Power-Saving Design for Server Farms with Response Time Percentile Guarantees

Shengquan Wang, Waqaas Munawar[†], Jun Liu, Jian-Jia Chen[†], and Xue Liu[♯]
University of Michigan-Dearborn, USA
[†]Karlsruhe Institute of Technology (KIT), Germany
[♯]McGill University, Canada
Email: shqwang@umd.umich.edu, munawar@kit.edu, juliu@umd.umich.edu, j.chen@kit.edu, xueliu@cs.mcgill.ca

***Abstract*—The expense of power cost in server farms has driven the recent power-aware development in both industry and academia. At the same time, a Service Level Agreement (SLA) of service performance between a customer and a service provider is demanded to meet the customer satisfaction. This paper investigates the queueing-theoretical power-saving design strategy for server farms under a given SLA, which in particular is measured in a certain level percentile of the job response time. We consider server farms with servers that are equipped with the capabilities of Dynamic Voltage Scaling (DVFS) and Dynamic Power Management (DPM). We adopt an M/G/1/PS server model, where the job service time distribution is assumed heavy-tailed, as discovered and validated by previous research. We propose a design strategy called *PowerTail* to minimize the power consumption under the given SLA. Our data confirms that the proposed *PowerTail* strategy indeed provides statistical guarantee in comparison with existing dynamic DVFS approaches and significantly outperforms the intuitive load-balancing strategy.**

I. Introduction

Power has become an essential problem for computing systems. The advanced hardware technology has improved the performance per dollar spent on hardware, but the performance per watt remains roughly flat over time [1]. It is reported that 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. Moreover, given the large number of servers in use today, the worldwide expenditure on enterprise power and cooling of these servers is estimated to surpass $30 billion [2].

From the customer side, Quality of Service (QoS) provided by server providers is one of the key factors to customer's satisfaction. A QoS guarantee is normally contracted with a Service Level Agreement (SLA) between a customer and a service provider. For performance-oriented SLAs, a common approach is to use mean value or plus variance of the response time. If the goal is set for some companies like Amazon.com to satisfy as many customers as possible rather than just the majority, a SLA with mean (even with variance) guarantee however does not yield to good user experiences. Amazon [3] proposes to express and measure SLAs at the 99.9th percentile of the distribution. The choice for 99.9% over an even higher percentile has been made based on a cost-benefit analysis which demonstrated a significant increase in cost to improve performance that much. Experiences with Amazon's production systems have shown that this approach provides a better overall experience compared to those systems that meet SLAs defined based on the mean. This motivates this research to study SLA with certain level *percentile guarantees*. To meet the performance constraint, the system has to guarantee that some percentile of arriving jobs are processed under a certain response time threshold.

To reduce the power consumption while maintaining a given SLA, power-aware and energy-efficient designs have been extensively explored in the literature. Operating with lower power consumption instead of the maximum power consumption indeed helps, but it has been shown that the electricity cost still remains significant [4]. We would like to keep performance boosting to ensure the customer satisfaction, while the power/energy consumption should also be reduced to minimize the total maintenance cost. There is a trade-off between the power consumption and the performance. Dynamic Voltage/Frequency Scaling (DVFS) has been widely adopted to study this trade-off. Specifically, for web servers, DVFS was applied to reduce the energy consumption with the minimal performance impact in [5], [6]. In [7], a queueing theoretic model was used to predict the optimal power allocation in a variety of scenarios with DVFS. An optimal speed scaling was investigated in [8] to balance the mean energy consumption and mean response time under PS scheduling.

Unfortunately, as shown in [9]–[11], depending on the architecture, the static power consumption when a server is idle could be still significant, up to 60% of the peak power consumption. Furthermore, if the power dissipation of power delivery and cooling sub-systems is also counted, the idle power consumption could increase by up to 50$\sim$100% [12]. Even though a server might sometimes be fully utilized, average server utilization is only 20$\sim$30% in typical data centers [5], [9], [13]. Therefore, one can also apply Dynamic Power Management (DPM) to turn the server to a sleep mode for reducing the energy consumption. There are two different strategies in applying DPM: transient or long term. With the transient strategy, a server will be put into sleep for a short

period of time during idle time, such as the *PowerNap* in [13] and the *PowerSleep* in our previous work [14]. An effective DPM scheme should consider when to wake the dormant server up and when to turn the server to the sleep mode. In [14], DVFS and DPM were considered jointly to decide which frequency to execute requests at and when to put the server into sleep mode. With the long-term strategy, the server will be deactivated into a deep sleep for a long period of time, such as the server consolidation approach [15], where multiple servers will run on one machine through virtual machine technology.

In most of the work in the literature, the SLAs are expressed with either the mean value of the response time in server systems [13], [14] or the hard deadline guarantee in real-time systems [16], [17]. Modeling servers as hard real-time systems can guarantee the worst-case response time, but it is usually too pessimistic. To meet the percentile guarantee, the approaches in the literature to meet the mean response time constraint cannot be applied.

Closely related to this paper, Bertini et al. [18] use deadline missing probability as the performance metrics in web server clusters and propose an online feedback control scheme to control the QoS. Specifically, their scheme first records the probability distribution for deadline misses from the run-time observation, and then, based on the heavy-tail response time characteristics, they decide a suitable speed of the server to maintain the *stability* of the response time guarantees. Therefore, even when the workload distribution is known, the study in [18] does not have any statistic guarantee. Moreover, Rusu et al. [] also study the power management problem in servers by modeling the system as a soft real-time system. The results in [19] construct a workload table to guide the system which servers should be activated to reduce the energy consumption. Then, the local DVFS is applied to further reduce the energy consumption in [19]. However, the speed selection in [19] is based on the average execution time, and, hence, also has no statistic guarantees.

Different from [18], [19], in this paper, we investigate queueing-theoretical power-saving design strategies for server farms under a given SLA that is measured in a certain level percentile of the job response time. This is more appealing as we discussed above. We make the following contributions in this paper:

- To the best of our knowledge, there is no previous work on this topic to minimize the expected power consumption while providing the percentile guarantee. We propose *PowerTail* design strategy, which applies DPM to activate/deactivate servers and DVFS to choose the execution speed for activated servers. We adopt an M/G/1/PS server model together with heavy-tailed job service time distribution. As M/G/1/PS model is used, we will implicitly assume single-core systems for the rest of this paper. With this design, we are able to minimize the overall power consumption of a server farm under the given SLA.
- To show the effectiveness, we consider the load-balancing strategy as the baseline scheme. We evaluate both *PowerTail* and this baseline scheme on real HTTP trace. Our results show that *PowerTail* significantly outperforms the baseline scheme.
- *PowerTail* is easy to be implemented in server farms due to its simplicity.

The rest of this paper is organized as follows: Section II describes the system model. In Section III, we present a detailed analysis of power consumption and response time. The *PowerTail* design strategy is introduced in Section IV. The optimal parameters required in *PowerTail* are derived in Section V. Section VI presents performance evaluation. Finally, we conclude the paper in Section VII.

II. System Model

We consider a server farm, which includes n homogeneous servers, as shown in Figure 1. The front end monitors the

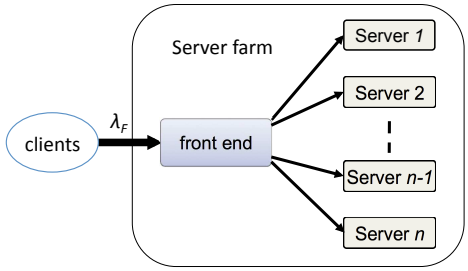


Fig. 1. A server farm.

requests from the clients and analyzes the request arrival rate for the next control period. Then, the dispatcher at the front end will distribute the job requests in the front-end queue to the back-end queue at servers according to a load allocation policy. However, we could assume a Poisson arrival [7], [8], [20], [21] of job requests at the front-end queue of the system with an arrival rate λ_F during a control period. To exploit the mechanism to activate and deactivate back-end servers dynamically, the control period is assumed to be sufficiently long to pay off the energy and timing overhead of the activation and deactivation of servers.

In this work, we need to determine the load allocation policy to minimize the overall mean power consumption under the given SLA. We also assume all jobs in each back-end queue are served with Process Sharing (PS) [1] job scheduling algorithm. Specifically, for Linux, the system call sched_setscheduler can be adopted to select the desired scheduling policy, including FCFS, Round-Robin (an approximation of PS), and Linux time-sharing scheduling. Once a server finishes serving a job, it will send a feedback to the corresponding client.

