A Framework of Stochastic Power Management Using Hidden Markov Model

Ying Tan, Qinru Qiu
Department of Electrical and Computer Engineering
Binghamton University, State University of New York
Binghamton, New York 13902, USA
{ying, qqiu}@binghamton.edu

Abstract - The effectiveness of stochastic power management relies on the accurate system and workload model and effective policy optimization. Workload modeling is a machine learning procedure that finds the intrinsic pattern of the incoming tasks based on the observed workload attributes. Markov Decision Process (MDP) based model has been widely adopted for stochastic power management because it delivers provable optimal policy. Given a sequence of observed workload attributes, the hidden Markov model (HMM) of the workload is trained. If the observed workload attributes and states in the workload model do not have one-to-one correspondence, the MDP becomes a Partially Observable Markov Decision Process (POMDP). This paper presents a framework of modeling and optimization for stochastic power management using HMM and POMDP. The proposed technique discovers the HMM of the workload by maximizing the likelihood of the observed attribute sequence. The POMDP optimization is formulated and solved as a quadraticly constrained linear programming (QCLP). Compared with traditional optimization technique, which is based on value iteration, the QCLP based optimization provides superior policy by enabling stochastic control.

1. Introduction

Dynamic power management (DPM) - a mechanism that selectively shut-off or slow-down those system components that are idle or underutilized – has become a popular technique for power reduction at system level. The effectiveness of power management relies heavily on the accuracy of the workload modeling and the efficiency of policy optimization techniques.

Workload modeling is a machine learning procedure that finds the intrinsic pattern of the incoming tasks based on the observed workload attributes. In [1], the idle intervals are modeled and predicated using the exponential-average approach. In [2], a regression function is used to predict the next task incoming time. The nature of the workload of a complex computing system is random and uncertain because it is determined by user context and the sophisticated hardware/software. Stochastic methods are naturally selected for modeling and optimization of such system.

Stochastic models such as Markov decision process ([4]), Semi-Markov decision process ([5]), and Partially Observable Markov Decision process ([6], [7]) have been investigated by previous DPM research. All of these models assume that the workload is intrinsically Markovian and this embedded Markov model can be reconstructed (or trained) based on the given observation sequence.

In most of the cases, the states of the embedded Markov model and the observed workload attributes do not have one-to-one correspondence because the workload is not only controlled by the hardware and software but also affected by user and environment. For example, user working style and user mood have significant impact of the workload distribution of a computer system. Consider a power manager (PM) of a wireless adapter. The observed workload attribute is the frequency and the size of the incoming/outgoing TCP/IP packets. During a teleconference, the user may choose to send a video image, send the audio message, share files or share the image on whiteboard. Different operations generate different communication requirements. An accurate workload model of the wireless adapter must reflect how the user switches from one operation to another. However, this information is not observable by the PM. A *hidden Markov model* (HMM) is an embedded stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observations [10]. Among all the stochastic models mentioned above, only the POMDP model is capable to provide the optimal control policy for an HMM [8]. How to train the HMM based on the observed information and how to find the optimal control policy are two major challenges. While most of the previous researches focus on the second challenge, this paper addresses both of them.

The authors of [6] consider an integrated circuit under process, voltage and temperature variations as a partially observable system. POMDP is applied to search for the optimal power management policy of such system. Online and offline algorithm have been proposed. For both algorithms, it is assumed that the state space of the HMM is attained from the pre-characterized temperature-performance relation. The offline algorithm assumes the availability of the entire HMM while the online algorithm estimates the transition and observation probability of the HMM using maximum-likelihood estimation. Both algorithms utilize value iteration for policy optimization. The authors of [7] discuss several partially observable scenarios of an on-chip system and their HMM modeling. However, the discussion focuses on the calculation of the observation probability instead of the overall HMM. The optimal policy is obtained using value iteration and the controller can be implemented as a finite state automaton. Although very effective and theoretically optimal, the value iteration algorithm is limited by high memory requirement. Therefore, it can find the optimal policy only for simple systems with a small state and observation space [11]. Furthermore, the value iteration algorithm only considers the deterministic policy, which turns the device on/off with probability 1. It is impossible to constrain the size of the controller. Sometimes, a finite state automaton with hundreds of states is needed to implement the controller.

