Power Saving Design 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
Computer Engineering and Networks Laboratory (TIK)
ETH Zurich, Switzerland
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*—The electricity cost becomes a very significant portion of the total cost of ownership for the maintenance of servers. Reducing the power consumption while maintaining the response time constraint has been one of important goals 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 an efficient technique 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. We propose a smart `PowerSleep` power-saving scheme, where a procrastination technique is adopted to carefully aggregate jobs to reduce the overhead of transitions in and out of the sleep mode. We also observe that `PowerSleep` might not always be a good choice due to the mode transition overhead when the server utilization is high. Instead we use `PowerIdle` power-saving scheme, which uses only DVS. By modeling the system with M/G/1/PS queuing model and further extensions, we present how to minimize the mean power consumption of the server under the given mean response time constraint for both power-saving schemes. 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, for maintaining a cluster with hundreds of servers, the electricity cost is significant. 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]. The advanced hardware technology has improved the performance per hardware dollar, but the performance per watt remains roughly flat over time [2]. As a result, the electricity cost of server clusters will exceed the hardware cost and become a very significant portion of the total cost of ownership for the maintenance of servers. By 2011, data centers in U.S. are expected to consume around 100 billion kWh per year [3], 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 of servers has been an active research [4]–[7]. 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 [8], [9] to reduce the energy consumption by applying DVS with minimal performance impact. In [10], 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 [11] to balance the mean energy consumption and mean response time under processor sharing 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 [12]–[14], 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 subsystems is counted, which could increase power consumption by 50∼100% [15]. Given the fact that average server utilization is only 20∼30% in typical data centers [8], [12], [16], 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 [16]. 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 [16], 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. 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 [16], one of the goals is to achieve fast transitions and reduce each transition duration. 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 a `PowerSleep` 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. Specifically, we introduce a *procrastination* technique for carefully aggregating jobs to reduce the number of mode transitions: When a new job arrives at the empty queue, the system could continue to stay in the sleep power mode for extra time to aggregate more jobs to reduce the mode transition frequency. We will show that `PowerSleep` outperforms `PowerNap` significantly, in particular when the single mode transition overhead is large. However, fundamentally, the mode transition overhead cannot be eliminated, `PowerSleep` might not work well sometimes when the server utilization is high. In this case, we better use the pure DVS design without using the sleep power mode, which we define as `PowerIdle` power-saving scheme. In `PowerIdle`, we will present how to efficiently choose the execution speed to minimize the mean power consumption under a given mean response time constraint. 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 [4], [7], [10], [11], [17]–[19]. 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. Both `PowerSleep` and `PowerIdle` power-saving schemes 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

In this section, we define the system model. Based on the system model, we will investigate the power consumption and the response time of jobs in the next section.

We adopt the M/G/1/PS server model [4], [7], [10], [11], [17]–[19]. 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 Process Sharing (PS) job 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 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. Then we have $r_l \leq r \leq 1$. 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, when the server is idle, 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 [16], 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 [10]. The server has the following power modes:

- *Running power mode:* In the running mode, the power consumption $P_R(r)$ by the server at a speed ratio r is

$$P_R(r) = \alpha[r - r_l]^\gamma + P_I, \quad (1)$$

  where $\gamma \geq 1$ and P_I is the the static power consumption. 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 [10];
- *Idle power mode:* In the idle mode, the server consumes the static power P_I;
- *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.

III. Power-Saving Schemes

In the following, we describe in details the power-saving schemes, i.e., `PowerSleep` and `PowerIdle`.

In the design of `PowerSleep`, we set the server in sleep mode once the queue is empty. In order to design a better power-saving strategy, we try to answer the following questions: (i) What speed ratio can be set in the running state? [1] (ii) When should the server be waken up? The straightforward scheme is to choose the full speed in the running state, set the server to sleep once the queue is empty and is waken up upon a new job arrival as shown in [16]. This approach works well but with some limitations and it is not efficient.