We assume that the system is equipped with DVFS and DPM for power management. A subset of servers will be activated by the DPM. We denote n_a as the number of the activated servers. For an activated server, with the DVFS we

[1]Our methodology can be applied to other job scheduling algorithms such as First-Come-First-Served (FCFS) as well.

can choose an execution speed (with a corresponding choice of the supply voltage) to serve jobs. We define r as the ratio of its execution speed 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 a server is activated, it is either (i) in the *running* mode executing jobs whenever its back-end queue is not empty, or (ii) in the *idle* mode at the lowest speed ratio r_l without executing any job whenever its back-end queue is empty.

In the literature of server performance, it has been shown by different research studies that M/G/1/PS server model can model the modern web servers well [7], [8], [20], [21]. In this paper, we adopt an M/G/1/PS server model for an activated server. We assume that the requests to each activated server still follow a Poisson arrival with an arrival rate λ, which is determined by the load allocation policy. 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 denote μ as the service rate of a single server. Then the mean job service time under the maximum speed is $\mathbb{E}[S] = \frac{1}{\mu}$. If the server runs at a speed ratio r in the running mode, we have $\mathbb{E}[S] = \frac{1}{r\mu}$. We denote ρ as the (absolute) server utilization with respect to the maximum speed, i.e.,

$$\rho = \frac{\lambda}{\mu}. \tag{1}$$

Then the relative server utilization with respect to the speed ratio r can be written as $\lambda\mathbb{E}[S] = \frac{\rho}{r}$.

In this work, we aim to choose a certain-level percentile of the response time as the SLA instead of the mean. Therefore we need more information on the job size in addition to the mean service time $\mathbb{E}[S]$. It has been widely recognized that heavy-tailed distributions are suitable for modeling job sizes in information service networks [22]–[25]. In particular, in [25], it has been shown that web file sizes and HTTP request times follow heavy-tailed distributions. Therefore we adopt heavy-tailed distribution for the job service time. In particular, we choose the Pareto distribution.[2] By the definition of Pareto distribution [26], we assume that the job service time S is lower-bounded by $\hat{x}$ when the server runs at the maximum speed, and the tail of S is

$$\mathbb{P}\{S \geq x\} = [\frac{x}{\hat{x}}]^{-\nu} \tag{2}$$

as $x \geq \hat{x}$, where $\nu > 1$. If the job service time is less than $\hat{x}$, the tail is 1. With the heavy-tailed distribution, we will more likely have a longer job service time than the light-tailed. From (2), we can obtain the mean value as $\mathbb{E}[S] = \frac{\nu}{\nu-1}\hat{x}$. Therefore, we have the following equality: $\frac{1}{\mu} = \frac{\nu}{\nu-1}\hat{x}$. Hence μ is uniquely determined by $\hat{x}$. When the server runs at the speed ratio r, the tail of S should be re-written as

$$\mathbb{P}\{S \geq x\} = [\frac{rx}{\hat{x}}]^{-\nu} \tag{3}$$

as $rx \geq \hat{x}$.

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. When the server is deactivated by the DPM, its power consumption is negligible. For an activated server equipped with DVFS, its power consumption depends on the mode the server is in (running or idle), and also the execution speed in use. In this paper, we adopt the model in [7]. The activated 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

$$P_R(r) = \alpha[r - r_l]^{\gamma} + P_I, \tag{4}$$

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 DVFS or for some applications, the linear rule could be applied.

A design strategy should address how many servers should be activated by the DPM and what speed ratio should be chosen by the DVFS for each activated servers. Note that, we assume that the control period is sufficiently long to compensate the energy consumption of the activation and deactivation energy overhead of servers. The timing overhead to activate and deactivate a server is assumed to be hidden in the control period. That is, if the front end requires to change the system configuration at time t, the activation request of a server should be issued at $t-\delta$, in which δ is the upper bound of the timing overhead to activate a server.

III. Power Consumption and Response Time Analysis for an Activated Server

Recall that our objective is to minimize the power consumption under the given response time SLA. First we conduct the analysis of power consumption and response time for an activated server based on the system model in Section II.

A. Power Consumption

We consider an activated server that will switch in two power modes alternatively: running and idle. We define π_R and π_I as the probabilities that the server is in running and idle mode respectively, where $\pi_R + \pi_I = 1$, then its mean power consumption can be written as:

$$\mathbb{E}[P] = P_R(r)\pi_R + P_I\pi_I. \tag{5}$$

According to (5), in order to calculate the mean power, we need to obtain the value of π_R. For an M/G/1/PS server with a speed ratio r at the running state, an arrival rate λ, and a mean service time $\mathbb{E}[S] = \frac{1}{\mu}$, by the traditional queueing theory [27] we have

$$\pi_R = \lambda\mathbb{E}[S] = \frac{\rho}{r}, \tag{6}$$

[2]Our methodology can work for other specific heavy-tailed distributions too.

where ρ is the absolute server utilization defined in (1). Applying (4) and (6) into (5), we have the following theorem regarding the power consumption:

Theorem 1: In an M/G/1 server with an absolute server utilization ρ and a speed ratio r at the running state, its mean power consumption is

$$\mathbb{E}[P] = \alpha[r - r_l]^\gamma \frac{\rho}{r} + P_I. \tag{7}$$

$\mathbb{E}[P]$ defined in (7) is an increasing function of r as $r_l \leq r \leq 1$ and of ρ too.

B. Response Time Analysis

The response time (also called the sojourn time) is defined as the total time spent by a job in waiting in the queue and executing on the server. For an M/G/1/PS server with heavy-tailed service time distribution, we have the following result for the tail of the response time [28]:

Lemma 1: In an M/G/1/PS server running at the maximum speed at the running state where the job service time S is heavy tailed, the tail of the response time R can be asymptotically approximated as

$$\mathbb{P}\{R \geq \hat{R}\} \sim \mathbb{P}\{S \geq [1-\rho]\hat{R}\} \tag{8}$$

as $\hat{R} \to \infty$.

Lemma 1 shows that the response time is also heavy-tailed if the service time is heavy-tailed. With this lemma, given the tail of the service time, we are able to obtain the tail of the response time for a relatively large response time threshold $\hat{R}$.

When the server runs at the speed ratio r and the heavy-tailed distribution of the service time S is defined as (3), then we apply (3) into Lemma 1, and obtain the tail of the response time as shown in the following theorem:

Theorem 2: In an M/G/1 server with a speed ratio r at the running state and the heavy-tailed distribution of the service time S is defined as (3), the tail of the response time can be asymptotically approximated as

$$\mathbb{P}\{R \geq \hat{R}\} \sim \mathbb{P}\{S \geq [1-\frac{\rho}{r}]\hat{R}\} = [\frac{\hat{x}}{\hat{R}}]^\nu [r-\rho]^{-\nu} \tag{9}$$

as $\hat{R} \to \infty$.

In this work, we aim to minimize the overall mean power consumption of the server farm under a given response time SLA. The SLA is normally expressed and measured at a low bound of a certain level percentile (for example, 99.9-th percentile) of the distribution of the response time. We could also express the SLA in terms of the upper bound of its tail (for example, $1 - 0.999 = 0.001$ tail for 99.9th percentile). If we denote ϵ as the upper bound, the SLA can be expressed as

$$\mathbb{P}\{R \geq \hat{R}\} \leq \epsilon, \tag{10}$$

where the value of $\hat{R}$ is usually relatively large in comparison to the mean value $\mathbb{E}[S]$, and ϵ is relatively small. The SLA defined in (10) is contracted between the customer and the service provider. Applying (9) in (10), we have $[\frac{\hat{x}}{\hat{R}}]^\nu [r-\rho]^{-\nu} \leq \epsilon$. By denoting

$$\eta \stackrel{\text{def}}{=} \frac{\hat{x}}{\hat{R}} \epsilon^{-\frac{1}{\nu}}, \tag{11}$$

The above equation can be re-written as

$$r \geq \rho + \eta. \tag{12}$$

(12) becomes the simplified version of the SLA constraint (10).

The analysis of power consumption and response time in this section establishes the foundations for our design in the next section.

