PowerSleep: A Smart Power-Saving Scheme with Sleep for Servers under Response Time Constraint

Shengquan Wang
Department of Computer
and Information Science
University of Michigan-Dearborn, USA
shqwang@umd.umich.edu

Jian-Jia Chen
Department of Informatics
Karlsruhe Institute of Technology, Germany
jchen@tik.ee.ethz.ch

Jun Liu
Department of Computer
and Information Science
University of Michigan-Dearborn, USA
juliu@umd.umich.edu

Xue Liu
Department of Computer Science
and Engineering
University of Nebraska Lincoln, USA
xueliu@cse.unl.edu

***Abstract*—Reducing the power consumption while maintaining the response time constraint has been an important goal in server system design. One of the techniques widely explored in the literature to achieve this goal is Dynamic Voltage Scaling (DVS). However, DVS is not efficient in modern systems where the overall power consumption includes a large portion of static power consumption. In this paper, we aim to reduce the static power consumption by Dynamic Power Management (DPM) with sleep model in addition to DVS. To maximize the sleep efficiency, we propose `PowerSleep`, a smart power-saving scheme by carefully choosing an execution speed for the server with DVS and sleep periods while putting the system in the sleep power mode with DPM. By modeling the system with M/G/1/PS queuing model and further significant extensions, we present how to minimize the mean power consumption of the server under the given mean response time constraint. Simulation results show that our smart `PowerSleep` scheme significantly outperforms the simple power-saving scheme which adopts sleep mode.**

***Keywords*-power saving; sleep; response time; M/G/1**

I. Introduction

Power-aware design has become a prominent design issue in server systems due to the rising energy utility bill. For example, for a high-performance server with 330 Watt power consumption, the annual energy cost of the server is around \$$214$, provided that the electricity costs \$0.074 per kWh. Even without considering the cost of the power delivery subsystems and the cooling facility, the electricity cost is significant in maintaining a cluster with hundreds of servers. Specifically, it has been shown that the electricity cost remains significant even if the server does not always operate with the maximum power consumption [1]. By 2011, data centers in U.S. are expected to consume around 100 billion kWh per year [2], in which the annual power cost is around \$7.4 billion.

At the same time, clients are very sensitive to the server performance. Delayed response to users will have negative effects for a hosting company including client frustrations and revenue loss. Mean response time of requests has been an important performance measure for servers. How to minimize the server mean response time or to meet the mean response time service level agreement (SLA) constraint for servers has been an active research [3]–[6]. Recently, how to reduce the power consumption while maintaining the mean response time constraint has received increasing attention. Low-power opportunity for web servers has been observed in [7], [8] to reduce the energy consumption by applying Dynamic Voltage Scaling (DVS) with minimal performance impact. In [9], a queueing theoretic model was used to predict the optimal power allocation in a variety of scenarios with DVS. An optimal speed scaling was investigated in [10] to balance the mean energy consumption and mean response time under Processor Sharing (PS) scheduling.

DVS is an efficient power-saving technique in the systems where the static power consumption is only a small portion of the overall power consumption. However, as shown in [11]–[13], the static power dissipation when a server is idle could reach up to 60% of the peak power, and is worsened if the power waste in power delivery and cooling sub-systems is counted, which could increase power consumption by 50$\sim$100% [14]. Given the fact that average server utilization is only 20$\sim$30% in typical data centers [7], [11], [15], reducing the power consumption for an idle server becomes practically important. To overcome the idleness of the server, consolidation technique is adopted in server clusters by using virtual machines to put several servers in one machine and reduce the number of active machines. However, a large fraction of servers exhibit frequent but brief bursts of activity, making dynamic consolidation and system shutdown difficult [15]. The other common approach to reducing power consumption for an idle server is to use Dynamic Power Management (DPM) (such as clock gating or power gating). In other words, to reduce the power consumption when a server is idle, we can turn the server from the active mode to the sleep mode. However, mode transitions between the active mode and the sleep mode introduce significant overheads in terms of time and energy.

In [15], a `PowerNap` scheme was proposed to handle the dominant idle time by quickly transitioning in and out of a low power sleep mode. Under `PowerNap`, the server runs at the maximum speed in executing jobs if there are jobs in queue, and is immediately put into the sleep mode once the queue is empty and is waken up once a new job arrives. However, mode transitions between the active mode and the sleep mode introduce a pure timing overhead for the server, which degrades the performance such as the mean response time of jobs. When the server utilization is low, the `PowerNap` scheme is shown superior to the pure DVS scheme. The higher the server utilization is, the more obvious the mode transition overhead has an impact on the system performance. Therefore, it is necessary in the design with the sleep mode to reduce the mode transition overhead as much as possible. In [15], one of the goals is to reduce each transition duration for fast mode transitions. However, due to the hardware limitation, we have no much space in this direction. Moreover, in general, the more inactive hardware components are in the sleep mode, the larger the timing overhead is required for mode transitions. Therefore, in addition to having fast mode transitions from the hardware aspects, we would also like to consider another direction: reducing the mode transition frequency from the software aspects. We could jointly consider DVS to change the execution speed of the server and DPM to change the power mode. We do not have to run the server at the maximum speed in executing jobs or wake up the server so greedily like `PowerNap` since the server might have to go to sleep again after a short period of job execution.

In this paper, we propose `PowerSleep`, a smart power-saving scheme. To minimize the mean power consumption while maintaining the mean response time constraint, we carefully choose an execution speed for the server with DVS and sleep periods while putting the system in the sleep power mode with DPM. We will show that `PowerSleep` outperforms `PowerNap` significantly, in particular when the single mode transition overhead is large. In the study of server performance, M/G/1/PS server model has been shown by different research studies that can model the modern web servers well [3], [6], [9], [10], [16]–[18]. To make the modeling and analysis even more accurate, in this paper, we adopt the M/G/1/PS model with some significant extensions as the mode transition overhead is taken into consideration.

The rest of this paper is organized as follows: Section II shows the system model. The design framework of `PowerSleep` will be described in details in Section III. Section IV presents detailed power consumption and response time analysis. The optimal design is described in Section V by showing how to minimize the mean power consumption under the given mean response time constraint. Section VI presents performance evaluation over simulated platforms. We will conclude the paper in Section VII.

II. System Model

We use DVS and DPM for the power management in the server. With DVS, we can choose an execution speed for the server (with a corresponding choice of the supply voltage) to serve jobs in the queue. We define r as the ratio of the execution speed of the server to its maximum speed. The speed ratio r is bounded by a lower bound r_l, i.e., $r_l \leq r \leq 1$. When the server is active, either it is (i) in the *running* mode while executing jobs, or (ii) in the *idle* mode at the lowest speed ratio r_l without executing any job. With DPM, the server can be set to the *sleep* mode. However, there are some timing overheads for the mode transitions between the running mode and the sleep mode [15], during which the server is in the *transition* mode.

The power consumption in our study is the system-level power, including the power consumed by the processor and all other components within the server. The power consumption depends on the mode the server is in (running, idle, sleep, or transition), and also the execution speed in use. In this paper, we adopt the power consumption model in [9]. The server has the following power modes:

- *Idle power mode:* In the idle mode, the server consumes the static power P_I;
- *Running power mode:* In the running mode, the power consumption $P_R(r)$ by the server at a speed ratio r is
$$P_R(r) = \alpha[r - r_l]^\gamma + P_I, \tag{1}$$
where $\gamma \geq 1$. The cubic rule is widely suggested in the literature for the processor power-to-speed relationship in the running mode, i.e., $\gamma = 3$. However, in server farms with DVS or for some applications, the linear rule could be applied. The reader can find more details on this in [9];
- *Sleep power mode:* In the sleep mode, the power consumption by the server is P_S, where $P_S \ll P_I$.
- *Transition power mode:* In the transition mode, the server also consumes power, which is defined as P_T. In this paper we assume the power consumption in the transition mode is equal to the one in the running mode, i.e., $P_T = P_R(r)$.

The different power modes provide the space for system designers to design efficient power-saving schemes.

