Power-Aware Speed Scaling in Processor Sharing Systems

Adam Wierman
Computer Science Department
California Institute of Technology

Lachlan L.H. Andrew
Centre for Advanced Internet Architectures
Swinburne University of Technology, Australia

Ao Tang
School of ECE
Cornell University

Abstract—**Energy use of computer communications systems has quickly become a vital design consideration. One effective method for reducing energy consumption is dynamic speed scaling, which adapts the processing speed to the current load. This paper studies how to optimally scale speed to balance mean response time and mean energy consumption under processor sharing scheduling. Both bounds and asymptotics for the optimal speed scaling scheme are provided. These results show that a simple scheme that halts when the system is idle and uses a static rate while the system is busy provides nearly the same performance as the optimal dynamic speed scaling. However, the results also highlight that dynamic speed scaling provide at least one key benefit — significantly improved robustness to bursty traffic and mis-estimation of workload parameters.**

I. INTRODUCTION

Power management is increasingly important in computer communications systems. Not only is the energy consumption of the internet becoming a significant fraction of the energy consumption of developed countries [1], but cooling is also becoming a major concern. Consequently, there is an important tradeoff in modern system design between reducing energy use and maintaining good performance.

There is an extensive literature on power management, reviewed in [2]–[4]. A common technique, which is the focus of the current paper, is dynamic speed scaling [5]–[8]. This dynamically reduces the processing speed at times of low workload, since processing more slowly uses less energy per operation. This is now common in many chip designs [9], [10]. In particular, speed scaling has been proposed for many network devices, such as switch fabrics [11], TCP offload engines [12], and OFDM modulation clocks [13].

This paper studies the efficacy of dynamic speed scaling analytically. The goal is twofold: (i) to elucidate the structure of the optimal speed scaling scheme, e.g., how should the speed depend on the current workload? (ii) to compare the performance of dynamic speed scaling designs with that of designs that use static processing speeds, e.g., how much improvement does dynamic speed scaling provide?

There are many analytic studies of speed scaling designs. Beginning with Yao et al. [14], the focus has been on either (i) the goal of minimizing the total energy used in order to complete arriving jobs by their deadlines, e.g., [15], [16], or (ii) the goal of minimizing the average response time of jobs, i.e., the time between their arrival and their completion of service, given a set energy/heat budget, e.g., [17]–[19].

Web settings typically have neither job completion deadlines nor fixed energy budgets. Instead, the goal is to optimize a tradeoff between energy consumption and mean response time. This model is the focus of the current paper. In particular, the performance metric considered is $\mathbb{E}[T] + \mathbb{E}[E]/\beta'$, where T is the response time of a job, E is the expected energy expended on that job, and β' controls the relative cost of delay.

This performance metric has attracted attention recently [16], [20]–[22]. The related analytic work falls into two categories: worst-case analyses and stochastic analyses. The former provide specific, simple speed scalings guaranteed to be within a constant factor of the optimal performance regardless of the workload, e.g., [16], [20], [21]. In contrast, stochastic results have focused on service rate control in the M/M/1 model under First Come First Served (FCFS) scheduling, which can be solved numerically using dynamic programming. One such approach [22] is reviewed in Section III-C. Unfortunately, the structural insight obtained from stochastic models has been limited.

Our work extends the stochastic analysis of dynamic speed scaling. We focus on the M/GI/1 queue under Processor Sharing (PS) scheduling, which serves all jobs currently in the system at equal rates. We focus on PS because it is a tractable model of current scheduling policies in CPUs, web servers, routers, etc. Based on the model (Section II) and the speed scaling we consider (Section III), our analysis makes three main contributions.

- We provide bounds on the performance of dynamic speed scaling (Section IV-A). Surprisingly, these bounds show that even an idealized version of dynamic speed scaling improves performance only marginally compared to a simple scheme where the server uses a static speed when busy and speed 0 when idle — at most a factor of 2 for typical parameters and often less (see Section V). Counterintuitively, these bounds also show that the power-optimized response time remains bounded as the load grows.

- We provide bounds and asymptotics for the speeds used by the optimal dynamic speed scaling scheme (Sections IV-B and IV-C). These results provide insight into how the speeds scale with the arriving load, the queue length, and the relative cost of energy. Further, they uncover a connection between the optimal stochastic policy and results from the worst-case community (Section IV).

- We illustrate through analytic results and numerical experiments that, though dynamic speed scaling provides limited performance gains, it dramatically improves robustness to mis-estimation of workload parameters and bursty traffic (Section VI).

II. MODEL AND NOTATION

In order to study the performance of dynamic speed scaling, we focus on a simple model: an M/GI/1 PS queue with controllable service rates, dependent on the queue length. In this model, jobs arrive to the server as a Poisson process with rate λ, have intrinsic sizes with mean $1/\mu$, and depart at rate $s_n \mu$ when there are n jobs in the system. Under static schemes, the (constant) service rate is denoted by s. Define the "load" as $\rho = \lambda/\mu$, and note that this ρ is not the fraction of time the server is busy.

The performance metric we consider is $\mathbb{E}[T] + \mathbb{E}[E]/\beta'$, where T is the response time of a job and E is the energy expended on a job. It is often convenient to work with the expected cost per unit time, instead of per job. By Little's law, this can be written as $z = \mathbb{E}[N] + \lambda \mathbb{E}[f(s)]/\beta'$, where N is the number of jobs in the system and $f(s)$ determines the power used when running at speed s.

The remaining piece of the model is to define the form of $f(s)$. Prior literature, with the notable exception of [20], has typically assumed that f is convex, and often, that f is a polynomial, specifically a cubic. That is because the dynamic power of CMOS is proportional to $V^2 f$, where V is the supply voltage and f is the clock frequency [4]. Operating at a higher frequency requires dynamic voltage scaling (DVS) to a higher voltage, nominally with $V \propto f$, yielding a cubic relationship.