IV. *PowerTail*: Power-Saving Design with Response Time Percentile Guarantees

In this section, we will present our design strategy, called *PowerTail*. In order to achieve a good design, we introduce a new term, called *mean energy consumption per job*. It can be expressed mathematically as

$$Q(\lambda, r) = \frac{\mathbb{E}[P]}{\lambda}. \tag{13}$$

In (13), $\mathbb{E}[P]$ is the mean power consumption for a server, and λ is the job arrival rate at the server. Therefore, $\frac{\mathbb{E}[P]}{\lambda}$ is the mean energy consumed by a job.

Our strategy to achieve a good design is to minimize Q, the energy consumption per job, under a given SLA. The optimization problem can easily be formulated as follows:

$$\text{minimize } Q(\lambda, r) \tag{14a}$$

$$\text{subject to } \mathbb{P}\{R \geq \hat{R}\} \leq \epsilon, \tag{14b}$$

$$\max\{r_l, \rho\} \leq r \leq 1. \tag{14c}$$

Inequality (14c) is based on the low bound of r ($r_l \leq r \leq 1$) and the stability condition of a server ($r > \rho$).

We define the optimal solution (λ_c, r_c) obtained from (14) as the *critical arrival rate* and the *critical speed ratio*, i.e.,

$$(\lambda_c, r_c) = \arg\min_{(\lambda,r)} \{Q(\lambda, r) \text{ in (14)}\}. \tag{15}$$

With the above critical arrival rate and critical speed ratio, we propose our design strategy *PowerTail*, which is described as follows.

> *PowerTail:* If λ_F is divisible by λ_c obtained in (15) and $n_a \leq n$, where $n_a = \frac{\lambda_F}{\lambda_c}$, we activate n_a servers and evenly distribute the load into these n_a servers. Each activated server is allocated with the exact critical arrival rate λ_c and runs at the running state with the exact critical speed ratio r_c obtained in (15).

The following theorem shows that *PowerTail* is an optimal design:

Theorem 3: PowerTail minimizes the overall mean power consumption of the server farm under the given SLA defined in (10).

Proof: With *PowerTail*, the overall job requests at the front-end queue is evenly distributed to n_a activated homogeneous servers. Each activated server is allocated with the critical job arrival rate λ_c and configured with the critical speed

ratio r_c. The rest $n - n_a$ servers are deactivated and consume negligible power. Then the overall mean power consumption of the server farm is

$$n_a \mathbb{E}[P] = \frac{\lambda_F}{\lambda_c}\mathbb{E}[P] = \lambda_F \frac{\mathbb{E}[P]}{\lambda_c} = \lambda_F Q(\lambda_c, r_c). \quad (16)$$

Since $Q(\lambda_c, r_c)$ in (16) is minimized, then the overall mean power consumption $n_a\mathbb{E}[P]$ is also minimized. ■

Theorem 3 shows that minimizing the overall mean power consumption is equivalent to minimizing the mean energy consumption per job.

In Theorem 3, notice that we have assumed that λ_F is divisible by λ_c and $n_a \leq n$ holds. If λ_F is not divisible by λ_c and $n_a < n$, where $n_a = \lfloor \frac{\lambda_F}{\lambda_c} \rfloor$, the *PowerTail* should be revised as follows:[3] We still allocate the critical arrival rate λ_c into n_a homogeneous servers and apply the critical speed ratio r_c to them, and allocate the residue arrival rate $\lambda_F - n_a\lambda_c$ into an additional server and apply the critical speed ratio r_c. In this case of *PowerTail*, although the additional server does not achieve the optimal power consumption, we still prefer *PowerTail* over an optimal design strategy because of the following three reasons:

1) The difference between the resulting overall mean power consumption with *PowerTail* and the optimal overall mean power consumption is negligible when n_a is relatively large (tens to hundreds or above of activated servers);
2) The method used in *PowerTail* – minimizing mean energy consumption per job – is very elegant. Instead of solving the optimization for the whole server farm, we only focus on a single server;
3) *PowerTail* is much easier to implement than the optimal design strategy. As the arrival rate λ_F at the front end changes, we only need to activate or deactivate more servers without changing the running speed ratio of activated servers as long as we have enough available servers.[4] However, with the optimal design strategy, for any change of λ_F, we need to re-solve the optimization problem and obtain a new arrival rate and a new speed ratio for each activated server, and then re-configure these parameters in the system.

The key in *PowerTail* design is to obtain the critical arrival rate λ_c and the critical speed ratio r_c defined in (14), which is the focus of the next section.

V. Obtaining Critical Utilization and Critical Speed Ratio in *PowerTail*

We observe that the mean energy consumption per job is focused on a single server and the utilization ρ in a single server is easier to picture than the arrival rate λ, such as the boundaries of ρ. In the following, we replace λ with $\rho\mu$ based on (1) and also $Q(\lambda, r)$ with $Q(\rho, r)$. We introduce the critical utilization ρ_c to replace the critical arrival rate λ_c.

Based on the power consumption in (7) and the simplified SLA constraint (12), the optimization problem (14) can be re-written as

$$\text{minimize } Q(\rho, r) = \frac{1}{\mu}[\frac{\alpha[r - r_l]^\gamma}{r} + \frac{P_I}{\rho}] \quad (17a)$$

$$\text{subject to } \max\{r_l, \rho + \eta\} \leq r \leq 1. \quad (17b)$$

The detailed discussion of deriving optimal $Q(\rho, r)$ in the above optimization problem can be found in Appendix A.

A. Main Result

The following theorems summarize the main results of the critical utilization ρ_c and its corresponding speed ratio r_c for both $\gamma = 1$ and $\gamma > 1$:

Theorem 4 (linear-power servers): The critical utilization $\rho = \rho_c$ and the critical speed ratio r_c for linear-power servers ($\gamma = 1$) are obtained as

$$(\rho_c, r_c) = \begin{cases} (\rho_m, \max\{r_l, \eta\}), & \text{if } 0 < \xi < r_l \\ (\rho^*, \max\{r_l, \xi\}), & \text{if } r_l \leq \xi < 1 \\ (\rho_r, 1), & \text{if } \xi \geq 1 \vee \xi \leq 0 \end{cases} \quad (18)$$

where ρ_m and ρ_r are defined in (30) and (27) respectively, ξ and ρ^* are defined respectively as[5]

$$\xi \stackrel{\text{def}}{=} \frac{\eta}{1 - [\frac{P_I}{\alpha r_l}]^{\frac{1}{2}}}, \text{ and } \rho^* \stackrel{\text{def}}{=} \xi - \eta. \quad (19)$$

Theorem 5 (Nonlinear-power servers): The critical utilization $\rho = \rho_c$ and the critical speed ratio r_c for nonlinear-power servers ($\gamma > 1$) are obtained as

$$(\rho_c, r_c) = \begin{cases} (\rho^*, \max\{r_l, \rho^* + \eta\}), & \text{if } \chi > 0 \\ (\rho_r, 1), & \text{if } \chi \leq 0 \end{cases} \quad (20)$$

where χ and ρ^* are defined respectively as

$$\chi \stackrel{\text{def}}{=} H(\rho_r) = \alpha[\rho_r - \rho_m]^{\gamma-1}[\gamma - 1 + r_l] - \frac{P_I}{\rho_r^2}, \quad (21)$$

$$\rho^* \stackrel{\text{def}}{=} \{\rho : H(\rho) = 0\}, \quad (22)$$

and $H(\rho)$ is defined in (34).

The proofs of Theorems 4 and 5 can be found in Appendices B and C respectively.

By Theorems 4 and 5, we observe that the critical utilization ρ_c and the critical speed ratio r_c can be achieved at either the boundaries or insider the feasible region. The key difference between the linear servers and nonlinear servers is that ρ_c and r_c will never fall at the left boundary for the latter case.

Also, ϵ or $\hat{R}$ can not be any positive value. By (11) with $\epsilon \leq 1$, a feasible η should satisfy: $\frac{\hat{x}}{\hat{R}} \leq \eta \leq 1$. And a feasible ϵ should satisfy: $\epsilon \geq (\frac{\hat{x}}{\hat{R}})^\nu$. We will demonstrate the feasible region in the evaluation section.

[3] If $\lambda_F > n\lambda_c$, it means that the servers are not enough to achieve the optimal configuration with *PowerTail*. In this case, load balancing with even job allocation to all servers is the optimal configuration, i.e., we choose $\frac{\lambda_F}{n}$ as the arrival rate for each server. We can easily obtain an optimal speed ratio for each server.

