Hierarchical Adaptive Dynamic Power Management

Zhiyuan Ren, Bruce H. Krogh, and Radu Marculescu
Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
{rzy|krogh|radum}@cmu.edu

Abstract

The main contribution of this paper is a novel hierarchical scheme for adaptive dynamic power management (DPM) under nonstationary service requests. We model the nonstationary arrival process of service requests as a Markov-modulated stochastic process in which the stochastic process for each modulation state models a particular stationary mode *of the arrival process. The bottom layer of our hierarchical architecture is a set of stationary optimal DPM policies, precalculated off-line for selected modes from policy optimization in Markov decision processes. The supervisory power manager at the top layer adaptively and optimally switches among these stationary policies on-line to accommodate the actual mode-switching arrival dynamics. Simulation results show that our approach, under highly nonstationary requests, can lead to significant power savings compared to previously proposed heuristic approaches.*

Keywords: *low-power design, hierarchical adaptive dynamic power management, nonstationary service requests.*

1. Introduction

Dynamic power management (DPM) has emerged as an effective means of saving power and enabling the use of portable devices (laptops, PDAs, etc.). At the very heart of this approach, lies the ability of exploiting the variability in the operation conditions that portable devices normally have to deal with. Indeed, many computing devices offer multiple operating states with different levels of power consumption and performance. DPM saves power by selectively switching a device into lower power states when there is no activity to reduce power consumption without severely affecting the overall performance of the device [3][8].

The framework illustrated in Fig. 1 is widely used (e.g., see [2]) to study the DPM optimization problem. Service requests arrive at the device according to a probabilistic model (that is, the probability distribution of the inter-arrival times) and are serviced by the device with another probabilistic model (the probability distribution of the service times). As shown in Fig. 1, the device serves one request at a time, and a buffer

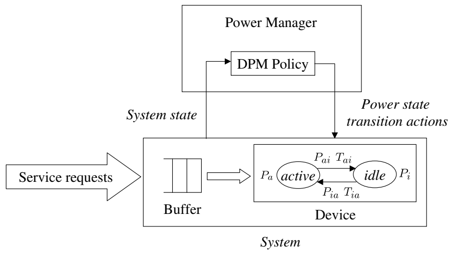
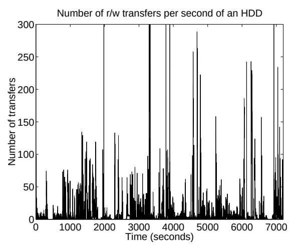


Figure 1. DPM of a device with two power states

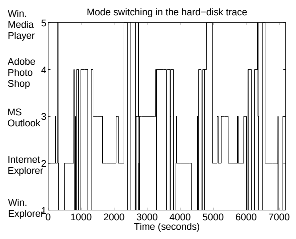
holds the service requests that cannot be serviced immediately. The power manager can command the device transit into different power states. Now, depending on the actual DPM policy, the power state transitions take random amounts of time that follow some probability distributions. If these arrival, service and transition processes are *stationary* random processes, then the DPM problem can be formulated as a Markov decision process (MDP) [1, 7, 10]. For this problem, a stationary optimal DPM policy can be computed to minimize the power consumption under performance (i.e., waiting time and loss probability) constraints.

In most cases, however, the service request arrival process is *not* stationary. For example, the service requests to a hard disk drive (HDD) can vary significantly, as the pattern of application accessing the disk changes. Figure 2(a) shows a two-hour long trace of service requests in which five applications (*Windows Explorer*, *Internet Explorer*, *Microsoft Outlook*, *Adobe Photoshop* and *Windows Media Player*), in a Windows 2000 environment, access the HDD alternatively for read/write operations (transfers). As we can see, the statistics of the arrival times of the transfers vary dramatically among these applications. As it turns out, this wide variation gives us ample opportunities for power savings.

The contribution of this paper is a novel hierarchical scheme for adaptive DPM under nonstationary service requests. We model the nonstationary arrival process of service requests as a *Markov-modulated stochastic process*, a *multi-mode* model in which each mode corresponds to a given workload (a stationary arrival process), while mode transitions are



(a)



(b)

Figure 2. A two-hour HDD trace: a) read/write per second b) application (mode) switching

modeled as a discrete-time Markov chain. From the multi-mode model of the service request arrival process, the two-level hierarchical scheme for DPM proposed in this paper is as follows. The *lower level* comprises a set of pre-computed stationary optimal DPM policies: each (randomized) policy selects the device operating state according to the current system state. The *upper level* selects the DPM policy to be applied at any time according to the current system state and the current mode of the arrival process (if the mode is known). The big advantages of this hierarchical DPM algorithm is its adaptiveness to changes in the pattern of the service requests and the easiness in practical implementation (as part of the OS, for instance) since it does *not* assume any prior knowledge of the parameters in the Markov-modulated model.