- *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?
- *The transition from the running state into the sleep state might be too costly.* If the empty-queue duration is short, the server is wasting time in transition without taking enough duration of sleep (or execution of jobs). How will we control the transition frequency?

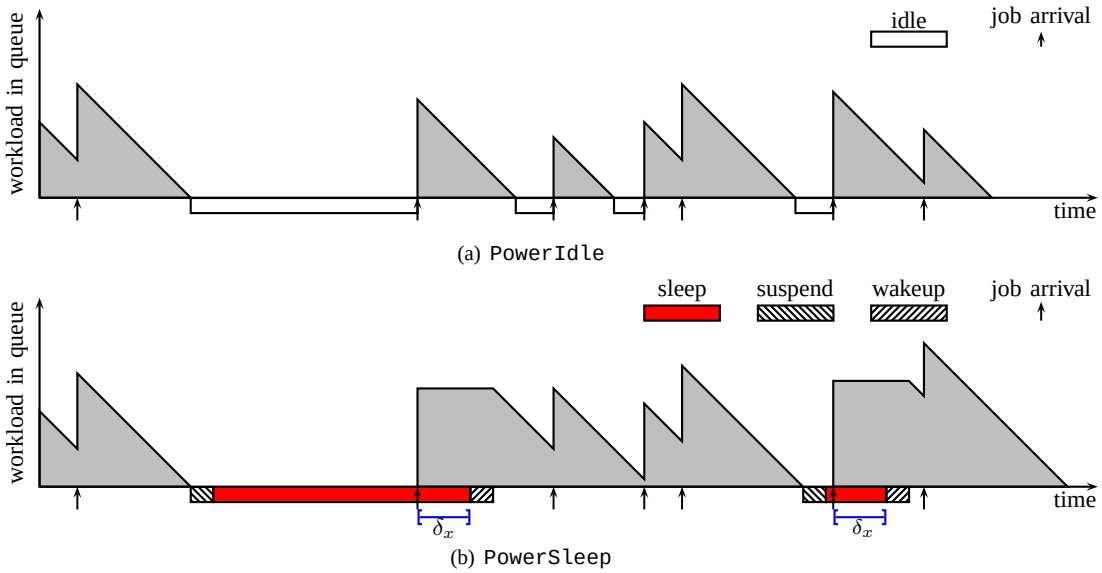
In the design of `PowerSleep`, we will address the above raised concerns. With `PowerSleep`, we utilize both DVS and DPM. In the running mode, a server will run at a constant speed ratio r. Once the queue is empty, the server will immediately be set to the sleep mode. It remains in the sleep mode if no new job arrives. When a job arrives at a sleeping server, it cannot be served immediately; rather the server is procrastinated in the sleep mode for an additional constant δ_x time units. Then the server will be waken up into the running mode. Recall that there are some timing overhead for the mode transitions between the running mode and the sleep mode. We denote δ_s and δ_w as the overheads for the *suspend* transition from the running mode to the sleep mode and the *wake-up* transition from the sleep mode back to the running mode respectively. In [16], 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 [16] is a special case of `PowerSleep` by setting $r = 1$ and $\delta_x = 0$.

`PowerIdle` is quite simple where only DVS is used: If there are jobs in the queue, the server will be in the running mode at a constant speed ratio r, otherwise the server stays in the idle mode. We ignore the mode transition overhead between the running mode and the idle mode since it is relatively small in comparison with the one between the running mode and the sleep mode.

Figure 1 illustrates the change of the server mode under `PowerSleep` and `PowerIdle` for the same given job arrivals. Under `PowerSleep` the server stays in three different modes: running, transition and sleep, while under `PowerIdle`, the server stays in two different modes alternatively: running and idle. As shown in Figure 1(b), the procrastination of the sleep period can aggregate job arrivals so as to reduce the mode transition frequency along with the wake-up transitions (sometimes suspend transitions as well). Slowing the execution speed can have the similar effect too. Therefore, choosing appropriate values of r and δ_x is the key to the efficiency of `PowerSleep` design.