To validate the polynomial form of f, we consider data from real 90 nm chips in Fig. 1. The voltage versus speed data comes from the Intel PXA [23], Pentium M 770 processor [24], and the TCP offload engine studied in [12] (specifically the NBB trace at 75°C in Fig 8.4.5). Interestingly, the dynamic power use of real chips is well modeled by a polynomial scaling of speed to power, but this polynomial is far from cubic. In fact, it is closer to quadratic, indicating that the voltage is scaled down less aggressively than linearly with speed. As a result, we will model the power used by running at speed s by

$$\lambda \frac{f(s)}{\beta'} = \frac{s^\alpha}{\beta} \tag{1}$$

where $\alpha > 1$ and β takes the role of β', but has dimension $(\text{time})^{-\alpha}$. The cost per unit time then becomes

$$z = \mathbb{E}[N] + \frac{s^\alpha}{\beta}. \tag{2}$$

We will often focus on the case of $\alpha = 2$ to provide intuition. Clearly, this is an idealized model since in reality only a few discrete speeds can be used.

The impact of the workload parameters ρ, β, and α can often be captured using one simple parameter $\gamma = \rho/\beta^{1/\alpha}$, which is a dimensionless measure. Thus, we will state our results in terms of γ to simplify their form. Also, it will often be convenient to use the the dimensionless unit of speed $s/\beta^{1/\alpha}$.

Though we focus on dynamic power in this paper, it should be noted that leakage power is increasingly important. It represents 20-30% of the power use of current and near-future chips [4]. However, analytic models for leakage are much less understood, and so including leakage in our analysis is beyond
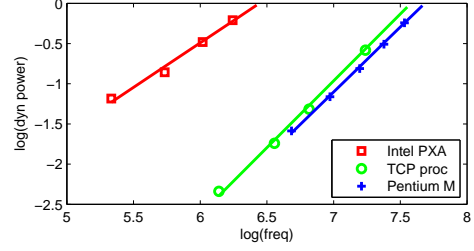

Fig. 1. Dynamic power for an Intel PXA 270, a TCP offload engine, and a Pentium M 770. The slopes of the fitted lines are 1.11, 1.66, and 1.62 respectively.

the scope of this paper and we leave the question of including both leakage and dynamic power for future work.

III. POWER-AWARE SPEED SELECTION

When provisioning processing speed in a power-aware manner, there are three natural thresholds in the capability of the server.

(i) *Static provisioning:* The server uses a constant static speed, which is determined based on workload characteristics so as to balance energy use and response time.

(ii) *Gated static provisioning:* The server "gates" its clock (setting $s = 0$) if no jobs are present, and if jobs are present it works at a constant rate chosen to balance energy use and response time.

(iii) *Dynamic speed scaling:* The server adapts its speed to the current number of requests present in the system.

The goal of this paper is to understand how to choose optimal speeds in each of these scenarios and to contrast the relative merits of each scheme. Clearly the expected cost is reduced each time the server is allowed to adjust its speed more dynamically. This must be traded against the costs of switching, such as a delay of up to tens of microseconds to change speeds [2]. The important question is "What is the magnitude of improvement at each level?" For our comparison, we will use idealized versions of each scheme. In particular, in each case we will assume that the server can be run at any desired speed in $[0, \infty)$ and ignore switching costs. Thus, in particular, the dynamic speed scaling is a significant idealization of what is possible in practice. However, our results will suggest that it provides very little performance improvement over an ideally-tuned gated static scheme.

In this section, we will derive expressions for the optimal speeds in cases (i) and (ii). For case (iii), we will describe a numerical approach for calculating the optimal speeds which is due to George and Harrison [22]. Though this numerical approach is efficient, it provides little structural insight into the structure of the dynamic speeds or the overall performance. Providing such results will be the focus of Section IV.

A. The optimal static speed

The simplest system to manage power is one which selects an optimal speed, and then always runs the processor at that speed. This case, which we call pure static, is the least power-aware scenario we consider, and will be used simply as a benchmark for comparison.

Even when the speed is static, the optimal design can be "power-aware" since the optimal speed can be chosen so that it trades off the cost of response time and energy appropriately. In particular, we can write the cost per unit time (2) as

$$ z = \frac{\rho}{s - \rho} + \frac{s^\alpha}{\beta}. $$

Then, differentiating and solving for the minimizer gives that the optimum s occurs when $s > \rho$ and $s^{\alpha-1}(s-\rho)^2 = \beta\rho/\alpha$.

B. The optimal static speed for a gated system

The next simplest system is when the processor is allowed two states: halted or processing. We model this situation with a server that runs at a constant rate except when there are no jobs in the system, at which point it sets $s = 0$, using zero dynamic power.

To determine the optimal static speed, we proceed as we did in the previous section. If the server can gate its clock, the energy cost is only incurred during the fraction of time the server is busy, ρ/s. The cost per unit time (2) then becomes

$$ z = \frac{\rho}{s - \rho} + \rho\frac{s^{\alpha-1}}{\beta}. $$

The optimum occurs when $s > \rho$ and

$$ 0 = \frac{dz}{ds} = -\frac{\rho}{(s-\rho)^2} + \rho\frac{(\alpha-1)s^{\alpha-2}}{\beta}, $$

which is solved when

$$ (\alpha-1)s^{\alpha-2}(s-\rho)^2 = \beta. \tag{3} $$

The optimal speed can be solved for explicitly for some α. For example, when $\alpha = 2$, $s_{gs} = \rho + \sqrt{\beta}$. In general, define

$$ G(\gamma;\alpha) = \sigma \quad \text{s.t. } \sigma > \gamma $$
$$ (\alpha-1)\sigma^\alpha(1 - \gamma/\sigma)^2 = 1. \tag{4} $$

With this notation, the optimal static speed for a server which gates its clock is $s_{gs} = \beta^{1/\alpha}G(\gamma;\alpha)$. We call this policy the "gated static" policy, and denote the corresponding cost z_{gs}.

