Stochastic Modeling and Optimization for Robust Power Management in a Partially Observable System

Qinru Qiu, Ying Tan, Qing Wu
Department of Electrical and Computer Engineering
Binghamton University, State University of New York
Binghamton, New York 13902, USA
{qqiu, ytan3, qwu}@binghamton.edu

Abstract

As the hardware and software complexity grows, it is unlikely for the power management hardware/software to have a full observation of the entire system status. In this paper, we propose a new modeling and optimization technique based on partially observable Markov decision process (POMDP) for robust power management, which can achieve near-optimal power savings, even when only partial system information is available. Three scenarios of partial observations that may occur in an embedded system are discussed and their modeling techniques are presented. The experimental results show that, compared with power management policy derived from traditional Markov decision process model that assumes the system is fully observable, the new power management technique gives significantly better performance and energy tradeoff.

1. Introduction

Power consumption has become one of the major roadblocks in the VLSI technology. Most of the state-of-the-art system modules are capable of trading power for performance or being put into sleep or low power mode to reduce power consumption. However, the effectiveness of the power management approach is highly dependent on the correct modeling of the system architecture and the application running on it, as well as the solution techniques that lead to robust power management policies.

Dynamic power management – which refers to selective shut-off or slow-down of system components that are idle or underutilized – is a particularly effective power reduction technique at the system level. Previous approaches to DPM can be classified into three major categories: timeout-based, predictive, and stochastic. A good survey about these techniques can be found in [1]. Among those techniques, the stochastic approaches are based on a solid theoretical foundation, and are thus able to deliver provable optimal power management policies.

Three stochastic models [2]-[4] are widely used in recent research works on stochastic power management [6]-[8]. All of them are based on Markov decision process (MDP). In [2], Benini et. al model the power managed system as a discrete-time Markov decision process. Each state of the Markov process correspond to a system state, which is characterized by the number of waiting requests, the current power mode of the service provider and the current request generation mode of the service requestor. In a computer system, the service requestor usually is the user software program and the service provider can be the processor or hard disk. The power management hardware/software monitors the state transition in the system and issues control commands periodically. Reference [3] models the similar system using the continuous-time Markov decision process. The new model enables the power manager works in an asynchronous and event-driven mode, and thus reduces the performance overhead. Reference [4] proposes a modeling technique based on the time-indexed semi-Markov decision process. It improves previous works by considering more general idle time distribution.

All of the above mentioned modeling and policy optimization techniques assume that the power manager has the perfect information of the current state of the system. Based on this information, the power manager finds the best power management action from a pre-computed table stored in the memory.

As the complexity of hardware and software grows, however, the assumption that the entire system is fully observable will not be true. Firstly, the power manager may not be able to detect the request mode change of the service requestor (i.e. the application software) immediately and accurately because there is no standard way for software to pass this information to the OS. Secondly, the power manager may not be able to detect the mode change of the service provider in time because, as the size of the hardware grows, the delay to transmit information from one functional block to another becomes non-negligible. Therefore, it is possible that the power manager observes one state while the system is actually in another. Such system is called *partially observable system* because the observed view only provides partial information of the system state. Because of those hidden Markov states, the system sometime appears to be non-stationary and non-Markovian to the power manager.

A robust power management approach should be able provide good energy – performance tradeoff, even if it has only partial information of the system. However, our experimental results show that the existing stochastic power

management based on Markov decision processes cannot work robustly in a partially observable system.

The modeling and optimization of a partially observable Markov decision process (POMDP) has been well developed and widely applied in the research of Artificial Intelligence [9][10]. In this work, we use POMDP to model and optimize a power managed system. Besides the observed system state, a power manager using POMDP maintains a *belief state* during the runtime. The belief state is the power manager's estimation of the current system state based on the history information. It provides sufficient information for the power manager to make power control decision.

To the best of our knowledge, this work is the first that gives formal modeling and optimization framework for stochastic power management in a partially observable system. Compared with power management policy derived from traditional MDP model that assumes the system is fully observable, the new technique gives significantly better energy performance tradeoff. In some test cases, the new power management technique can achieve near-optimal power saving as if the system is fully observable.