The system in this work is based on the M/G/1/PS server model. We consider a Poisson job arrival with an arrival rate λ and we assume that jobs follow a generalized service time distribution with a given mean value $\mathbb{E}[S]$ when executing at the maximum speed. We also assume all jobs in the queue are served with the PS scheduling algorithm, where PS approximates very well the Round-Robin job scheduling algorithm used in Linux. Our methodology can be applied to other job scheduling algorithms such as First-Come-First-Served (FCFS) as well. We assume the mean job execution time $\mathbb{E}[S] = \frac{1}{\mu}$ under the maximum speed. If the server runs at a speed ratio r in the running mode, we have $\mathbb{E}[S] = \frac{1}{r\mu}$.

We denote

$$\rho = \frac{\lambda}{\mu}. \tag{2}$$

The relative server utilization with respect to the speed ratio r can be written as

$$\lambda \mathbb{E}[S] = \frac{\rho}{r}, \tag{3}$$

while ρ is the (absolute) server utilization with respect to the maximum speed. Note that this paper considers a coarse-grained model for scheduling by assuming that the execution time is proportional to $\frac{1}{r}$. If the memory or I/O peripheral accesses make the frequency scaling dis-match the frequency, speeding up the CPU frequency by a factor of 2 may only reduce the execution time by 30%. The definitions in (2) and (3) become invalid.

III. The Design Framework of `PowerSleep`

In order to design a better power-saving strategy, we try to answer the following questions: (i) When should the server start to sleep? (ii) When should the server be waken up? (iiii) What speed ratio can be set in the running state? The straightforward scheme is to set the server to sleep once the queue is empty and is waken up upon a new job arrival and choose the full speed in the running state as shown in [15]. This approach works but with some limitations and it is not efficient due to the following concerns:

C1. *The transition from the running power mode into the sleep power mode might be too costly.* If the idle-queue duration (or non-idle-queue duration) is short, the server is wasting time in transition without taking enough duration of sleep (or execution of jobs).

C2. *The average response time threshold would be always violated if the wake-up transition time is beyond the average response time threshold.* All jobs after the wake-up will experience at least the wake-up transition delay. This is another big drawback of this simple approach.

C3. *The full speed might not be the best choice in saving power.* This is well-accepted in the study of pure DVS in the literature. If we also consider DPM, how will speed affect the power-saving design?

In the design of `PowerSleep`, we introduce the following constant parameters of time periods to overcome the above raised concern by utilizing both DVS and DPM:

- Idle period threshold δ_h: It is the minimum length of the idle-queue duration before the server is put into the sleep power mode, i.e., if the idle-queue duration is shorter than δ_h, the server remains in the idle power mode not in the sleep power mode, and then goes back to the running power mode for the new job arrivals. δ_h is a very important parameter to tradeoff the benefit of the sleep power mode and the cost of the transition overhead in(/out of) the sleep mode.
- Sleep period threshold δ_e: It is the maximum length of the period during which the server can stay in the sleep power mode continuously. Once the sleep duration exceeds δ_e, the server needs to be put back to the idle power mode or the running power mode. This gives the server the chance to be put back to the running power mode precautiously so that the job can be served shortly with reduced response time. This works better under the long idle period of the job queue. δ_e is a very important parameter to tradeoff the gain between the sleep power mode and the response time.
- Procrastination sleep period δ_x: If the job arrives earlier before the expiration of δ_e, the server will be procrastinated in the sleep mode for δ_x period so that jobs can be batched together to reduce the short idle periods of the job queue. This works better under the short idle period of the job queue.

Now we are able to address the concerns raised in the beginning of this section: For C1 and C2, δ_h and δ_e are introduced for these purposes respectively; for C3, we will study the effect of the value of the speed ratio r to the power saving design in Sections IV and V;

With the above pre-determined constant parameters of time periods (δ_h, δ_x, δ_e), and of speed ratio r, `PowerSleep` can be described as the following steps:

i) Once the queue is empty, the server intends to hold on in the idle power mode for δ_h time unit;
ii) If a new job arrives before the expiration of the δ_h time unit, the server will immediately serve the new job. Otherwise, the server will be set to the sleep power mode and then it will be enforced to stay in the sleep power mode for δ_e time unit;
iii) If a new job arrives before the expiration of the δ_e time unit, the server will remain in the sleep power mode for a procrastination sleep period δ_x time unit (counted from the arrival of the new job). In other words, the job will wait for δ_e time units. Otherwise, the server will be put in the idle power mode again until a new job arrives;
iv) Once the server is in the running power mode, the server runs at a constant speed ratio r in the running power mode serving jobs in the queue until the queue become empty;
v) Once the queue is empty, repeat Step i.

The above procedure will be looped as jobs come to and leave the queue.

Recall that there are timing overheads for the mode transitions between the running power mode and the sleep power mode. We denote δ_s and δ_w as the overheads for the *suspend* transition from the running power mode to the sleep power mode and the *wake-up* transition from the sleep power mode back to the running power mode respectively. In [15], it is assumed that these two types of transition duration are equal, but we consider a general case. Note that each suspend transition could not be stopped once a suspend transition is initiated. The wake-up transition will follow the procrastination sleep period before serving a new job. Also note that `PowerNap` in [15] is a special case of `PowerSleep` by setting $r = 1$, $\delta_h = \delta_x = 0$, and $\delta_e = \infty$.

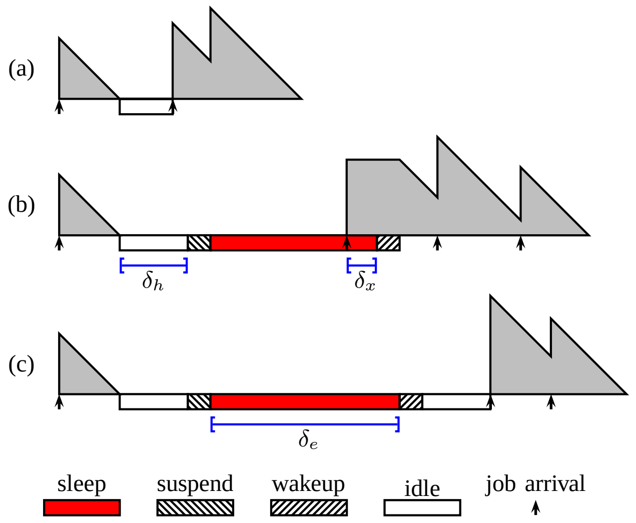


Fig. 1. Scenarios for `PowerSleep`

Figure 1 illustrates the change of the power mode with `PowerSleep` under different scenarios, where the X-axis is the time line and the Y-axis is the workload in the queue. We define δ_i as the length of a preceding idle-queue duration for a job arrival:

(a) $\delta_i \le \delta_h$: After a new job arrives, the server will immediately serve the new job;
(b) $\delta_h < \delta_i < \delta_h + \delta_s + \delta_e$: After a new job arrives, the server will remain in the sleep power mode for extra δ_x time units, and then wake up to serve;
(c) $\delta_i \ge \delta_h + \delta_s + \delta_e$: After the server is waken-up, the server will stay in the idle power mode until a new job arrives.

In Scenario (a), since the idle-queue period is short, it is not worth putting the server into the sleep power mode. In Scenario (b), the new job arrives before the expiration of the pre-defined δ_e period, then we adjust the sleep period with an extra procrastination sleep period δ_x. In Scenario (c), the idle-queue period is long, then the server will stay in the idle power mode before a new job arrives.

As we can see in Figure 1, if δ_h is too short, the server wastes time in mode transition out of sleep mode; if δ_h is too long or δ_e is too short, the server does not take the advantage of the long idle-queue period; if δ_e is too long and the wakeup transition overhead is big, the mean response time for job might be not tolerable. At the same time, the execution speed r will also affect the idle-queue duration. Therefore, choosing appropriate values of δ_h, δ_x, δ_e, and r is the key to the efficiency of the design of `PowerSleep`.

IV. Power Consumption and Response Time Analysis

Recall that our objective is to minimize the power consumption under the mean response time constraint. First we need to perform the power consumption and response time analysis, which depends on the power-saving scheme used in the server. Under `PowerSleep`, we will extend the tradition M/G/1/PS queueing theory to consider the sleep and transition modes.