This work presents a complete framework for POMDP based power management, including novel modeling and optimization techniques. First, the HMM of the workload is trained from the observation sequences using Baum-Welch algorithm [10]. The policy optimization problem is then formulated and solved using quadraticly constrained linear programming (QCLP). The technical contribution of this work can be characterized as the followings.

1. The modeling technique iteratively utilizes the Baum-Welch algorithm to find the local optimal HMM that maximizes the likelihood of the observed sequence.
2. No prior knowledge of the state space, transition probability or observation probability is assumed during model construction.
3. To demonstrate the effectiveness of the proposed modeling technique, several real-life traces of the workload attributes of the PC hard disk system are investigated and their HMMs are trained. Compared with other modeling techniques, the average likelihood of the observed sequences under the HMM model is 65.4% higher.
4. The QCLP based formulation is able to find the optimal power management policy of large systems with hundreds of states.
5. The generated power management policy is represented as a fixed-size stochastic controller. The size of the controller is defined by user.
6. When applied to the power management of hard disk system, the proposed policy optimization technique gives better energy-performance tradeoffs than some heuristic policies and the value-iteration based policy.

The remainder of this paper is organized as follows. Section 2 gives the background of HMM and POMDP model. Section 3 discusses the model construction and the QCLP based policy optimization. Our experimental results and analysis are presented in Section 4. Finally, we conclude our work in Section 5.

2. Background

In this section, we briefly introduce some necessary backgrounds of hidden Markov models and POMDP model.

2.1. Hidden Markov models

A hidden Markov model (HMM) is an embedded stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations [10]. An HMM can be characterized as a 4-tuple: $\lambda = \{S, O, \mathbf{P}, \mathbf{B}\}$, where $S = \{s_1, s_2, \ldots, s_N\}$ denotes a finite set of states, $O = \{o_1, o_2, \ldots, o_M\}$ denotes a finite set of observations, **P** is an $N \times N$ matrix with its ijth entry denotes the state transition probability $p_{ij} = P[s_j \mid s_i]$ and **B** is an $M \times N$ matrix with its ijth entry denotes the observation probability $b_j(k) = P[o_k \mid s_j]$. Through the parameters **P** and **B**, the uncertainties in the system can be modeled.

We adopt the notation in [10] and denote an observation sequence and a model as O and λ respectively. To apply HMM to model real-world applications, three basic problems are of great interests and should be solved properly.

1. How well a given HMM model matches a given observation sequence. This can be interpreted as how to efficiently compute the probability $P(O|\lambda)$.
2. How to unveil the hidden part of the model. That is, if we have a observations sequence and a HMM, how to find the real state sequence that best "explains" the observations.
3. How to determine the optimal model $\lambda = \{S, O, \mathbf{P}, \mathbf{B}\}$ to maximize the probability $P(O|\lambda)$

In this work, we are more interested in the first the third problem. We adopt the HMM to model the process of the incoming workloads, and try to find the HMM that maximizes the likelihood of the observed sequence of workload attributes. Given several HMMs trained based on the same observation sequence, we compare their accuracy by comparing the probability $P(O|\lambda)$. Forward-backward algorithm and Baum-Welch algorithm are well established techniques to solve the above mentioned problems. For more details of these two algorithms, refer to [10].

2.2. POMDP

The Partially Observable Markov Decision Process (POMDP) combines the strength of the HMMs and the Markov Decision Process (MDP). A POMDP can be defined as a tuple $\lambda = \{S, O, \mathbf{P}, \mathbf{B}, A, R\}$, where

- S denotes a finite set of *states*.
- O denotes a finite set of *observations*.
- **P** specifies the probability of the system transition from state s to state s', given that action a is taken at state s, *i.e.* $p_{ij}^{a} = P[s_j \mid s_i, a]$.
- **B** denotes the *observation function,* i.e. $b_{ik}^{a} = P[o_k \mid s_i, a]$.
- A denotes a finite set of *actions*.
- $R(s, a)$ denotes a *reward function*. The reward could be negative if there is a cost when taking action a in the state s.

The first five parameters form an HMM whose transition and observation probability is controlled by the selected actions, while the last two parameters are special for Markov decision process.