In order to minimize the mean power consumption under the mean response time constraint, we could apply either `PowerSleep` or `PowerIdle`. Usually `PowerSleep` outperforms `PowerIdle`, in particular when the server utilization is low. However, as we will show later, in certain circumstance, `PowerIdle` is better than `PowerSleep` when the server encounters higher workload demands. That is, to meet the mean response time constraint for a stringent server, putting the server to the sleep mode could increase the mean power consumption. The server should then take the better one between the two solutions to minimize the

[1] This question should be answered in `PowerIdle` design too.



(a) `PowerIdle`

(b) `PowerSleep`

Figure 1. Examples for `PowerIdle` and `PowerSleep`

mean power consumption.

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 revise the tradition M/G/1/PS queueing theory because of the sleep and transition modes. Based on the result of `PowerSleep`, we can easily obtain the result of `PowerIdle`.

A. Main Results

Under `PowerSleep`, the server will stay in three different modes: running, transition, and sleep. With sleep and transition modes, the traditional queueing theory cannot be applied directly here. Instead we adopt a *Queue with Starter* model [20], [21]: 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, sleep or transition), which we define as π_R, π_S, and π_T 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

$$\pi_R = \lambda \mathbb{E}[S], \tag{2}$$

$$\pi_T = [1 - \lambda \mathbb{E}[S]] \frac{\lambda[\delta_s + \delta_w]}{1 + \lambda \mathbb{E}[T_X]}, \tag{3}$$

$$\pi_S = [1 - \lambda \mathbb{E}[S]] \left[1 - \frac{\lambda[\delta_s + \delta_w]}{1 + \lambda \mathbb{E}[T_X]}\right]. \tag{4}$$

The proof of Lemma 1 is included in the appendix.

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

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

$$\mathbb{E}[P] = P_R(r)\pi_R + P_T\pi_T + P_S\pi_S, \tag{5}$$

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

Next we will investigate the response time under `PowerSleep`. It is shown [20] 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 [20], 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

$$\mathbb{E}[R_X] = \frac{\mathbb{E}[T_X] + \frac{1}{2}\lambda\mathbb{E}[T_X^2]}{1 + \lambda\mathbb{E}[T_X]}. \quad (6)$$

Therefore, we have the following result:

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

$$\mathbb{E}[R] = \frac{\mathbb{E}[S]}{1 - \lambda\mathbb{E}[S]} + \mathbb{E}[R_X], \quad (7)$$

where $\mathbb{E}[R_X]$ is defined in (6).

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

$$\rho = \frac{\lambda}{\mu}. \quad (8)$$

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

$$\lambda\mathbb{E}[S] = \frac{\rho}{r}, \quad (9)$$

while ρ is the (absolute) server utilization with respect to the maximum speed.

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 (3), (4), and (6). 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 empty-queue period T_I before a new job arrival. Based on this definition, T_X can be written as:

$$T_X = [\delta_s - T_I]^+ + \delta_x + \delta_w. \quad (10)$$

where $z^+ = \max\{0, z\}$. If the empty-queue period T_I is shorter than the suspend transition period δ_s, the server needs to wait the completion of the suspend transition and goes back to the wakeup transition right away, therefore in this case $T_X = \delta_s - T_I + \delta_x + \delta_w$; otherwise, the server take $T_X = \delta_x + \delta_w$ time period before the next service.

By the definition of T_I, we know that T_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 (10), we have

$$\begin{aligned}\mathbb{E}[T_X] &= \int_0^\infty T_X \lambda e^{-\lambda t} dt \\ &= \int_0^{\delta_s} [\delta_s - t]\lambda e^{-\lambda t} dt + \delta_x + \delta_w \\ &= \delta - \frac{1}{\lambda}, \end{aligned} \quad (11)$$

$$\begin{aligned}\mathbb{E}[T_X^2] &= \int_0^\infty T_X^2 \lambda e^{-\lambda t} dt \\ &= \int_0^{\delta_s} [\delta - t]^2 \lambda e^{-\lambda t} dt + \int_{\delta_s}^\infty [\delta_x + \delta_w]^2 \lambda e^{-\lambda t} dt \\ &= [\delta - \frac{1}{\lambda}]^2 + \frac{1}{\lambda^2} - \sigma[\sigma + 2\delta_s], \end{aligned} \quad (12)$$

where σ and δ are defined as

$$\sigma = \frac{1}{\lambda} e^{-\lambda\delta_s} \text{ and } \delta = \delta_s + \delta_x + \delta_w + \sigma. \quad (13)$$