A previous approach of DPM for nonstationary requests is proposed in [4], in which the power manager directly estimates the arrival statistics and calculates the DPM policy by interpolating a set of pre-characterized policies designed for a set of statistical values. The approach in [4] is similar to our approach in that it uses pre-calculated policies. The fundamental difference between the approach in [4] and ours is that the authors in [4] rely on "identifying the mode and then apply the stationary optimal policy corresponding to the mode if there is one available" (this is called *mode-matching*). In contrast, our approach enables a more complicated policy switching. More precisely, mode-matching is optimal only when the mode evolution process is *slow* enough because the stationary optimal policies are designed under the assumption that the mode remains constant. As we will see later in the paper, when the mode switching dynamics is taken into consideration, it may be better to use one of the policies designed for a different mode. This is the intuition behind the power savings that we observe compared to the approach in [4].

To fully validate our approach, we provide simulations of DPM in HDDs for which the service request arrivals are modeled as a Markov modulated Poisson process [5]. We compare our optimal switching to the sub-optimal mode-matching, and demonstrate that, indeed, our approach can lead to significant power savings.

The following section presents the formulation of DPM problem as an MDP when the service request is stationary, which serves as a starting point for our hierarchical formulation for the nonstationary case. Section 3 presents hierarchical adaptive DPM scheme and Sect. 4 presents results of several simulation studies applying our scheme to the HDD problem. The concluding section summarizes the contribution of this paper.

2. Stationary DPM Policies

Conventional power management is based on a simple time out policy [6]. The weakness of this approach is that the time-out value does not use any models of the service request arrival process (called the *usage pattern* in this paper) for the device. A more formal approach in [1] based on stochastic optimization overcomes the shortcomings of the time-out approach, where the power manager solves an MDP problem to achieve the optimization.

To get a better intuition about power management, we simulate DPM on an HDD with the power states in Fig. 1 and a buffer of length one. The HDD parameters used in our simulation are as follows: $P_a = 2.1W; P_i = 0.65W; P_{ai} = P_{ia} = 1.4W$. $T_{ai} = T_{ia} = 0.4s$ are expected values of the random power state transition times with exponential distributions. The service rate is 1.0 request/second (r/s) for the active HDD modeled as an exponential server[1].

In a more formal sense, when the service request arrives as a Poisson process, the system shown in Fig. 1 can be modeled as a continuous-time MDP, which can be discretized by sampling the continuous-time system at times when *request-arrival*, *service-completion* and *power-state transition-completion* events occur (see Fig. 1).

The discrete-time MDP is built on a discrete-time controlled Markov chain denoted as $\mathbf{S} = \{S_n, n \geq 0\}$. The state space of the Markov chain is $\Sigma = \{(n_{hq}, n_{ps}), n_{hq} = 0, 1, 2, n_{ps} = 1, 2\}$, where n_{hq} is the total number of requests in the buffer and HDD, and n_{ps} is the power state of the HDD (1 for *active* and 2 for *idle*). The action space is $\mathcal{A} = \{GO-ACTIVE, GO-IDLE\}$. At each decision epoch $n \geq 0$, a power state transition action U_n is taken from $\mathcal{A}$ and applied to the Markov chain. If an action $U_n = \alpha \in \mathcal{A}$ is taken at state $S_n = \sigma$, a power consumption of $f(\sigma, \alpha)$ incurs with two performance measurements $f^w(\sigma, \alpha)$ (waiting-time related), $f^l(\sigma, \alpha)$ (loss-probability related); the system then transits to state $S_{n+1} = \sigma' \in \Sigma$ with probability $p^\alpha(\sigma, \sigma')$. A *policy* c is a mapping $c : \Sigma \to \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(A)$ denotes the set of all the probability distributions over a given finite set A. Under a policy c, action $\alpha \in \mathcal{A}(\sigma)$ is applied to the system with probability $[c(\sigma)]_\alpha$ when the system state is $\sigma \in \Sigma$.

