3-26-2018

Investigation of LSTM Based Prediction for Dynamic Energy Management in Chip Multiprocessors

Milad Ghorbani Moghaddam
Marquette University

Wenkai Guan
Marquette University

Cristinel Ababei
Marquette University, cristinel.ababei@marquette.edu

Investigation of LSTM Based Prediction for Dynamic Energy Management in Chip Multiprocessors

Milad Ghorbani Moghaddam
Electrical and Computer Engr., Marquette University, Milwaukee WI
Wenkai Guan
Electrical and Computer Engr., Marquette University, Milwaukee WI
Cristinel Ababei
Electrical and Computer Engr., Marquette University, Milwaukee WI

IEEE Keywords

Energy consumption, Heuristic algorithms, Computers, Energy management, Optimization, Artificial neural networks, Logic gates

Abstract:

In this paper, we investigate the effectiveness of using long short-term memory (LSTM) instead of Kalman filtering to do prediction for the purpose of constructing dynamic energy management (DEM) algorithms in chip multi-processors (CMPs). Either of the two prediction methods is employed to estimate the workload in the next control period for each of the processor cores. These estimates are then used to select voltage-frequency (VF) pairs for each core of the CMP during the next control period as part of a dynamic voltage and frequency scaling (DVFS) technique. The objective of the DVFS technique is to reduce energy consumption under performance constraints that are set by the user. We conduct our investigation using a custom Sniper system simulation framework. Simulation results for 16 and 64 core network-on-chip based CMP architectures and using several benchmarks demonstrate that the LSTM is slightly better than Kalman filtering.

SECTION I.

Introduction

Continuous increase in computational demands has made chip multiprocessors (CMPs) the work horse of most computing systems including portable devices, desktop computers, servers and datacenters. While CMPs provide great computational capabilities, they do face the problem of increased energy consumption, especially in the case of mobile devices and datacenters. In mobile devices, the energy consumption affects directly the battery life. Thus, it is desirable to find ways to save energy in order to prolong the battery life. In datacenters or warehouse scale computers, huge amounts of energy are consumed and this increases the electricity and operation costs as well as the environmental pollution [1]. For example, it has been estimated that by 2020, the energy consumption in datacenters will increase to 140 billion kilowatt-hours, imposing $13 billion per year in electricity bills in American businesses as well as emitting nearly 150 million metric tons of carbon pollution annually [2]. Because servers in these datacenters consume a significant portion of the energy consumed, it would be beneficial on many fronts to develop means to reduce energy consumption of CMPs that are used in datacenter servers.

Dynamic voltage and frequency scaling (DVFS) is one of the most popular techniques to enable optimizations of energy consumption in processors. Energy consumption is related to the clock frequency and the square of the voltage supply. DVFS takes advantage of these relations to control the energy consumption by changing dynamically the voltage-frequency (VF) pairs of the processor as a whole or of the individual cores. While support for DVFS at the processor level is common place today, per-core DVFS support has only recently started to be studied and supported by a few commercial multicore processors. Several recent studies have shown the benefits of per-core or per-cluser-of-cores DVFS capabilities [3]–[4][5].

Under the assumption that such per-core DVFS will become standard in future multicore processors, in this paper, we investigate the use of long short-term memory (LSTM) based prediction in dynamic energy management (DEM) for chip multiprocessors. Specifically, we propose a new DVFS based energy management algorithm. The objective of this algorithm is to reduce energy consumption under performance constraints, which are set as a performance loss threshold by the user. We test the

proposed algorithm on several benchmarks using a custom system simulation framework that uses the Sniper tool.

The remainder of this paper is organized as follows. The next section reviews related literature. Then, we present background information on LSTM and on a performance loss estimation method called delayed instruction count (DIC) that we later use in this paper. The proposed dynamic energy management algorithm is then presented in section V. Section VI reports simulation results and a comparison to a similar method that uses Kalman filtering as the prediction approach. We summarize our findings in the conclusion section VII.

SECTION II.

Related Work

There has been a lot of work done on methods to find voltage-frequency (VF) pairs in algorithms that employed DVFS for energy optimization in processors. These methods include also machine learning. Examples of such studies used online learning [6], artificial neural networks [7], [8], supervised learning [9], and reinforcement learning [10], [11]. Many of these methods were used in heuristic algorithms for energy optimization. However, algorithms that employed game theory, convex optimization, and combinatorial optimization were studied too [12]–[13][14]. Most of these algorithms were proposed in the context of homogeneous (i.e., formed by identical cores) processors. However, DVFS based energy and temperature management of heterogeneous processors has been studied as well. For example, the study in [15] proposed DVFS and temperature- and performance-aware task assignment strategies for heterogeneous processors that maximize energy savings, while maintaining the temperature at safe levels.