The following lemma bounds G. The proof is deferred to Appendix A.

Lemma 1. *For $\alpha \le 2$,[1]*

$$ \gamma + \sqrt{\frac{\gamma^{2-\alpha}}{\alpha-1}} \le G(\gamma;\alpha) \le (\alpha-1)^{-1/\alpha} + \frac{2}{\alpha}\gamma \tag{5} $$

and the inequalities are reversed for $\alpha \ge 2$.

Note that the first inequality becomes tight for $\gamma^\alpha \gg 1$ and the second becomes tight for $\gamma^\alpha \ll 1$. Further, when $\alpha = 2$ both become equalities, giving $G(\gamma;2) = \gamma + 1$.

[1]Equations (6), (12) and (22), and the conditions on α here have been corrected since the version presented at INFOCOM'09.

C. Optimal dynamic speed scaling

A popular alternative to static power management is to allow the speed to adjust dynamically to the number of requests in the system. The task of designing an optimal dynamic speed scaling scheme in our model can be viewed as a stochastic control problem.

We start with the following observation, which simplifies the problem dramatically. An M/GI/1 PS system is well-known to be insensitive to the job size distribution. This still holds when the service rate is queue-length dependent since the policy still falls into the class of *symmetric policies* introduced by Kelly [25]. As a result, the mean response time and entire queue length distribution are affected by the service distribution through only its mean. Thus, we can consider an M/M/1 PS system. Further, the mean response time and entire queue length distribution are equivalent under all non-size-based service distributions in the M/M/1 queue [25]. Thus, to determine the optimal dynamic speed scaling scheme for an M/GI/1 PS queue we need only consider an M/M/1 FCFS queue.

The "service rate control" problem in the M/M/1 FCFS queue has been studied extensively [22], [26], [27]. In particular, George and Harrison [22] provide an elegant solution to the problem of selecting the state-dependent processing speeds to minimize a weighted sum of an arbitrary "holding" cost with a "processing speed" cost. Specifically, the optimal state-dependent processing speeds can be framed as the solution to a stochastic dynamic program, to which [22] provides an efficient numerical solution. In the remainder of this section, we will provide an overview of this numerical approach. The core of this approach will form the basis of our derivation of bounds on the optimal speeds in Section IV.

We will describe the algorithm of [22] specialized to the case considered in this paper, where the holding cost in state n is simply n. Further, we will generalize the description to allow arbitrary arrival rates, λ. The solution starts with an estimate z of the minimal cost per unit time, including both the occupancy cost and the energy cost. As in [22], [27], [28], the minimum cost of returning from state n to the empty system is given by the dynamic program

$$ v_n = \inf_{s \in A} \left\{ \frac{1}{\lambda + \mu s}\left[\lambda\frac{f(s)}{\beta'} + n - z\right] + \frac{\mu s}{\lambda + \mu s}v_{n-1} + \frac{\lambda}{\lambda + \mu s}v_{n+1} \right\} $$

where A is the set of available speeds. We will usually assume $A = [0,\infty)$. With the substitution $u_n = \lambda(v_n - v_{n-1})$, this can be written as [22], [28]

$$ u_{n+1} = \sup_{s \in A}\left\{z - n - \lambda\frac{f(s)}{\beta'} + \frac{su_n}{\rho}\right\}. \tag{6} $$

Two additional functions are defined. First,

$$ \phi(u) = \sup_{x \in A}\{ux/\rho - \lambda f(x)/\beta'\} \tag{7} $$

Second, the minimum value of x which achieves this supremum, normalized to be dimensionless, is

$$\psi(u) = \beta^{-1/\alpha} \min\{x : ux/\rho - \lambda f(x)/\beta' = \phi(u)\}. \quad (8)$$

Note that under (1),

$$\phi(u) = (\alpha - 1)\left(\frac{u}{\alpha\gamma}\right)^{\alpha/(\alpha-1)}, \qquad \psi(u) = \left(\frac{u}{\alpha\gamma}\right)^{1/(\alpha-1)}.$$

Given the estimate of z, u_n satisfy

$$u_1 = z \quad (9a)$$
$$u_{n+1} = \phi(u_n) - n + z. \quad (9b)$$

The optimal value of z can be found as the minimum value such that $(u_n)_{n=1}^{\infty}$ is an increasing sequence. This allows z to be found by an efficient binary search, after which u_n can in principle be found recursively.

The optimal speed in state n is then given by

$$\frac{s_n^*}{\beta^{1/\alpha}} = \psi(u_n). \quad (10)$$

This highlights the fact that $\gamma = \rho/\beta^{1/\alpha}$ provides the appropriate scaling of the workload information because the cost z, normalized speed $s\beta^{-1/\alpha}$ and variables u_n depend on λ, μ and β only through γ.

Note that this "forward" approach advocated in [22] is numerically unstable (Appendix B). We suggest that a more stable way to calculate u_n is to start with a guess for large n, and work backwards. Errors in the initial guess decay exponentially as n decreases, and are much smaller than the accumulated roundoff errors of the forward approach. This backward approach is made possible by the bounds we derive in Section IV.