The objective is to minimize the average power consumption under two performance constraints: the average waiting-time limit of service requests in the buffer b_w and the loss-probability limit of incoming service requests b_l; that is, to

1 The power-specific parameters are chosen to be comparable to those from [11] for an HDD. Note that the *active* power state in this paper is in fact the combination of *active* and *performance idle* states in [11]; the *idle* state in this paper is the *low power idle* state in [11].

solve the following constrained optimization problem

$$\begin{aligned} \min_{c} \quad & J_f(c) \\ \text{s.}t. \quad & J_{f^w}(c) \leq b^w; \quad J_{f^l}(c) \leq b^l, \end{aligned} \tag{1}$$

where $J_f(c)$, $J_{f^w}(c)$ and $J_{f^l}(c)$ are averages of power, waiting time and loss probability respectively, under policy c.

The five applications in the trace shown in Fig. 2 generate five stationary workloads to the hard disk. Each workload is associated with an application accessing the disk alone indefinitely. We model each workload as a Poisson process with rates $\lambda_i,\ i = 1, 2, \cdots, 5$, given in the last column of Table 1.

By solving the MDP problem in (1) for each stationary case, we obtain five stationary optimal policies $c_i,\ i = 1, 2, \cdots, 5$ (which corresponds to *Windows Explorer*, *Internet Explorer*, *Microsoft Outlook*, *Adobe Photoshop* and *Windows Media Player*, respectively). Table 1 illustrates the five stationary optimal policies designed under limits $b_w = 0.2$s and $b_l = 0.02$. Each entry in Table 1 specifies probabilities for choosing actions $GO-ACTIVE$ and $GO-IDLE$ at a particular system-state. For instance, under the second stationary case corresponding to application *Internet Explorer* (c_2), when the buffer is empty and the HDD is idle, (i.e. at state (0,2)), the policy activates the HDD with probability 0.14.

The table only shows the actions for states (0,1) and (0,2), i.e., when there is no service request in the system, because all five policies agree on $GO-ACTIVE$ action for other states. The second column from right in Table 1 gives the power consumption under the five stationary cases.

State	*(0,1)*	*(0,2)*	Power	λ (r/s)
c_1	(0.67 0.33)	(0.06 0.94)	1.69W	0.0989
c_2	(0.54 0.46)	(0.14 0.86)	1.51W	0.0336
c_3	(0.61 0.39)	(0.25 0.75)	1.67W	0.0881
c_4	(0.45 0.55)	(0.68 0.32)	1.73W	0.1158
c_5	(0.43 0.57)	(0.75 0.25)	1.77W	0.1344

Table 1. Optimal policies for five stationary cases

The assumption on the stationary request-arrival process can be invalidated, and this is particularly the case when the applications are accessing the HDD alternatively as shown in Fig. 2(b); in such cases, the statistics of the accessing vary dramatically among the applications. Next, we address this issue and propose an hierarchical algorithm to solve it.

3. Hierarchical DPM

In this section, we introduce our hierarchical DPM scheme for the HDD operating under nonstationary service requests. We model the nonstationary request arrival process as a Markov-modulated stochastic process. This is a *multi-mode* model in which each *mode* corresponds to a given workload (a stationary arrival process) and mode transitions are modeled as a discrete-time Markov chain. In the HDD scenario,

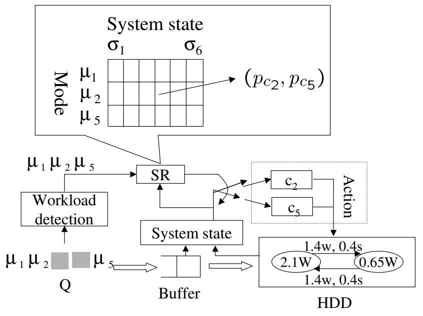


Figure 3. Our hierarchical adaptive DPM scheme

a *mode* is associated with a particular pattern of application accessing the HDD, and this is assumed to produce a stationary process of service request arrivals. Specifically in our simulations, we use Markov-modulated Poisson process to model the stochastic process illustrated in Fig. 2(a), using five workloads $\lambda_i,\ i = 1, 2, \cdots, 5$. The modulated process is the result of a jumping process among the five workloads shown in Fig. 2(b). We assume the workload at the sampling times (when request-arrival, service-completion and power-state transition completion events occur) forms a discrete-time Markov chain with transition probability matrix Q. We call the modulated process with a matrix Q a *usage pattern.* When Q is an identity matrix, we have the stationary case.