While the above previous studies focused mainly on the cores inside a CMP, recent studies focused also on the interconnects and the shared last level caches (collectively called the uncore) to estimate the performance of the CMP and use that in DVFS based energy optimization algorithms. For example, the study in [16] uses the number of cache misses while the study in [17] uses the number of non-speculative reads that result in last-level cache misses (called leading loads), and the study in [18] extends that for variable memory access latencies. Similarly, the authors in [19] take into consideration the off-chip (L2) I-cache misses and off-chip (L2) D-cache load misses in their estimation processes. The study in [20] proposed a DVFS policy for the uncore. The policy uses a technique similar to the TCP Vegas congestion control and was shown to result in significant energy savings. These methods based on last-level cache miss cycles and on non-pipelined stall cycles can be very sensitive to the accuracy of counting misses and stalls. The study in [21] addressed this issue by estimating performance losses due to frequency throttling by using a concept called delayed instruction count (DIC). They used the estimations in a Kalman filtering technique to predict the workload in the next control period and to identify VF pairs that can help reduce energy consumption without violating the performance constraint set by the user. However, the quality of the results in that approach depends on the actual prediction technique that is employed. As such, here, we investigate the use of long short-term memory (LSTM) as an alternative prediction technique to Kalman filtering.

SECTION III.

Background on Long Short-Term Memory (LSTM) Model

The neural network (NN) is a popular model in machine learning. The idea behind it is to model the human brain as a network of neurons (nodes) to mimic the learning process of the human brain on computers. The simplest and the most popular neural network architecture is the feedforward neural network. In feedforward NNs the information is transferred through the network from one layer to the next in the forward direction only and no cyclic connections exist between layers, as illustrated in the simplified diagram from Fig. 1.a. NNs can be employed to construct models that capture input-output relationships. After typically training these models with labeled training data (formed by known input-output pairs), the NN model can be used to infer or predict the output for new inputs or features.

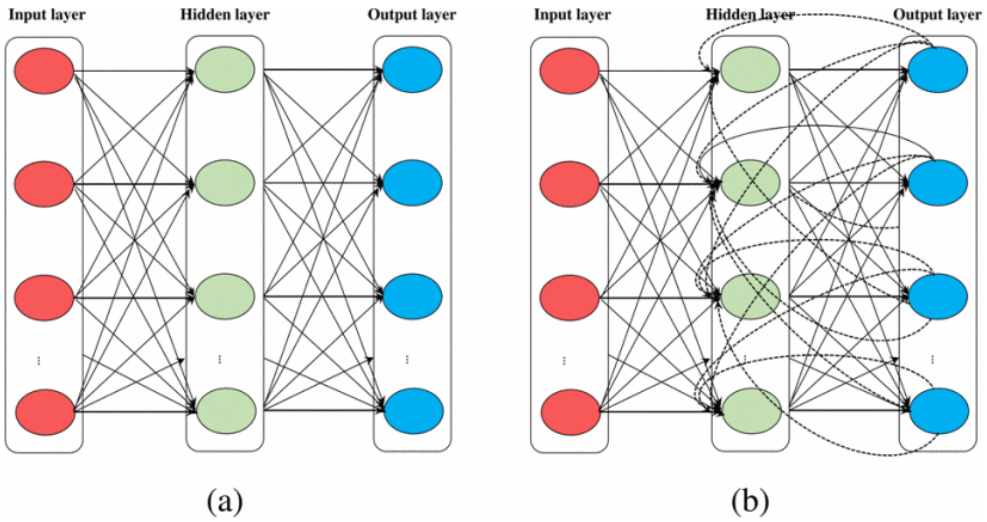

Fig. 1. Simplified diagrams of two types of neural networks (a) Feedforward neural network and (b) Recurrent neural network.

Because feedforward NNs do not have any cycles or loops, their temporal modeling capability is rather limited. Therefore, in situations where the prediction of the output must depend on long histories of the input feature sequence, the recurrent neural network (RNN) can represent a better model. The RNN model includes cyclic connections between different layers as illustrated in the simplified diagram from Fig. 1.b. The challenge that the RNN model faces though is that it can be difficult to train standard RNNs to solve problems that require learning long-term temporal dependencies. This is because the gradient of the loss function decays exponentially with time; this is known as the vanishing gradient problem. To address this problem, the long short-term memory (LSTM) was proposed [22]. The LSTM network is an RNN that uses special units in addition to the standard units. LSTM units include *memory cells* that can store information for long periods of time in addition to special units called gates that control the flow of information. In other words, these gates are used to determine what to store as well as when to allow reads, writes and erasures of information into/from cells.

Fig. 2 shows the simplified diagrams of the three different cells used by feedforward NNs, RNNs, and LSTM networks. It can be observed that the LSTM cell is more complex. The added complexity is due to the input, forget and output gates that decide whether to let new inputs in, erase the present cell state, and let the state impact the output at a given time step. These gates are activated through weighted signals connected to an activation function. These weighted signals are adjusted during the learning process. That is, the cells learn when to allow data to enter, leave or be deleted through the iterative process of making guesses, backpropagation of errors, and adjustment of weights via the gradient descent technique [23].

SECTION IV.

Background on Performance Loss Estimation

We build our work on the performance loss estimation technique proposed in [21]. This technique uses the new concept of delayed instruction count (DIC). Therefore, in this section, we briefly describe that technique.

