Improving the Efficiency of Power Management Techniques by Using Bayesian Classification

Hwisung Jung and Massoud Pedram

University of Southern California
Dept. of Electrical Engineering

USC

Overview

- Introduction
- Motivation
- Learning-based DPM Framework
- Stochastic Policy Optimization
- Experimental Results
- Conclusion

Introduction

- Power management becomes a first-order concern
  - More functional blocks in SOC are being built with DVFS capability
  - Different clock and voltage domains exist on the same chip
- Challenges of dynamic power management (DPM)
  - Intricate trade-off between power saving and performance loss
  - Power manager (PM) can become a heavy duty task
    - Monitors the workloads of multiple processors
    - Analyzes the information to make decisions
    - Issues DVFS commands to each processor
- DVFS-enabling techniques depend on:
  - Configuration of voltage/frequency control circuits
  - Efficiency of workload prediction mechanisms

Some Relevant Prior Work

- A. Iyer, et al. (ICCAD 2002)
  - Online DVFS technique by utilizing an interface queue
- R. Kumar, et al. (Micro 2003)
  - Analytical model for power management under performance constraints
- Q. Wu, et al. (HPCA 2005)
  - Voltage island-based power management technique
- E. Chung, et al. (ICCAD 1999)
  - Power management technique based on adaptive control mechanism
- G. Dhiman, et al. (ICCAD 2006)
  - Machine learning based power management technique

Proposed Approach

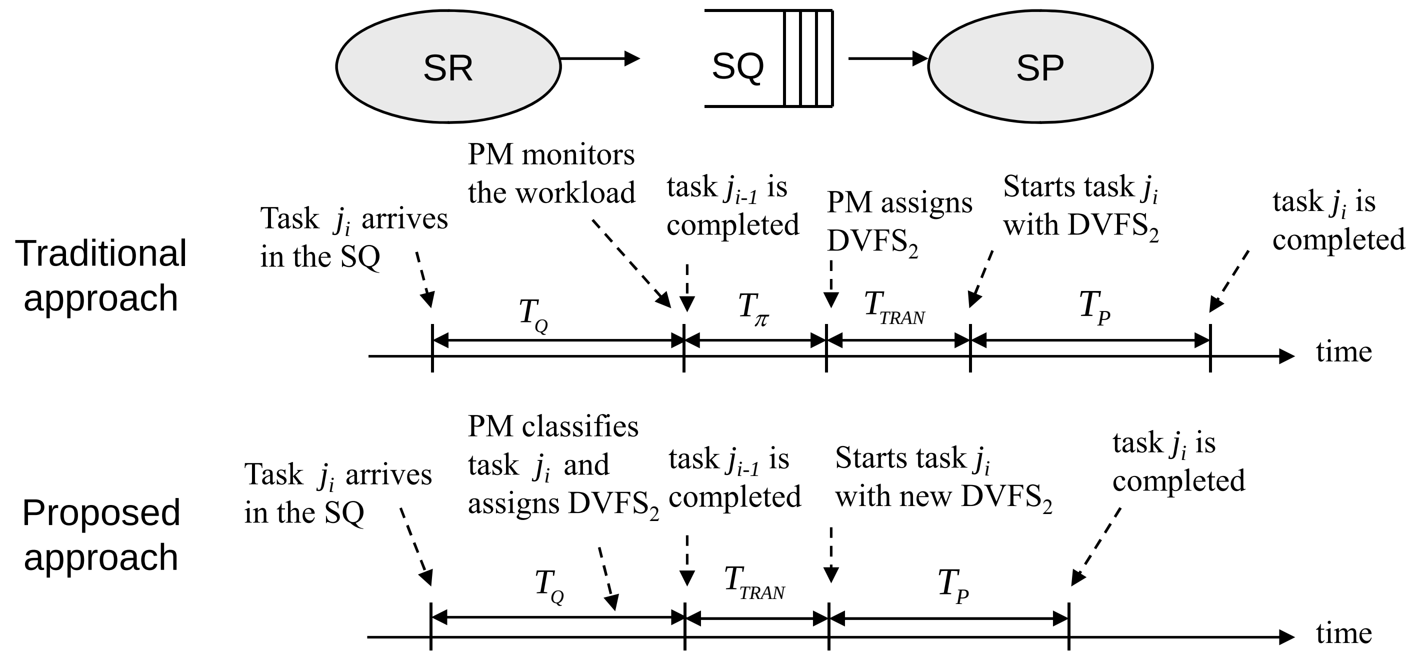
- Traditional approaches for DPM
  - Non-negligible overhead for power management policy computations
  - The PM needs to control each processor individually