A. Extended M/G/1/PS Queue with Starter

Under `PowerSleep`, the server will stay in four different modes: running, transition, sleep, and idle. With sleep and transition modes, the traditional queueing theory cannot be applied directly here. Instead we adopt a *Queue with Starter* model [19], [20]: the server is "turned off" whenever the queue becomes empty. When a job arrives at an empty queue, it cannot be served immediately; rather the server requires an additional amount of time T_X (called a *starter*) to start from "cold" before it can serve the new first job. Jobs which arrive to a "hot" server (i.e., one with at least one job either in service or in the queue) will join the queue and be served in turn as in a simple queueing system. Starter T_X under `PowerSleep` includes the wake-up transition plus the procrastination sleep period δ_x and may also include the remaining portion of a suspend transition.

Since the server has different power consumption in different modes, we need to obtain the probability that the server is in each mode (running, transition, sleep, and idle), which we define as π_R, π_T, π_S, and π_I respectively. Under the *Queue with Starter* model, these probabilities can be obtained with the following lemma:

Lemma 1: In an M/G/1 server under `PowerSleep` with Starter T_X, a job arrival rate λ, and a generalized service time distribution with a given mean value $\mathbb{E}[S]$, we have

$$\pi_R = \lambda \mathbb{E}[S], \tag{4}$$

$$\pi_T = [1 - \lambda \mathbb{E}[S]] \frac{\lambda[\delta_s + \delta_w] e^{-\lambda \delta_h}}{1 + \lambda \mathbb{E}[T_X]}, \tag{5}$$

$$\pi_S = [1 - \lambda \mathbb{E}[S]] \frac{e^{-\lambda \delta_h} [\lambda \delta_x + e^{-\lambda \delta_s} [1 - e^{-\lambda \delta_e}]]}{1 + \lambda \mathbb{E}[T_X]}, \tag{6}$$

$$\pi_I = 1 - \pi_R - \pi_T - \pi_S. \tag{7}$$

Proof: In order to compute the probability that the server is in each mode, we focus on a *cycle* in the server, which is a time interval that begins at an instant when the queue becomes empty and ends at the first time thereafter when the queue is empty again, after at least one job has been served. We will study the time spent by the server in each mode during a cycle. Let the random variable T_C have the distribution of the length of a cycle. Also let random variables T_R, T_T, T_S, and T_I have the distributions of the lengths of time spent in running, transition, sleep, and idle transition modes in a cycle respectively. Then the probabilities can be written as

$$\pi_R = \frac{\mathbb{E}[T_R]}{\mathbb{E}[T_C]}, \pi_T = \frac{\mathbb{E}[T_T]}{\mathbb{E}[T_C]}, \pi_S = \frac{\mathbb{E}[T_S]}{\mathbb{E}[T_C]}, \pi_I = \frac{\mathbb{E}[T_I]}{\mathbb{E}[T_C]}. \tag{8}$$

In the following, we will study how to obtain $\mathbb{E}[T_C]$, $\mathbb{E}[T_R]$, $\mathbb{E}[T_T]$, $\mathbb{E}[T_S]$, and $\mathbb{E}[T_I]$.

Under the *Queue with Starter* model, we can obtain the value of $\mathbb{E}[T_C]$. We define Starter T_X in a cycle T_C as a virtual job. Starter T_X will be the first job in a busy period T_B including T_X. Then by the relationship between the first job and the following busy period in [21, pp. 65], [20], we have

$$\mathbb{E}[T_B] = \frac{\mathbb{E}[T_X] + \mathbb{E}[S]}{1 - \lambda\mathbb{E}[S]}. \tag{9}$$

We know $T_C = T_B + \delta_i$, where δ_i is a idle period in an ordinary M/G/1 model, which follows the exponential distribution with a mean value $\frac{1}{\lambda}$. Therefore, the mean duration of a cycle can be written as

$$\mathbb{E}[T_C] = \frac{\mathbb{E}[T_X] + \frac{1}{\lambda}}{1 - \lambda\mathbb{E}[S]}. \tag{10}$$

We know that the probability that the server is in the running mode under `PowerSleep` is always equal to the server's server utilization $\lambda\mathbb{E}[S]$, then the mean running duration in a cycle is

$$\mathbb{E}[T_R] = \lambda\mathbb{E}[S]\mathbb{E}[T_C]. \tag{11}$$

When $\delta_i > \delta_h$, the system will also experience transition and sleep mode. The mean transition time is

$$\begin{align} \mathbb{E}[T_T] &= \int_{\delta_h}^{\infty} [\delta_s + \delta_w]\lambda e^{-\lambda t} dt \tag{12} \\ &= [\delta_s + \delta_w] e^{-\lambda\delta_h}. \tag{13} \end{align}$$

The mean sleep time is

$$\begin{align} \mathbb{E}[T_S] = &\int_{\delta_h}^{\delta_h+\delta_s} \delta_x \lambda e^{-\lambda t} dt \\ &+ \int_{\delta_h+\delta_s}^{\delta_h+\delta_s+\delta_e} [t - \delta_h - \delta_s + \delta_x]\lambda e^{-\lambda t} dt \\ &+ \int_{\delta_h+\delta_s+\delta_e}^{\infty} [\delta_e + \delta_x]\lambda e^{-\lambda t} dt \tag{14} \\ = &\frac{1}{\lambda} e^{-\lambda\delta_h}[\lambda\delta_x + e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]]. \tag{15} \end{align}$$

We also know that $T_C = T_R + T_S + T_T + T_I$, then the mean idle time can be written as

$$\mathbb{E}[T_I] = \mathbb{E}[T_C] - \mathbb{E}[T_R] - \mathbb{E}[T_S] - \mathbb{E}[T_T]. \tag{16}$$

Applying (10), (11), (13), (15), and (16) into (8), we have the result in Lemma 1. ■

With the probabilities in Lemma 1, we can easily obtain the mean power consumption as shown in the following lemma:

Lemma 2: In an M/G/1 server under `PowerSleep`, the mean power consumption of the server is

$$\mathbb{E}[P] = P_R(r)\pi_R + P_T\pi_T + P_S\pi_S + P_I\pi_I, \tag{17}$$

where $P_R(r)$ defined in (1), and π_R, π_T, π_I, and π_S are defined in Lemma 1.

Next we will investigate the response time under `PowerSleep`. It is shown in [19] that the additional delay in a queue introduced by a starter is independent of the response time in the system without starters. Using this independence property, it is then easy to calculate the total response time in the system with starters: it is simply the sum of the response time in the queue without starters plus the additional delay R_X introduced by starter. By the traditional M/G/1/PS queue theory [22], the mean response time of a job in an M/G/1/PS server without starters is $\frac{\mathbb{E}[S]}{1-\lambda\mathbb{E}[S]}$. By [19], in an M/G/1/PS system with a job arrival rate λ and Starter T_X, the mean additional delay introduced by Starter is

$$\mathbb{E}[R_X] = \frac{\mathbb{E}[T_X] + \frac{1}{2}\lambda\mathbb{E}[T_X^2]}{1 + \lambda\mathbb{E}[T_X]}. \tag{18}$$

We summarize it in the following lemma:

Lemma 3: In an M/G/1/PS server under `PowerSleep` with Starter T_X, a job arrival rate λ, and a generalized service time distribution with a given mean value $\mathbb{E}[S]$, the mean response time of a job is

$$\mathbb{E}[R] = \frac{\mathbb{E}[S]}{1 - \lambda\mathbb{E}[S]} + \mathbb{E}[R_X], \tag{19}$$

where $\mathbb{E}[R_X]$ is defined in (18).

With Lemmas 2 and 3, we obtain the mean power consumption and mean response time. We observe that both of them rely on the values of $\mathbb{E}[S]$, $\mathbb{E}[T_X]$, and $\mathbb{E}[T_X^2]$, which depend on the values of δ_h, δ_x, δ_e, and r. In the following, we aim to obtain the explicit formula for each one.