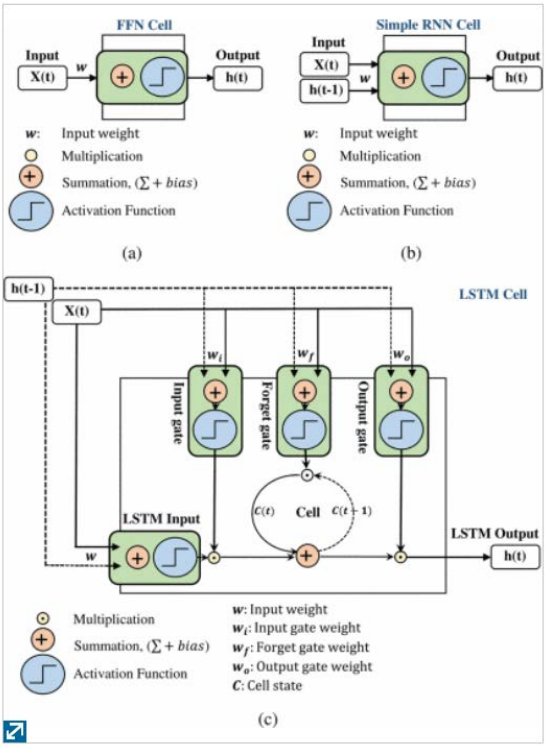

Fig. 2. Simplified diagrams of three different cells (a) Feedforward NN cell, (b) RNN cell, and (c) LSTM cell.

The main idea of the DVFS based energy optimization for CMPs is to periodically assess the CMP system at runtime and decide about how to change the voltage and frequency of each core. The voltage and frequency are changed usually as a pair and they take only a relatively small number of values. Energy can be reduced by lowering voltage and frequency, but that usually comes at the expense of some performance degradation, especially when the application at hand is workload

intensive during the so called region of interest (ROI) where all cores performs useful work all the time. However, in reality there are differences among applications, and these differences make the performance degradation to be sensitive to the changes in VF settings.

Having an accurate method to estimate at runtime the performance loss during the next control period for a given VF pair is very important to the success of implementing an effective DVFS strategy that can maximize energy reductions while offering guarantees that performance is not degraded beyond a user set threshold. The study in [21] introduced such a technique to estimate the performance loss and we use it in this paper as well. The technique uses the concept of delayed instructions count (DIC), which is used to derive an equation to calculate dynamically the amount of performance loss (PL) that would be incurred if we were to switch the current VF pair for a given core for the next control period to a throttling VF pair. The amount of performance loss is estimated with respect to the reference case, which is always that of the highest frequency. While the details of the derivation are available in [21], here we only present the final formula that we later use inside the DVFS based energy optimization algorithm. This formula is given by the following expression:

$$PL = \sum_{P=1}^{N} \frac{T_{P_{delay}}}{T} = \sum_{P=1}^{N} \frac{I_{P_{Done}} \times \left(\frac{CPI_P\left(\frac{f_H}{f_P}-1\right)}{f_H}\right)}{T} \quad (1)$$

The main idea is that in each control period the technique estimates the time, $T_{P_{delay}}$, that would be needed to execute the additional number of instructions that could have been executed if the highest clock frequency were used. In the above equation, PL represents the total performance loss that is incurred over all the control periods that the region of interest (ROI) of the benchmark is split in. N is the total number of control periods, which are indexed by the variable index P. All control periods have a fixed duration denoted as T. CPI_P is the average number of CPU cycles per instruction when the CPU is not stalled and does useful work during period P. $I_{P_{done}}$ represents the number of instructions executed during period P when the core operates at a particular clock frequency f_P. f_H is the highest frequency among all supported VF pairs.

SECTION V.

Using LSTM Prediction for Dynamic Energy Management

Dynamic energy management is achieved by continuously monitoring the CMP and periodically changing the VF pairs for each core such that energy consumption is reduced while performance is not degraded or degraded within the limit set by the user. By default, all the cores in the CMP are assumed to operate at the highest frequency to get the best performance. Then, during periods when the workload is low, frequency can be throttled to save energy with little or no impact on performance. The method discussed in the previous section provides us with equation (1) that can be used to estimate the performance loss suffered in the control period that just finished execution. However, to make an informed decision about what VF pair to use during the next incoming control period, we need to extend equation (1) to be able to estimate the performance loss during the next period. That

will help us to identify VF pairs for the next control period. Thus, for a given performance constraint, if we could predict the incoming workload for each core of the CMP, we would be able to better guess the best sets of VF pairs for the all the cores that can reduce energy consumption without violating the performance constraint.

The method described in [21] proposed a Kalman filtering approach to predict the workload in the next control period. Our objective in this paper is to investigate other prediction approaches. Specifically, we are interested in the use of the LSTM model due to its ability to capture history in time series.

The block diagram of the dynamic energy management (DEM) scheme investigated in this paper is shown in Fig. 3. The scheme is implemented as a control loop inside our customized Sniper simulation framework. For each application or benchmark, the system simulator is halted periodically. At each stop, statistics about the performance counters (i.e., number of instructions executed by each core and CPI values) are collected and fed into the algorithm. The algorithm records the last statistics for a moving window of *w* past control periods. It sends this information to the LSTM predictor that predicts the workload for the next control period based on the characteristics of the recorded past.

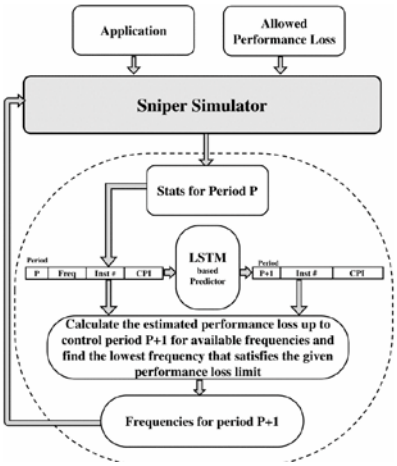

Fig. 3. Block diagram of the DVFS based dynamic energy management scheme as implemented inside our custom sniper simulator.