The multi-mode arrival dynamics results in a multi-mode MDP model. The state of a controlled Markov process $\mathbf{X}$ has two variables, the *system state* $\mathbf{S}$, defined in the previous section, and the *mode* $\mathbf{M}$ assuming values in a set $\mathcal{M} = \{\mu_1, \mu_2, \cdots, \mu_5\}$ corresponding, for instance, to the five applications at hand. At the n^{th} decision epoch, if a power state transition action $U_n = \alpha \in \mathcal{A}(\sigma)$ is taken at state $S_n = \sigma$, the incurred power is now $f_\mu(\sigma, \alpha)$ and the performance measurements are $f^w_\mu(\sigma, \alpha)$ (waiting-time related), $f^l_\mu(\sigma, \alpha)$ (loss-probability limited); and the system transits to state $S_{n+1} = \sigma' \in \Sigma$ with probability $p^\alpha_\mu(\sigma, \sigma')$, where $M_n = \mu$ is the mode at time n. The mode transits from $M_n = \mu$ to $M_{n+1} = \mu'$ with probability $q^\alpha_{\sigma\sigma'}(\mu, \mu')$.

An *observable* mode corresponds to an application deemed important enough for the operating system to inform the power manager when such an application is accessing the disk. Otherwise, the power manager faces an *unobservable mode.*

The *policy switching scheme* is a triplet $(\mathcal{C}, \mathcal{M}_t, SR)$, where: $\mathcal{C} = \{c_1, c_2, \cdots, c_{|\mathcal{C}|}\}$ is a set of available policies; $\mathcal{M}_t \subseteq \mathcal{M}$ is the set of *observable modes*; and $SR : \Sigma \times \mathcal{M}_t \rightarrow \mathcal{P}(\mathcal{C})$ is a (non-deterministic) *switching rule.* The scheme operates as follows. Each time the system mode is in $\mathcal{M}_t$, the switching rule SR chooses a policy from $\mathcal{C}$ and applies it to the system. This chosen policy stays applied until the next time the system mode is in $\mathcal{M}_t$ again. For a given SR, policy $c \in \mathcal{C}$ is applied to the system with probability $[SR(\sigma, \mu)]_c$ when the system state is $\sigma \in \Sigma$ and the mode is $\mu \in \mathcal{M}_t$.

The hierarchical scheme is illustrated in Fig. 3 with the observable mode set $\mathcal{M}_t = \{\mu_1, \mu_2, \mu_5\}$ and the policy set $\mathcal{C} = \{c_2, c_5\}$. Thus, for this example, when *Windows Ex-*

plorer, *Internet Explorer* and *Windows Media Player* are accessing the disk (modes μ_1, μ_2, μ_5 above), the power manager has to choose between the stationary optimal policies designed for *Internet Explorer* or *Windows Media Player* (policies $c_2 or c_5$ above). When *Microsoft Outlook* and *Adobe Photoshop* are accessing the disk, no policy switching decision is made, previously selected policies stay applied.

For the higher-level policy switching, the power manager solves an *optimal switching rule problem*: given the policy set $\mathcal{C}$ and the mode set $\mathcal{M}_t$, find an SR with minimum long run average power under the performance constraints; that is:

$$\begin{aligned} \min_{SR} \quad & J_f(SR) \\ \text{s.t.} \quad & J_{f^w}(SR) \le b^w; \quad J_{f^l}(SR) \le b^l, \end{aligned} \tag{2}$$

where $J_f(SR)$, $J_{f^w}(SR)$ and $J_{f^l}(SR)$ are averages of power, waiting time and loss probability under switching rule SR.

The process $\mathbf{X}$ running under an SR is *not* generally Markovian since knowing the current system mode and state does not imply knowing the current policy. The process $\mathbf{Y}$ embedded in the switching epochs (times when the system mode is in $\mathcal{M}_t$) is Markovian, however. For a sample path of $\mathbf{X}$: $\omega = \{X_0, X_1, X_2, \cdots,\}$ running under a particular SR, $\{Y_i = X_{T_i}, i = 0, 1, 2, \cdots\}$ forms an embedded Markov chain with stationary transition probabilities, where $T_0 = 0$ and $T_i = \min_t\{t > T_{i-1}, M_t \in \mathcal{M}_t\}$, $i = 1, 2, \cdots$. A new MDP can be formulated on process $\mathbf{Y}$, where at each state $(\sigma, \mu) \in \Sigma \times \mathcal{M}_t$ an action is the selection of a policy in $\mathcal{C}$. When policy c is selected and applied to the system, $\mathbf{Y}$ transits to state $(\sigma', \mu') \in \Sigma \times \mathcal{M}_t$ with probability $\tilde{p}^c\,[(\sigma, \mu), (\sigma', \mu')]$. (See [9] for further details.)