B. Main Result

In order to evaluate the performance of power consumption and response time under `PowerSleep`, we need to obtain $\mathbb{E}[T_X]$ and $\mathbb{E}[T_X^2]$ used in Lemmas 2 and 3. First we have to find the formula for Starter T_X. By the definition of a starter in *Queue with Starter*, Starter T_X under `PowerSleep` includes the wake-up transition plus the procrastination sleep period δ_x and may also include the remaining portion of a suspend transition, which depends on the preceding idle-queue period δ_i before a new job arrival. Based on the Figure 1, we observe that there are following cases:

(a) $\delta_i \leq \delta_h$: After a new job arrives, the server will immediately serve the new job, and there is no starter;
(b) $\delta_h < \delta_i < \delta_h + \delta_s$: A new job arrives during the suspend transition, and the length of the starter is $\delta_h + \delta - \delta_i$;
(c) $\delta_h + \delta_s < \delta_i < \delta_h + \delta_s + \delta_e$: A new job arrives during the enforced sleep period δ_e, and the length of the starter is $\delta_x + \delta_w$;
(d) $\delta_h + \delta_s + \delta_e < \delta_i < \delta_h + \delta + \delta_e$: A new job arrives during the procrastination or wakeup period, and the length of the starter is $\delta_h + \delta + \delta_e - \delta_i$;
(e) $\delta_i \geq \delta_h + \delta + \delta_e$: A new job arrives at the idle mode, and there is no starter.

We summarize it with the following formula:

$$T_X = \begin{cases} \delta_h + \delta - \delta_i, & \text{if } \delta_h < \delta_i < \delta_h + \delta_s \\ \delta_x + \delta_w, & \text{if } \delta_h + \delta_s < \delta_i < \delta_h + \delta_s + \delta_e \\ \delta_h + \delta + \delta_e - \delta_i, & \text{if } \delta_h + \delta_s + \delta_e < \delta_i < \delta_h + \delta + \delta_e \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

where δ is defined as

$$\delta = \delta_s + \delta_x + \delta_w. \tag{21}$$

The preceding idle-queue period δ_i is the same as the idle period defined in an ordinary M/G/1 model, which follows the exponential distribution with a mean value $\frac{1}{\lambda}$. Therefore, for T_X defined in (20), its mean value and variance can be obtained as:

$$\begin{aligned}\mathbb{E}[T_X] = & \int_{\delta_h}^{\delta_h+\delta_s} [\delta_h + \delta - t]\lambda e^{-\lambda t} dt \\ & + \int_{\delta_h+\delta_s}^{\delta_h+\delta_s+\delta_e} [\delta_x + \delta_w]\lambda e^{-\lambda t} dt \\ & + \int_{\delta_h+\delta_s+\delta_e}^{\delta_h+\delta+\delta_e} [\delta_h + \delta + \delta_e - t]\lambda e^{-\lambda t} dt \qquad (22) \\ = & e^{-\lambda\delta_h}[\delta - \frac{1}{\lambda} + \frac{1}{\lambda}[e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}] + e^{-\lambda[\delta+\delta_e]}]], \qquad (23)\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[T_X^2] = & \int_{\delta_h}^{\delta_h+\delta_s} [\delta_h + \delta - t]^2\lambda e^{-\lambda t} dt \\ & + \int_{\delta_h+\delta_s}^{\delta_h+\delta_s+\delta_e} [\delta_x + \delta_w]^2\lambda e^{-\lambda t} dt \\ & + \int_{\delta_h+\delta_s+\delta_e}^{\delta_h+\delta+\delta_e} [\delta_h + \delta + \delta_e - t]^2\lambda e^{-\lambda t} dt \qquad (24) \\ = & e^{-\lambda\delta_h}[[\delta - \frac{1}{\lambda}]^2 + \frac{1}{\lambda^2}[1 - 2e^{-\lambda[\delta+\delta_e]}] \\ & + \frac{2}{\lambda}[\delta_x + \delta_w - \frac{1}{\lambda}]e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]]. \qquad (25)\end{aligned}$$

Hence $1 + \lambda\mathbb{E}[T_X]$ and $\mathbb{E}[T_X] + \frac{\lambda}{2}\mathbb{E}[T_X^2]$ in Lemmas 2 and 3 can be written as

$$1 + \lambda\mathbb{E}[T_X] = e^{-\lambda\delta_h}\sigma, \tag{26}$$

$$\mathbb{E}[T_X] + \frac{\lambda}{2}\mathbb{E}[T_X^2] = e^{-\lambda\delta_h}[\frac{\lambda}{2}\delta^2 + [\delta_x + \delta_w]e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]], \tag{27}$$

where

$$\sigma = e^{\lambda\delta_h} + \lambda\delta - 1 + e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}] + e^{-\lambda[\delta+\delta_e]}. \tag{28}$$

Therefore, the probabilities defined in Lemma 1 can be written as

$$\pi_R = \frac{\rho}{r}, \tag{29}$$

$$\pi_T = [1 - \frac{\rho}{r}]\frac{\lambda[\delta_s + \delta_w]}{\sigma}, \tag{30}$$

$$\pi_S = [1 - \frac{\rho}{r}]\frac{\lambda\delta_x + e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]}{\sigma}, \tag{31}$$

$$\pi_I = 1 - \pi_R - \pi_T - \pi_S, \tag{32}$$

and $\mathbb{E}[R_X]$ in Lemma 3 can be written as

$$\mathbb{E}[R_X] = \frac{\frac{\lambda}{2}\delta^2 + [\delta_x + \delta_w]e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]}{\sigma}. \tag{33}$$

Recall that we assume $P_T = P_R(r)$. [1] Applying (3), (23) and (25) into Lemmas 2 and 3, with further mathematical manipulation, we have the following theorem:

Theorem 1 (`PowerSleep`*):* In an M/G/1/PS server under `PowerSleep`, the mean power consumption is

$$\begin{aligned}\mathbb{E}[P] = & \alpha[r - r_l]^\gamma[\frac{\rho}{r} + [1 - \frac{\rho}{r}]\frac{\lambda[\delta_s + \delta_w]}{\sigma}] + P_I \\ & - [P_I - P_S][1 - \frac{\rho}{r}]\frac{\lambda\delta_x + e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]}{\sigma}, \qquad (34)\end{aligned}$$

and the mean response time of a job is

$$\mathbb{E}[R] = \frac{1}{\mu[r - \rho]} + \frac{\frac{\lambda}{2}\delta^2 + [\delta_x + \delta_w]e^{-\lambda\delta_s}[1 - e^{-\lambda\delta_e}]}{\sigma}, \tag{35}$$

where δ and σ are defined in (21) and (28) respectively.

C. Remarks

Even though we focus in our study by far on the PS job scheduling police and the heavy-tailed job service time distribution, the proposed methodology can still work for other settings. We observe that in the analysis for `PowerTail`, the power consumption analysis is based on M/G/1 queueing model and is independent of the underlying scheduling policy and the job service time distribution. This will not hold for the response time analysis. We need to make some corresponding changes.

For instance, if First-Come-First-Served (FCFS) is adopt, with M/G/1/FCFS queueing theory, the mean response time of a job without starters will be revised as $\frac{\lambda\mathbb{E}[S^2]}{2[1-\lambda\mathbb{E}[S]]} + \mathbb{E}[S]$ if the second moment $\mathbb{E}[S^2]$ is also known in addition to $\mathbb{E}[S]$.