- Develop a learning-based power management framework
  - Improve the decision-making strategy which minimizes the overhead of PM
    - Quickly analyze available input features
    - Accurately predict the current system state
    - Select an optimal action to minimize a user-specified cost function

- Key features of the proposed framework
  - Supervised-learning based DPM
  - Accurate predict ion of the system state

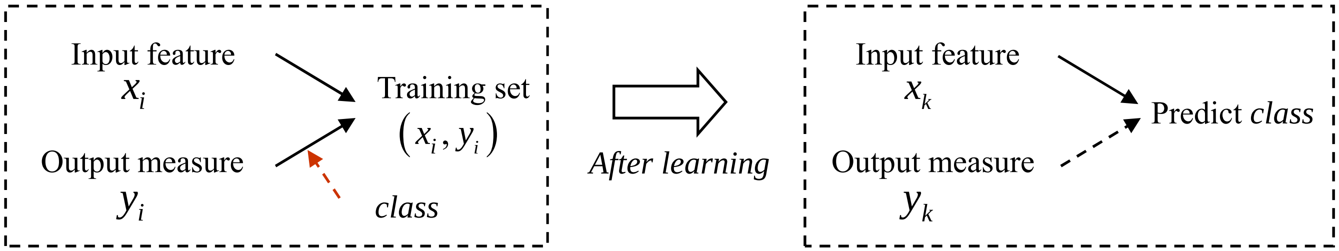
A First-Order Explanation

- Proposed DPM framework
  - Incoming tasks are labeled with the state information (power and delay)
  - DVFS values are assigned to these tasks while in the service queue (SQ), i.e., before it reaches the service provider (SP)
  - The overall task processing time is thus reduced



Learning-based DPM Framework (1/5)

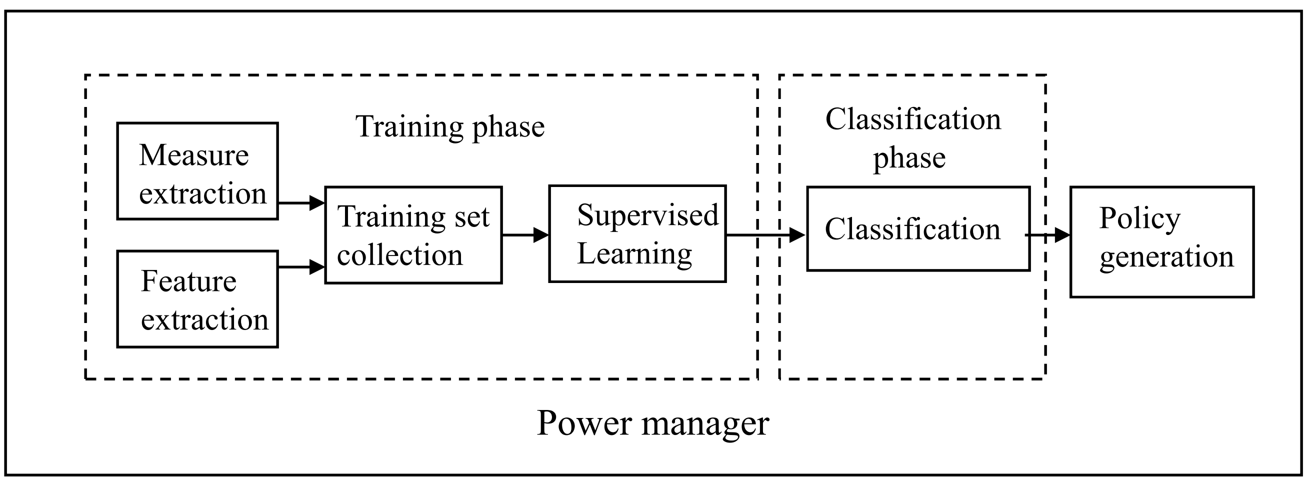
- Background on supervised learning
  - A technique for discovering relations and extracting relevant knowledge
  - Typically used to construct a *self-improving* decision-making strategy



  - *x*: input feature (quantifiable feature of the system under considerations)
  - *y*: output measure (categorical measure labeled with a pre-defined class)
  - (*x*,*y*): elements of the training set
  - Training a PM involves finding a mapping from input features to output measures
  - PM predicts the class of an output measure when a new input feature is given

Learning-based DPM Framework (2/5)

- Top-level organization of the proposed PM
  - Learning framework consists of *training* and *classification* phases
  - Classification is based on the Bayesian approach
  - The goal is to discover the relations b/w input features and output measures and to predict the performance level by using classification