Recall that we assume $P_T = P_R(r)$. [2] Applying (9), (11) and (12) into Lemmas 2 and 3, with further mathematical manipulation, we have the following theorem:

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

$$\mathbb{E}[P] = P_S + [P_R(r) - P_S]\left[\frac{\rho}{r} + [1 - \frac{\rho}{r}]\frac{\delta_s + \delta_w}{\delta}\right], \quad (14)$$

and the mean response time of a job is

$$\mathbb{E}[R] = \frac{1}{\mu[r - \rho]} + \frac{1}{2}\left[\delta - \frac{\sigma[\sigma + 2\delta_s]}{\delta}\right]. \quad (15)$$

where ρ is defined in (8), and σ and δ are defined as

$$\sigma = \frac{1}{\rho\mu} e^{-\rho\mu\delta_s} \text{ and } \delta = \delta_s + \delta_x + \delta_w + \sigma. \quad (16)$$

In Theorem 1, we observe that $\mathbb{E}[P]$ is a decreasing functions in terms of δ_x and r, and $\mathbb{E}[R]$ is an increasing function in terms of δ_x but a decreasing function in terms of r.

Under `PowerIdle`, the server stays in only two different modes alternatingly: running and idle. So, the formulas of power consumption and response time can be easily derived by setting $\delta_s = \delta_w = 0$ and $P_S = P_I$ in Theorem 1. With further mathematical manipulation, we can obtain the following theorem:

Theorem 2: In an M/G/1/PS server under `PowerIdle`, the mean power consumption is

$$\mathbb{E}[P] = \alpha[r - r_l]^\gamma \frac{\rho}{r} + P_I, \quad (17)$$

and the mean response time of a job is

$$\mathbb{E}[R] = \frac{1}{\mu[r - \rho]}, \quad (18)$$

where ρ is defined in (8).

In Theorem 2, 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., $r > \rho$.

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

B. Remarks

If other job scheduling algorithms are adopted in the server, the methodology for the power consumption and response time analysis can still work with some corresponding changes. Based on the analysis for both `PowerSleep` and `PowerIdle`, the power consumption keeps the same for all work-conserving scheduling algorithms including PS and FCFS. However, the response time depends on the scheduling algorithm. For instance, if FCFS is used, with M/G/1/FCFS queueing theory, the mean response time of a job in the system without starter will be revised as $\frac{\lambda\mathbb{E}[S^2]}{2[1-\lambda\mathbb{E}[S]]}+\mathbb{E}[S]$, where the second moment $\mathbb{E}[S^2]$ is needed in addition to $\mathbb{E}[S]$. The addition delay $\mathbb{E}[R_X]$ in `PowerSleep` will remain the same. The mean response time in `PowerSleep` and `PowerIdle` could be revised accordingly.

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 Sections V-B and V-A by only considering those available speeds.

V. Optimal Design

In this section, we will study the optimal design for power-saving schemes of both `PowerSleep` and `PowerIdle`, 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:

$$\text{minimize } \mathbb{E}[P] \tag{19a}$$

$$\text{subject to } \mathbb{E}[R]\le\hat{R}, \tag{19b}$$

$$\max\{r_l,\rho\}\le r\le 1. \tag{19c}$$

Inequality (19c) is based on the low bound of r ($r_l\le r\le 1$) and the stability condition of a server ($r>\rho$).

