects, we further show the average energy-efficiency over the first 100 running iterations (epochs) and after 100th epochs. A high first-100-average energy-efficiency indicates that the system can learn quickly, and a high after-100-average result suggests that the system finally learns a better policy. From the figure, we can observe that the MQL-based method provides higher energy-efficiency in both execution intervals, which verifies the effectiveness of MQL.

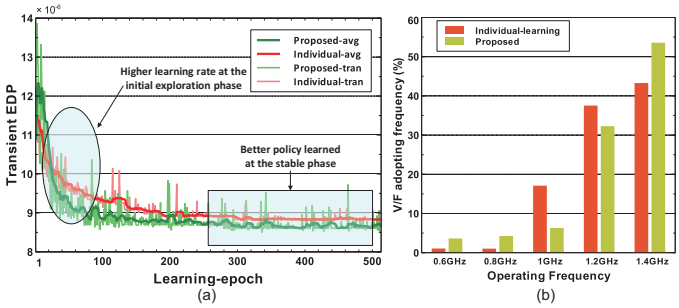

Fig. 5. (a) EDP (lower the better) transient of US application on 32-core system; (b) Percentage of each operating-frequency being adopted.

To illustrate the learning process in more detail, we show the EDP transient diagram of the US application on 32-core system in Fig. 5(a). The system EDP gradually goes down as more learning instances happen. We can observe that the MQL-based approach (green lines) converges more quickly than the individual-learning approach (red lines). And

ultimately, the proposed method reaches a lower EDP level than the individual method. The statistics of different V/F levels that are selected are shown in Fig. 5(b), and we can easily see the differences in the distributions of selected-actions. This is because taking the states of more cores into consideration could make an agent forgo the locally best action for a globally better one. Another interesting point is that in Fig. 5(a), at the very beginning, the EDP could even rise, this is because the learning agents are exploring different policies without any prior model or knowledge, and there must be some policies resulting in bad energy-efficiency.

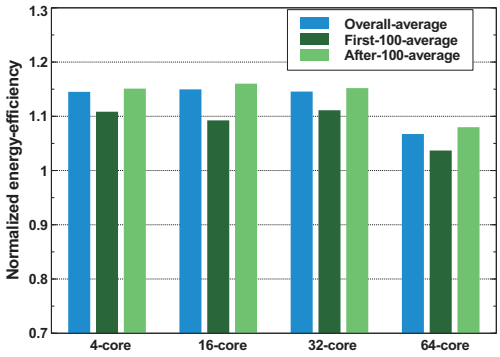

Fig. 6. Energy-efficiency improvement on different system scales.

To evaluate the scalability of the proposed method, Fig. 6 shows the normalized average energy-efficiency with respect to the individual-learning method on different system scales. MQL-based method shows better performance on all four system scales, and on average improves the energy-efficiency by 12.6% over the four scales.

VI. Conclusion and Future Work

Energy-efficiency plays a critical role in modern mutli-core system. Being able to adapt to varying system conditions, learning-based DVFS control techniques is presented as promising solutions to improve system energy-efficiency at runtime. However, these techniques are known to be non-scalable in multi-core systems to provide globally-optimized solutions. For this reason, we propose to apply Modular Reinforcement Learning technique to adaptively decide the operating point for each core to maximize global energy-efficiency. Experimental results on multiple applications and different system scales show that the proposed approach can improve energy-efficiency by up to 28% compared to individual-learning method.

In the future work, we plan to adaptively construct modular structure at runtime based on application profiling and OS scheduling information. Moreover, we will investigate the performance of different modular constructing rules to make appropriate trade-offs between module counts and learning quality.

References

[1] Z. Lai, K. T. Lam, C. L. Wang, J. Su, Y. Yan, and W. Zhu, "Latency-aware dynamic voltage and frequency scaling on many-core architectures for data-intensive applications," in *Cloud Computing and Big Data (CloudCom-Asia), 2013 International Conference on*, 2013, pp. 78–83.

[2] C. Isci, A. Buyuktosunoglu *et al.*, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, 2006, pp. 347–358.

[3] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Low Power Electronics and Design, 2007 ACM/IEEE International Symposium on*.

[4] 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*, ser. ICCAD '06.

[5] G. Theocharous, S. Mannor, N. Shah, P. Gandhi, B. Kveton, S. Siddiqi, and C.-H. Yu, "Machine learning for adaptive power management," *Intel Technology Journal*, vol. 10, no. 4, Nov. 2006.

[6] Y. Tan, W. Liu, and Q. Qiu, "Adaptive power management using reinforcement learning," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*.

[7] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, vol. 29, no. 9, Sep. 2010.

[8] W. Liu, Y. Tan, and Q. Qiu, "Enhanced q-learning algorithm for dynamic power management with performance constraint," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE)*.

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

[10] A. Das, A. Kumar *et al.*, "Workload uncertainty characterization and adaptive frequency scaling for energy minimization of embedded systems," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15, 2015.

[11] R. A. Shafik, S. Yang, A. Das, L. A. Maeda-Nunez *et al.*, "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.

[12] Y. Wang and M. Pedram, "Model-free reinforcement learning and bayesian classification in system-level power management," *IEEE Transactions on Computers*, vol. PP, no. 99, 2016.

[13] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[14] 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.

[15] 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*, ser. DATE '15, 2015.

[16] M. Otoom, P. Trancoso, H. Almasaeid, and M. Alzubaidi, "Scalable and dynamic global power management for multicore chips," in *Proceedings of the 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures*.

[17] S. Whitehead, J. Karlsson, and J. Tenenberg, *Learning Multiple Goal Behavior via Task Decomposition and Dynamic Policy Merging*. Boston, MA: Springer US, 1993, pp. 45–78.

[18] N. Ono and K. Fukumoto, *A modular approach to multi-agent reinforcement learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 25–39.

[19] N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of splash-2 and parsec," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, 2009, pp. 86–97.

[20] R. K. V. Maeda, P. Yang, X. Wu *et al.*, "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*, ser. AISTECS '16, 2016.

[21] S. Li, J. H. Ahn *et al.*, "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*, Dec 2009, pp. 469–480.

[22] Z. Wang, W. Liu, J. Xu *et al.*, "A Case Study on the Communication and Computation Behaviors of Real Applications in NoC-based MPSoCs," in *IEEE Computer Society Annual Symp. VLSI*, 2014.