Learning-based DPM Framework (3/5)

- Key functions of the proposed PM
  - **Feature extraction**: choose the input feature
    (e.g., characteristics of the tasks, and the state of the SQ)
  - **Measure extraction**: choose the output measure
    (e.g., power dissipation and execution time of the tasks)
  - **Training set generation**: assemble input-output data into a training set
  - **Supervised learning**: map the input to output based on training set
  - **Classification**: select the most likely class given the input
  - **Policy generation**: map the output classes to actions
- Our proposed DPM framework generally requires
  - Training
  - Classification
  - Policy generation

Learning-based DPM Framework (4/5)

- Input feature and output measure extraction
  - PM gathers input features, which affect performance level of the system
    - Type of tasks (e.g., high-priority or low-priority)
    - State of the SQ (e.g., queue occupancy)
    - Arrival rate of tasks (e.g., 0 < arrival rate < 1)
  - PM collects output measures
    - Power dissipation
    - Execution time of task
  - Class is considered as an attribute chosen from enumerated type sets:
    - $L_1 = \{pow_1, pow_2, pow_3\}$
    - $L_2 = \{exe_1, exe_2, exe_3\}$

Input features			Output measures	
Task type	Queue occupancy	Arrival rate of task	Power dissipation	Execution time
high-priority	med	low	pow_2	exe_1
high-priority	low	med	pow_3	exe_1
low-priority	med	low	pow_2	exe_3
low-priority	low	med	pow_1	exe_3
high-priority	low	med	pow_1	exe_1
low-priority	med	med	pow_2	exe_2
low-priority	med	med	pow_1	exe_2
low-priority	med	high	pow_2	exe_2
low-priority	med	med	pow_1	exe_1

Learning-based DPM Framework (5/5)

- Classification
  - Classification task is the assignment of Maximum A Posteriori (MAP)
  - Input feature *x* and the prior class *l* assignment are given

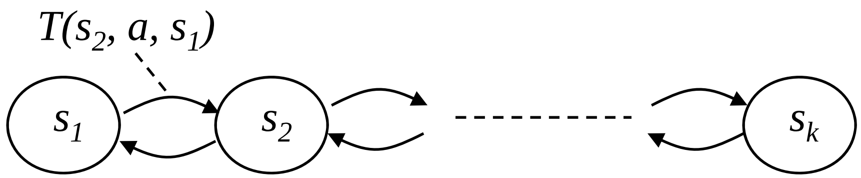
$$
\begin{aligned}
y_{MAP} &= \arg\max_{l} Prob(y_i = l \mid x_1, x_2, \ldots, x_n) \\
&= \arg\max_{l} \frac{Prob(x_1, x_2, \ldots, x_n \mid y_i = l) \cdot Prob(y_i = l)}{Prob(x_1, x_2, \ldots, x_n)} \\
&= \arg\max_{l} Prob(y_i = l) \cdot \prod_{j=1}^{n} Prob(x_j \mid y_i = l)
\end{aligned}
$$

- Example: Suppose a new input (x_1=*low*, x_2=*med*, x_3=*med*) is given, and
  - $P(y_1=pow_1)P(x_1=low,x_2=med,x_3=med \mid y_1=pow_1)$ = 1/6
  - $P(y_1=pow_2)P(x_1=low,x_2=med,x_3=med \mid y_1=pow_2)$ = 1/12
  - $P(y_1=pow_3)P(x_1=low,x_2=med,x_3=med \mid y_1=pow_3)$ = 1/144
  - Thus, MAP class of power level for the given input feature is pow_1

Stochastic Policy Optimization (1/4)

- Develop autonomous decision-making
  - Map the output classes to actions
    - Actions commanded by PM lead to quantifiable costs
  - Devise a policy for issuing a command that minimizes the expected cost

- Target processor has *k* states, each characterized by a power-delay product (PDP)
  - Each state is annotated by its PDP value
  - PM chooses an action from voltage-frequency set, $A = \{a_1, \ldots a_n\}$
  - State transition probability, $T(s', a, s) = Prob(s' \mid a, s)$



Stochastic Policy Optimization (2/4)

- Find an optimal action which minimizes the total energy dissipation
  - Assume that predicted classes for output measures are *p* and *d*
    - $p = [p_- \; p_+]$ and $d = [d_- \; d_+]$
  - The expected cost of current state, where *a* is given in state *s*

$$C(s,a) \in \left[p_- \cdot d_- + e(s,a) \quad p_+ \cdot d_+ + e(s,a) \right]$$

  - *e*(*s*,*a*): the expected energy dissipation to transit from state *s* to some next state under action *a*
  - We define a scalar cost function

$$C(s,a) = \frac{p_- \cdot d_- + p_+ \cdot d_+}{2} + e(s,a)$$

Stochastic Policy Optimization (3/4)