Given the power consumption and response time analysis summarized in Theorems 1 and 2, we can adopt specific optimization approaches to obtain the optimal design for `PowerSleep` and `PowerIdle` individually.

A. PowerSleep

To minimize $\mathbb{E}[P]$, we can first fix r and consider variable δ_x only. By the linear relationship between δ and δ_x defined in (16), it will be equivalent to consider δ only.

Based on (14), $\mathbb{E}[P]$ is a decreasing function with respect to δ. Therefore, to minimize $\mathbb{E}[P]$, we only need to maximize δ. By denoting

$$\tilde{R}=\hat{R}-\frac{1}{\mu[r-\rho]}, \tag{20}$$

the constraint condition can be revised as: $\delta^2-2\tilde{R}\delta-\sigma[\sigma+2\delta_s]\le 0$. Then the optimization problem can be simplified as:

$$\text{maximize } \delta \tag{21a}$$

$$\text{subject to } \delta^2-2\tilde{R}\delta-\sigma[\sigma+2\delta_s]\le 0, \tag{21b}$$

$$\delta\ge\delta_s+\delta_w+\sigma, \tag{21c}$$

where Constraint (21c) comes from the definition of δ in (16).

Constraint (21b) is a quadratic inequality. Therefore, the maximal δ is achieved at $\delta=\delta^*$, where

$$\delta^*=\tilde{R}+\sqrt{\tilde{R}^2+\sigma[\sigma+2\delta_s]}, \tag{22}$$

as $\delta^*\ge\delta_s+\delta_w+\sigma$.

We apply the value of $\delta=\delta^*$ into $\mathbb{E}[P]$ defined in (14), then $\mathbb{E}[P]$ has only one variable r, which is constrained by (19c) and $\delta^*\ge\delta_s+\delta_w+\sigma$. The optimal $\mathbb{E}[P^*]$ and r^* can be obtained with an ordinary optimization tool. In our evaluation, we use `fmincon` tool in Matlab. Moreover, by (22), we can obtain δ^* and hence δ_x^* too. We summarize the result in the following theorem:

Theorem 3: In an M/G/1/PS server under `PowerSleep`, there exist optimal values of r^* and δ_x^* such that the power consumption can be minimized as $\mathbb{E}[P^*]$ while the mean response time is below the mean response time threshold $\hat{R}$.

Under `PowerSleep`, the upper-bound of the feasible server utilization is ρ_u

$$\rho_u=1-\frac{2}{\mu\left[2\hat{R}-[\delta_s+\delta_w+\sigma]+\frac{\sigma[\sigma+2\delta_s]}{\delta_s+\delta_w+\sigma}\right]}, \tag{23}$$

which is obtained by considering the constraints in (21b) and (21c).

B. PowerIdle

Under `PowerIdle`, based on (17) in Theorem 2, $\mathbb{E}[P]$ is an increasing function in terms of r. Obviously the minimal power consumption is achieved at the smallest possible r. By (19b), (19c) and (18), we have $r\ge r^*$, where

$$r^*=\max\{r_l,\frac{1}{\mu\hat{R}}+\rho\}. \tag{24}$$

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

Theorem 4: 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 $r^*\le 1$, where r^* is defined in (24).

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

$$\rho_u=1-\frac{1}{\mu\hat{R}}. \tag{25}$$

If $r^*\le 1$ and $\hat{R}\le\frac{1}{\mu}$, 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 $\hat{R} \leq \frac{1}{\mu}$.

VI. Performance Evaluation

This section presents performance evaluation of the proposed power-saving schemes `PowerSleep` and `PowerIdle` with optimal design, in comparison with the baseline scheme `PowerNap` [16]. The power consumption model used in the evaluation is based on the power profile of servers in [10], 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 two cases:

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

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 [16], where the mean job execution time on the server at the maximum speed is $\frac{1}{\mu} = 0.038$ sec. 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` (by applying Theorem 4), and `PowerSleep` (by applying Theorem 3).