A POMDP consists of a set of decisions over an infinite sequence of states. At each stage, an action is chosen by the control agent based on the history of observations. The objective of the agent is to maximize/minimize the expected discounted sum of rewards/costs. The discount factor, $0 \leq \gamma < 1$, is introduced to maintain finite sums over the infinite time horizon.

To avoid remembering the entire histories, finite-state controllers are used to represent POMDP policies [11]. To distinguish from the state in an HMM or POMDP, in the rest of the paper, we will refer the state of the controller as *node* and denote it as q. The node transition of the controller is based on the observation sequence and the agent determines its action based on the controller node. The action selection and controller state transition are stochastic, in order to make up for limited memory. The finite state controller can formally be defined by the tuple (Q, ψ, η), where Q is the finite set of controller nodes, ψ is the action selection model $P(a \mid q)$ that specifies the selection probability of action a for controller node q, and η is the node transition model $P(q' \mid q, a, v)$ that specifies the node transition probability from q to q' under action a and observation v.

3. System modeling and policy optimization

In this section, we propose a framework of modeling and optimization for the power management of a partially observable system. Our model is general enough to be applied to many systems and can handle large problems with hundreds of states. The optimal policy obtained by our framework is stochastic instead of deterministic [7]. To the best of our knowledge, this is the first time that the stochastic policy is used for power management in a partially observable system.

In order not to lose generality, consider a system composed of three components: service requestor (SR), service provider (SP), and service queue (SQ). The SR generates service requests that need to be treated by the SP. The service requests are first stored in the SQ, and will then be processed by the SP. The service time of the requests is a random variable. Such service requester-provider based architecture generally exists in many real world computing systems. For example, the SR may be the software applications that require reading data from or writing data into the hard disk, the SP may be the hard disk, and the SQ may be the read or write queue implemented in the OS. The power manager monitors the state of the system which is the joint state of the three components (SR, SQ and SP), and issues appropriate actions to the SP.

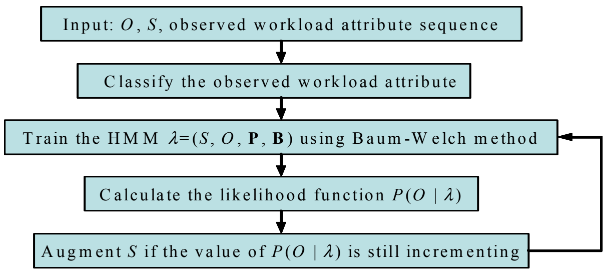
3.1. System modeling

We assume that the underlying system states are Markovian, as assumed in [4] and [12], which means the transition of the states solely depends on the current state, but has nothing to do with the history of the system states. However, the system may appear to be non-Markovian to the power manager as a result of the incomplete or noisy observations made by the power manager.

The service requester, in general, presents the highest uncertainty and is the least observable among the three system components. This is why the workload modeling is always challenging. This paper will focus on the HMM modeling of SR. The rest of the system (including SP and SQ) will be modeled in the similar way as the previous works [4].

Given an observation sequence O and a finite set of states, the transition probabilities and the observation probabilities of an HMM λ with locally maximized $P(O|\lambda)$ can be trained using Baum-Welch method [10]. In this work, we iteratively augment the state set and train the HMM model using Baum-Welch method until the likelihood function $P(O|\lambda)$ reaches the local maxima. The likelihood function can be calculated using forward-backward method [10].

Figure 1 gives the flow of our modeling algorithm for the SR. The input of this algorithm is a sequence of observed workload attributes, the user defined observation set and the initial state set. The observed workload attributes are first classified into different states in the observation set. This step usually involves information aggregation in order to reduce the complexity of the training process. After the observation sequence is generated, the HMM is trained and evaluated. If the likelihood of the observation sequence is still improving, then we augment the state set S and repeat the previous steps.