The LSTM model itself is rather simple. It is constructed with just one hidden layer of 4 LSTM blocks or neurons and the sigmoid activation function is used for each block. To use the predictor, the LSTM model is first trained. Training data include CPI and instruction count and are collected and organized as input features for a moving window of *w*=20 in order for the prediction to take into consideration the past 20 data sequences. This is similar to the Kalman filter configuration used in [21] against which we will compare later on. The collection process is done during separate runs of the custom Sniper simulation framework and without any DEM algorithm. The model is trained with 20,000 samples collected at intervals of 1 ms during simulations of only a small number of benchmarks on architectures with 16 and 64 cores. The training samples are collected from all individual cores in these architectures.

The DEM algorithm utilizes the predicted CPI and instruction count to estimate the performance loss using equation (1). This estimations are then used by the heuristic that decides the actual VF pair to be used for each core in the next control period. These VF pairs reduce energy consumption without degrading performance beyond the user set threshold. The pseudocode of the heuristic is described in Fig. 4. In each control period, *P*, the CPI and the instruction count for the current period (*CPIP*,*IP*) as well as for the next control period (*CPIP*+1,*IP*+1), as predicted by the LSTM predictor, are passed to the heuristic algorithm. The algorithm estimates the performance loss for the available frequencies listed in ascending order and selects the lowest frequency that satisfies the performance constraints and that lead to maximum energy savings.

Algorithm: Dynamic Energy Management using LSTM based predictions

Input:
γ: acceptable performance loss ratio threshold; $I_{P_{done}}$, CPI_P for just ended control period
Output:
(V_{P+1}, f_{P+1}) VF pairs for all cores for next control period
Definitions:
T each control period duration
$I_{(P+1)_{done}}$, CPI_{P+1} predicted with LSTM-based predictor
$FreqList$: list of available frequencies sorted in ascending order (f_H is the highest frequency)

$T_{delay} = 0$
$T_{ref} = 0$
if end of *control period* index P **then**
  for each *core* in CMP **do**

    $T_{ref} += T$
    $T_{delay} += \frac{I_{P_{done}} \times (\frac{f_H}{f_P} - 1) \times CPI_P}{f_H}$
    $Freq\_set = False$

    for $Freq$ in $FreqList$ **do**

      $T_{ref_{next}} = T_{ref} + T$

      $PredT_{delay} = T_{delay} +$
      $\frac{I_{(P+1)_{done}} \times (\frac{f_H}{Freq} - 1) \times CPI_{P+1}}{f_H}$

      $PredPerfLoss_{P+1} = PredT_{delay} / T_{ref_{next}}$

      if $PredPerfLoss_{P+1} < \gamma$ **then**
        $f_{P+1} = Freq$
        $Freq\_set = True$
      end if
    end for
    if $Freq\_set = False$ **then**
      $f_{P+1} = f_H$
    end if
  end for
end if

Fig. 4. Pseudocode of the DEM algorithm. This algorithm is implemented as a callable routine inside our modified sniper CMP simulator. The parameter *γ* is set by the user.

SECTION VI.

Simulation Results

A. Simulation Setup

For our simulation setup, we leverage existing simulations tools. These include the Sniper system simulator [24] that is integrated with the McPAT [25] power calculator. The machine learning library Keras [26] is integrated in our simulation framework and employed to build and train the LSTM predictor. To

speed-up the training process, we take advantage of the acceleration provided by a K20c Tesla GPU that we have available on the workstation with an eight core processor that runs Linux Ubuntu 16.04.

We conduct simulations using sixteen Parsec and Splash2x benchmarks [27] in order to investigate the performance of the DEM algorithm for two different CMP architectures with 16 and 64 cores. Each of these architectures uses a regular mesh network-on-chip (NoC) for communication. It should be noted that in our simulations we focus on the region of interest (ROI) portion of the benchmarks; the ROI contains most of the computations of a given benchmark. The architectural configuration parameters utilized in our custom Sniper based simulations are shown in Table I. The simulation framework is implemented such that Sniper is stopped periodically (1 ms intervals) and the algorithm from Fig. 4 is called as a routine that finds the VF pairs for each core for the next control period.

Table I Architectural configuration parameters.

Parameter	Value
Technology node	45nm
Core	Intel X86 Gainstown
Core CPU model	Out of order (Detailed CPU)
Frequencies(f)	2GHz downto 1GHz, with 100MHz step
VDDs	[f>=1.8G:1.2V],[1.8G>f>=1.5G:1.1V],[1.5G>f>=1G=1V]
Cores/socket	1
Transition latency	2000 ns
Branch predictor	2 bit counter
Reorder buffer	80-entries
L1ICache/1core	32KB
L1DCache/1core	64KB
L2/1core	256KB
L3/4cores	8MB
Network	2D regular mesh, 1 router per core
Link bandwidth	64 bits

B. Results

The dynamic energy management algorithm described in Fig. 3 is investigated for several different application criticality levels, indicated as the tolerable performance loss (PL) percentage. Specifically, we focus on six different PL values including 5%, 10%, 20%, 30%, 40%, and 50%. For example, if the user sets a value of *PL*=20%, it means that the objective of the DEM algorithm is to reduce the energy consumption as much as possible without degrading the performance with more than 20% compared to the case when no DEM is implemented and all cores operate at the highest clock frequency all the time.