For the new MDP, the transition and cost parameters $\tilde{p}$ and h can be estimated along the sample path of the system. Let $\{Y_0, Y_1, \cdots, Y_N\} = \{(S_{T_0}, M_{T_0}), (S_{T_1}, M_{T_1}), \cdots, (S_{T_N}, M_{T_N})\}$ be the full-state observations at switching times and let $\{C_0, C_1, \cdots, C_{N-1}\}$ be the policies chosen at switching times. We use capitalized notations for the policies because they are random variables assuming values in $\mathcal{C}$. We have the following maximum-likelihood estimation for the $\tilde{p}$-quantities,

$$\begin{aligned} \hat{\tilde{p}}^c[(\sigma, \mu), (\sigma', \mu')] &= \frac{1}{\sum_{n=0}^{N-1} \mathbf{1}_{\{Y_n=(\sigma,\mu)\}} \mathbf{1}_{\{C_n=c\}}} \\ &\times \sum_{n=0}^{N-1} \left[\mathbf{1}_{\{Y_n=(\sigma,\mu)\}} \mathbf{1}_{\{C_n=c\}} \mathbf{1}_{\{Y_{n+1}=(\sigma',\mu')\}} \right], \end{aligned} \tag{3}$$

where indicator function $\mathbf{1}_{\{\cdot\}}$ is one when the condition in $\{\cdot\}$ is true, zero otherwise. For a (power or performance) function g, let the observed power or performance along the sample path be $\{F_{T_0}, F_{T_0+1}, \cdots, F_{T_1}, F_{T_1+1}, \cdots, F_{T_N}\}$. We have the maximum-likelihood estimation for the h-quantities,

$$\begin{aligned} \hat{h_g}[(\sigma, \mu), c] &= \frac{1}{\sum_{n=0}^{N-1} \mathbf{1}_{\{Y_n=(\sigma,\mu)\}} \mathbf{1}_{\{C_n=c\}}} \\ &\times \left[\mathbf{1}_{\{Y_n=(\sigma,\mu)\}} \mathbf{1}_{\{C_n=c\}} \sum_{m=T_n}^{T_{n+1}-1} F_m \right]. \end{aligned} \tag{4}$$

There are many ways to generate a good sample path for the estimation. The basic requirement is to use each state-policy pair $[(\sigma, \mu), c]$ a sufficient number of times along the sample path so that the parameter estimates are adequate. Using these estimates, the power manager uses the following algorithm to compute the adaptive optimal switching rule.

Algorithm 1 *(Adaptive Optimal Switching Rule)*

1. *Estimate* $|\mathcal{C}||\Sigma \times \mathcal{M}_t|^2$ *$\tilde{p}$-quantities and* $(I+2)|\mathcal{C}||\Sigma \times \mathcal{M}_t|$ *h-quantities in (5) from a sample path of the system by using (3) and (4).*
2. *Choose* $\delta > 0$. *Choose* $r_0 > 0$ *and set k:=0;*
3. *At the* k^{th} *step, solve*

$$\begin{aligned} &\max \sum_{\sigma,\mu,c} \left\{ \hat{h_f}[(\sigma, \mu), c] - r_k \hat{h_t}[(\sigma, \mu), c] \right\} z[(\sigma, \mu), c], \\ &s.t. \sum_{\sigma,\mu,c} \left\{ \delta_{[(\sigma,\mu)(\sigma',\mu')]} - \hat{\tilde{p}}^c[(\sigma, \mu), (\sigma', \mu')] \right\} z[(\sigma, \mu), c] = 0, \\ &\qquad \textit{for every } (\sigma', \mu') \in \Sigma \times \mathcal{M}_t, \\ &\qquad \sum_{\sigma,\mu,c} z[(\sigma, \mu), c] = 1, \\ &\sum_{\sigma,\mu,c} \left\{ \hat{h_{f^i}}[(\sigma, \mu), c] - b^i \hat{h_t}[(\sigma, \mu), c] \right\} z[(\sigma, \mu), c] \le -B, \\ &\qquad i = 1, 2, \cdots, I, \\ &\qquad z[(\sigma, \mu), c] \ge 0, \quad \textit{for every } \sigma \in \Sigma, \mu \in \mathcal{M}_t, c \in \mathcal{C}, \end{aligned} \tag{5}$$