Input: O, S, observed workload attribute sequence
Classify the observed workload attribute
Train the HMM $\lambda=(S, O, \mathbf{P}, \mathbf{B})$ using Baum-Welch method
Calculate the likelihood function $P(O \mid \lambda)$
Augment S if the value of $P(O \mid \lambda)$ is still incrementing

Figure 1. SR modeling algorithm

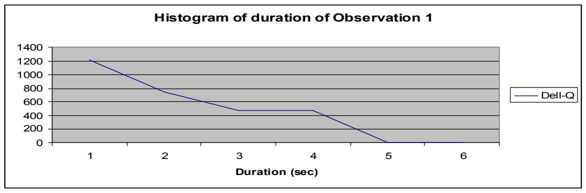
We use a power-managed hard disk as our example of system modeling. We traced three Windows XP systems for their hard disk access requests. All the data were collected using the performance monitoring facility which is built in Windows XP. We created a specific counter of our own in the monitoring tool, and logged all the data in a text file. The minimum time interval between two data reporting by the monitoring tool is 1 second. This interval is chosen to maintain the balance between the accuracy of workload information and the overhead introduced by the monitoring tool. The monitored attribute of hard disk workload is the number of transfers per second. It is the addition of the number of reads per second and the number of writes per second. Table 1 shows the characteristics of the transfer sequences on the three systems.

Table 1. Characteristics of the SR observation sequences.

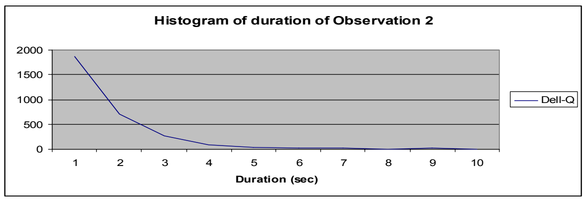
Name of sequence	Duration (in sec)	Largest # of transfers/sec	Smallest # of transfers/sec	Average # of transfers/sec
Dell-Q	10000	1033.124	1.971134	7.984103
Dell-L	8640	428.7265	0.80002	3.99056
Dell-T	10000	273.2839	1.969355	4.889311

We classify the observed transfer rates into five levels, each corresponding an observation in SR. O_1 and O_2 represent the transfers at a rate smaller than 3/sec, larger than 3/sec but smaller than 20/sec, respectively. O_3, O_4 and O_5 represent the transfers at a rate larger than 20/sec but smaller than 50/sec, larger than 50/sec but smaller than 100/sec, larger than 100/sec, respectively. The quantization is not equalized as most of the observed transfer rate is below 50/sec.

Figure 2 shows the probability distributions of the durations of O_1 and O_2. As we can see, the distribution of O_2 fits nicely to an exponential distribution, while the distribution of O_1 is totally different from the exponential distribution. The duration of the other three observations have the similar distribution as O_2. The distribution of O_1 is special because it indicates the time when the SR is almost idle, since the transfers in O_1 are at a rate below 3/sec. As a matter of fact, most of the disk transfer during this time is due to the monitoring tool itself. The non-exponential distribution of O_1 indicates that observation sequence appears to be non-Markovian to the power manager and there maybe some hidden states in the workload model.



(a) Histogram of duration of observation 1.



(b) Histogram of duration of observation 2.

Figure 2. Distributions of the durations of two observations in SR.

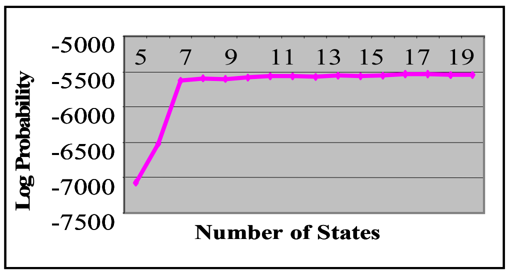


Figure 3. The size of S has impact on the likelihood of observation.

Figure 3 shows how the likelihood of the observation sequence changes when we keep on augmenting the size of the state set of the HMM. In the plot, the x-axis represent the number of states in the HMM and the y-axis represent $log[P(O|\lambda)]$. As we can see, the likelihood the observation first increase then remains stable as the size of S increases. Table 2 lists the best $P(O|\lambda)$ of the trained HMM, the $P(O|\lambda)$ of uniform selection and the $P(O|\lambda)$ of trained HMM with less states.

Table 2. Log[$P(O|\lambda)$] for trained HMM and uniform selection.