First, we compare the DEM algorithm to the case when no DEM algorithm is used at all. Fig. 5 and Fig. 6 show the results for the energy reduction, the total performance degradation, and the energy-delay-product (EDP) on the selected benchmarks for 16 and 64 core CMP architectures. These plots show that while the DEM algorithm reduces the energy consumption, it keeps the total performance loss under the desired threshold fairly well. The energy savings increase as the tolerable performance degradation is increased suggesting that the DEM algorithm provides a good mechanism to trade off

performance versus energy consumption. However, for some benchmarks the performance degradation is slightly larger than expected. This is due for the most part to the prediction errors of the LSTM based predictor.

Most importantly, we note that the EDP for the majority of the benchmarks is improved. The EDP data points are also summarized in Table II. There are however instances where the EDP worsened. That is because we focus only on the ROI of the benchmarks where all the cores are fully utilized and busy almost all the time and thus there is little or no room to improve the performance via frequency throttling. Essentially, it is unlikely to improve both energy consumption and performance in such experimental setups because all cores are working all the time. Another interesting aspect is that beyond a PL threshold of 40% the EDP is not improved anymore as seen in Table II. In other words, the DEM algorithm can offer benefits only when the PL threshold set by the user is less than 40%; beyond that, the performance degrades too much compared with how much energy is saved.

Table II Average EDP improvement of data from fig. 5.c and fig. 6.c

PL	Avg. EDP improvement 16 core (%)	Avg. EDP improvement 64 core (%)
5%	12.02	10.66
10%	12.08	9.98
20%	13.01	6.60
30%	9.36	9.14
40%	11.92	4.29
50%	1.31	-0.02
Avg.	9.95	6.77

Next, we compare the DEM algorithm against the algorithm presented in [21], where Kalman filtering was used as the prediction technique. Fig. 7 and Fig. 8 compare the energy reduction, the total performance degradation, and the energy-delay-product. The EDP data points are also summarized in Table III. We note however that when we use the LSTM technique the results are generally slightly and not significantly better than the case when Kalman filtering is used for prediction purposes.

Table III Average EDP improvement of data from fig. 7.c and fig. 8.c

PL	Avg. EDP improvement 16 core (%)	Avg. EDP improvement 64 core (%)
5%	-0.32	2.08
10%	0.31	-0.94
20%	1.05	1.95
30%	0.34	1.26
40%	0.10	0.14
50%	0.69	1.29
Avg.	0.36	0.96

SECTION VII.

Conclusion

We investigated the effectiveness of using long short-term memory (LSTM) for workload prediction in chip multiprocessors for the purpose of constructing dynamic energy management (DEM) algorithms based on dynamic voltage and frequency scaling under performance constraints. Simulation results using several benchmarks were reported for 16 and 64 core network-on-chip based CMP architectures. These results demonstrated that although the LSTM model can be used to construct an effective DEM algorithm that provides a good mechanism to trade off performance versus energy consumption, it is only slightly better than the Kalman filtering approach for prediction. We suspect that the LSTM could be improved by exploring different model topologies with more hidden layers and units per layer; this will be investigated in our future work.