A. Fixed 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 $\frac{10}{\mu} = 0.38$ sec. Correspondingly, the optimal r^* and/or δ_x^* for `PowerSleep` and `PowerIdle` are shown in Figure 3. Note that the maximum feasible server utilization ρ under `PowerIdle` is $\rho_u = 0.9$ defined by (25) and $\rho_u = 0.864$ under `PowerNap` and `PowerSleep` defined by (23).

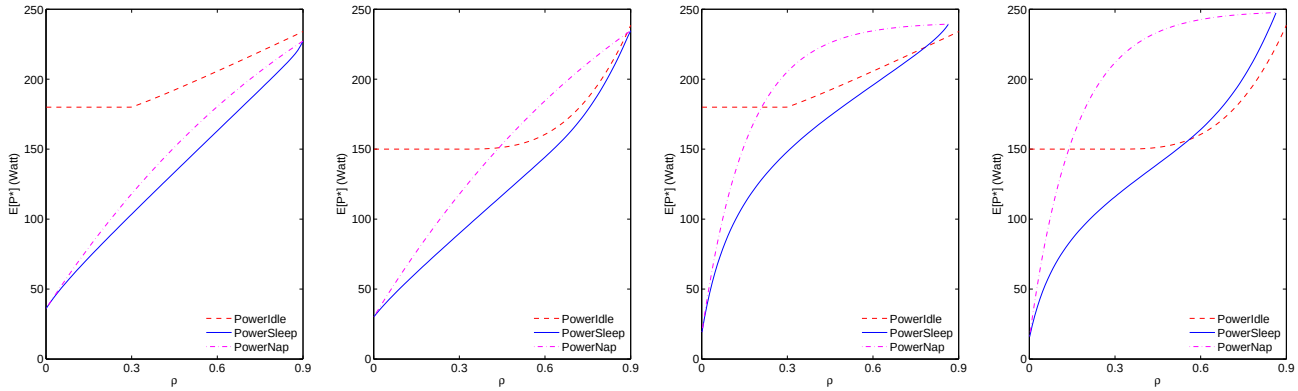
We first compare the power consumption under `PowerSleep` and `PowerNap`.

- 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 $\rho = 0.2$ and $\delta_s = \delta_w = 0.1$ sec, for linear-power servers, `PowerSleep` has 28.6% power reduction rate over `PowerNap`; for cubic-power servers, the power reduction rate is 46.2%.
- 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 technique 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, under `PowerSleep`, the optimal speed ratio is very close to 1 and the procrastination sleep period δ_x is very close to 0, which implies the optimal solution under `PowerSleep` has only marginal improvement over `PowerNap`.

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 $\rho = 0.2$ and $\delta_s = \delta_w = 0.1$ sec, for linear-power servers, `PowerSleep` has 30.1% power reduction rate over `PowerIdle`; for cubic-power servers, the power reduction rate is 35.0%. `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 0.4, 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. As stated in Theorem 4, 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 0.4, 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.

Moreover, as shown in Figure 3, when $\hat{R}$ is fixed, the optimal speed ratio r^* under `PowerSleep` is a non-decreasing function with respect to the server utilization



(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

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

ρ, whereas the procrastination sleep period δ_x is a non-increasing function with respect to ρ except the case when ρ is small in Figure 3(c). 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.

B. Fixed Server Utilization

Figure 4 presents the mean power consumption with respect to the different mean response time constraint $\hat{R}$ under `PowerNap`, `PowerIdle`, and `PowerSleep` when the server utilization is fixed as $\rho = 0.3$. $\hat{R}$ varies from R_{min} to $10 * R_{min}$, where R_{min} is the minimum response time as $\rho = 0.3$. The trends for their corresponding optimal r^* 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 `PowerSleep` and `PowerIdle` 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 `PowerNap` to reduce power consumption. For `PowerSleep` and `PowerIdle`, as $\hat{R}$ increases, the optimal choice of r and δ_x takes effects on the power consumption. However, if $\hat{R}$ is very large, the power consumption will not be subject to the mean response time constraint, the power consumption will converge at a certain level. 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 used in previous subsection.