- Policy generation deals with the cost function
  - A dynamic programming technique is used to solve the problem since it exhibits the property of optimal substructure cost
- The optimal cost is defined as:
  - The expected discounted sum of cost that an agent accrues

$$\Psi^*(s) = \min_{\pi} E\left(\sum_{t=0}^{\infty} \gamma^t \cdot c(t) \right)$$

  - γ: a discount factor, $0 \leq \gamma < 1$
  - *c(t)*: cost at time *t*
- In our problem setup, the cost function is defined as

$$\Psi^*(s) = \min_{a} \left(C(s,a) + \gamma \sum_{s' \in S} T(s',a,s) \Psi^*(s') \right) \quad \forall s \in S$$

Stochastic Policy Optimization (4/4)

- Given the cost function, the optimal action can be obtained by

$$\pi^*(s) = \arg\min_a \left(C(s,a) + \gamma \sum_{s' \in S} T(s',a,s)\Psi^*(s') \right)$$

- One way to solve optimal decision problem is to use value iteration method
  - Value iteration method consists of a recursive update of the value function to choose an action

1: initialize $\Psi(s)$ arbitrarily
2: loop until a stopping criterion is met
3: loop for $\forall s \in S$
4: loop for $\forall a \in A$
5: $Q(s,a) = C(s,a) + \gamma \sum_{s' \in S} T(s',a,s)\Psi(s')$
6: $\Psi(s) = \min_a Q(s,a)$
7: end loop
8: end loop
9: end loop

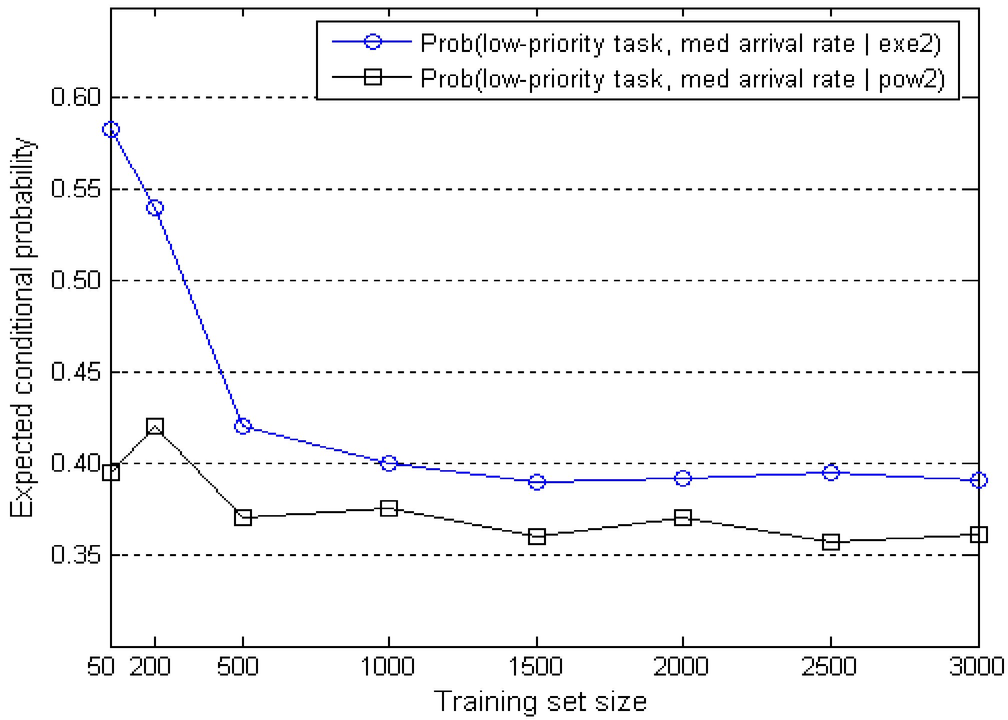
The value iteration algorithm

Experimental Results (1/4)