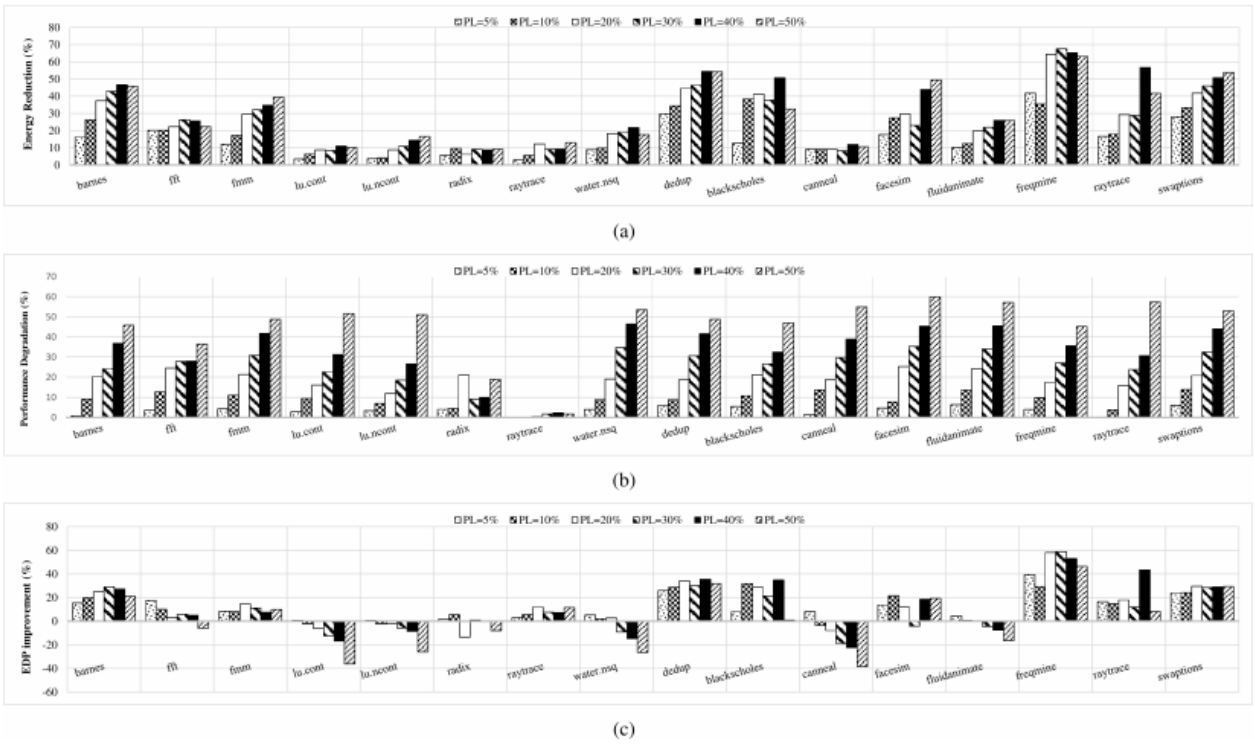

Fig. 5. Simulation results for the 16 core CMP: (a) Energy reduction percentages, (b) Performance degradation percentages, and (c) EDP improvement percentages. Comparison is done versus the case when no DEM is used.

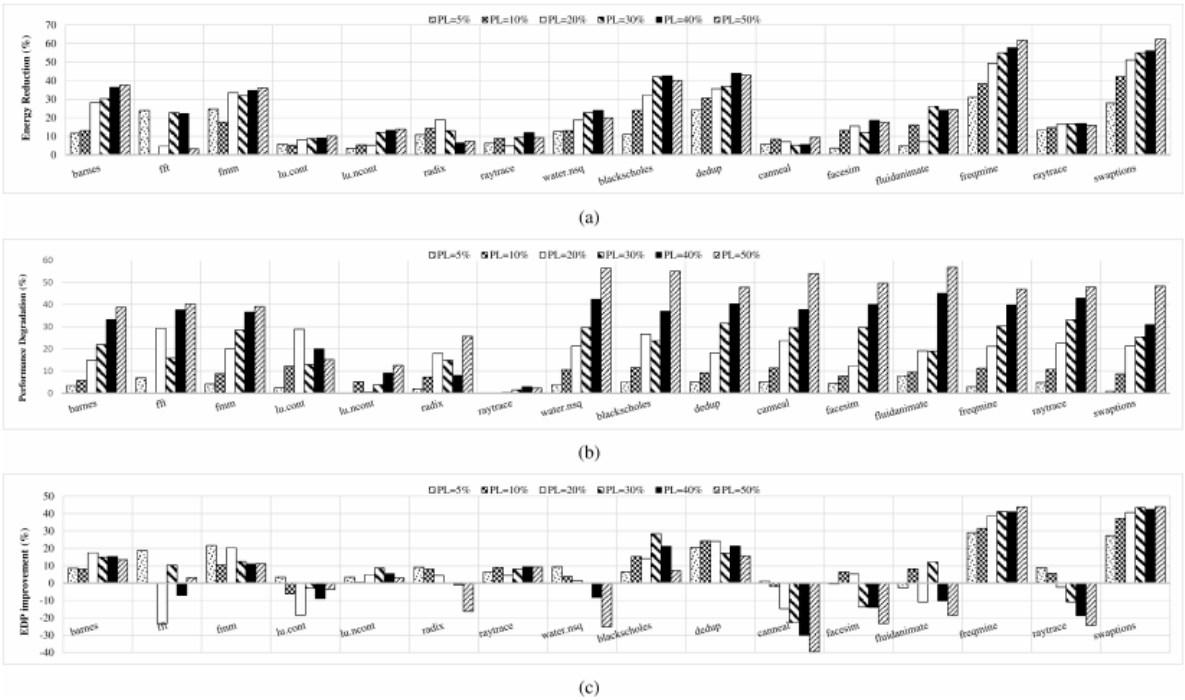

Fig. 6. Simulation results for the 64 core CMP: (a) Energy reduction percentages, (b) Performance degradation percentages, and (c) EDP improvement percentages. Comparison is done versus the case when no DEM is used.