IV. BOUNDS ON OPTIMAL DYNAMIC SPEED SCALING

In the prior section, we presented the optimal designs for the cases of static, gated static and dynamic speed scaling. In the first two cases, the optimal speeds were presented more-or-less explicitly, however in the third case we presented only a recursive numerical algorithm for determining the optimal dynamic speed scaling. Even though this approach provides an efficient means to calculate s_n^*, it is difficult to gain insight into system design. In this section, we provide results exhibiting the structure of the optimal dynamic speeds and the performance they achieve.

The main results of this section are summarized in Table I. The bounds on z for arbitrary α are essentially tight (i.e., agree to leading order) in the limits of small or large γ. Due to the complicated form of the general results, we illustrate the bounds for the specific case of $\alpha = 2$ to provide insight. In particular, it is easy to see the behavior of s_n and z as a function of γ and n in the case of $\alpha = 2$. This leads to interesting observations. For example, it illustrates a connection between the optimal stochastic policy and policies analyzed in the worst-case model. In particular, Bansal, Pruhs and Stein [21] showed that, when nothing is known about future arrivals, a policy that gives speeds of the form $s_n = (n/(\alpha - 1))^{1/\alpha}$

is constant-competitive, i.e., in the worst case the total cost is within a constant of optimal. This matches the asymptotic behavior of the bounds for $\alpha = 2$ for large n. This behavior can also be observed for general α (Lemma 7 and Theorem 4).

A. Bounds on cost

We start the analysis by providing bounds on z in this subsection, and then using the bounds on z to bound s_n^* above and below (Sections IV-B and IV-C).

Recall that z_{gs} is the total cost under gated static.

Theorem 2.

$$\max\left(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}\right)$$
$$\leq z \leq z_{gs} = \frac{\gamma}{G(\gamma;\alpha) - \gamma} + \gamma G(\gamma;\alpha)^{\alpha-1}$$

Proof: The optimal cost z is bounded above by the cost of the gated static policy, which is simply

$$z_{gs} = \frac{\gamma}{G(\gamma;\alpha) - \gamma} + \gamma G(\gamma;\alpha)^{\alpha-1}. \quad (15)$$

Two lower bounds can be obtained as follows.

In order to maintain stability, the time-average speed must satisfy $\mathbb{E}[s] \geq \rho$. But $z > \mathbb{E}[s^\alpha]/\beta \geq (\mathbb{E}[s])^\alpha/\beta$ by Jensen's inequality and the convexity of $(\cdot)^\alpha$. Thus

$$z > \frac{\mathbb{E}[s^\alpha]}{\beta} \geq \frac{\rho^\alpha}{\beta} = \gamma^\alpha. \quad (16)$$

For small loads, this bound is quite loose. Another bound comes from considering the minimum cost of processing a single job of size X, with no waiting time or processor sharing. It is optimal to serve the job at a constant rate [14]. Thus

$$\frac{z}{\lambda} \geq \mathbb{E}_X\left[\min_s\left(\frac{X}{s} + \frac{s^\alpha}{\beta}\frac{X}{s}\right)\right].$$

The right hand side is minimized for $s = (\beta/(\alpha - 1))^{1/\alpha}$ independent of X, giving $z \geq \rho\beta^{-1/\alpha}\alpha(\alpha - 1)^{(1/\alpha)-1}$. Thus

$$z \geq \max\left(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}\right). \quad (17)$$

∎

The form of the bounds on z are complicated, so it is useful to look at the particular case of $\alpha = 2$.

Corollary 3. *For $\alpha = 2$, gated static has cost within a factor of 2 of optimal. Specifically,*

$$\max(\gamma^2, 2\gamma) \leq z \leq z_{gs} = \gamma^2 + 2\gamma. \quad (18)$$

Proof: For $\alpha = 2$, $G(\gamma; 2) = \gamma + 1$. Hence (15) gives

$$z_{gs} = \frac{\gamma}{(\gamma + 1) - \gamma} + \gamma(\gamma + 1) = \gamma^2 + 2\gamma, \quad (19)$$

which establishes the upper bound.

The lower bound follows from substituting $\alpha = 2$ into (17):

$$z \geq \max(\gamma^2, 2\gamma). \quad (20)$$

The ratio of z_{gs} to the lower bound on z has a maximum value of 2 at $\gamma = 2$, and hence gated static is within a factor of 2 of the true optimal scheme. ∎

TABLE I
BOUNDS ON TOTAL COSTS AND SPEED AS A FUNCTION OF THE NUMBER $n \geq 1$ OF JOBS IN THE SYSTEM.

For any α,

$$\max\left(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}\right) \leq z \leq \frac{\gamma}{G(\gamma;\alpha)-\gamma} + \gamma G(\gamma;\alpha)^{\alpha-1} \qquad \text{Theorem 2} \qquad (11)$$

$$\sigma_n \leq \frac{s_n^*}{\beta^{1/\alpha}} \leq \left(\frac{1}{\alpha}\min_{\sigma>\gamma}\left(\frac{n+\sigma^\alpha-\gamma^\alpha}{(\sigma-\gamma)} + \frac{\gamma}{(\sigma-\gamma)^2}\right)\right)^{1/(\alpha-1)} \qquad \text{Theorems 8 and 4} \qquad (12)$$

where σ_n satisfies $\sigma_n^{\alpha-1}((\alpha-1)\sigma_n - \alpha\gamma) \geq n - (\gamma/(G(\gamma;\alpha)-\gamma) + \gamma G(\gamma;\alpha)^{\alpha-1}$

For $\alpha = 2$,

$$\max\left(\gamma^2, 2\gamma\right) \leq z \leq \gamma^2 + 2\gamma \qquad \text{Corollary 3} \qquad (13)$$

$$\gamma + \sqrt{n-2\gamma} \leq \frac{s_n^*}{\sqrt{\beta}} \leq \gamma + \sqrt{n} + \min\left(\frac{\gamma}{2n}, \frac{3}{2}\left(\frac{\gamma}{4}\right)^{1/3}\right) \qquad \text{Corollaries 9 and 5} \qquad (14)$$

For $\alpha = 2$ and $n < 2\gamma$, a lower bound on s_n results from linear interpolation between $\max(\gamma/2, 1)$ at $n = 1$ and γ at $n = 2\gamma$.