The authors of [12] consider the similar problem and proposed a hierarchical power management solution for a system with partially observable service requestor. It first calculates a set of policy which only considers the state transition of the service provider and service queue. Then an algorithm is proposed to select one of the pre-calculated policies whenever the service requestor reaches an observable state. The policy will not change if the service requestor is in an unobservable state. Our approach is different in the way that the power manager estimates when it enters an unobservable state based on the belief state. Therefore, even if the power manager keeps on seeing the same observation, it may change the power control action.

The remainder of this paper is organized as follows: section 2 gives the background of POMDP model and its policy optimization technique. Section 3 discusses the model construction for a power managed system using POMDP. How to implement the POMDP based power manager is also discussed. Section 4 presents several simulation results of the new power management technique. Finally, Section 5 provides the conclusions of the work.

2. Background on POMDP

A traditional MDP can be characterized using four parameters.

- A finite state space, S
- A finite set of *actions*, A
- A *transition model*, $P(s'|s, a)$, where $s', s \in S$ and $a \in A$. It specifies the probability that the system will switch to next state *s'* given that the current system state is *s* and current action is *a*.
- A *reward function*, $r(s, a)$, where $s \in S$ and $a \in A$. It specifies the reward that the system receives when it is staying in state *s* and choosing action *a*.

A *policy* $\pi = \{< s, a > | a \in A, s \in S\}$ is the set of state-action pairs for all the states in an MDP. It specifies the actions for different states. An optimal policy is the one that gives maximum/minimum average reward/cost.

The POMDP is a generalized Markov decision process (MDP). It does not make assumption that the states are fully observable. In addition to the above four parameters, a POMDP has two more parameters:

- An *observation set*, Z. It specifies a set of states that is observable to the decision maker
- An *observation function*, $P(z | s, a)$, where $z \in Z$, $s \in S$ and $a \in A$. It specifies the *observation probability* that the system is at state *s* and action *a* is taken while the decision maker observes *z*.

In a POMDP system, the environment appears to be non-stationary and non-Markovian to the decision maker. The best policy is also not stationary with respect to the observed state. In order to choose an action, the decision maker has to refer to all the historical information that includes the initial state, the history of observed states, and the actions that have been performed. Keeping all of the information in memory will be impossible. However, it has been proved that all the useful information about the system history can be summarized by a *belief state*, which is a *sufficient statistic* [10] for the decision making.

A belief state, *b*, is a vector with |*S*| entries. The entries of the vector represent the probability distribution over all states. It is the "belief view" of the environment that the decision maker maintains during the runtime. The belief state is updated every time after the controller selects an action and makes an observation. The update uses the following equation:

$$b_z^a(s') = \frac{\sum_s P(s', z | s, a) b(s)}{\sum_s \sum_{s'} P(s', z | s, a) b(s)}, \text{ for all } s' \in S \quad (1)$$

where *b* is the current belief state, *a* is the selected action, *z* is the observed state, and $P(s', z | s, a) = P(s' | s, a) P(z | a, s')$.

Because the probability of next belief state depends only on the current belief state, the POMDP can be transformed into a belief space MDP. The belief space MDP has continuous state because the belief states are vectors in continuous domain. The transition probability that the belief space MDP switches from state *b* to *b'* can be calculated as the following equation:

$$P(b' | b, a) = \sum_{z \in Z} P(z | b, a) I(b', b_z^a), \quad (2)$$

where

$$I(b', b_z^a) = \begin{cases} 1 & \text{if } b' = b_z^a \\ 0 & \text{otherwise} \end{cases} \text{ and}$$

$$P(z | b, a) = \sum_s \sum_{s'} b(s) P(s' | s, a) P(z | s', a).$$

In other words, the probability of a belief state is the summation of the probabilities of all the observations that would lead to this belief state.

The reward function $r(b, a)$ of the belief space MDP is the expectation of $r(s, a)$ over all the states. It can be calculated as the following equation:

$$r(b, a) = \sum_{s \in S} b(s) r(s, a) \quad (3)$$

With the help of the belief state, the problem of policy optimization for a POMDP is transformed into the problem of policy optimization for a continuous-state MDP. The later can be solved using the v*alue iteration*. In the next, we will give a brief introduction of value iteration.