In the performance evaluation, we will also consider two baseline power-saving schemes: `PowerIdle` and `PowerNap`, *both of which are special cases of* `PowerSleep`. With `PowerIdle`, only DVS is adopted to change the execution speed. The server stays in only two different modes alternatively: running and idle. Therefore, the power consumption and the response time can be easily derived by setting $\delta_h = \infty$ in Theorem 1. `PowerNap` in [15] is a special case of `PowerSleep` by setting $r = 1$, $\delta_h = \delta_x = 0$, and $\delta_e = \infty$ in Theorem 1. The formulas are shown as in the following:

Corollary 1 (`PowerIdle` *and* `PowerNap`*):* In an M/G/1/PS server under `PowerIdle`, the mean power consumption is

$$\mathbb{E}[P] = \alpha[r - r_l]^\gamma\frac{\rho}{r} + P_I, \tag{36}$$

and the mean response time of a job is

$$\mathbb{E}[R] = \frac{1}{\mu[r - \rho]}. \tag{37}$$

In an M/G/1/PS server under `PowerNap`, the mean power consumption is

$$\begin{aligned}\mathbb{E}[P] = & P_S + [\alpha[1 - r_l]^\gamma + P_I - P_S] \\ & [1 - [1 - \rho]\frac{1}{\lambda[\delta_s + \delta_w]e^{\lambda\delta_s} + 1}], \qquad (38)\end{aligned}$$

[1]Other transition power consumption model can be applied here too.

and the mean response time of a job is

$$\mathbb{E}[R] = \frac{1}{\mu[1-\rho]} + \frac{\frac{\lambda}{2}[\delta_s+\delta_w]^2 e^{\lambda\delta_s} + \delta_w}{\lambda[\delta_s+\delta_w]e^{\lambda\delta_s}+1}. \qquad (39)$$

In Corollary 1, under `PowerIdle`, we observe that $\mathbb{E}[P]$ is an increasing function in terms of r as $\gamma \geq 1$, and $\mathbb{E}[R]$ is a decreasing function in terms of r. The necessary condition for the stability of the system is that the relative server utilization has to be less than 1, i.e., $r > \rho$. Under `PowerNap`, both $\mathbb{E}[P]$ and $\mathbb{E}[R]$ are constant for given ρ.

V. Optimal Design

In this section, we will study the optimal design for power-saving scheme `PowerSleep`, in order to minimize the mean power consumption under a given mean response time constraint, where we choose a mean response time threshold $\hat{R}$. The optimization problem can easily be formulated as follows:

$$\text{minimize } \mathbb{E}[P] \qquad (40a)$$

$$\text{subject to } \mathbb{E}[R] \leq \hat{R}, \qquad (40b)$$

$$\max\{r_l, \rho\} \leq r \leq 1. \qquad (40c)$$

Inequality (40c) is based on the low bound of r ($r_l \leq r \leq 1$) and the stability condition of a server ($r > \rho$).

The optimization problem defined in (40) can in general be solved with a Lagrangian function

$$L = \mathbb{E}[P] + \chi\mathbb{E}[R]. \qquad (41)$$

Through Lagrangian function (41), we notice that the following three problem settings are dual to each other: (i) Minimizing the power consumption subject to response time threshold (in this paper), (ii) Minimizing the response time subject to power consumption budget [9], and (iii) Minimizing the combination of the response time and power consumption [10]. Therefore, our method can be easily applied to the other two problem settings.

We set $\frac{\partial L}{\partial r} = 0$, $\frac{\partial L}{\partial \delta_h} = 0$, $\frac{\partial L}{\partial \delta_e} = 0$, and $\frac{\partial L}{\partial \delta_x} = 0$. Together with (40b) where the optimal value will be achieved at the boundary, we are able to obtain 5 variables (r, δ_h, δ_e, δ_x, and χ. We denote r^*, δ_h^*, δ_e^*, and δ_x^* as the corresponding optimal values. We summarize the main result in the following theorem:

Theorem 2 (`PowerSleep`*):* In an M/G/1/PS server under `PowerSleep`, the minimal power consumption $\mathbb{E}[P^*]$ under the mean response time threshold $\hat{R}$ can be achieved with an optimal configuration of r^*, δ_h^*, δ_e^*, and δ_x^*.

As most commercial computers nowadays only have discrete number of available speeds, it is also important to decide the speed for execution for such cases. By sequential search, it is quite straightforward to extend the results in Section V by only considering those available speeds.

Under `PowerNap`, all variables (r, δ_h, δ_e, and δ_x) are fixed. There is no option for optimal design. Under `PowerIdle`, based on (36) in Corollary 1, $\mathbb{E}[P]$ is an increasing function in terms of r. Obviously the minimal power consumption is achieved at the smallest possible r. By (40b), (40c) and (37), we have $r \geq r^*$, where

$$r^* = \max\{r_l, \frac{1}{\mu\hat{R}} + \rho\}. \qquad (42)$$

Therefore, we have the following corollary on the optimal result:

Corollary 2 (`PowerIdle`*):* In an M/G/1/PS server under `PowerIdle`, the minimal power consumption under the mean response time threshold $\hat{R}$ is $\mathbb{E}[P^*] = \alpha[r^* - r_l]^\gamma \frac{\rho}{r^*} + P_I$ as $r^* \leq 1$, where r^* is defined in (42).

The mean response time threshold will always be violated as $r^* > 1$ or $\hat{R} > \frac{1}{\mu}$. The upper-bound of the feasible server utilization is

$$\rho_u = 1 - \frac{1}{\mu\hat{R}}. \qquad (43)$$

If $r^* \leq 1$ and $\hat{R} \leq \frac{1}{\mu}$, the feasible ρ is in $[0, \rho_u]$. By denoting $\rho_m = [r_l - \frac{1}{\mu\hat{R}}]^+$ as an intermediate value of ρ, we consider the following cases for the value of $\mathbb{E}[P^*]$:

- As $\rho \in [0, \rho_m]$, we have $r^* = r_l$, then $\mathbb{E}[P^*] = P_I$ keeps constant.
- As $\rho \in [\rho_m, \rho_u]$, we have $r^* = \frac{1}{\mu\hat{R}} + \rho$, then $\mathbb{E}[P^*] = \alpha[r^* - r_l]^\gamma \frac{\rho}{r^*} + P_I$, which increases as ρ increases.

Therefore, $\mathbb{E}[P^*]$ is a non-decreasing function in terms of ρ as $r^* \leq 1$ and $\hat{R} \leq \frac{1}{\mu}$.

Remarks: As the workload in servers changes over time, we have to calculate the optimal solution for (40) dynamically. To reduce the on-line overhead, one reasonable approach is to build a look-up table so that the system only has to refer to the table for selecting the suitable configuration without violating the constraints. The optimal configuration with r^*, δ_h^*, δ_e^*, and δ_x^* for specified ρ and $\hat{R}$ can be calculated in an off-line manner, and stored in the look-up table. In general, we can assume that the mean response time threshold is specified during the design time. Therefore, the look-up table only has to be built for different requests rates or workload characteristics. With the look-up table, the on-line overhead will become negligible.

VI. Performance Evaluation

This section presents performance evaluation of the proposed power-saving scheme `PowerSleep` with optimal design, in comparison with the baseline scheme `PowerNap` [15] and `PowerIdle`. The optimal design for `PowerIdle` can be obtained by setting $\delta_s = \delta_w = 0$ and $P_S = P_I$ used in `PowerNap`. Recall that `PowerNap` in [15] is a special case of `PowerSleep` by setting $r = 1$, $\delta_h = \delta_x = 0$, and $\delta_e = \infty$.

The power consumption model used in the evaluation is based on the power profile of servers in [9], which includes two cases:

- *Linear-power server*: $\gamma = 1$, $r_l = 0.4$, $\alpha = 100$ Watt, $P_I = 180$ Watt, and
- *Cubic-power server*: $\gamma = 3$, $r_l = 0.4$, $\alpha = 455$ Watt, $P_I = 150$ Watt.

As PowerSleep model requires timing overhead for mode transition, we consider the following cases:

- $\delta_s = \delta_w = 0.01$ sec, $P_S = \frac{1}{5}P_I$, and
- $\delta_s = \delta_w = 0.1$ sec, $P_S = \frac{1}{10}P_I$, and
- $\delta_s = \delta_w = 0.5$ sec, $P_S = \frac{1}{20}P_I$,

where the first one has higher power consumption in the sleep mode but requires faster transitions.

Our performance evaluation focuses on the power management of a web server with web applications. For fair comparison, we adopt the application specification from [15], where the mean job execution time on the server at the maximum speed is $\frac{1}{\mu} = 0.028$ sec [2]. We consider two cases by (i) fixing the mean response time constraint $\hat{R}$ and varying the server utilization ρ, and (ii) fixing the server utilization ρ and varying the mean response time constraint $\hat{R}$. For each configuration, we report the optimal mean power consumption under PowerNap, PowerIdle, and PowerSleep.