VII. Conclusion and Future Work

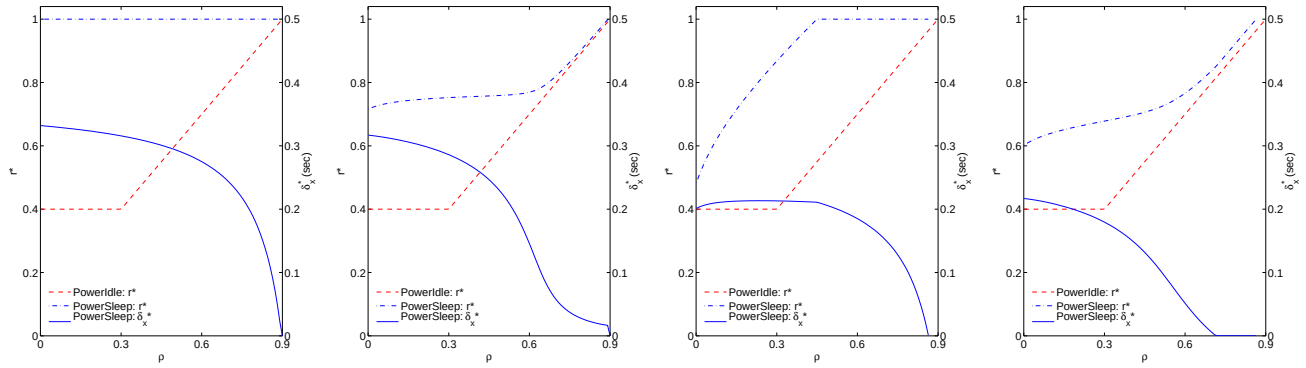
This paper explores how to minimize the mean power consumption in a server under the mean response time constraint to reduce the power cost. We consider two power-saving schemes, where `PowerSleep` applies both DVS and DPM to put the server to a low-power sleep mode while `PowerIdle` applies only DVS to choose an execution speed. 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 procrastination 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.

There are several open questions we want to address in our future work.

- A globally fixed constant speed in the running state might not be the best choice in saving power. A dynamic speed relying on the preceding idle-queue duration could reduce the power consumption while maintaining the performance requirement.
- The average response time threshold would be always violated if the wake-up transition time is beyond the average response time threshold. When the queue gets empty, the server can wait for a short-period waiting in the idle state in hope that a job arrives soon enough to avoid the transition to the sleep state.

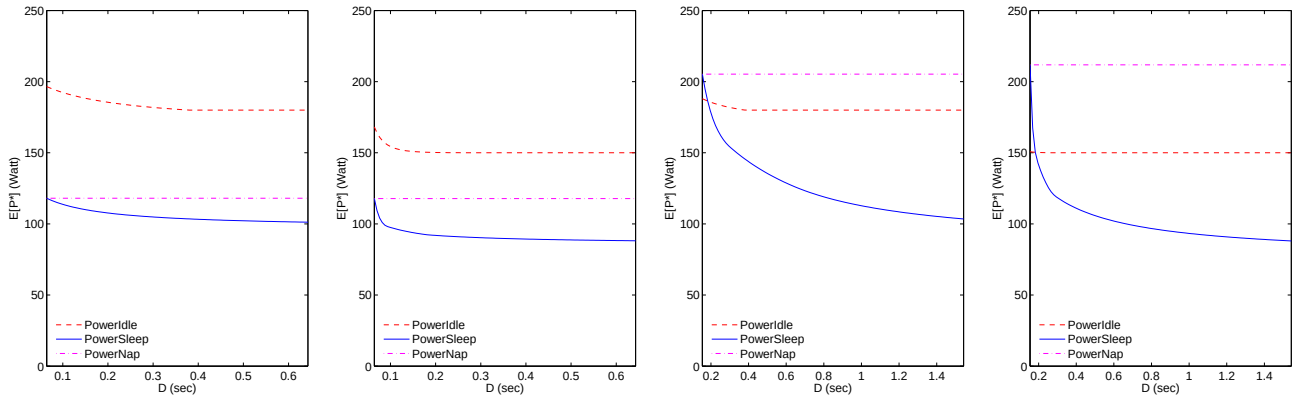
A common approach in the industry for forming a performance-oriented SLAs is to describe it using mean value of the response time, which is the focus in this paper. Some companies like Amazon [23], feel these metrics are not good enough to have a good user experience. To address this issue, at Amazon, SLAs are expressed and measured at the 99.9-th percentile of the distribution. In this case, the knowledge of the service time distribution of jobs is needed in order to obtain such tail probability.