It is perhaps surprising that such an idealized version of dynamic speed scaling provides such a small magnitude of improvement over a simplistic policy such as gated static. In fact, the bound of 2 is very loose when γ is large or small. Further, empirically, the maximum ratios for typical α are below 1.1 (see Fig. 3). Thus there is little to be gained by dynamic scaling in terms of *mean cost*. However, Section VI shows that dynamic scaling dramatically improves robustness.

A second interesting observation about Corollary 3 is that the expected response time under these power aware schemes remains bounded as the arrival rate λ grows. Specifically, by (16),

$$\mathbb{E}[T] = \frac{z}{\lambda} - \frac{\mathbb{E}[s^2/\beta]}{\lambda} \leq \frac{2}{\mu\sqrt{\beta}}.$$

This is a marked contrast to the standard M/GI/1 queue.

B. Upper bounds on the optimal dynamic speeds

We now move to providing upper bounds on the optimal dynamic speed scaling scheme.

Theorem 4. *For all n and α,*

$$u_n \leq \gamma\frac{n+\sigma^\alpha-\gamma^\alpha}{\sigma-\gamma} + \frac{\gamma^2}{(\sigma-\gamma)^2} \qquad (21)$$

for all $\sigma > 0$, whence

$$\frac{s_n^*}{\beta^{1/\alpha}} \leq \left(\frac{1}{\alpha}\min_{\sigma>\gamma}\left(\frac{n+\sigma^\alpha-\gamma^\alpha}{\sigma-\gamma} + \frac{\gamma}{(\sigma-\gamma)^2}\right)\right)^{1/(\alpha-1)}. \qquad (22)$$

In particular, for $\sigma = \gamma + n^{1/\alpha}$,

$$u_n \leq n^{(\alpha-1)/\alpha}\gamma\left(1+(1+\gamma)^\alpha\right) + \gamma^2 \qquad (23)$$

which is concave in n.

Proof: As explained in [28], (6) can be rewritten as

$$u_n = \rho\min_{s_n}\left[\frac{s_n^\alpha/\beta + n + u_{n+1} - z}{s_n}\right]. \qquad (24)$$

Unrolling the dynamic program (24) gives a joint minimization over all s_n

$$u_n = \rho\min_{s_n}\frac{1}{s_n}\Bigg[s_n^\alpha/\beta + n - z$$
$$+ \rho\min_{s_{n+1}}\frac{1}{s_{n+1}}\left[s_{n+1}^\alpha/\beta + (n+1) - z + u_{n+2}\right]\Bigg]$$
$$= \min_{s_i, i\geq n}\sum_{i=n}^{\infty}\left(\prod_{j=n}^{i}\frac{\rho}{s_j}\right)(s_i^\alpha/\beta + i - z). \qquad (25)$$

An upper bound can be found by taking any (possibly suboptimal) choice of s_{n+i} for $i \geq 1$, and bounding the optimal z. Taking $s_i = \sigma\beta^{1/\alpha} > 0$ for all $i \geq n$ gives

$$u_n \leq \min_{\sigma>0}\frac{\gamma}{\sigma}\sum_{j=0}^{\infty}\left(\frac{\gamma}{\sigma}\right)^j(\sigma^\alpha + (n+j) - z)$$
$$= \gamma\min_{\sigma>0}\left[\frac{n+\sigma^\alpha-z}{\sigma-\gamma} + \frac{\gamma}{(\sigma-\gamma)^2}\right].$$

Since $z \geq \gamma^\alpha$ from (17), equation (21) follows. With (10), this establishes (22).

For $n = 0$, (23) holds since $u_0 = 0$. Otherwise, it follows from the inequality $\sigma^\alpha = n(1 + \gamma n^{-1/\alpha})^\alpha \leq n(1+\gamma)^\alpha$ and the fact that $n^{-2/\alpha} \leq 1$. ∎

By specializing to the case when $\alpha = 2$, we can provide some intuition for the upper bound on the speeds.

Corollary 5. *For $\alpha = 2$,*

$$\frac{s_n^*}{\beta^{1/\alpha}} \leq \sqrt{n} + \gamma + \min\left(\frac{\gamma}{2n}, \frac{3}{2}\left(\frac{\gamma}{4}\right)^{1/3}\right). \qquad (26)$$

Proof: Factoring the difference of squares in the first term of (21) and canceling with the denominator yields

$$u_n \leq \frac{\gamma n}{\sigma-\gamma} + \left[2\gamma^2 + \gamma(\sigma-\gamma)\right] + \frac{\gamma^2}{(\sigma-\gamma)^2}. \qquad (27)$$

One term of (27) is increasing in σ, and two are decreasing. Minimizing pairs of these terms gives upper bounds on u_n.

A first bound can be obtained by setting $\sigma - \gamma = \sqrt{n}$, which minimizes the sum of the first two terms, and gives

$$u_n \leq 2\gamma\sqrt{n} + 2\gamma^2 + \frac{\gamma^2}{n}.$$

By (10), this gives a bound on the optimal speeds of

$$\frac{s_n^*}{\sqrt{\beta}} \leq \sqrt{n} + \gamma + \frac{\gamma}{2n}. \tag{28}$$

A second bound comes by minimizing the sum of the second and third terms, when $\sigma - \gamma = (2\gamma)^{1/3}$. This gives

$$u_n \leq \frac{\gamma n}{(2\gamma)^{1/3}} + 2\gamma^2 + \gamma(2\gamma)^{1/3} + \frac{\gamma^2}{(2\gamma)^{2/3}}$$

which, upon division by 2γ, gives

$$\frac{s_n^*}{\sqrt{\beta}} \leq \frac{n}{2} \left(\frac{1}{2\gamma}\right)^{1/3} + \gamma + \frac{3}{2} \left(\frac{\gamma}{4}\right)^{1/3}. \tag{29}$$

The minimum of the right hand sides of (28) and (29) is a bound on s_n.