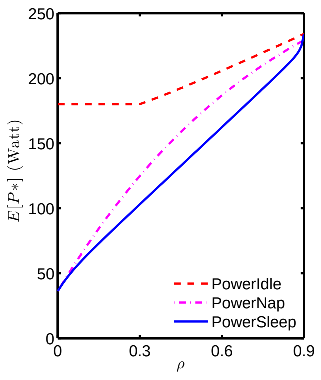
A. Evaluation Results by Fixing Mean Response Time Constraint

Figure 2 presents the optimal mean power consumption with respect to the varying server utilization ρ under PowerNap, PowerIdle, and PowerSleep when the mean response time constraint $\hat{R}$ is fixed as $\frac{10}{\mu} = 0.28$ sec. Correspondingly, the optimal δ_h^*, δ_x^*, δ_e^*, and r^* for PowerSleep are shown in Figure 3. Note that, when $\delta_s^* = \delta_x^* = 0$, the value of δ_h^* could be arbitrary, and hence, δ_h^* is set to 0 for such cases when presenting Figure 3. Given the fact that average server utilization is only 20$\sim$30% in typical data centers [7], [11], [15], in particular we also report in Table I the detailed power consumption improvement of PowerSleep over PowerNap and PowerIdle for a server with $\delta_s = \delta_w = 0.1$ sec as $\rho = 0.20, 0.25$, and 0.30.

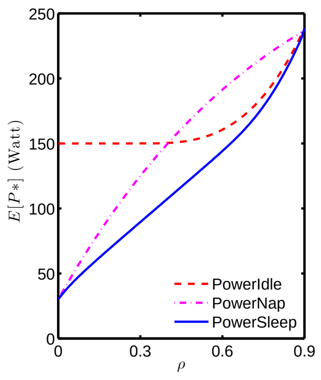
We first compare the power consumption under PowerSleep and PowerNap.

- As shown in Figure 2, PowerSleep is always better than PowerNap since PowerNap is a special case of PowerSleep. In particular, the improvement is significant when δ_s and δ_w are large. For instance, as $\rho = 0.25$ and $\delta_s = \delta_w = 0.1$ sec, for linear-power servers, PowerSleep has 33.7% power reduction rate over PowerNap; for cubic-power servers, the power reduction rate is 49.6% as shown in Table I.
- However, for both linear-power and cubic-power servers, the power consumption under PowerNap is close to that under PowerSleep when the server utilization is either very low (less than 0.05) or very high (more than 0.85). When ρ is very low, the transition power consumption dominates the mean power consumption. For such cases, the server usually is put to the transition mode right after serving one or two jobs. The procrastination idea behind PowerSleep might help, but the low utilization also implies the less probability to aggregate more jobs for joint execution. Similarly, when the server utilization is very high, in order to aggregate jobs for joint execution by procrastination, one has to use a higher speed ratio, which consumes more power (especially for the cubic-power server) for job execution. Therefore, when ρ is very high, PowerSleep performs exactly the same as PowerIdle, as shown in Figure 2(d) for $\rho \geq 0.63$, Figure 2(e) for $\rho \geq 0.31$, and Figure 2(f) for $\rho \geq 0.44$. This is because the optimal speed ratio is very close to

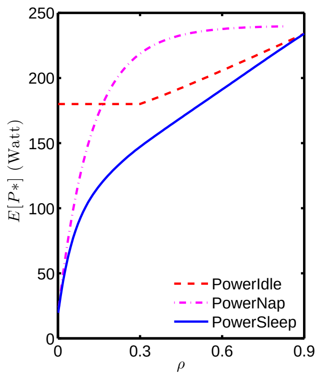
[2]In [15], the mean busy interval and idle interval for the measured web workload are 0.038 sec and 0.106 sec. With the traditional queueing theory, the mean job execution time is $\frac{0.038*0.106}{0.038+0.106} = 0.028$ sec.



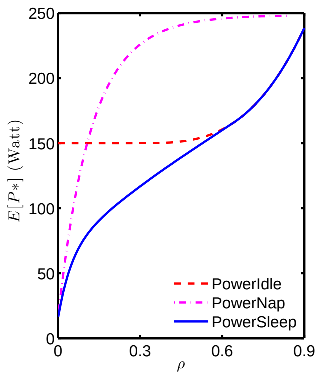
(a) linear-power server with $\delta_s = \delta_w = 0.01$ sec



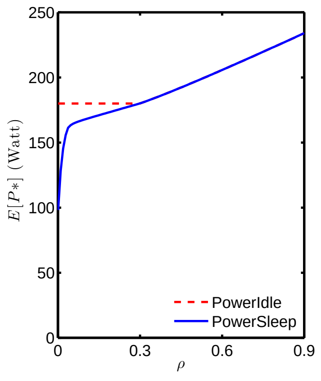
(b) cubic-power server with $\delta_s = \delta_w = 0.01$ sec



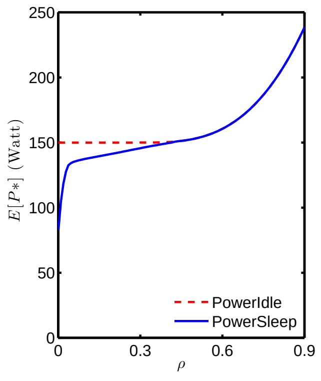
(c) linear-power server with $\delta_s = \delta_w = 0.1$ sec



(d) cubic-power server with $\delta_s = \delta_w = 0.1$ sec



(e) linear-power server with $\delta_s = \delta_w = 0.5$ sec



(f) cubic-power server with $\delta_s = \delta_w = 0.5$ sec

Fig. 2. Power consumption comparison for $\hat{R} = \frac{10}{\mu}$.

TABLE I
POWER CONSUMPTION IMPROVEMENT OF `PowerSleep` OVER `PowerNap` AND `PowerIdle` FOR A SERVER WITH $\delta_s = \delta_w = 0.1$ SEC.

ρ	Linear-power Server		Cubic-power Server	
	Over `PowerNap`	Over `PowerIdle`	Over `PowerNap`	Over `PowerIdle`
0.20	33.9%	28.5%	50.1%	33.2%
0.25	33.7%	23.0%	49.6%	27.5%
0.30	32.7%	18.3%	48.4%	22.2%

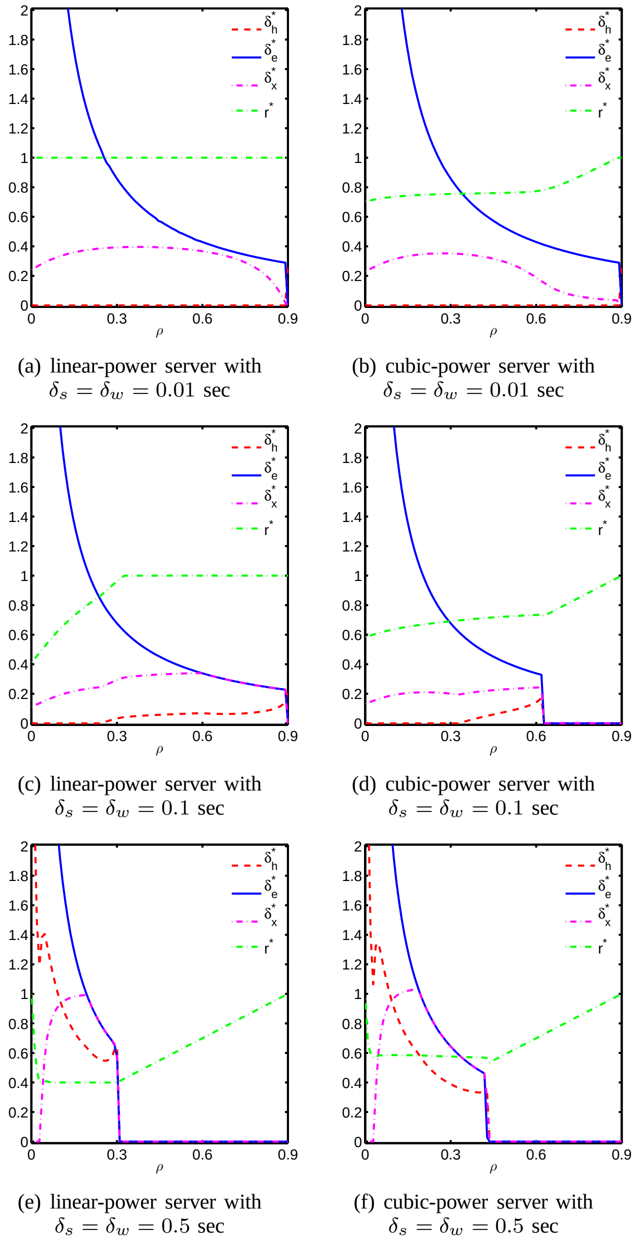
1 and there is no space for procrastination sleep period due to the response time constraint. Therefore, as shown in Figures 3(d), 3(e), and 3(f), the optimal solutions would set $\delta_s^* = \delta_x^* = 0$. which shows the equivalence of `PowerSleep` to `PowerIdle` when ρ is large enough.