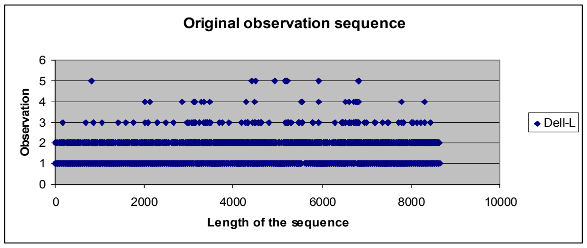
Name of sequence	Best Trained HMM	Uniform Selection	Trained HMM with less states
Dell-Q	-5534.3	-16094.4	-7062.9
Dell-L	-4398.8	-6209.3	-4504.2
Dell-T	-6738.5	-16094.4	-8300.6

Given the HMM of the SR, a synthesized workload can be generated. Figure 4 shows the synthesized observation sequence that is generated from the HMM model for Dell-L as well as the original observation sequence that is collected from the same machine. We can see from the figure that the generated sequence reflects the original sequence very well.

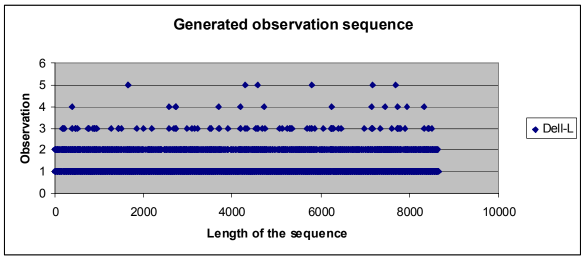
The system consists of three components, SR, SQ and SP, the system state can be represented as the composition of the states of each component. It is represented by a triplet (s, r, q) where $s\in S$, $r\in R$, and $q\in Q$. The probability to switch from state (s, r, q) to (s', r', q') under power control action a can be calculated as:

$$P^a((s,r,q),(s',r',q')) = P^a(s,s')\times P(r,r')\times P_{r,s}(q,q')$$

where $P^a(s,s')$ is the probability for SP to switch from s to s' under action a, $P(r,r')$ is the probability for SR to switch from r to r' and $P_{r,s}(q,q')$ is the probability for SQ to switch from q to q' when SR is in state r and SP is in state s. $P_{r,s}(q,q')$ depends on the SR incoming rate and the SP service rate.



(b) Original observation sequence



(b) Generated observation sequence

Figure 4. Observation sequence of SR.

3.2. Policy Optimization

The policy optimization is a constrained optimization which minimizes the power consumption with the respect of a given performance constraint. The expected latency of a request as well as the request loss rate are the most widely used performance criteria. These two criteria are not independent to each other. Given an SQ with fixed length, the average request latency can be controlled to a certain degree if the request loss rate is under control. Furthermore, a lost request will have more significant impact to the overall performance of a general purpose computing system than a delayed request. Therefore, we consider the request loss rate as the only performance constraint in this work. Note that the QCLP framework allows us to formulate the request latency as another performance constraint, however, at the cost of increased complexity.

The power management policy is represented as a stochastic controller (Q, ψ, η). To further control the complexity of the optimization problem, we assume that state set Q of the controller is fixed while the state transition function ψ and observation function η are unknown. Fixing the state set may limit the degree of freedom of the optimization. However, this will be compensated by using stochastic action at each state.

We consider power and loss rate as two cost functions of the system. For each cost, there is a value function associated with each node-state pair (q, s), where $q\in Q$ and $s\in S$. The value function gives the average discounted total cost if the system starts from state s and the controller starts from node q. It can be calculated using Bellman equation:

$$V(q,s) = \Sigma_a P(a\mid q)[R(s,a) + \gamma\Sigma_{s'}P(s'|s,a)\Sigma_o b(o\mid s',a)\Sigma_{q'}P(q'|q,a,o)V(q',s')]$$

We denote the value function of power consumption cost as $y(q,s)$. To calculate $y(q, s)$, the reward function $R(s,a)$ is defined as the power consumption of the system at state s when action a is selected and it will be denoted as $power(s, a)$. We also denote the value function of request loss rate as $v(q,s)$. To calculate $v(q,s)$, the reward function $R(s,a)$ is defined by a function $l(s)$, which is given as the following:

$l(s)=\begin{cases}1/\mu & \text{if } SQ \text{ is full}\\ 0 & \text{otherwise}\end{cases}$, where μ is the request generating rate which is determined by the state of SR.