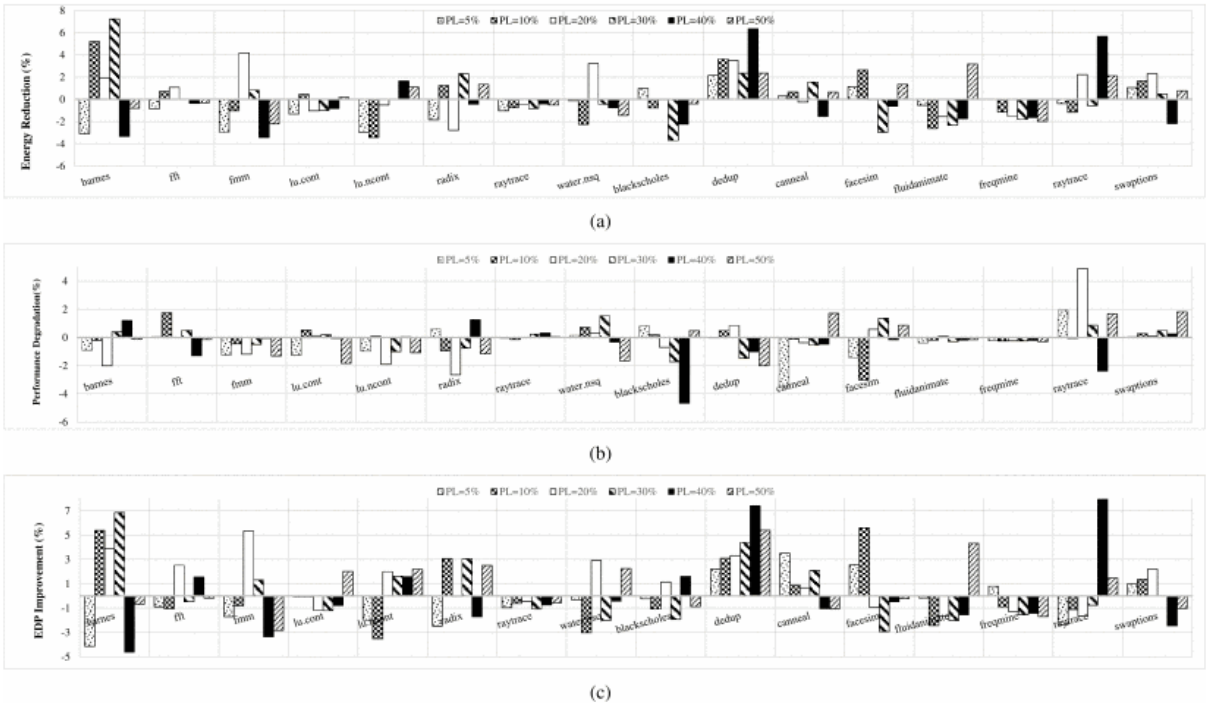

Fig. 7. Simulation results for the 16 core CMP: (a) Energy reduction percentages, (b) Performance degradation percentages, and (c) EDP improvement percentages. Comparison is done versus the DEM algorithm that uses Kalman filtering for prediction from [21].

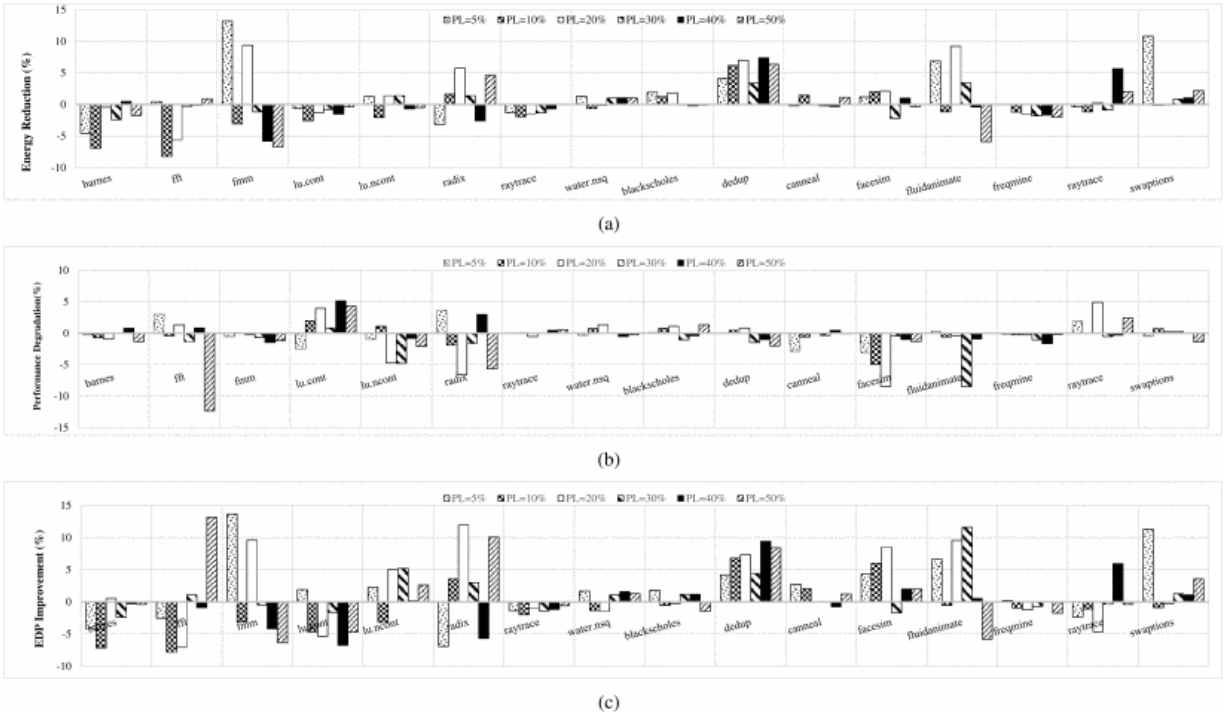

Fig. 8. Simulation results for the 64 core CMP: (a) Energy reduction percentages, (b) Performance degradation percentages, and (c) EDP improvement percentages. Comparison is done versus the DEM algorithm that uses Kalman filtering for prediction from [21].