- In the case of $\delta_s = \delta_w = 0.5$ sec as shown in Figure 2(e) and Figure 2(f), we observe that there are no feasible solutions for `PowerNap`. by the design of `PowerNap`, the response time will include one wakeup time $\delta_w = 0.5$ sec, which is larger than the response time threshold $\hat{R} = \frac{10}{\mu} = 0.28$ sec. Under this scenario, `PowerSleep` can still work.

Next, we compare the power consumption under `PowerSleep` and `PowerIdle`.

- `PowerSleep` outperforms `PowerIdle` when the server utilization is low as shown in Figure 2. In particular, the improvement is significant when δ_s and δ_w are large. For instance, as $\rho = 0.25$ and $\delta_s = \delta_w = 0.1$ sec, for linear-power servers, `PowerSleep` has 23.0% power reduction rate over `PowerIdle`; for cubic-power servers, the power reduction rate is 27.5% as shown in Table I. `PowerIdle` might outperform `PowerSleep` when the server utilization is high, such as the scenarios in Figure 2(c) with $\rho > 0.8$ and Figure 2(d) with $\rho > 0.55$.
- When the server utilization is lower than $r_l - \frac{1}{\mu\hat{R}} = 0.3$, the mean power consumption under `PowerIdle` is a constant since the server would like to execute at the lower bound r_l of the speed ratio. The optimal solution of `PowerIdle` always tries to use the minimal speed to meet the mean response time constraint. When the server utilization is high than 0.3, the mean power consumption under `PowerIdle` becomes a linear (cubic, respectively) function for the linear-power (cubic-power, respectively) server. Note that the results in Figure 2 also suggests that power management for the server should be `PowerIdle` and `PowerSleep` dynamically, depending on the server utilization.

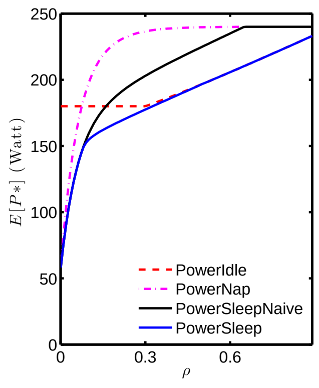
The trends in Figure 3 show that the optimal solution under `PowerSleep` have to jointly choose the speed ratio and the procrastination sleep period such that the mean power consumption in running, transition, and sleep modes is balanced.



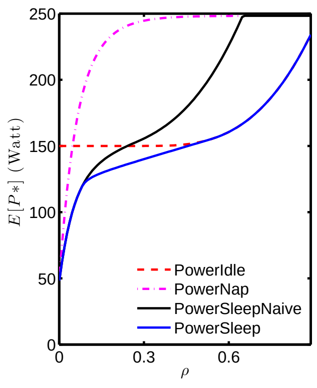
(a) linear-power server with $\delta_s = \delta_w = 0.01$ sec
(b) cubic-power server with $\delta_s = \delta_w = 0.01$ sec
(c) linear-power server with $\delta_s = \delta_w = 0.1$ sec
(d) cubic-power server with $\delta_s = \delta_w = 0.1$ sec
(e) linear-power server with $\delta_s = \delta_w = 0.5$ sec
(f) cubic-power server with $\delta_s = \delta_w = 0.5$ sec

Fig. 3. Optimal δ_h^*, δ_x^*, δ_e^*, and r^* for $\hat{R} = \frac{10}{\mu}$.

We also conducted the performance comparison of `PowerSleep` in this manuscript and the one in the conference version [23]. In order to differentiate them, we rename the one in the conference version as `PowerSleepNaive`. `PowerSleepNaive` is a special case of `PowerSleep` by setting $\delta_h = 0$ and $\delta_e = \infty$. The performance difference between `PowerSleepNaive` and `PowerSleep` depends on the sleep transition overhead (δ_s, δ_w) and the arrival rate ρ. We have conducted a comprehensive evaluation under different scenarios in terms of different sleep transition overheads and



(a) linear-power server with $\delta_s = \delta_w = 0.2$ sec



(b) cubic-power server with $\delta_s = \delta_w = 0.2$ sec

Fig. 4. Power consumption comparison between `PowerSleepNaive` and `PowerSleep` for $\hat{R} = \frac{10}{\mu}$.

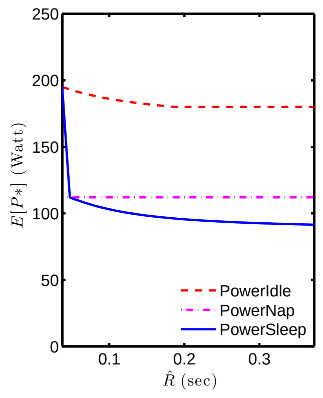
different arrival rates for both linear-power and cubic-power models. Due to the similarity, here we only show the following typical scenarios:

- $\delta_s = \delta_w = 0.01$ sec: In this scenario, the performance is the same for both `PowerSleepNaive` and `PowerSleep` in either linear-power server or cubic-power server, which is the same as shown in Figures 2(a) and 2(b). Since the transition overhead is relatively small, the optimal idle period threshold in `PowerSleep` is set as $\delta_h = 0$. Hence `PowerSleep` is reduced to `PowerSleepNaive`.
- $\delta_s = \delta_w = 0.2$ sec: The comparison is shown in Figure 4. In this scenario, the transition overheads ($\delta_s = \delta_w$) is shorter than the mean response time constraint $\hat{R} = 0.28$, but relatively large. Then δ_h is positive, i.e., it is better to keep the server in the idle power mode when the idle-queue duration is short. We observe that `PowerSleep` outperforms `PowerSleepNaive` for the high arrival rate.
- $\delta_s = \delta_w = 0.5$ sec: In this scenario, the transition overhead ($\delta_s = \delta_w$) is larger than the mean response time constraint $\hat{R} = 0.28$. `PowerSleepNaive` is infeasible since each job shall experience at least the wake-up transition delay δ_w and the mean response time inevitably exceeds the threshold. But in `PowerSleep`, we could choose a proper δ_e and put the server back from the sleep power mode to the idle or running power mode early and precautiously, which results in better feasibility of the timing constraint as shown in Figures 2(e) and 2(f).

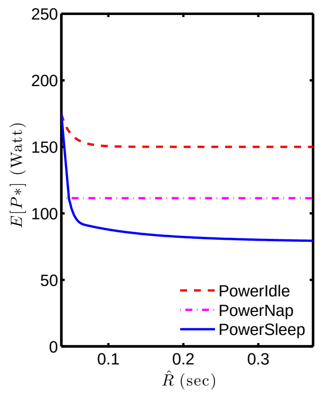
B. Evaluation Results by Fixing Server Utilization

Given the fact that average server utilization is only 20∼30% in typical data centers [7], [11], [15], we consider the case that the server utilization is fixed as $\rho = 0.25$. Figure 5 presents the mean power consumption with respect to different mean response time constraints $\hat{R}$ under `PowerNap`, `PowerIdle`, and `PowerSleep` when $\rho = 0.25$. $\hat{R}$ varies from $\hat{R}_{min}$ to $10 * \hat{R}_{min}$, where $\hat{R}_{min}$ is the minimum response time as $\rho = 0.25$. The trends for their corresponding optimal r^*, δ_h^*, δ_e^*, and δ_x^* are skipped here due to similarity with Figure 3.