For variables: $x(q',a,q,o), y(q,s), v(q,s)$

Minimize $\sum_s b_0(s)y(q_0,s)$

Subject to quadratic constraints:

$$y(q,s)=\sum_a\left[\begin{array}{l}\left(\sum_{q'}x(q',a,q,o)\right)power(s,a)+\\ \gamma\sum_{s'}P(s'|s,a)\sum_o O(o|s')\sum_{a'}x(q',a,q,o)y(q',s')\end{array}\right],\quad \forall q,s \qquad (1)$$

$$\forall q,s,v(q,s)=\sum_a\left[\begin{array}{l}\left(\sum_{q'}x(q',a,q,o)\right)l(s)+\\ \gamma\sum_{s'}P(s'|s,a)\sum_o O(o|s')\sum_{a'}x(q',a,q,o)v(q',s')\end{array}\right],\forall q,s \qquad (2)$$

Subject to linear constraints:

$$\sum_s b_0(s)v(q_0,s)\le V_{\lim} \qquad (3)$$

$$\forall q,o,\sum_{q',a}x(q',a,q,o)=1 \qquad (4)$$

$$\forall q,o,a,\sum_{q'}x(q',a,q,o)=\sum_{q'}x(q',a,q,o_k) \qquad (5)$$

Figure 5. QCLP formulation of policy optimization

Let $b_0(s)$ denote the initial state distribution of the POMDP, q_0 denote the initial node of the controller, and $x(q', a, q, o)$ represent $P(q', a|q, o)$. Figure 5 gives the QCLP formulation of the optimization problem. The objective is to minimize the expected value function of power consumption. In this formulation, Equation (1) and (2) specify how the value function of power consumption and request loss rate are calculated. Equation (3) specifies the performance constraint on loss rate. Equation (4) specifies that the summation of the node transition probability should be 1 while Equation (5) specifies that the action selection probability at different node is independent to the observation.

The action selection probability $P(a|q)$ can be calculated as $P(a \mid q) = \sum_{q'} x(q', a, q, o)$. While the node transition probabilities $P(q'|q, a, o)$ can be calculated using the following equation:

$$x(q', a', q, o) = P(a \mid q) P(q' \mid q, a, o).$$

It is believed that the optimization complexity of QCLP problems primarily depends on the controller size, not the size of the POMDP [11], so this QCLP algorithm can be used to solve POMDP models for large state space problems.

It is proved that an optimal solution of the QCLP results in an optimal stochastic controller for the given size and initial state distribution [11]. For this paper, we use a free nonlinearly constrained optimization solver SNOPT available on the NEOS server [13]. It implements an algorithm finding solutions by a method called sequential quadratic programming (SQP). For more information on the algorithm of SQP, please refer to reference [15]. The POMDP model and QCLP problem are described using a standard optimization language AMPL [16].

4. Experimental Results and Analysis

The effectiveness of our proposed POMDP modeling and stochastic policy optimization is evaluated in terms of energy saving, average latency and service queue overflow rate by a series of experiments.

We first consider a typical partially observable situation with three service requester states (SR), five service queue states (SQ) and two service provider states (SP). The SQ and SP states are fully observable to the power manager, while there are some hidden SR states. The hidden states are those states that are totally unobservable to the power manager. We do not have enough information to distinguish these states from each other. For a set of hidden states $H=\{h_0, h_1, \ldots, h_n\}$, there is only one observation z. The observation function is defined as: $P(z \mid h_i) = 1$, $\forall h_i \in H$. The transition of the SR states is illustrated in Figure 6. The number associate with each arrow is the probability that SR transits from the source state to the sink state.

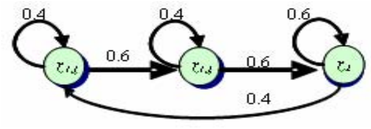


Figure 6. Illustration of the SR states of the system.