Given a policy π, a *value function* $V^{\pi}(b)$ is the discounted expected reward that the system receives if it starts from a belief state b. It is defined as

$$V^{\pi}(b) = E_{b,\pi}\left[\sum_{n=1}^{\infty}\lambda^{n-1} r(b_n, \pi(b_n))\right],$$

Where λ is the discount factor which is less than 1 and $\pi(b_n)$ gives the action for state b_n under policy π. The value function for the optimal policy can be obtained numerically by iterating functions (4)~(6).

$$V_{n+1}^{a,z}(b) = \lambda P(z \mid b,a) V_n(b_z^a) \quad (4)$$

$$V_{n+1}^{a}(b) = r(b,a) + \sum_z V_{n+1}^{a,z}(b) \quad (5)$$

$$V_{n+1}(b) = \max_a V_{n+1}^{a}(b) \quad (6)$$

It can be proved that $V_n(b)$ converges to the optimal value function [9]. The following theorem gives the stopping condition for the iteration [9]. It also shows how to construct the policy from the value functions.

Theorem 1 Let π be the policy given by

$$\pi(b) = arg\max_a \left\{ r(b,a) + \lambda \sum_z \lambda P(z \mid b,a) V_n\left(b_z^a\right) \right\}, \quad (7)$$

if $\max_b \left| V_n(b) - V_{n-1}(b) \right| \le \eta$, then $\max_b \left| V^{\pi}(b) - V^{*}(b) \right| \le \frac{2\eta\lambda}{1-\lambda}$, where $V^{*}(b)$ is the value function of the optimal policy.

To calculate the value of functions (4)~(6) is difficult because b is a variable in a continuous space with $|S|$ dimensions. However, a nice property of the V_{n+1}^{a} and $V_{n+1}^{a,z}$ is that they are piece-wise-linear. Each of these functions is a convex surface formed by a set of hyperplanes in the $|S|$ dimension space. Therefore, each of these functions can be represented by a set of vectors that characterize the hyperplanes. The operations on V_{n+1}^{a} and $V_{n+1}^{a,z}$ can be transformed to the operations on those hyperplanes. Different algorithms have been developed for value iteration. For more detailed information please refer to reference [10].

3. System modeling and policy implementation

The proposed modeling and optimization method can be applied to more complex system. However, in this paper, we will focus on the modeling of the power managed system with a single service provider.

We adopt the system configuration in [2] and [3] and model the system as a composition of three components: Service Requestor (SR), Service Provider (SP), and Service Queue (SQ). The SR generates service requests for the SP. The SQ buffers the service requests. The SP provides service to the requests in a first-in-first-serve manner. In a real system, the SR may be a software application, the SP may be the processor, and the SQ may be the ready queue that is implemented in OS. The power manager monitors the states of the three components and issues state-transition commands to the SP.

Similar to [2] and [3], we assume that the power managed system is Markovian, i.e. the next state of the system depends only on its current state. The history information does not impact the behavior of the system. Note that the system may appear to be non-Markovian to the power manager or user because some states are not observable. If the power manager does not have the complete information of the system, then the system needs to be modeled as a POMDP. To build the POMDP model, we first need to construct its embedded MDP, then to characterize the observation set and the observation function.

3.1 Embedded MDP model

The embedded MDP model of the power managed system is constructed in the similar way as reference [2]. Here we use the discrete-time model because, to maintain the belief state, the power manager needs to observe the system periodically. The SR is modeled as an MDP $\mathcal{R}$ with state set $R=\{r_i, \text{ s.t. } i=0, 1, \ldots, R\}$. Different states associate with different request generating modes which generate service request at different rates. The state transition probability can be obtained by software profiling. The SP is also modeled as an MDP $\mathcal{S}$ with state set $S=\{s_i, \text{ s.t. } i=0, 1, \ldots, S\}$. Different states associate with different power modes. The state transition probability is determined by the power control actions and the power mode switching time. The SQ is also modeled as an MDP $\mathcal{Q}$ with state set $Q=\{q_i, \text{ s.t. } i=0, 1, \ldots, Q\}$. State q_i indicates that there are i requests in the service queue. The state transition probability is determined by the request incoming rate and the request service rate. The system state is the composition of SR, SQ and SP. The system state is 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.

3.2 Observation set and observation functions

In this paper, we assume that the power control action taken by the power manager does not affect the observation of the system state. Therefore, the observation function $P(z \mid s,a)$ can be reduced to $P(z \mid s)$.

We can classify the type of partial observation of the power manager into three classes: hidden states, delayed observation, and noisy observation.