[4] The power-law characteristics of the job service time does not change dramatically on servers. Otherwise, we need to reconfigure all these parameters.

[5] If the denominator of ξ in (19) is zero, we assume $\xi = +\infty$ and include it in the case of $\xi > 1$.

B. Remarks

Note that the control period is assumed sufficiently long to compensate the energy consumption of the activation and deactivation energy overhead of servers. If the control period of the front end for estimating the arrival rate is long, to compensate the bursty request arrivals, which is unpredicted, we need to spare some idle servers to handle this issue. In *PowerTail*, the strategy is to spare one back-end server, which can be used to accommodate the unpredicted requests. The strategy can be easily extended to allow the designers to spare more servers. However, the more spare servers are used for accommodating bursty requests, the more additional energy the system consumes.

Moreover, for systems that allow only discrete speeds, the easiest extension is to run at the closest speed that is higher than the desired speed. Such approaches have been adopted in many papers in the literature, e.g., [19]. Another alternative is to allow the possibility to use two speeds (the closest speeds higher and lower than the desired speed) with linear combination to achieve the desired speed, e.g., similar to [29]. The former can still maintain the statistic guarantee, but consumes more energy. The latter has less energy consumption, but requires high scheduling overhead.

Even though we focus in our study by far on the PS job scheduling policy, the proposed methodology still works for other scheduling policies. 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. This will not hold for the response time analysis. We need to make some corresponding changes. For instance, for FCFS scheduling policy, it has been shown [30] that the tail of the response time is regularly varying of index $1-\nu$ iff the service time is regularly varying of index $-\nu$ as shown in (2). The tail probability of the response time in the server running at the maximum speed can be written as $\mathbb{P}\{R \geq \hat{R}\} \sim \frac{\rho}{1-\rho}\frac{\mu\hat{x}}{\nu-1}\hat{R}^{1-\nu}$ as $\hat{R} \to \infty$. If the server runs at the speed ratio r, then we have

$$\mathbb{P}\{R \geq \hat{R}\} \sim \frac{\rho}{r-\rho}\frac{\mu\hat{x}}{\nu-1}[r\hat{R}]^{1-\nu}. \tag{23}$$

as $\hat{R} \to \infty$.

In the evaluation, we will consider an intuitive baseline design strategy, called *PowerEven*. With *PowerEven*, the load is evenly allocated to all servers. Therefore, the arrival rate at each server is $\frac{\lambda_F}{n}$. The absolute utilization for each server is

$$\rho_{PB} = \frac{\lambda}{\mu} = \frac{\lambda_F}{n\mu}. \tag{24}$$

We need to determine the speed ratio r to minimize the mean power consumption under the given SLA. The optimization problem can be formulated as

$$\text{minimize } \mathbb{E}[P] = \alpha[r-r_l]^\gamma\frac{\rho_{PB}}{r} + P_I \tag{25a}$$

$$\text{subject to } \max\{r_l, \rho_{PB}+\eta\} \leq r \leq 1. \tag{25b}$$

Since $\mathbb{E}[P]$ is an increasing function of r, the minimum power consumption will be achieved at $r_{PB} = \rho_{PB} + \eta$, where ρ_{PB} is defined in (24).

TABLE I
SERVER PROFILES.

	γ	r_l	α (Watt)	P_I Watt
Type-A server	1	0.4	100	180
Type-B server	1.2137	0.5161	12.3977	36.7040
Type-C server	3	0.4	455	150

TABLE II
MISS RATES BY VARYING ρ ON SERVER OF TYPE-B.

ρ	Miss rate (%)			Average Power (Watt)		
	RUSU	YDS	*PowerTail*	RUSU	YDS	*PowerTail*
0.1	0.75	5.28	0.39	36.81	37.04	37.21
0.2	1.69	11.77	0.59	36.96	37.41	37.72
0.3	3.02	17.82	0.97	37.19	37.79	38.23
0.4	6.88	25.23	1.52	37.52	38.21	38.74
0.5	9.95	35.10	1.93	37.93	38.67	39.25
0.6	13.93	47.00	4.26	38.29	39.19	39.75

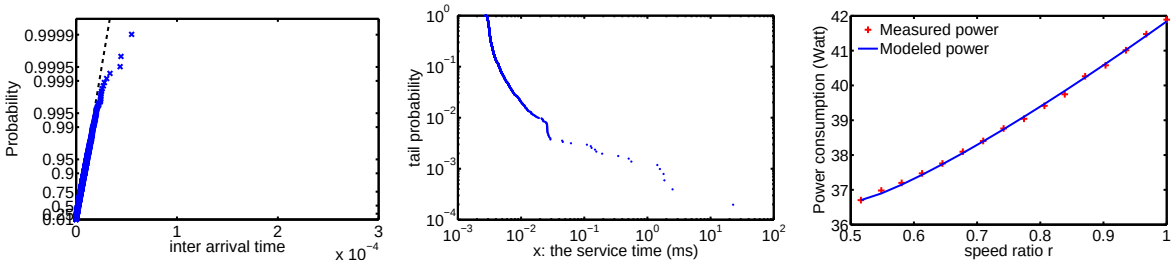
VI. PERFORMANCE EVALUATION

In this section, we present performance evaluation of our proposed *PowerTail* design. For the job requests from clients, we consider a real 24-hour HTTP trace of accesses on January 20th, 2011 from IRCache system [31]. The trace is with a minimum arrival rate 5.4 per second and a maximum rate 238.5 per second. We consider the first one-minute trace at the 12th hour with 5074 requests and the arrival rate 84.6 per second. Based on [32], the service time of a job request is proportional to the size of the requested file on web servers. We adopt the power-law distribution fitting tool in [33] to obtain the parameters ν and $\hat{x}$ of Pareto distribution for the service time. Figures 2(a) and 2(b) show the Poisson arrival fitting and power-law distribution fitting of the one-minute trace with $\nu = 2.2211$ and $\hat{x} = 0.00658$. Then we obtain $\mu = 83.5521$. We also verify the heavy-tailed phenomenon of the job response time for heavy-tailed job service time as shown in Lemma 8 and we include the result in Appendix D.

We use three types of servers to evaluate the performance of *PowerTail*: the one used in our lab (Intel i3 dual core processor based server with 6GB of RAM equipped with DVFS capability with option to switch between 16 frequencies between 1.6GHz and 3.1GHz), and the other two from [7] (IBM BladeCenter with DFS and DVFS+DFS). We measure the power consumption for the one used in our lab. Based on the measured power, we obtain the modeled power as shown in Figure 2(c). The power profiles of all three types of servers are shown in Table I, where Type-B is the one used in the previous experiment. We enable hyper-threading for Type-B, in which the results are very similar to the results when hyper-threading is disabled.

A. Comparing to Dynamic DVFS Approaches

There have been many results in the literature by applying DVFS to change the speed dynamically to reduce the energy consumption under the performance constraints. Such



(a) Poisson arrival (b) Power-law of the service time (c) Measured and modeled power.

Fig. 2. Measurement of the trace and the server power consumption.

approaches have been widely explored in soft and hard real-time systems. When the earliest deadline first (EDF) policy [34] is applied, the results by Yao et al. [35] can be used to schedule jobs in the on-line or off-line fashion to reduce the energy consumption and meet the hard real-time constraints. In our model, we consider all the jobs are with a response time constraint. Therefore, EDF scheduling also implies FCFS scheduling in our case. Unfortunately, the optimality of the EDF policy holds when no job misses the deadline. For soft real-time tasks, when a job with heavy execution time comes, applying EDF for scheduling may be bad, as the Domino effect for deadline misses may occur. This leads to significant percentage of response time violations. Therefore, even when running at the maximum speed, the response time violation can be very significant.