The result then follows from the fact that

$$\frac{3}{2} \left(\frac{\gamma}{4}\right)^{1/3} \leq \frac{\gamma}{2n} \Rightarrow \frac{n}{2} \left(\frac{1}{2\gamma}\right)^{1/3} \leq \sqrt{n},$$

which follows from taking the square root of the first inequality and rearranging factors. ∎

C. Lower bounds on the optimal dynamic speeds

Finally, we prove lower bounds on the dynamic speed scaling scheme. We begin by bounding the speed used when there is one job in the system. The following result is an immediate consequence of Corollary 3 and (9a).

Corollary 6. *For $\alpha = 2$,*

$$\max\left(\frac{\gamma}{2}, 1\right) \leq \frac{s_1^*}{\sqrt{\beta}} \leq \frac{\gamma}{2} + 1. \tag{30}$$

Observe that the bounds in (30), like those in Corollary 3, are essentially tight for both large and small γ, but loose for γ near 1, especially the lower bound.

Next, we will prove a bound on s_n^* for large n.

Lemma 7. *For sufficiently large n,*

$$\frac{s_n^*}{\beta^{1/\alpha}} > \left(\frac{n}{\alpha - 1}\right)^{1/\alpha}. \tag{31}$$

Proof: Rearrange (9b) as

$$\frac{u_n}{\alpha\gamma} = \left(\frac{n - z + u_{n+1}}{\alpha - 1}\right)^{(\alpha-1)/\alpha} \geq \left(\frac{n}{\alpha - 1}\right)^{(\alpha-1)/\alpha}$$

where the inequality uses the fact that the u_n is non-decreasing [22] hence unbounded as n is unbounded, whence $u_{n+1} - z > 0$ for large n. Applying $s_n^* = \beta^{1/\alpha}(u_n/(\alpha\gamma))^{1/(\alpha-1)}$ gives (31). ∎

This result highlights the connection between the optimal stochastic policy and prior policies analyzed in the worst-case model that we mentioned at the beginning of this section. Specifically, combining (31) with (23) and (10) shows that speeds chosen to perform well in the worst-case are asymptotically optimal (for large n) in the stochastic model. However, note that the probability of n being large is small.

Next, we can derive a tighter, albeit implicit, bound on the optimal speeds.

Theorem 8. *The scaled speed $\sigma_n = s_n^*/\beta^{1/\alpha}$ satisfies*

$$\sigma_n^{\alpha-1}\big((\alpha - 1)\sigma_n - \alpha\gamma\big) \geq n - \frac{\gamma}{G(\gamma; \alpha) - \gamma} - \gamma G(\gamma; \alpha)^{\alpha-1}.$$

Proof: Note that $u_n \leq u_{n+1}$ [22]. Thus by (9b)

$$u_n \leq \frac{\alpha - 1}{(\alpha\gamma)^{\alpha/(\alpha-1)}} u_n^{\alpha/(\alpha-1)} - n + z. \tag{32}$$

By (10), this can be expressed in terms of s_n^* as

$$\alpha\gamma \left(\frac{s_n^*}{\beta^{1/\alpha}}\right)^{\alpha-1} \leq (\alpha - 1)\frac{(s_n^*)^\alpha}{\beta} - n + z$$

whence

$$\left(\frac{s_n^*}{\beta^{1/\alpha}}\right)^{\alpha-1} \left((\alpha - 1)\frac{s_n^*}{\beta^{1/\alpha}} - \alpha\gamma\right) \geq n - z$$

and the result follows from (15) since $z \leq z_{gs}$. ∎

For $\alpha = 2$, the above theorem can be expressed more explicitly as follows.

Corollary 9. *For $\alpha = 2$ and any $n \geq 2\gamma$,*

$$\frac{s_n^*}{\beta^{1/\alpha}} \geq \gamma + \sqrt{n - 2\gamma}. \tag{33}$$

Proof: For $\alpha = 2$, (32) can be solved explicitly, giving

$$u_n \geq 2\gamma^2 + \sqrt{4\gamma^4 + 4\gamma^2(n - z)},$$

since $u_n \geq 0$. By (10),

$$\frac{s_n^*}{\beta^{1/\alpha}} \geq \gamma + \sqrt{(n - z) + \gamma^2} \tag{34}$$

and substituting $z \leq 2\gamma + \gamma^2$ from (18) gives the result. ∎

There are two important observations about the above corollary. First, the corollary only applies when $s^* \geq \rho$, and hence after the mode of the distribution. However, it also proves that the mode occurs at $n \leq 2\gamma$. Second, the corollary only applies when $n \geq 2\gamma$. In this case, we can simplify the upper bound on s_n in (28) and combine it with (33) to obtain:

$$\sqrt{n - 2\gamma} + \gamma \leq \frac{s_n^*}{\sqrt{\beta}} \leq \sqrt{n} + \gamma + \frac{1}{4}. \tag{35}$$

When this form holds, it is tight for large n and/or large γ.

Finally, note that in the case when $n < 2\gamma$ the only bounds we have on the optimal speeds are $s_n^* \geq s_1^* \geq \sqrt{\beta}\max(\gamma/2, 1)$, which follow from Corollary 6 and the fact that s_n^* is increasing in n [22]. The following lemma proves that an improved lower bound can be attained by interpolating linearly between $\max(\gamma/2, 1)$ and γ.