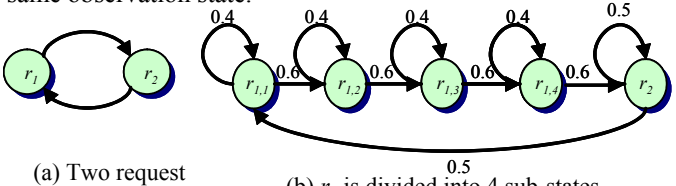
3.2.1 Partial observation due to hidden states

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: $P(z \mid h_i) = 1$, $\forall h_i \in H$.

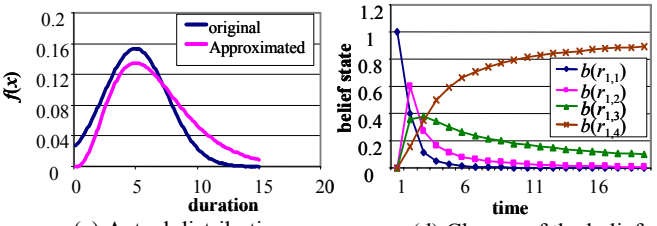
The hidden states may be different request generation modes of the SR. The previous stochastic power management framework assumes that the power manager is able to detect which mode the SR is currently in and then make a decision correspondingly. This is possible if we modify the program and embed some instructions to inform the power manager about the current requester mode. However, this is not the way that current software was developed. The processor may be able to observe certain request modes by using side channel information, such as the context switching from one process to another. It is not able to differentiate the request mode within one process.

The hidden states sometimes are added to facilitate the construction of the embedded MDP. For example, in reference [5], a "stage method" is proposed to approximate the non-exponential service or inter-arrival time. Based on the "stage method", any state that has random duration with non-exponential distribution can be decomposed into a set of parallel/serial connected sub-states whose duration follows exponential distribution. However, to the power manager, these sub-states appear to be the same. The traditional stochastic power management cannot perform robustly in this situation. However, if the system is modeled as a POMDP, with the help of the belief state, the power manager can estimate which hidden state the system is currently in. The following example shows how the power manager changes the belief state even though it keeps on seeing the same observation state.



(a) Two request states in SR.

(b) r_1 is divided into 4 sub-states.



(c) Actual distribution vs. approximated distribution.

(d) Change of the belief state along the time.

Figure 1 Detection of hidden state using POMDP.

Example 1: Assume that an SR has two request modes, r_1 and r_2. The time that it stays in r_1 follows normal distribution with mean 5 and variance 2.6. The time that it stays in r_2 follows exponential distribution with mean 2. To model the normal-distributed duration, we divide r_1 into 4 sub-states: $r_{1,1}$~$r_{1,4}$. The duration of each sub-state follows exponential distribution with mean 1/0.6. Figure 1 (a) and (b) give the original state transition diagram and the transformed state transition diagram of the SR respectively. The four stages $r_{1,1}$~$r_{1,4}$ are hidden states. They share one observation r_1. The observation function is $P(z = r_1 \mid r_{1,i}) = 1$, i=1~4. The transformed model provides a fairly close approximation of the normal distribution. Figure 1 (c) shows the comparison of the probability density function of the time that the SR stays in r_1 and time that the SR stays in either one of $r_{1,1}$~$r_{1,4}$. The belief state is a 1×5 vector $(b(r_{1,1}), b(r_{1,2}), b(r_{1,3}), b(r_{1,4}), b(r_2))$. The ith element of the vector is the probability that the system is in state i according to the power manager's estimation. Assume that the system start from state $r_{1,1}$. Therefore, the starting belief state is $(1,0,0,0,0)$. Figure 1 (d) shows the changing of the belief state along the time when the power manager keeps on observing state r_1. As we can see, the probability $b(r_{1,4})$ increases while $b(r_{1,1})$ decreases. Therefore, although the power manager sees the same observation for all the time, it "believes" more and more that the SR is currently in state $r_{1,4}$ instead of $r_{1,1}$ as the time goes by. There will be different power control actions associated with different sub-states. Obviously, the power manager that maintains a belief state has more information on how to select those actions while the tradition power manager does not.