- The technique is applied to a multicore processor
  - Dynamic load balancing block guarantees in-order delivery
  - TCP/IP-related task (e.g., TCP segmentation and checksum offloading)
  - Power and delay measurement with TSMC 65nmLP library
  - a_1 = [1.00V, 150MHz], a_2 = [1.08V, 200MHz], a_3 = [1.20V, 250MHz]
- Define a set of input feature / output measure
  - Input measure = {occupancy state of SQ, arrival rate of task}
  - Output measure = {power dissipation [mW], execution time [nS]}
- The classes of output measure
  - pow_1 = [34 41.0], pow_2 = (41.0 47.0], pow_3 = (47.0 54.0]
  - exe_1 = [14.1 21.5], exe_2 = (21.5 28.5], exe_3 = (28.5 35.7]

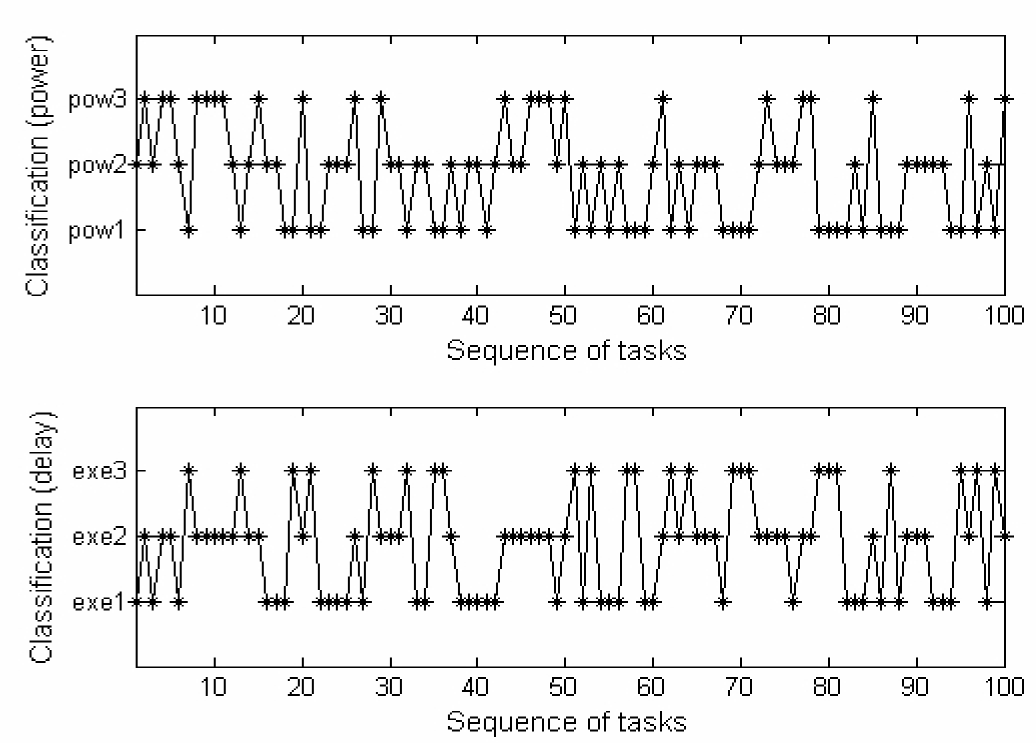
Experimental Results (2/4)

- Selection of the training set size
  - The training set size affects the classification accuracy
  - Results show that a training set size of 1000 is adequate for our purpose



Experimental Results (3/4)

- Classification of tasks
  - Randomly generates 100 tasks that arrive in the SQ of the processor
  - After predicting the system state, the PM chooses the best action to issue



Experimental Results (4/4)

- Investigate energy-efficiency of the proposed technique
  - Greedy - apply a greedy DPM strategy as follows
    - Use the lowest a_1 value for low workload (i.e., arrival rate)
    - Likewise, use a_2 and a_3 for med and high workloads
  - Bayesian - apply the optimal actions based on the proposed technique

- Generate a number of tasks (e.g., 50, 100, 150, and 200)
  - Randomly select the priority and arrival rate of tasks

Processor	Number of tasks			Total energy (normalized)		Energy saving Over Greedy
	High-pri	Low-pri	Total	Greedy	Bayesian	
Proc1	27	23	50	77.8	64.1	17.6%
Proc2	52	48	100	170.9	113.8	33.4%
Proc3	67	83	150	258.4	175.7	32.1%
Proc4	103	97	200	340.9	231.5	32.0%

Conclusion

- Addressed the problem of system-level DPM
  - Reduce the computational overhead and latency associated with regular activities of the PM
- Proposed a supervised learning based DPM framework
  - Enable the PM to predict the performance state of incoming tasks by using a Bayesian classification technique
  - Reduce the overhead of traditional decision-making strategy
- Experimental results demonstrate that the proposed technique achieves system-wide energy savings up to 28.7% (average) compared to a greedy approach