The focus of this paper is also on one server. For systems with multiple homogeneous servers, our approaches in this paper can be extended to decide the number of servers



(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

Figure 3. Optimal r^* and δ_x^* under `PowerSleep` and `PowerIdle` for $\hat{R} = \frac{10}{\mu}$.



(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

Figure 4. Power consumption comparison for $\rho = 0.3$.

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.

REFERENCES

[1] APC (American Power Conversion), "Determining total cost of ownership for data center and network room infrastructure," 2003.

[2] L. A. Barroso, "The price of performance," *Queue*, vol. 3, no. 7, pp. 48–53, 2005.

[3] U.S. Environmental Protection Agency (EPA), "Report to congress on server and data center energy efficiency, public law 109-431," 2007.

[4] 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*, 2004.

[5] 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*, 2003.

[6] L. Sha, X. Liu, Y. Lu, and T. Abdelzaher, "Queueing model based network server performance control," in *IEEE Real-Time Systems Symposium*, 2002.

[7] 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.

[8] 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.

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

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

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

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

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

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

[15] 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.

[16] 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.

[17] L. Wang and Y. Lu, "Efficient power management of heterogeneous soft real-time clusters," in *IEEE Real-Time Systems Symposium*, 2008.

[18] 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.

[19] 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 Real-Time Systems Symposium*, 2007.

[20] 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.

[21] 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.

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

[23] D. Hastorun, M. Jampani, G. Kakulapati, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *ACM Symposium on Operating Systems Principles*, 2007.

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

APPENDIX

PROOF OF LEMMA 1

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. We can observe in Figure 1 that a cycle under `PowerSleep` is different from the one under `PowerIdle` for the same workload. Let the random variable T_C have the distribution of the length of a cycle. Also let random variables T_R, T_S and T_T have the distribution of the length of time spent in running, sleep and transition modes in a cycle respectively. Then the probabilities can be written as

$$\pi_R = \frac{T_R}{T_C}, \pi_S = \frac{T_S}{T_C}, \pi_T = \frac{T_T}{T_C}. \tag{26}$$

In the following, we will study how to obtain T_C, T_R, T_S, and T_T.

Under the *Queue with Starter* model, we can obtain the mean value of 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 [24, pp. 65], [21], we have

$$\mathbb{E}[T_B] = \frac{\mathbb{E}[T_X] + \mathbb{E}[S]}{1 - \lambda\mathbb{E}[S]}. \tag{27}$$

We know $T_C = T_B + T_I$, where T_I is a empty-queue 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

$$\mathbb{E}[T_C] = \frac{\mathbb{E}[T_X] + \frac{1}{\lambda}}{1 - \lambda\mathbb{E}[S]}. \tag{28}$$

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

$$\mathbb{E}[T_R] = \lambda\mathbb{E}[S]\mathbb{E}[T_C]. \tag{29}$$

It is obvious that the mean transition time is

$$\mathbb{E}[T_T] = \delta_s + \delta_w. \tag{30}$$

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

$$\mathbb{E}[T_S] = \mathbb{E}[T_C] - \mathbb{E}[T_R] - \mathbb{E}[T_T]. \tag{31}$$

Applying (28), (29), (30), and (31) into (26), we have the result in Lemma 1.