3.2.2 Partial observation due to delayed observation

Sometime, because of bus contention or system busy, the power manager is not able to obtain the accurate system status information in time. More specifically, after the system state change from x_1 to mode x_2, it still appears to be x_1 to the power manager for a short time. To model this situation in POMDP, for each (x_1, x_2) pair, we divide the state x_2 into two sub-states: $x_{2,1}$ and $x_{2,2}$. The state $x_{2,1}$ is always observed as x_1 while $x_{2,2}$ is always observed as x_2. Hence, the observation function is $P(z = x_1 \mid x_{2,1}) = 1$ and $P(z = x_2 \mid x_{2,2}) = 1$.

Let $P(b \mid a)$ and $P'(b \mid a)$ denote the transition probability from state a to state b of the original model and the POMDP model respectively. The following set of heuristic rules can be used to update the state transition probabilities for the new model:

$$P'(x_{2,1} \mid x_1) = P(x_2 \mid x_1);$$
$$P'(x_{2,2} \mid x_1) = 0;$$
$$P'(x_{2,2} \mid x_{2,1}) = (1 - P'(x_{2,1} \mid x_{2,1})) \cdot P(x_2 \mid x_2);$$
$$P'(x \mid x_{2,1}) = (1 - P'(x_{2,1} \mid x_{2,1})) \cdot P(x \mid x_2);$$
$$P'(x_{2,2} \mid x_{2,2}) = P(x_2 \mid x_2);$$
$$P'(x \mid x_{2,2}) = P(x \mid x_2);$$

where x is all the next state of x_2, $P'(x_{1,2} \mid x_{1,2})$ is $(1 - 1/d)$ if the average observation delay is d.

Example: Consider the SR model in Figure 2 (a). Assume that there is always a delay for the power manager to detect the transition from r_1 to r_2. The average delay is 1.25 time steps. The corresponding POMDP model is given in Figure 2. In order to verify if the POMDP model keeps the characters of the original model, we compare the cumulative distribution function of the duration of r_2 in the original model and the duration of $r_{2,1}$ and $r_{2,2}$ in the POMDP model.

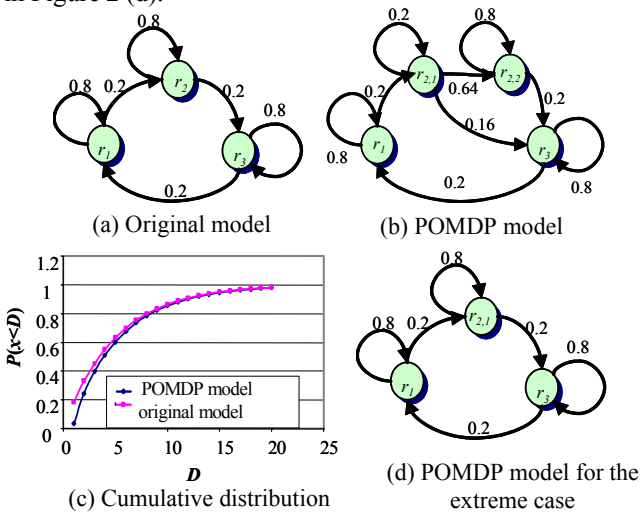
The duration of r_2 in the original model is a random variable with exponential distribution with mean 1/0.2. Hence its cumulative distribution function is $F_{r_2}(t) = 1 - e^{-0.2t}$.

The duration of $r_{2,1}$ and $r_{2,2}$ are both exponential distributions with mean 1/0.8 and 1/0.2 respectively. Denote their probability density function as $f_{r_{2,1}}(t)$ and $f_{r_{2,2}}(t)$. Also denote their cumulative distribution function as $F_{r_{2,1}}(t)$ and $F_{r_{2,2}}(t)$. The cumulative distribution function of the time that the system stays in either $r_{2,1}$ and $r_{2,2}$ can be calculated as

$$F(t) = 0.8 \cdot \int_0^\infty f_{r_{2,1}}(x) F_{r_{2,2}}(t-x)dx + 0.2 \cdot F_{2,1}(t)$$

$$= 0.2(1 - e^{0.8t}) + 0.8(1 - 1.33e^{-0.2t})$$

Figure 2 (c) shows the comparison of the cumulative distribution function of the time that the SR stays in r_2 and the time that the SR stays in either $r_{2,1}$ or $r_{2,2}$. As we can these two random variables follows very similar distribution. For the extreme case when the information of mode switching is lost indefinitely, the r_2 will become a hidden state. It will always be observed as r_1. The state transition diagram for the POMDP model of the extreme case is given in Figure 2 (d).