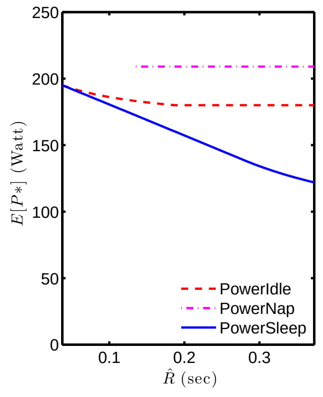
As the mean response time constraint $\hat{R}$ increases, the power consumption decreases for `PowerIdle` and `PowerSleep` schemes but keeps constant for `PowerNap`. In `PowerNap`, $r = 1$ and $\delta_x = 0$ are fixed, the response time keeps constant as the server utilization is fixed. Therefore, with a looser mean response time constraint, there is no space for



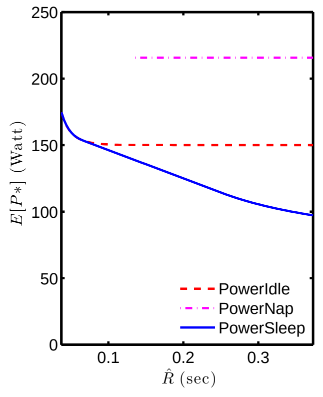
(a) linear-power server with $\delta_s = \delta_w = 0.01$ sec



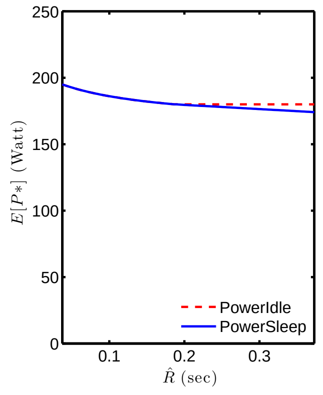
(b) cubic-power server with $\delta_s = \delta_w = 0.01$ sec



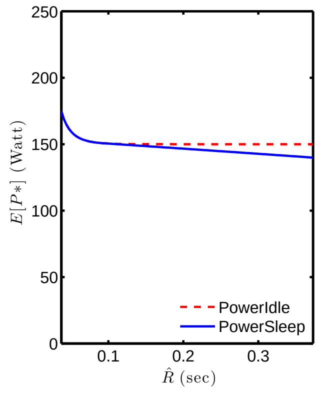
(c) linear-power server with $\delta_s = \delta_w = 0.1$ sec



(d) cubic-power server with $\delta_s = \delta_w = 0.1$ sec



(e) linear-power server with $\delta_s = \delta_w = 0.5$ sec



(f) cubic-power server with $\delta_s = \delta_w = 0.5$ sec

Fig. 5. Power consumption comparison for $\rho = 0.25$.

PowerNap to reduce power consumption. For PowerIdle and PowerSleep, as $\hat{R}$ increases, the optimal choice of r and δ_x takes effects on the power consumption. We also observe that PowerSleep outperforms PowerIdle for all cases with small δ_s and δ_w and for cases with large $\hat{R}$ and large δ_s and δ_w. The fundamental reason is same as the one explained in the previous subsection.

VII. Conclusion

This paper explores how to minimize the mean power consumption in a server under the mean response time constraint for reducing the power cost. We proposed PowerSleep, a smart power-saving schemes, which applies both DVS and DPM to put the server to a low-power sleep mode. By adopting the extended M/G/1/PS queuing model for job arrival and execution, we present how to jointly decide the execution speed for jobs and the sleep period such that the mean response time constraint is satisfied and the mean power consumption is minimized. Simulation results reveal the effectiveness and efficiency of the proposed schemes with comparisons to two baseline schemes PowerNap and PowerIdle.

The focus of this paper is on one server. For systems with multiple homogeneous servers, our approaches in this paper can be extended to decide the number of servers to evenly assign jobs (workload). When the number of activated servers is small, there might not exist a feasible solution to meet the mean response time constraint or the execution speeds are too large so that the mean power consumption is too high. On the other hand, when the number of activated servers is large, the activated servers might waste too much power for mode transitions since the server utilization might be too low. Therefore, we have to activate a proper number of servers and evenly distribute the server utilization. Moreover it is also not difficult to extend the schemes and approaches in this paper to consider systems with multiple low-power modes for idling, e.g., standby/sleep/shutdown.

Acknowledgment

An earlier version of this work was published in the Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS), July 2010 [23]. This work is sponsored in part by NSF CAREER Grant No. CNS-0746906, the European Community's Seventh Framework Programme FP7/2007-2013 project Predator (Grant 216008), NSERC Discovery Grant 341823-07, NSERC Strategic Grant STPGP 364910-08, and FQRNT grant 2010-NC-131844.

References

[1] APC (American Power Conversion), "Determining total cost of ownership for data center and network room infrastructure," 2003, http://www.apcmedia.com/salestools/CMRP-5T9PQG_R3_EN.pdf.

[2] U. EPA, "Report to congress on server and data center energy efficiency public law 109-431. us environmental protection agency energy star program."

[3] A. Kamra, V. Misra, and E. Nahum, "Yaksha: a self-tuning controller for managing the performance of 3-tiered web sites," in *IEEE International Workshop on Quality of Service (IWQOS)*, June 2004, pp. 47–56.

[4] X. Liu, L. Sha, Y. Diao, S. Froehlich, J. L. Hellerstein, and S. Parekh, "Online response time optimization of apache web server," in *IEEE International Workshop on Quality of Service (IWQOS)*, 2003, pp. 461–478.

[5] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher, "Queueing model based network server performance control," in *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, 2002, pp. 81–90.

[6] X. Liu, J. Heo, L. Sha, and X. Zhu, "Queueing-model-based adaptive control of multi-tiered web applications," *IEEE Transactions on Network and Service Management*, vol. 5, no. 3, pp. 157–167, September 2008.

[7] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The case for power management in web servers," *Power Aware Computing*, pp. 261–289, 2002.

[8] V. Sharma, A. Thomas, T. F. Abdelzaher, K. Skadron, and Z. Lu, "Power-aware QoS management in web servers," in *IEEE International Real-Time Systems Symposium*, 2003.

[9] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *ACM SIGMETRICS*, 2009.

[10] A. Wierman, L. L. H. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *IEEE INFOCOM*, 2009.

[11] L. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.

[12] X. Fan, W. Weber, and L. Barroso, "Power provisioning for a warehouse-sized computer," in *International symposium on Computer architecture*, 2007.

[13] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," in *International Conference on Autonomic Computing*, 2007.

[14] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling" cool": temperature-aware workload placement in data centers," in *USENIX Annual Technical Conference*, 2005.

[15] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *International conference on Architectural support for programming languages and operating systems*, 2009.

[16] L. Wang and Y. Lu, "Efficient Power Management of Heterogeneous Soft Real-Time Clusters," in *Real-Time Systems Symposium, 2008*, 2008, pp. 323–332.

[17] X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive Control of Multi-Tiered Web Applications Using Queueing Predictor," in *IEEE/IFIP Network Operations and Management Symposium*, 2006.

[18] J. Heo, D. Henriksson, X. Liu, and T. Abdelzaher, "Integrating Adaptive Components: An Emerging Challenge in Performance-Adaptive Systems and a Server Farm Case-Study," in *IEEE International Real-Time Systems Symposium*, 2007.

[19] H. Levy and L. Kleinrock, "A queue with starter and a queue with vacations: delay analysis by decomposition," *Operations Research*, vol. 34, no. 3, pp. 426–436, 1986.

[20] P. Welch, "On a generalized M/G/1 queuing process in which the first customer of each busy period receives exceptional service," *Operations Research*, vol. 12, no. 5, pp. 736–752, 1964.

[21] L. Takacs, *Introduction to the Theory of Queues*. Oxford University Press New York, 1962.

[22] L. Kleinrock, *Queueing Systems Volume II: Computer applications*. Wiley Interscience, 1976.

[23] S. Wang, J. Chen, J. Liu, and X. Liu, "Power saving design for servers under response time constraint," in *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, 2010, http://www.umd.umich.edu/~shqwang/papers/ECRTS10-sleep.pdf.