By using the PS scheduling, Rusu et al. [19] develop a local DVFS algorithm for controlling the speeds of execution. The local DVFS scheduling policy inside a server is set to $U = \sum_i \frac{C_i}{D_i}$, where C_i is the average execution time and D_i is the time remaining (from now) to the deadline (response time + arrival time) of job i. The server periodically updates the value U and sets the speed the closest higher speed than U. The speed scaling period suggested in [19] is 10 ms to cover any default Linux kernel.[6]

We evaluate the above approaches with DVFS and our approach by selecting a suitable constant speed for ensuring the percentile response time guarantee for a server with simulations. The processor uses a process sharing with Round Robin job scheduling with time slot 1 ms. We consider the traces under different approaches, denoted by RUSU, in Rusu et al. [19] and the on-line algorithm (OPT) in Yao et al. [35], denoted by YDS.[7] As RUSU and YDS are not well-defined to deal with response time violations, in our implementations, YDS algorithm runs at the maximum speed if any job in the ready queue misses the deadline, whereas RUSU runs at the maximum speed if U is less than or equal to 0. In *PowerTail*, only a constant speed (the closest higher speed if the desired speed is not provided) is used to run jobs in a server. We report the experimental results as follows: Table II presents the simulation results for the Type-B server (by using the discrete speed settings) by changing ρ with a fixed $\hat{R} = 60$ ms. Table III presents the simulation results for the Type-C server for different percentile response time requirements, in which $\hat{R}$ changes when $\rho = 0.6$.

YDS, as it inherits the EDF policy, suffers from the Domino effect and has poor percentile response time guarantees. As the server has to speed up to the maximum speed after subsequent deadline misses in YDS, the system would have to run at the maximum speed more frequently, and therefore YDS would consume more energy consumption than the other two approaches when such cases happen (e.g., $\hat{R} \geq 100$ ms in Table III). As shown in the tables, to guarantee the percentile, the system should choose the speed carefully. Even though the statistic analysis shows that the server can guarantee 95% response time when running at the maximum speed, speed scaling decisions should be taken with care.

The miss rate of RUSU increases when the system becomes more stringent, i.e., larger ρ or less $\hat{R}$. The reason is that the less stringent settings will have more possibility for $U \leq r_\ell$, which will be enforced to r_ℓ. Therefore, the chance of running at too slow speeds becomes less. We have also observed that RUSU would have higher miss rates when r_ℓ is set smaller, in which the results are omitted due to space limitation. The miss rates of RUSU observed in the tables can be up to 13.93%, while *PowerTail* has miss rates up to 4.26%.

In general, the miss rate of the *PowerTail* approach satisfies the percentile requirement. However, as *PowerTail* does not perform DVFS dynamically but runs at a constant speed for the ease of management and analysis, it consumes more power than RUSU, as shown in both tables.

B. Comparing to Load Balancing Design

We also compare *PowerTail* with a baseline design strategy *PowerEven*, which applies the load balancing allocation policy. We use the same job profile of the one-minute except that the job size is scaled in each type of server with $\mu = 83.5521$ per second at the maximum speed. For fair comparison, we need to decide the number of required servers for a peak demand: We consider a worst-case arrival rate 500 per second, and fix $\hat{x} = 0.00658$ sec, $\frac{\hat{x}}{\hat{R}} = 0.02$, and $\epsilon = 0.001$, then the

[6]We have also evaluated different settings of the speed scaling period. The miss rates of RUSU also depend on the setting as well. A bad setting (e.g., 2 seconds) would result in up to 50% deadline misses.

[7]To run YDS, we provide the actual execution time immediately when a job arrives, which is needed in YDS. This is beneficial for YDS. We only consider one server here without applying on/off policies in the server in this setting.

TABLE III
MISS RATES BY VARYING $\hat{R}$ ON SERVER OF TYPE-C.

$\hat{R}$ (ms)	Miss rate (%)			Average Power (Watt)		
	RUSU	YDS	*PowerTail*	RUSU	YDS	*PowerTail*
50	9.46	27.69	1.83	170.58	181.12	191.48
60	12.34	35.08	1.93	175.45	187.61	198.59
80	10.92	41.27	2.50	172.19	193.04	196.31
100	7.80	43.95	1.58	176.17	195.43	191.32
200	2.62	49.11	0.65	173.56	200.19	182.74
400	1.22	50.47	0.24	172.70	202.51	178.13
600	0.85	50.43	0.22	173.55	203.22	176.60
1000	0.63	49.35	0.22	172.18	203.55	175.44

minimum number of required servers under *PowerTail* for all three types of servers is 11.[8] We assume that both *PowerTail* and *PowerEven* use 11 servers.

There are three parameters to evaluate the performance of *PowerTail*: the arrival rate λ_F at the front-end queue, the tail upper-bound ϵ, and the ratio of the minimum response time to the guaranteed maximum response time $\frac{\hat{x}}{\hat{R}}$. Both ϵ and $\frac{\hat{x}}{\hat{R}}$ will determine the value of η defined in (11). In the following, we will fix two of these parameters and vary the other. Based on these settings, we compare *PowerTail* with *PowerEven*.

1) Varying $\frac{\hat{x}}{\hat{R}}$: In this case, we assume the arrival rate $\lambda_F = 84.6$ per second. We choose $\epsilon = 0.001$. Based on the feasible region of $\frac{\hat{x}}{\hat{R}}$, given the value of ϵ, $\frac{\hat{x}}{\hat{R}}$ is upper-bounded by $\epsilon^{\frac{1}{\nu}} = 0.0446$ and it could not reach the upper bound. In our evaluation, we fix $\hat{x} = 0.00658$ sec and vary $\frac{\hat{x}}{\hat{R}}$ from 0.00001 to 0.04. We obtain the mean power consumption of the server farm with *PowerTail* and *PowerEven* under all three types of servers. The result is shown in Figure 3. The corresponding critical speed ratio and the critical utilization with *PowerTail* are shown in Figure 4.

In Figure 3, we observe that $\mathbb{E}[P]$ increases with respect to $\frac{\hat{x}}{\hat{R}}$ for both *PowerTail* and *PowerEven*. However, *PowerTail* outperforms *PowerEven* significantly. For example, when $\frac{\hat{x}}{\hat{R}} = 0.02$, *PowerTail* consumes only 21.0%, 19.5%, and 24.2% of power by *PowerEven* for Type-A, Type-B, and Type-C servers respectively. In this case, *PowerTail* only activates 2 servers out of 11 servers for all three types. As $\frac{\hat{x}}{\hat{R}}$ increases, for a fixed $\hat{x}$, $\hat{R}$ decreases. That means we have a tight response time requirement. In order to achieve this goal, we need to increase the processing capacity. For instance, we activate more servers, which results in more power consumption. Eventually, all servers will be activated, in that case, *PowerTail* will be same as *PowerEven*.

For $\gamma = 1$, we have $\alpha r_l \leq P_I$ and hence $\xi < 0$ by (19). Based on Theorem 4, $r_c = 1$ holds always, which is observed in Figure 4(a). For $\gamma = 1.2137$, we also have $r_c = 1$, which is also observed in Figure 4(b). For $\gamma = 3$, $\chi > 0$ as $\frac{\hat{x}}{\hat{R}} < 0.0171$ and $\chi < 0$ as $\frac{\hat{x}}{\hat{R}} > 0.0171$. Therefore, based on Theorem 5, $r_c = 1$ as $\frac{\hat{x}}{\hat{R}} > 0.0171$, which is also observed in Figure 4(c).

[8]Recall that the job size is scaled in each type of server with $\mu = 83.5521$ per second at the maximum speed, therefore the total numbers of servers under the peak demand are the same.

TABLE IV
COMPARISON OF *PowerTail* VS. THE OPTIMAL DESIGN.

λ $(/s)$	Type-A Server	Type-B Server	Type-C Server
100	$3, 1.4\%$	$3, 1.4\%$	$3, 11.8\%$
200	$5, 0.7\%$	$5, 0.7\%$	$5, 6.9\%$
300	$7, 0.4\%$	$7, 0.4\%$	$7, 3.9\%$
400	$9, 0.2\%$	$9, 0.2\%$	$9, 2.0\%$
500	$11, 0.1\%$	$11, 0.1\%$	$11, 0.8\%$

In Figure 4, we also observe that ρ_c is a (piecewise) linear function of $\frac{\hat{x}}{\hat{R}}$, which is based on Theorems 4 and 5 and (11).

2) Varying ϵ: In this case, we still assume the arrival rate $\lambda_F = 84.6$ per second. We choose $\frac{\hat{x}}{\hat{R}} = 0.02$. Based on the feasible region of ϵ, given the value of $\frac{\hat{x}}{\hat{R}}$, ϵ is lower-bounded by $[\frac{\hat{x}}{\hat{R}}]^{\nu} = 1.68e-4$ and it could not reach the lower bound. In our evaluation, we vary ϵ from 0.0002 to 0.1. We obtain the mean power consumption of the server farm with *PowerTail* and *PowerEven* under all three types of servers. The result is shown in Figure 5.