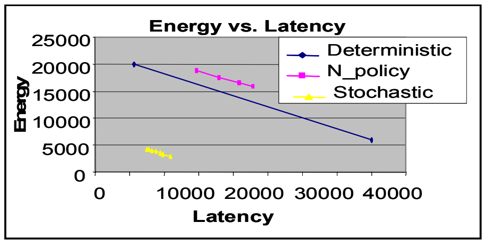
The three SR states $r_{1,1}$, $r_{1,2}$ and r_2 associate the request generating rates 0.2, 0.2 and 0.8, respectively. The state r_2 can only be reached from state $r_{1,2}$, while it cannot go back to state $r_{1,2}$ but goes to state $r_{1,1}$ only. $r_{1,1}$ and $r_{1,2}$ are two hidden states and they are observed as the same SR state by the power manager. The SP is considered as a hard disk drive (HDD) in the simulation with two power modes, *sleep* and *active*. Table 3 lists the characteristics of the HDD. P_{active} and P_{sleep} are the power consumption while the HDD is in *active* state and *sleep* state. P_{on} and P_{off} refer to the power consumed when SP transits from and to the *sleep* state. We also consider the time for transition, which are denoted as T_{on} and T_{off}, corresponding to the required time when switching from and to the *sleep* state. The service rate of SP is 0.7. The SQ can hold up to 4 waiting requests, so there are five different states of SQ including the state when there is no waiting request in the queue.

Table 3. Characteristics of the hard disk drive (HDD).

Device	P_{active}	P_{sleep}	P_{on}	P_{off}	T_{on}	T_{off}
HDD	2.0W	0.6W	2.8W	2.1W	1/0.5	1/0.9

The system is modeled as a QCLP problem and is described by a standard optimization language AMPL [16]. We use a web based solver SNOPT to solve this nonlinearly constrained optimization problem. The output of SNOPT consists of three components, the optimal value of variables $x(q', a, q, o)$, q', $q \in Q$, $a \in A$ and $o \in O$, the optimal value of the objective function and the values of other variables. Given the values of $x(q', a, q, o)$, node transition probability $P(q'|q, a, o)$ and action selection probability $P(a|q)$ of the controller can be calculated. A simulator is developed to test the effectiveness of the controller. The simulator reports the discounted total energy, latency and request loss during 10000 cycles of simulation. Our experiments show good correlation between the simulated results of these three parameters and their theoretical value function that is defined by equation (1). Therefore, in this paper, we only report the theoretical value.

In the first experiment, we compare our optimized stochastic policy with the N-policy and the deterministic policy [7]. The N-policy keeps SP in sleep state until there are N service requests waiting in the queue, and it keeps the SP in active state until the number of service requests in the queue goes back to zero. The deterministic policy is generated using a value iteration software which is available online [14]. It allows SP to either go to active state or go to sleep state with probability 1. The comparison of these three policies for the same system configuration is shown in Figure 7. By varying the number N, we get a set of different energy-performance tradeoffs for the N-policy. In our case, the service queue is only capable of 4 waiting requests, so the N varies from 1 to 4. Similarly, different energy-performance tradeoffs can be obtained by varying the performance weight of the deterministic policy as described in [7]. The weight varies from 0.1 to 1.0 with the step of 0.1. However, those 10 performance weights only give out two different determinist policies. The loss rate of the stochastic policy is constrained to be equal to that of the N-policy.



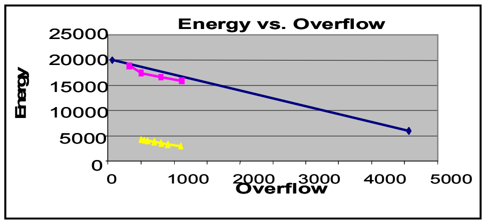
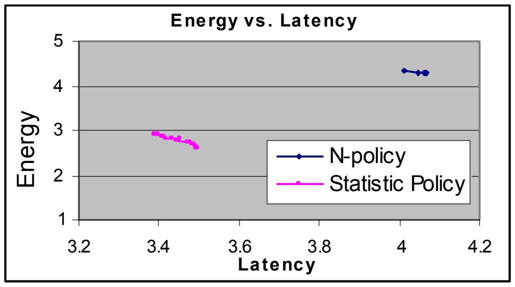


Figure 7. Comparison of three policies.

Figure 7 shows that the deterministic policy leads the SP to be either in sleep state or in active state for the most of the time, resulting in the system to either have a very large power consumption and small queue latency, or have a very small power consumption and an unacceptable large overflow. N-policy also provides sub-optimal power management because it does not consider the hidden states and the future impact of the current decision on SP action.

As shown in figure 7, the QCLP formulated policy optimization finds the stochastic controller that allows the system to precisely meet the performance requirement (i.e. the loss rate constraint) at minimum energy dissipation.