(a) Original model (b) POMDP model

(c) Cumulative distribution (d) POMDP model for the extreme case

Figure 2 POMDP model of a delayed observation.

3.2.2 Partial observation due to observation noise

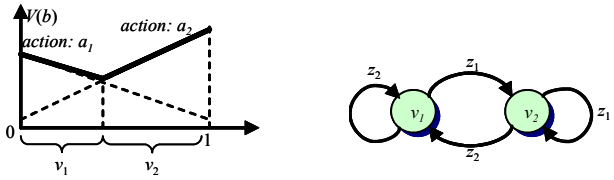
The above mentioned two types of partial observation are the most common ones in an embedded system. There will be many other reasons that a power manager will obtain inaccurate information of the system. We will not further classify them. Instead, we assume that they are all caused by random noise during the observation. For this type of partial observation, if state x has $p_{x,y}$ probability to be observed as state y, then we will have the observation function as $P(z = y \mid x) = p_{x,y}$. Note that $\sum_{y \in X} P(z = y \mid x) = 1$, where X is the entire state set of the system.

3.2 Implmentation of POMDP power manager

Given the POMDP model, the optimal policy can be calculated using value iteration as discussed in Section 2. The result of the value iteration is a value function of the belief state. In each time step, the power manager first updates the belief state using equation (1), and then calculates the best action using equation (7). Complex computing is involved in this procedure which increases the overhead of the power manager.

It has been shown [13] that the believe space can be partitioned into regions such that the same action will be chosen for all belief states within this region. Furthermore, given the optimal action and the resulting observation, all belief states in one partition transform to new belief states that are in the same partition. Based on the partition, a *policy graph* can be constructed. Each vertex in the policy graph associates with a partition and each edge associates with an observation. If there is the edge from vertex x to vertex y and it is associated with observation z, then all belief state in partition x transform to new belief states in partition y if z is observed.

Example: Figure 3 (a) shows the value function of a two state POMDP. It is the convex surface that is formed by two dashed lines. The surface can be partitioned into two regions, v_1 and v_2. The corresponding best actions for the two regions are a_1 and a_2 respectively. If the system is in region v_1 and a z_1 is observed, then it will switch to v_2. If the system is in region v_2 and a z_2 is observed, then it will switch to v_1. The corresponding policy graph is given in Figure 3 (b). The best action can be determined based on the policy graph and the observation.



(a) Value function of a two-state POMDP (b) Policy graph

Figure 3 Example of policy graph.

The policy graph can be implemented as a finite state controller. In this way, the implementation and computation complexity of the power manager is reduced significantly.

4. Experimental results

A set of simulations has been performed to evaluate the performance and energy saving of the POMDP based power management. Three scenarios are compared:

1. The system is fully observable. The policy is derived from an MDP model.
2. The system is partially observable. However, the power management policy is derived from an MDP model which assumes that the system is fully observable.
3. The system is partially observable. The power management policy is derived from a POMDP model.

The above mentioned three scenarios will be denoted as "MDP+full", "MDP+partial" and "POMDP" in the rest of the paper.

The service provider that is considered in the simulation is a hard disk drive (HDD) with two power modes, sleep and active. The power consumption is 2.0W, 0.6W, 1.2W and 1.2W when the HDD is active, sleeping, switching from active to sleep, and switching from sleep to active. When the SP is active, the service rate is 0.7. The queue can hold up to 4 waiting requests. The SR has three states, r_1, r_2, and r_3 and the corresponding request generation rates are 0.1, 0.3 and 0.8 respectively. The transition probabilities among the three request modes are given in the following matrix: $P = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.3 & 0.3 & 0.4 \end{bmatrix}$. The request mode r_3 is not observable. Furthermore, it will have equal probability to be observed as r_1 or r_2.

We use an open source POMDP solver [14] for POMDP policy optimization. The optimal MDP policy is obtained using the same software by setting the observation set equal to the system state. In all the experiments, we consider only the deterministic policy because it is the only type of policy that is supported by the value iteration algorithm. The reward is defined as the weighted sum of the average power consumption and the average latency of each request. The discounted factor is set to be 0.95.

We vary the weight of the latency and obtain a set of policies with different energy latency tradeoffs. Figure 4 gives the comparison of the energy-latency tradeoff curves of the three scenarios. As we can see, the POMDP policy gives near optimal energy-delay tradeoff as if the system is fully observable. It out performs the MDP policy in a partially observable system.