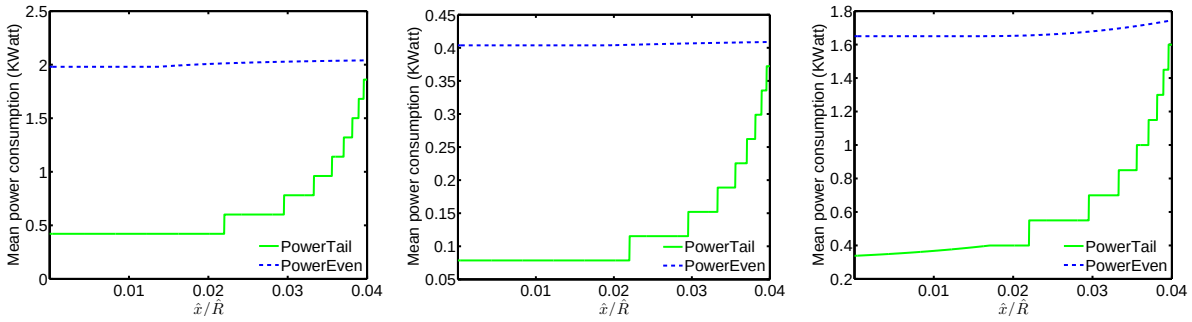
Based on Theorems 4 and 5, *PowerTail* design depends on the value of η. By (11), η is an increasing function of $[\frac{\hat{x}}{\hat{R}}]^{\nu}$, but a decreasing function of ϵ. Therefore, we observe that in Figure 5 $\mathbb{E}[P]$ decreases with respect to ϵ for both *PowerTail* and *PowerEven*. *PowerTail* still outperforms *PowerEven* significantly. For example, when $\epsilon = 0.001$, *PowerTail* consumes only 21.0%, 19.5% and 24.2% of power by *PowerEven* for Type-A, Type-B, and Type-C servers respectively. In this case, *PowerTail* only activates 2 servers out of 11 servers for all types. When ϵ is extremely small, *PowerTail* is worse due to the fact that *PowerTail* is not optimal for this case. Due to the symmetricity of this case with the previous one, the reason behind it can be drawn easily. We also skip the result of the critical speed ratio and the critical utilization.

3) Varying λ: In this case, we set $\frac{\hat{x}}{\hat{R}} = 0.02$ and $\epsilon = 0.001$. We vary λ from 0 per second to 500 per second. We obtain the mean power consumption of the server farm with *PowerTail* and *PowerEven* under all three types of servers. The result is shown in Figure 6. The overall mean power consumption is an almost-linear function of λ under *PowerTail*. The steps in the curve is due to the server with non-optimal configuration. *PowerTail* aim to activate more servers for a higher arrival. For *PowerEven*, since all servers are always activated, each server will consume the power in the idle mode, which is relatively large.

In this case, we also evaluate the gap between *PowerTail* and the optimal design as shown in Table IV. For each table entry, the first value is the number of servers activated by *PowerTail* and the second value is the gap between *PowerTail* and the optimal design. In all cases, as the request arrival rate increases *PowerTail* performs very closely to the optimal design.

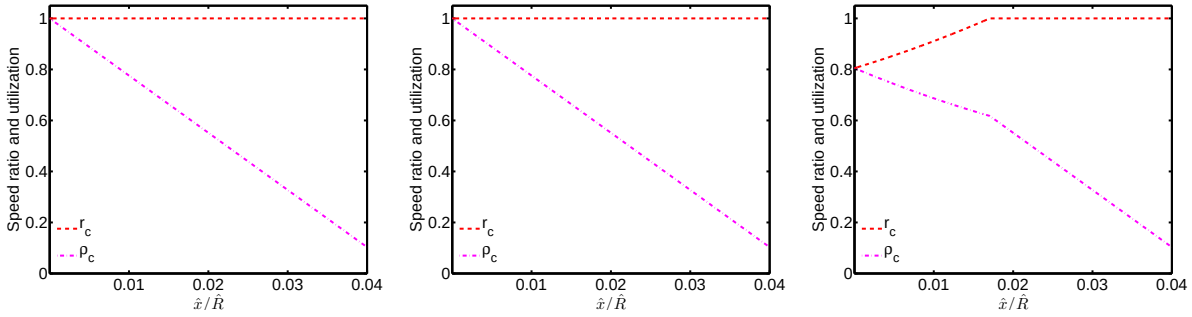
VII. CONCLUSION

In this paper, we studied power consumption optimization in server farm under response time SLA. The SLA in study was measured at a certain level percentile or tail of the job response time, which is more appealing than the commonly-used mean



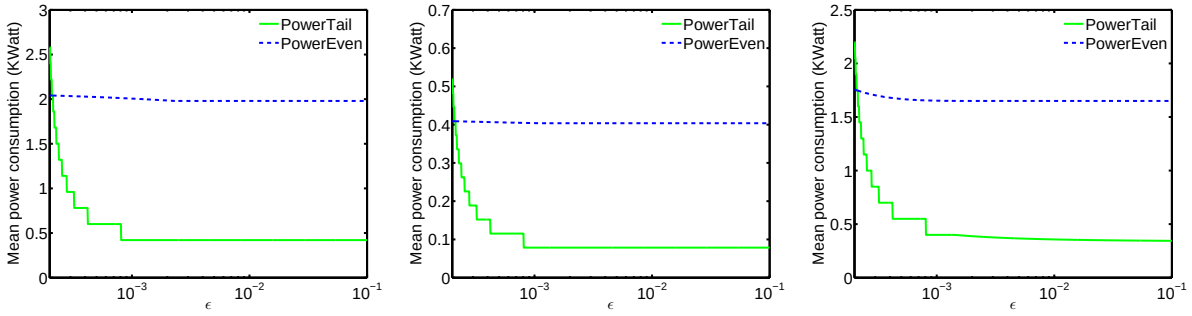
(a) Type-A servers ($\gamma = 1$) (b) Type-B servers ($\gamma = 1.2137$) (c) Type-C servers ($\gamma = 3$)

Fig. 3. Power consumption comparison for $\epsilon = 0.001$ and $\lambda = 84.6$ per second.



(a) Type-A servers ($\gamma = 1$) (b) Type-B servers ($\gamma = 1.2137$) (c) Type-C servers ($\gamma = 3$)

Fig. 4. The values of (ρ_c, r_c) with *PowerTail* for $\epsilon = 0.001$ and $\lambda = 84.6$ per second.



(a) Type-A servers ($\gamma = 1$) (b) Type-B servers ($\gamma = 1.2137$) (c) Type-C servers ($\gamma = 3$)

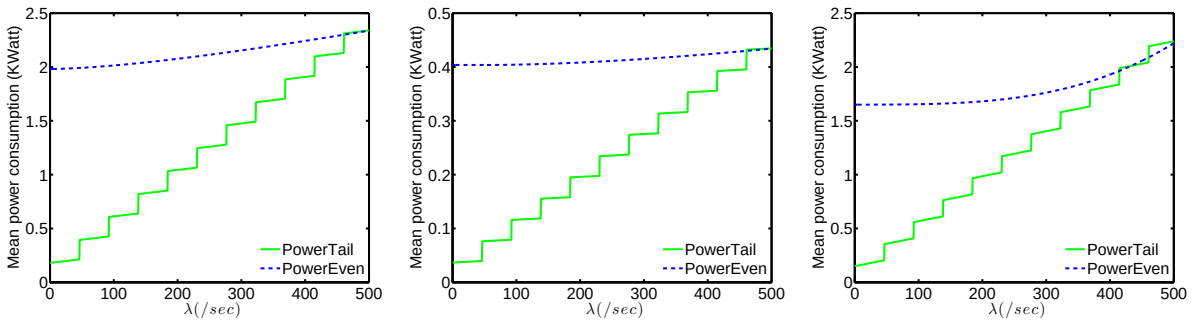
Fig. 5. Power consumption comparison for $\frac{\hat{x}}{\bar{R}} = 0.02$ and $\lambda = 84.6$ per second.

value. We adopted an M/G/1/PS server model, where the job service time distribution is assumed heavy-tailed. We proposed a simple and elegant design strategy *PowerTail*. It is based on our observation that minimizing the overall mean power consumption under a given SLA is equivalent to minimizing the mean energy consumption per job. We investigated how to find the optimal critical utilization and speed ratio setting for *PowerTail* design. Our evaluation results have showed that our *PowerTail* strategy significantly outperforms the intuitive load-balancing strategy *PowerEven*. We have compared our results with other approaches with greedy local DVFS [19]. For future work, we will also consider to adopt control theory to change the CPU frequency in servers under percentile guarantees, by which we will compare for fair comparisons with [18].