Lemma 10. *The sequence u_n is strictly concave increasing.*

Proof: Let $P(n)$ be the proposition

$$u_{n+1} - u_n \geq u_n - u_{n-1}. \tag{36}$$

Strict concavity of (u_n) is equivalent to there being no n for which $P(n)$ holds. Since (u_n) is non-decreasing [22] and there exists an upper bound on (u_n), (23), with gradient tending to 0, it is sufficient to show that $P(n)$ implies $P(n + 1)$. If so, then any local non-concavity would imply convexity from that

point onwards, in which case its long-term gradient would be positive and bounded away from zero and hence u_n would eventually violate the upper bound (23).

By (9b), $u_{n+1} - u_n = \phi(u_n) - \phi(u_{n-1}) - 1$. With this identity, $P(n)$ is equivalent to

$$\phi(u_n) - \phi(u_{n-1}) - (u_n - u_{n-1}) \geq 1.$$

This implies $u_{n-1} \neq u_n$ and

$$\left(\frac{\phi(u_n) - \phi(u_{n-1})}{u_n - u_{n-1}} - 1 \right) (u_n - u_{n-1}) \geq 1. \qquad (37)$$

Note that the first factor is positive, since the second factor is positive. Since ϕ is convex, there is a subgradient g defined at each point. This gives

$$\left(\frac{\phi(u_n) - \phi(u_{n-1})}{u_n - u_{n-1}} \right) \leq g(u_n) \leq \left(\frac{\phi(u_{n+1}) - \phi(u_n)}{u_{n+1} - u_n} \right).$$

This and (36) imply that both of the factors of (37) increase when going from $P(n)$ to $P(n + 1)$, establishing $P(n + 1)$, and the strict concavity of (u_n). Since it is also non-decreasing [22], the result follows. ∎

V. COMPARING STATIC AND DYNAMIC SCHEMES

To this point, we have only provided analytic results. We now use numerical experiments to contrast static and dynamic schemes. In addition, these experiments will illustrate the tightness of the bounds proven in Section IV on the optimal dynamic speed scaling scheme.

We will start by contrasting the optimal speeds under each of the schemes. Figure 2 compares the optimal dynamic speeds with the optimal static speeds. Note that the bounds on the dynamic speeds are quite tight, especially when the number of jobs in the system, n, is large. For reference, the modes of the occupancy distributions are about 1 and 5, close to the points at which the optimal speed matches the static speeds. Note also that the optimal rate grows only slowly for n much larger than the typical occupancy. This is important since the range over which DVS is possible is limited [4].

Although the speed of the optimal scheme differs significantly from that of gated static, the actual costs are very similar, as predicted by the remark after Corollary 3. This is shown in Fig. 3. The bounds on the optimal speed are also very tight, both for large and small γ. Part (a) shows that the lower bound is loosest for intermediate γ, where the weights given to power and response time are comparable. Part (b)
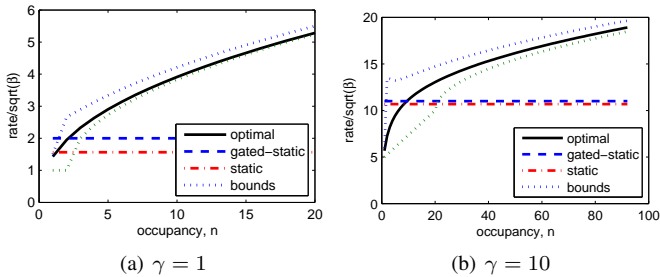

(a) Absolute costs, $\alpha = 2$
(b) Ratio of cost for gated static to optimal, z_{gs}/z.
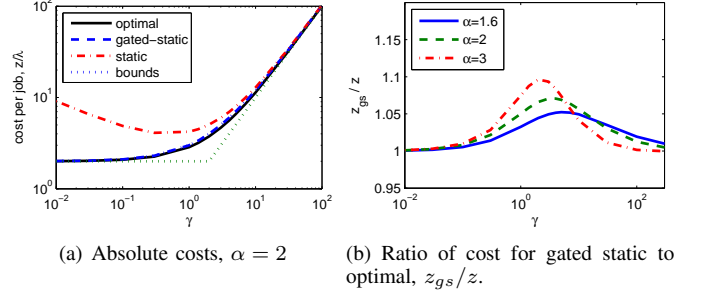
Fig. 3. Cost z vs energy-aware-load γ.

shows that the gated static (i.e., the upper bound) has very close to the optimal cost.

In addition to comparing the total cost of the schemes, it is important to contrast the mean response time and mean energy use. Figure 4 shows the breakdown. A reference load of $\rho = 3$ with delay-aversion $\beta = 1$ and power scaling $\alpha = 2$ was compared against changing ρ for fixed γ, changing β for fixed ρ and changing α. Note $\gamma = 3$ was chosen to maximize the ratio of z_{gs}/z. The second scenario shows that when γ is held fixed, but the load ρ is reduced and delay-aversion is reduced commensurately, the energy consumption becomes negligible.

VI. ROBUST POWER-AWARE DESIGN

We have seen both analytically and numerically that (idealized) dynamic speed scaling only marginally reduces the cost compared to the simple gated static. This raises the question of whether dynamic scaling is worth the complexity. This section illustrates one reason: *robustness*. Specifically, dynamic schemes provide significantly better performance in the face of bursty traffic and mis-estimation of workload.

We focus on robustness with respect to the load, ρ. The optimal speeds are sensitive to ρ, but in reality this parameter must be estimated, and will be time-varying.