   where $-B < 0$ *is a margin introduced to ensure the feasibility of the switching rule to the original inequality constraints.*
4. *With the solution* z^* *to (5), an* SR_k^* *can be constructed as follows: at state* $(\sigma, \mu) \in \Sigma \times \mathcal{M}_t$, *policy* c_i *is applied with probability* $\frac{z^*[(\sigma,\mu),c_i]}{\sum_{c \in \mathcal{C}} z^*[(\sigma,\mu),c]}$. *Also with* z^*, *calculate* $r_{k+1} = \frac{\sum_{\sigma,\mu,c} h_f[(\sigma,\mu),c] z^*[(\sigma,\mu),c]}{\sum_{\sigma,\mu,c} h_t[(\sigma,\mu),c] z^*[(\sigma,\mu),c]}$.
5. *If* $|r_{k+1} - r_k| \le \delta$, *exit; otherwise, set* $r_k := r_{k+1}$ *and* $k := k+1$, *go to step 3.*

To give a little bit of intuition behind the Algorithm 1 above, we should say that the constrained MDP optimization problem in (1) has a fractional objective function. That is, the objective function is a ratio of two long run average costs. This constrained MDP optimization problem is equivalent to a fractional programming problem in which the objective function is a ratio of two linear functions and constraints are all linear. Algorithm 1 solves this fractional programming problem through iterations, and a linear program is solved in each iteration (step 3 above). In step 4, the optimal switching rule is constructed from the solution to the fractional programming problem.

Referring to Algorithm 1, it is important to note that we do *not* assume that the arrival statistics under these modes are known. Also, we do *not* directly estimate these statistics in our scheme. In our adaptive algorithm, only an agreement on what modes are observable is needed. This contrasts to the approach in [4] where the arrival statistics need to be directly estimated, although this may be problematic.

4. Experimental Results

In this section, we show that our hierarchical adaptive DPM approach performs better than the approach proposed in [4]. When comparing the two approaches, we focus on the intrinsic difference of the two switching methodologies, i.e., the optimal switching vs. non-optimal mode-matching. When calculating the optimal switching rule, we estimate statistics along a sample path long enough to achieve high confidence and accuracy. On the other hand, when implementing the approach in [4], we assume perfect knowledge of arrival rates for the policy interpolations.

Through statistical analysis, the trace in Fig. 2 gives the following mode transition probability matrix

$$Q = \begin{bmatrix} 0.9791 & 0.0028 & 0.0084 & 0.0076 & 0.0021 \\ 0.0011 & 0.9930 & 0.0048 & 0.0004 & 0.0007 \\ 0.0071 & 0.0086 & 0.9800 & 0.0021 & 0.0022 \\ 0.0100 & 0 & 0.0008 & 0.9875 & 0.0017 \\ 0.0111 & 0.0044 & 0.0067 & 0 & 0.9778 \end{bmatrix}.$$

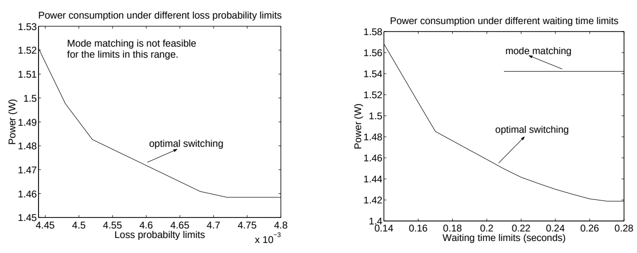
Under this usage pattern, observable mode set $\mathcal{M}_t = \{\mu_1, \mu_2, \mu_5\}$ and policy set $\mathcal{C} = \{c_2, c_5\}$, Table 2 gives the optimal switching rule for waiting-time and loss-probability limits $b_w = 0.2s$, $b_l = 0.02$. Each entry in Table 2 gives the mode, system-state and the probabilities for choosing policies c_2 and c_5 for that mode and system-state pair. For instance, when *Internet Explorer* accesses the hard disk (that is, mode μ_2) and the disk buffer becomes empty (that is, the system-state $(0, 1)$ which is the first bold entry in the μ_2 row in Table 2), the power manager applies the policy c_5 (instead of the mode-matching c_2) which puts the disk to idle with a probability of 0.57 according to Table 1. The optimal switching rule consumes a power of $1.46W$. The mode matching rule based on approach in [4] in this case chooses controller c_2 for mode μ_2, c_5 for μ_5, and c_2 with an interpolated probability 0.65 for μ_1. The mode matching rule consumes a power of $1.54W$, and *cannot* satisfy the waiting time limit of $0.2s$.