Moreover, for systems with heterogeneous servers with discrete speeds, the approximation algorithms in [36] can be adopted to decide the activation of servers and the speeds of the servers, as the algorithms in [36] only require a performance/energy table. For future work, we would also like to consider multi-core servers. The proposed framework can still be applied while the queueing theory should be extended to M/G/k/PS server model.

ACKNOWLEDGMENT

This work is sponsored in part by NSF CAREER Grant No. CNS-0746906, Baden Wuerttemberg MWK Juniorprofessoren-Programms, NSERC Discovery Grant



(a) Type-A servers ($\gamma = 1$) (b) Type-B servers ($\gamma = 1.2137$) (c) Type-C servers ($\gamma = 3$)

Fig. 6. Power consumption comparison for $\frac{\hat{x}}{R} = 0.02$ and $\epsilon = 0.001$.

341823-07, FQRNT grant 2010-NC-131844, and CFI Leaders Opportunity Fund 23090.

REFERENCES

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

[2] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power" struggles: coordinated multi-level power management for the data center," in *ACM ASPLOS*, 2008.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, p. 220, 2007.

[4] 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.

[5] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, *The case for power management in web servers*. Kluwer Academic Publishers, 2002, pp. 261–289.

[6] 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.

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

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

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

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

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

[12] 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.

[13] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *ACM ASPLOS*, 2009.

[14] S. Wang, J. Chen, J. Liu, and X. Liu, "Power saving design for servers under response time constraint," in *Euromicro Conference on Real-Time Systems*, 2010.

[15] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. Shin, "Performance evaluation of virtualization technologies for server consolidation," *HP Laboratories Technical Report*, 2007.

[16] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," in *IEEE Real-Time Systems Symposium*, 2001.

[17] J.-J. Chen, H.-R. Hsu, and T.-W. Kuo, "Leakage-aware energy-efficient scheduling of real-time tasks in multiprocessor systems," in *IEEE Real-time and Embedded Technology and Applications Symposium*, 2006.

[18] L. Bertini, J. Leite, and D. Mossé, "Statistical QoS guarantee and energy-efficiency in web server clusters," in *Euromicro Conference on Real-Time Systems*, 2007.

[19] C. Rusu, A. P. Ferreira, C. Scordino, and A. Watson, "Energy-efficient real-time heterogeneous server clusters," in *IEEE Real Time Technology and Applications Symposium*, 2006.

[20] A. Kamra, V. Misra, and E. Nahum, "Yaksha: a self-tuning controller for managing the performance of 3-tiered web sites," in *IEEE IWQoS*, 2004.

[21] 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.

[22] W. Leland and T. Ott, "Load-balancing heuristics and process behavior," *ACM SIGMETRICS Performance Evaluation Review*, vol. 14, no. 1, pp. 54–69, 1986.

[23] M. Harchol-Balter and A. Downey, "Exploiting process lifetime distributions for dynamic load balancing," *ACM Transactions on Computer Systems*, vol. 15, no. 3, pp. 253–285, 1997.

[24] M. Crovella and A. Bestavros, "Self-similarity in world wide web traffic," *ACM SIGMETRICS Performance Evaluation Review*, vol. 24, no. 1, pp. 160–169, 1996.

[25] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," in *ACM SIGMETRICS*, 1998.

[26] M. Harchol-Balter, "The Effect of Heavy-Tailed Job Size Distributions on Computer System Design." in *ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics, Engineering and Statistics*, 1999.

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

[28] A. Zwart and O. Boxma, "Sojourn time asymptotics in the M/G/1 processor sharing queue," *Queueing systems*, vol. 35, no. 1, pp. 141–166, 2000.

[29] E. Bini, G. C. Buttazzo, and G. Lipari, "Speed modulation in energy-aware real-time systems," in *Euromicro Conference on Real-Time Systems*, 2005.

[30] J. Cohen, "Some results on regular variation for distributions in queueing and fluctuation theory," *Journal of applied probability*, vol. 10, no. 2, pp. 343–353, 1973.

[31] "Web caching project," http://www.ircache.net, trace: sd.sanitized-access.20110120.

[32] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing server energy and operational costs in hosting centers," in *ACM SIGMETRICS*, 2005.

[33] A. Clauset, C. Shalizi, and M. Newman, "Power-Law Distributions in Empirical Data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009, the implementation is accessible via http://tuvalu.santafe.edu/~aaronc/powerlaws.

[34] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[35] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Symposium on Foundations of Computer Science*, 1995.

[36] J.-J. Chen, K. Huang, and L. Thiele, "Power management schemes for heterogeneous clusters under quality of service requirements," in *ACM Symposium on Applied Computing*, 2011.

APPENDIX A
DERIVING OPTIMAL $Q(\rho, r)$

Based on (17b), the feasible value of ρ should satisfy:

$$0 \leq \rho \leq \rho_r, \tag{26}$$

where ρ_r is the upper bound of the feasible ρ and defined as

$$\rho_r \stackrel{\text{def}}{=} 1 - \eta. \tag{27}$$

In order to obtain the optimal $Q = Q(\rho, r)$, we first consider Q as a function of r. We obtain its first partial derivative:

$$\frac{\partial Q}{\partial r} = \frac{\alpha}{\mu}[r - r_l]^{\gamma-1}\frac{[\gamma-1]r + r_l}{r^2}. \tag{28}$$

We observe that $\frac{\partial Q}{\partial r} > 0$ holds for any feasible r. Therefore Q is an increasing function of r. The minimal Q will be achieved at the critical speed r_c, which is the lower bound of the feasible r. Based on (17b), we have

$$r_c = \max\{r_l, \rho + \eta\}. \tag{29}$$

We apply (29) into (17a) and then Q becomes a function of ρ only. Next, we need to find out an optimal ρ to minimize Q, which is defined as the *critical utilization* ρ_c. Notice that r_c that was just applied in (17a) has a max operator as shown in (29). In order to remove this max operator, we divide the feasible region of ρ in (26) into the following sub-intervals: $[0, \rho_m]$ and $[\rho_m, \rho_r]$, where

$$\rho_m \stackrel{\text{def}}{=} \max\{0, r_l - \eta\}. \tag{30}$$

We have $r_l \geq \rho + \eta$ in $[0, \rho_m]$ and $r_l \leq \rho + \eta$ in $[\rho_m, \rho_r]$.

In the sub-interval $[0, \rho_m]$, by (29) we obtain the critical speed ratio as $r_c = r_l$. Applying it in (17a), we obtain the minimum Q as

$$Q = \frac{P_I}{\mu\rho}. \tag{31}$$

Q in (31) is a decreasing function of ρ. If the critical utilization ρ_c falls in this sub-interval, it will fall at $\rho = \rho_m$. Since Q is a continuous function at $\rho = \rho_m$, we could exclude the case of this sub-interval by including $\rho = \rho_m$ in the other sub-interval.

In the following, we focus on the second sub-interval $[\rho_m, \rho_r]$, which includes $\rho = \rho_m$. By (29) we obtain the critical speed ratio as $r_c = \rho + \eta$. Applying it in (17a), we obtain the minimum Q as

$$Q = \frac{1}{\mu}[\frac{\alpha[\rho - \rho_m]^\gamma}{\rho + \eta} + \frac{P_I}{\rho}]. \tag{32}$$

In order to obtain the minimal Q in (32), we study its first partial derivative, which can be obtained as

$$\frac{\partial Q}{\partial \rho} = \frac{H(\rho)}{[\rho + \eta]^2}, \tag{33}$$

where

$$H(\rho) = \alpha[\rho - \rho_m]^{\gamma-1}[[\gamma-1][\rho+\eta] + r_l] - P_I[1 + \frac{\eta}{\rho}]^2. \tag{34}$$

In order to find the minimum Q, we need to study the properties of $H(\rho)$: If $H(\rho) \geq 0$, then $\frac{\partial Q}{\partial \rho} \geq 0$ and hence Q defined in (32) is increasing; If $H(\rho) \leq 0$, then Q is decreasing; If there exists a ρ^* such that $H(\rho^*) = 0$, then Q will have a critical value at $\rho = \rho^*$. Based on these cases, we can obtain the critical utilization ρ_c.