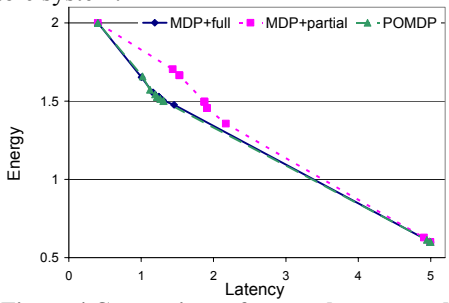


Figure 4 Comparison of energy latency tradeoff

Table 1 gives the comparison of the average latency (D), loss rate (L) and power (P). The first column gives the latency weight in the reward function. Columns 2~4 give the percentage increasing of latency, loss rate and power of POMDP policy comparing with MDP policy under full observation. Columns 5~7 give the same information of MDP policy under partial observation comparing with MDP policy under full observation. When the weight of latency is low, both POMDP and MDP policy choose to power on the HDD all the time. Therefore, these two policies work equally well. For the rest of the cases, comparing with fully observable system, the POMDP policy either trades power for performance or vise versa. The MDP policy does not work robustly in a partially observable system. For two of those cases, it leads to lower performance and higher energy. Note that the optimal MDP policies are the same when the latency weight equals to 0.6 and 0.7. Therefore the data of MDP+partial are exactly the same for these two cases.

Table 1 Comparison of different power management policy.

weight	POMDP			MDP+partial		
	ΔD (%)	ΔL(%)	ΔP(%)	ΔD(%)	ΔL(%)	ΔP(%)
0.1	0	0	0	0	0	0
0.2	0	0	0	0	0	0
0.3	0	0	0	0	0	0
0.4	−0.8	−1.1	1.8	−1.6	−2.7	4.2
0.5	0.7	4.5	−0.3	47.0	463.6	−3.3
0.6	−1.9	16.7	−0.2	50.9	561.1	−2.1
0.7	−4.0	11.1	0.9	50.9	561.1	−2.1
0.8	−12.5	−31.3	6.6	31.4	581.3	7.1
0.9	−60.0	−90.0	21.0	42.8	920	3.1

5. Conclusions

In this paper, we propose a new modeling and optimization technique based on partially observable Markov decision process (POMDP) for robust power management. Three scenarios of partial observations which may occur in an embedded system are discussed and their modeling techniques are presented. The simulation results show that, the POMDP policy provides near optimal energy performance tradeoff similar as the MDP policy in a fully observable system.

6. References

[1] 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.

[2] 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.

[3] Q. Qiu, Q Wu and M. Pedram, "Stochastic modeling of a power-managed system-construction and optimization," *IEEE Transactions on Computer-Aided Design*, Vol. 20, pp. 1200-1217, October 2001.

[4] 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.

[5] 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.

[6] P. Rong and M. Pedram, "Determining the optimal timeout values for a power-managed system based on the theory of Markovian processes: Offline and online algorithms," *Proc. of Design Automation and Test in Europe*, Mar. 2006.

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

[8] T. Simunic, S.P. Boyd, P. Glynn, "Managing power consumption in networks on chips," *IEEE Transactions on Very Large Scale Integration Systems*, Vol 12, Issue 1, pp. 96-107, January 2004.

[9] W. Zhang, "Algorithms for Partially Observable Markov Decision Processes," *Ph.D thesis*, 2001.

[10] A.R. Cassandra, "Exact and Approximate Algorithms for Partially Observable Markov Decision Processes," *Ph.D thesis*, 1994.

[11] E.Y. Chung, L. Benini, A. Bogliolo, Y.H. Lu; G. De Micheli, "Dynamic power management for nonstationary service requests," *IEEE Transactions on Computers*, Vol. 51, Issue 11, pp. 1345 – 1361, November 2002.

[12] Z. Ren, B.H. Krogh, R. Marculescu, "Hierarchical adaptive dynamic power management," *IEEE Transactions on Computers*, Vol. 54, Issue 4, pp. 409-420, April 2005.

[13] D. Braziunas, "POMDP Solution Methods," *technical report, Department of Computer Science, University of Toronto,* 2003.

[14] http://pomdp.org/pomdp/code/index.shtml