The asterisks in Table 2 indicate switching decisions that do *not* correspond to a simple mode-matching policy; that is, the policies selected by the optimal switching rule for the cases with an asterisk are *not* the stationary policies that were designed for those modes even though those stationary policies are available for policy switching. This demonstrates that mode-matching is *not* always the best thing to do. The reason is that the policies are designed for *stationary* processes, i.e., under the assumption that the mode will remain constant. When the mode switching dynamics are taken into account, it turns out that for some cases it is better to use one of the policies designed for a different mode.

For the usage pattern in the HDD trace, Fig. 4 shows the power consumption of different optimal switching rules obtained for different loss probability and waiting time limits. Figure 4(a) shows that, with looser performance bounds, more power can be saved by the optimal rule. For a comparison, Fig. 4(b) also shows the power consumption of the mode-matching power management scheme. The power consumption of the mode-matching switching rule is a constant because it does not change under different performance limits. Note that the mode-matching cannot achieve loss probability limits in the range in Fig. 4(a), and waiting time limits less than 0.2s in Fig. 4(b).

Mode	μ_1	μ_1	μ_1	μ_1	μ_1	μ_1
State	(0,1)	(0,2)	(1 1)	(1 2)	(2 1)	(2 2)
Prob.	(0 1)	(0.57 0.43)	(0 1)	(0 1)	(0 1)	(0 1)
Mode	μ_2	μ_2	μ_2	μ_2	μ_2	μ_2
State	(0,1)	(0,2)	(1 1)	(1 2)	(2 1)	(2 2)
Prob.	**(0 1)***	(1 0)	(0 1)*	(0 1)*	(0 1)*	(0 1)*
Mode	μ_5	μ_5	μ_5	μ_5	μ_5	μ_5
State	(0,1)	(0,2)	(1 1)	(1 2)	(2 1)	(2 2)
Prob.	(0 1)	(0 1)	(0 1)	(0 1)	(0 1)	(0 1)

Table 2. Optimal switching rule



(a) (b)

Figure 4. Power consumption under different (a) loss probability limits (b) waiting time limits

To further demonstrate the effectiveness of our hierarchical switching scheme, Fig. 5 gives the power consumption of the optimal switching and mode matching for 50 Q matrices that were generated randomly. For 9 out of the 50 patterns considered in this experiment, the mode-matching scheme cannot achieve the 0.2s waiting time or the 0.02 loss probability limits (these are the infeasible cases identified in Fig. 5). For the other 41 patterns, the power savings of optimal switching from mode matching is around 10%.

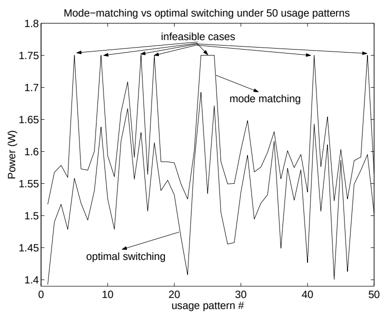
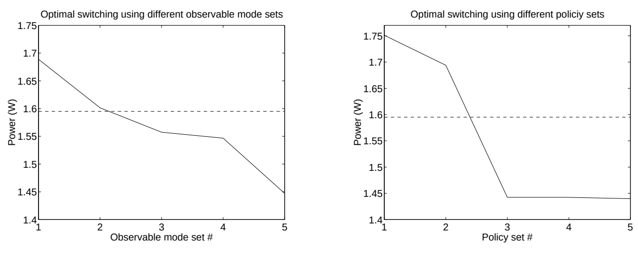


Figure 5. DPM under 50 nonstationary cases

We also consider the effect of different observable mode



(a) (b)

Figure 6. Power consumption under different (a) observable mode sets (b) policy sets

sets $\mathcal{M}_t$'s and policy sets $\mathcal{C}$'s on the performance of our adaptive switching rule. Intuitively, the optimal switching rule can achieve more power savings when more observable modes and more policies are used. On the other hand, more observable modes and more policies result in more computations for obtaining the optimal switching rule. Therefore, the selection of these two has to take both power savings and computation burden into account. One idea is to start with coarse sets and introduce more modes and policies into the sets gradually until a desired level of power consumption is achieved. The following simulations demonstrate this idea.