Our POMDP modeling and QCLP policy optimization can also be applied to large systems with hundreds of system states, which is not possible to be solved using value iteration based techniques. We assume a system with ten different SR states, five SQ states and two SP states, and half of the SR states are hidden states that cannot be seen by the power manager. The ten SR states represent ten different incoming request rates: {0.05, 0.1, 0.2, 0.3, ... 0.9}. The parameters of SP are the same as the previous experiments, and the values are listed in table 3. Figure 8 shows the performance of the optimal stochastic policies compared with N-policy. Clearly, stochastic policy provides better energy-performance tradeoffs.



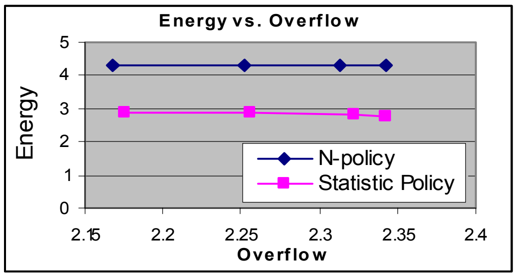


Figure 8. Performance of stochastic policy on large model.

5. Conclusions

In this paper we present a complete framework for POMDP based power management, including novel modeling and optimization techniques. To model a partially observable system, the Baum-Welch algorithm is adopted to derive the Hidden Markov Model of such system. And the policy optimization problem is solved by a Quadratically Constrained Linear Programming formulation. Experimental results shows that the average accuracy of the trained HMM model is 65.4% higher than other modeling techniques. And the statistic policy optimization achieves significant performance improvement than deterministic policy and N-policy. Our framework can also be applied to very large systems.

6. References

[1] C.H. Hwang, A.C. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-driven Computation," *Proceedings of International Conference on Computer-Aided Design*, Nov. 1997.

[2] M. Srivastava, A Chandrakasan, and R Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 4, pp. 42-55, March 1996.

[3] L. Benini, A. Bogliolo, G. De Micheli, "A survey of design techniques for system-level dynamic power management," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 8, pp. 299-316, June 2000.

[4] L. Benini, G. Paleologo, A. Bogliolo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Transactions on Computer-Aided Design*, Vol. 18, pp. 813–33, June 1999.

[5] T. Simunic, L. Benini, and G. De Micheli, "Event-driven power management," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 20, pp. 840-857, July 2001.

[6] H. Jung, M. Pedram, "Dynamic power management under uncertain information," in Proceedings of Design Automation & Test in Europe (DATE'07), Apr, 2007.

[7] Q. Qiu, Y. Tan, and Q, Wu, "Stochastic Modeling and Optimization for Robust Power Management in a Partially Observable System", Proceedings of *Design Automation and Test in Europe (DATE'07),* Apr 2007.

[8] 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, pp. 299-312, November 2006.

[9] L. Benini, A. Bogliolo, and G. De Micheli, "A Survey of Design Techniques for System-level Dynamic Power Management", *IEEE Transactions on Very Large Scale Integrated Systems*, Vol. 8, Issue 3, pp.299-316, 2000.

[10] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol.77, No.2, February 1989.

[11] C. Amato, D.S. Bernstein, and S. Zilberstein, "Solving POMDPs Using Quadratically Constrained Linear Programs", *Proceedings of International Symposium on Artificial Intelligence and Mathematics (AI&MATH'06)*, Florida, 2006.

[12] Q. Qiu, Q. Wu, M. Pedram, "Dynamic Power Management of Complex Systems Using Generalized Stochastic Petri Nets," *Proceedings of the Design Automation Conference*, June 2000.

[13] http://www-neos.mcs.anl.gov/

[14] http://www.kanungo.us/software/software.html#umdhmm

[15] P.E. Gill, W. Murray, and M. Saunders, "Snopt: An SQP algorithm for large-scale constrained optimization," *SIAM Review*, pp. 99-131, 2005.

[16] R. Fourer, D.M. Gay, and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Thomason, 2003.

[17] P. Rong and M. Pedram, "Hierarchical dynamic power management with application scheduling," *Proc. of Symp. on Low Power Electronics and Design*, Aug. 2005.

[18] T Simunic, L Benini, P Glynn and G. D. Micheli, "Event-driven power management," *IEEE Transactions on Computer-Aided Design*, Vol. 20, pp. 840-857, Jul. 2001.