References

1 *Report to Congress on server and data center energy efficiency*, 2007, [online] Available: https://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Fina11.pdf.

2 J. Whitney, P. Delforge, *Data center efficiency assessment - scaling up energy efficiency across the data center industry: evaluating key drivers and barriers*, 2014, [online] Available: https://www.nrdc.org/sites/default/files/data-center-efficiency-assessment-IP.pdf.

3 A.A. Sinkar, H.R. Ghasemi, M.J. Schulte, U.R. Karpuzcu, N.S. Kim, "Low-cost per-core voltage domain support for power-constrained high-performance processors", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 4, pp. 747-758, Apr. 2014.

4 R. Jevtic, H.-P. Le, M. Blagojevic, S. Bailey, K. Asanovic, E. Alon, B. Nikolic, "Per-core DVFS with switched-capacitor converters for energy efficiency in manycore processors", *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 4, pp. 723-730, May 2015.

5 R. Schone, T. Ilsche, M. Bielert, D. Molka, D. Hackenberg, "Software controlled clock modulation for energy efficiency optimization on Intel processors", *IEEE Int. Workshop on Energy Efficient Supercomputing (E2SC)*, 2016.

6 J.F. Martinez, E. Ipek, "System-level power management using online learning", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 676-689, May 2009.

7 J.F. Martinez, E. Ipek, "Dynamic multicore resource management: a machine learning approach", *IEEE Micro*, vol. 29, no. 5, pp. 8-17, Sep. 2009.

8 J.Y. Won, X. Chen, P. Gratz, J. Hu, V. Soteriou, "Up by their bootstraps: online learning in artificial neural networks for CMP uncore power management", *IEEE Int. Symposium on High-Performance Computer Architecture*, 2014.

9 H.S. Jung, M. Pedram, "Supervised learning based power management for multicore processors", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 9, pp. 1395-1408, Sep. 2010.

10 D.C. Juan, D. Marculescu, "Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors", *ACM/IEEE Int. Symposium on Low Power Electronics and Design*, 2012.

11 H. Shen, Y. Tan, J. Lu, Q. Wu, Q. Qiu, "Achieving autonomous power management using reinforcement learning", *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 2, pp. 1-32, March 2013.

12 D. Puschini, F. Clermidy, P. Benoit, G. Sassatelli, L. Torres, "Dynamic and distributed frequency assignment for energy and latency constrained MP-SoC", *ACM/IEEE Design Automation and Test in Europe Conference and Exhibition (DATE)*, 2009.

13 S. Murali, A. Mutapcic, D. Atienza, R. Gupta, S. Boyd, L. Benini, G. De Micheli, "Temperature control of high-performance multi-core platforms using convex optimization", *ACM/IEEE Design Automation and Test in Europe Conference and Exhibition (DATE)*, 2008.

14 G.Y. Pan, J. Yang, J.Y. Jou, B.C. Lai, "Scalable global power management policy based on combinatorial optimization for multiprocessors", *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 14, no. 4, pp. 70, Dec. 2015.

15 S. Sharifi, A.K. Coskun, T.S. Rosing, "Hybrid dynamic energy and thermal management in heterogeneous embedded multiprocessor SoCs", *ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2010.

16 G. Keramidas, V. Spiliopoulos, S. Kaxiras, "Interval-based models for run-time DVFS orchestration in superscalar processors", *ACM Int. Conference on Computing Frontiers*, 2010.

17 B. Rountree, D.K. Lowenthal, M. Schulz, B.R. de Supinski, "Practical performance prediction under dynamic voltage frequency scaling", *Int. Green Computing Conference and Workshops*, 2011.

18 R. Miftakhutdinov, E. Ebrahimi, Y.N. Patt, "Predicting performance impact of DVFS for realistic memory systems", *Int. Symposium on Microarchitecture (MICRO)*, 2012.

19 S. Eyerman, L. Eeckhout, "A counter architecture for online DVFS profitability estimation", *IEEE Trans. on Computers*, vol. 59, no. 11, pp. 1576-1583, March 2010.

20 J.Y. Won, P.V. Gratz, S. Shakkottai, J. Hu, "Resource sharing centric dynamic voltage and frequency scaling for CMP cores uncore and memory", *ACM Trans. on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, pp. 69, Sep. 2016.

21 M.G. Moghaddam, C. Ababei, "Dynamic energy management for chip multiprocessors under performance constraints", *Microprocessors and Microsystems*, 2017.

22 S. Hochreiter, J. Schmidhuber, "Long short-term memory", *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Nov. 1997.

23 *A beginner's guide to recurrent networks and LSTMs*, 2017, [online] Available: https://deeplearning4j.org/lstm.html.

24 T.E. Carlson, W. Heirman, L. Eeckhout, "Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation", *Int. Conf. for High Performance Computing Networking Storage and Analysis*, 2011.

25 S. Li, J.H. Ahn, R.D. Strong, J.B. Brockman, D.M. Tullsen, N.P. Jouppi, "McPAT: an integrated power area timing modeling framework for multicore and manycore architectures", *Int. Symposium on Microarchitecture (MICRO)*, 2009.

26 *Keras: the Python deep learning library*, 2017, [online] Available: https://keras.io/.

27 *PARSEC and Splash2 benchmarks*, 2017, [online] Available: http://parsec.cs.princeton.edu.