Figure 6(a) shows the power consumption of our scheme under the usage patterns in Fig. 2 when the policy set is $\mathcal{C} = \{c_2, c_5\}$ and there are five observable mode sets $\mathcal{M}_t$'s:

$$\{\mu_5\},\ \{\mu_5, \mu_4\},\ \{\mu_5, \mu_4, \mu_3\},\ \cdots,\ \{\mu_5, \mu_4, \mu_3, \mu_2, \mu_1\}$$

(In Fig. 6(a), mode 1 corresponds to the observable mode set $\{\mu_5\}$, mode 2 to $\{\mu_5, \mu_4\}$, etc.) More power savings are achieved through introducing more modes into $\mathcal{M}_t$. As we can see, mode set 3 consisting of $\mathcal{M}_t = \{\mu_5, \mu_4, \mu_3\}$ is needed to achieve a desired power level of less than $1.6W$. The desired power of about $1.55W$ is found by looking at the stationary case with the same arrival rate as the average arrival rate of the nonstationary usage pattern.

Figure 6(b) shows the power consumption when the observable mode is $\mathcal{M}_t = \{\mu_1, \mu_2, \mu_5\}$ and there are five available optimal control policies $\mathcal{C}$'s:

$$\{c_5\},\ \{c_5, c_4\},\ \{c_5, c_4, c_3\},\ \cdots,\ \{c_5, c_4, c_3, c_2, c_1\}$$

As we can see, the policy set 3 (that is, $\mathcal{C} = \{c_5, c_4\, c_3\}$) is needed in order to achieve a desired power level of around $1.45W$ (which is below the threshold of $1.6W$). This shows the possibility of deriving a "feedback" mechanism to adaptively adjust the set of "important" modes and policies based on online optimization performance.

5. Conclusions

This paper presents a novel hierarchical scheme for adaptive dynamic power management under nonstationary service requests. We derive a multi-mode approach based on a model of Markov-modulated stochastic process of the nonstationary request arrival process. In this scheme each mode is associated with a stationary arrival process. The power manager adaptively switches among a set of stationary optimal policies to accommodate the stochastic mode-switching arrival dynamics. The simulation results show that our hierarchical scheme performs better than the mode matching based approach proposed previously.

The mode switching paradigm illustrated for the HDD application in this paper can be applied to many situations where the demand for power-consuming services is nonstationary. Examples include voltage and frequency scaling in embedded processors to accommodate different mixes of tasks in a multi-tasking environment, and power management strategies for wireless sensor networks based on the type of network traffic generated by the current task (e.g., tracking versus monitoring).

For any application, the key to applying our hierarchical method is to associate sources of nonstationarity with specific operating conditions identified as modes. In general, the number of modes to be used is determined by identifying a set of operating conditions that can be conveniently analyzed off-line to determine a set of stationary DPM policies. The online optimization then finds the best way to use these polices for the actual nonstationary demand.

Acknowledgments: The authors would like to thank Jingcao Hu of CMU for help in preparing the final version of the manuscript.

References

[1] L. Benini, A. Bogliolo, G. Paleologo, and G.D. Micheli, Policy optimization for dynamic power management, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 18, 813-833, 1999.

[2] L. Benini, A. Bogliolo, and G.D. Micheli, A survey of design techniques for system-level dynamic power management, *IEEE Transactions on VLSI Systems*, Vol. 8, 299-315, 2000.

[3] L. Benini and G.D. Micheli, *Dynamic Power Management: Design Techniques and CAD Tools,* Norwell, MA: Kluwer, 1997.

[4] E. Y. Chung, L. Benini, A. Bogliolo, and G.D. Micheli, Dynamic power management for nonstationary service requests, in *Proc. Design and Test in Europe (DATE)*, 77-81, 1999.

[5] W. Fischer and K. Meier-Hellstern, The Markov-modulated Poisson process (mmpp) cookbook, *Performance Evaluation*, vol. 18, pp. 149–171, 1992.

[6] P. Greenawalt, Modeling power management for hard disks, *Internat. Workshop on Modeling, Analysis, and Simulation for Computer and Telecommunications Systems,* pp. 62-65, 1994.

[7] Q. Qiu and M. Pedram, Dynamic power management based on continuous-time Markov decision processes, in *Proc. of Design Automation Conference*, pp. 555-561, June 1999.

[8] M. Pedram and J. Rabaey, Eds., *Power Aware Design Methodologies,* Norwell, MA: Kluwer, 2002.

[9] Z. Ren, Aggregation and multi-mode switching in Markov decision processes, Ph.D. Dissertation, Dept. of ECE, Carnegie Mellon University, Pittsburgh, PA, April 2002.

[10] T. Šimunić, L. Benini, P. Glynn and G. De Micheli, Event-drivenpPower management, *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems,* Vol. 20, no. 7, July 2001.

[11] Storage Systems Division, IBM Corp., Adaptive Power Management for Mobile Hard Disks, *www.almaden.ibm.com/almaden/mobile_hard_drives.html*, January, 1999.