APPENDIX B
PROOF OF THEOREM 4 WITH $\gamma = 1$

We set $\gamma = 1$ in (34), then we have

$$H(\rho) = \alpha r_l - P_I[1 + \frac{\eta}{\rho}]^2. \tag{35}$$

Consider ξ defined in (19). If $\xi \leq 0$, then $\alpha r_l < P_I$ and hence $H(\rho) < 0$ holds always. In this case, by (33) Q is always decreasing. Then the critical utilization will fall at the right boundary, i.e., $\rho_c = \rho_r$, as demonstrated in first plot of Figure 7(a).

If $\xi > 0$, then $\alpha r_l > P_I$. $H(\rho) \geq 0$ means $\rho \geq \rho^*$ and $H(\rho) \leq 0$ means $\rho \leq \rho^*$, where ρ^* is defined as

$$\rho^* = \frac{\eta}{[\frac{\alpha r_l}{P_I}]^{\frac{1}{2}} - 1} = \xi - \eta. \tag{36}$$

We consider the following cases:

- If $0 < \xi < r_l$, then $\rho^* < \rho_m$ and hence $H(\rho) \geq 0$ holds for $\rho \in [\rho_m, \rho_r]$. In this case, Q is increasing in this sub-interval. The minimum Q is achieved at the left boundary, i.e., $\rho_c = \rho_m$, as demonstrated in second plot of Figure 7(a).
- If $r_l \leq \xi < 1$, then $\rho_m \leq \rho^* \leq \rho_r$. $H(\rho) \leq 0$ holds for $\rho \in [\rho_m, \rho^*]$ and $H(\rho) \geq 0$ holds for $\rho \in [\rho^*, \rho_r]$. Hence Q is decreasing first and then increasing in this sub-interval. The minimum Q is achieved inside the sub-interval, i.e., $\rho_c = \rho^*$, as demonstrated in third plot of Figure 7(a), where ρ^* is defined in (36).
- If $\xi \geq 1$, then $\rho^* > \rho_r$ and hence $H(\rho) \leq 0$ holds for $\rho \in [\rho_m, \rho_r]$. In this case, Q is always decreasing in this sub-interval. The minimum Q is achieved at the right boundary, i.e., $\rho_c = \rho_r$, as demonstrated in first plot of Figure 7(a).

With the above obtained critical utilization ρ_c, based on (29), we can also obtain the critical speed ratio r_c. We combine the last case $\xi \geq 1$ with the case $\xi \leq 0$. We can summarize the above observations in Theorem 4.

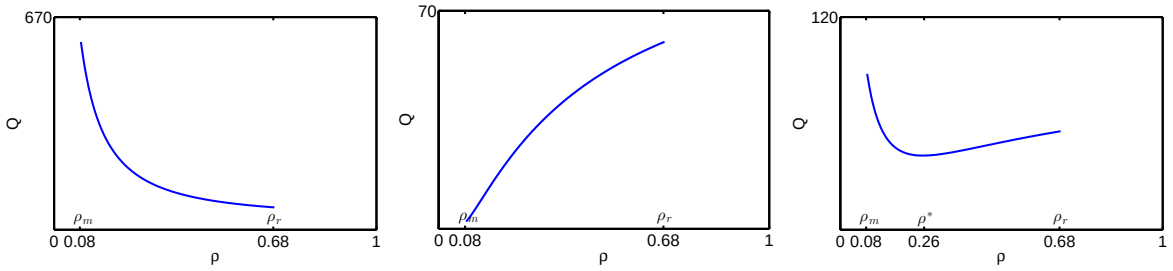
APPENDIX C
PROOF OF THEOREM 5 WITH $\gamma > 1$

We consider the left boundary of the sub-interval $[\rho_m, \rho_r]$, i.e., $\rho = \rho_m$. In this case, we have

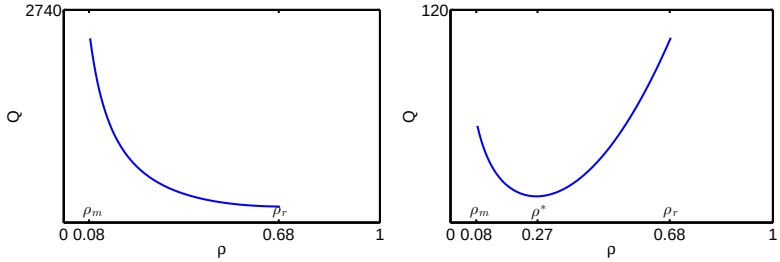
$$H(\rho_m) = -P_I[\frac{r_l}{\rho_m}]^2 < 0. \tag{37}$$

From (34), we also observe that $H(\rho)$ is strictly increasing in the sub-interval $[\rho_m, \rho_r]$. We consider the right boundary of the sub-interval $[\rho_m, \rho_r]$, i.e., $\rho = \rho_r$.

Consider χ defined in (21). If $\chi \leq 0$, then $H(\rho) \leq 0$ holds through the sub-interval $[\rho_m, \rho_r]$. By (33) we know that $\frac{\partial Q}{\partial \rho} \leq 0$ also holds through $[\rho_m, \rho_r]$. Hence Q is always decreasing



(a) Linear-power server



(b) Nonlinear-power server

Fig. 7. Q as a function of ρ in the sub-interval $[\rho_m, \rho_r]$.

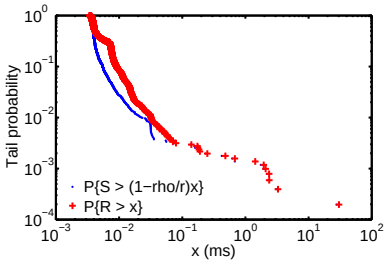
in this sub-interval. The minimum Q is achieved at $\rho_c = \rho_r$, as demonstrated in first plot of Figure 7(b).

If $\chi > 0$, then we the following observations: $H(\rho_m) < 0$, $H(\rho_r) > 0$, and $H(\rho)$ is increasing in the sub-interval $[\rho_m, \rho_r]$. Therefore there exists one and only one $\rho = \rho^*$ such that $H(\rho^*) = 0$ holds. Then $H(\rho) \leq 0$ holds for $\rho \in [\rho_m, \rho^*]$ and $H(\rho) \geq 0$ holds for $\rho \in [\rho^*, \rho_r]$. Hence, by (33) Q is decreasing first and then increasing in this sub-interval as demonstrated in second plot of Figure 7(b). The minimum Q is achieved inside the sub-interval, i.e., $\rho_c = \rho^*$, which is defined in (22).

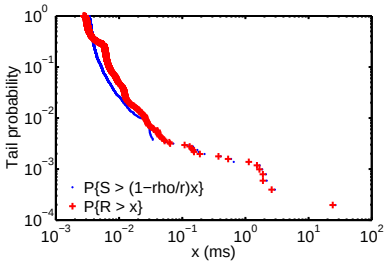
With the above obtained critical utilization ρ_c, based on (29), we can also obtain the critical speed ratio r_c. We can summarize the above observations in Theorem 5.

Appendix D
Verifying Heavy-Tailed Phenomenon

The purpose of the following experiment is to verify the asymptotical approximation of the job response time in Theorem 2. In our verification, we scale the one-minute trace to run it for 6 minutes. We scale the job service time so that we have $\mu = 83.5521$ per second at the maximum speed on the server. We run it on a Type-B server. For evaluating the correctness of single-core systems, we only enable one core in the server. We choose two different frequencies: 2.5GHz and 3.1GHz to examine the response time performance. The result is shown in Figure 8. For 2.5GHz, for the tail probability to reach 0.001, the service needs to reach 1.9268 second for the approximated value and 2.0879 second for the measured value. For 3.1GHz, for the tail probability to reach 0.001, the service needs to reach 1.8472 second for the approximated value and 1.6683 second for the measured value. Both cases confirm that the tail of the job response time can be asymptotically approximated as a function of the tail of the job service time as shown in Theorem 2.



(a) 2.5GHz



(b) 3.1GHz

Fig. 8. Verifying the asymptotical approximation of the job response time.