It is easy to see the problems mis-estimation of ρ causes for static speed designs. If the load is not known, then the selected speed must be satisfactory for all possible anticipated loads. Consider the case that it is only known that $\rho \in [\underline{\rho}, \bar{\rho}]$. Let $z(\rho_1 | \rho_2)$ denote the expected cost per unit time if the arrival rate is ρ_1, but the speed was optimized for ρ_2. Then, the robust design problem is to select the speed ρ' such that

$$\min_{\rho'} \max_{\rho \in [\underline{\rho}, \bar{\rho}]} z(\rho | \rho').$$

The optimal design is to provision for the highest foreseen load, i.e., $\max_{\rho \in [\underline{\rho}, \bar{\rho}]} z(\rho | \rho') = z(\bar{\rho} | \rho')$. However, this is wasteful in the typical case that the load is less than $\bar{\rho}$. The fragility of static speed designs is illustrated in Fig. 5, which shows that when speed is underprovisioned, the server is unstable, and when it is overprovisioned the design is wasteful.

Optimal dynamic scaling is not immune to mis-estimation of ρ, since s_n^* is highly dependent on ρ. However, because the speed adapts to the queue length, dynamic scaling is more robust. Figure 5 shows this improvement.

This robustness is improved further by the speed scaling scheme, which we term "linear", that scales the server speed

(a) $\gamma = 1$
(b) $\gamma = 10$

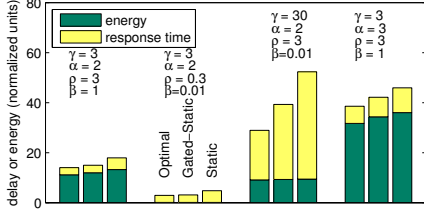Fig. 2. Rate vs n, for $\alpha = 2$ and different energy-aware-load, γ.

Fig. 4. Breakdown of $\mathbb{E}[T]$ and $\mathbb{E}[s^\alpha]$, for several scenarios.

in proportion to the queue length, i.e., $s_n/\beta^{1/\alpha} = n$. Note that under this scaling the queue is equivalent to an M/GI/∞ queue with homogeneous servers. Figure 5 shows that linear scaling provides significantly better robustness than the optimal dynamic scheme; indeed, the "optimal" scheme is only optimal for designs with $\rho \in [7, 14]$; even then, its cost is only slightly lower than that of linear scaling. The (significant) price that linear scaling pays is that it requires very high processing speed when the occupancy is high, which may not be supported by the hardware.

We now compare the robustness analytically in the case of $\alpha = 2$. First, we will show that if ρ is known, the cost of the linear scheme is exactly the same as the cost of the gated static scheme, and thus within a factor of 2 of optimal (Theorem 11). Then, we will show that when the target load differs from the actual load, the linear scheme significantly reduces the cost (Theorem 12). In particular, the linear scaling scheme has cost independent of the difference between the design and actual ρ. In contrast, the cost of gated static grows linearly in this difference, as seen in Fig. 5.

Theorem 11. *When $\alpha = 2$, $z_{gs} = z_{lin}$. Thus, $z_{lin} \le 2z$.*

Proof: If the speed in state n is kn then

$$\mathbb{E}[N] = \frac{\rho}{k} \qquad \mathbb{E}[s_n^2] = \sum_{n=0}^\infty (kn)^2 \frac{(\rho/k)^n}{n!} e^{-\rho/k} = \rho k + \rho^2,$$

and so the total cost is optimized for $k = \sqrt{\beta}$. In this case,

$$z_{lin} = \mathbb{E}[N] + \frac{\mathbb{E}[s_n^2]}{\beta} = \frac{\rho}{\sqrt{\beta}} + \left(\frac{\rho}{\sqrt{\beta}} + \frac{\rho^2}{\beta}\right)$$
$$= \gamma^2 + 2\gamma,$$

which is identical to the cost for gated static. By Corollary 3, this is within a factor of 2 of z. ∎

Theorem 12. *Consider a system designed for target load ρ' that is operating at load $\rho = \rho' - \epsilon$. When $\alpha = 2$,*

$$z_{lin} = \frac{\rho^2}{\beta} + 2\frac{\rho}{\sqrt{\beta}} \tag{38}$$

$$z_{gs} = z_{lin} + \frac{\rho}{\beta}\left(\frac{\epsilon^2}{\sqrt{\beta} + \epsilon}\right). \tag{39}$$

Proof: The optimal rates for the linear policy are $s_n = n\sqrt{\beta}$, independent of ρ'. Thus its cost is always (38).

The optimal speed for gated static in this case is $s_n = \rho' + \sqrt{\beta}$ for $n \ne 0$. When operated at actual load ρ, this gives

$$\mathbb{E}[N] = \frac{\rho}{\sqrt{\beta} + \rho' - \rho} \qquad \frac{\mathbb{E}[s^2]}{\beta} = \frac{\rho\rho'}{\beta} + \frac{\rho}{\sqrt{\beta}}$$
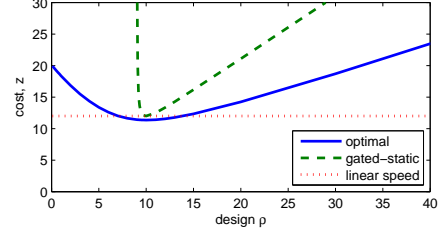

Fig. 5. Cost at load $\rho = 10$, when speeds are designed for "design ρ", using $\beta = 1$, $\alpha = 2$.

and

$$z_{gs} = \frac{\mathbb{E}[s^2]}{\beta} + \mathbb{E}[N] = \frac{\rho^2 + \epsilon\rho}{\beta} + \frac{\rho}{\sqrt{\beta}} + \frac{\rho}{\sqrt{\beta} + \epsilon}.$$

We can further relate z_{gs} to z_{lin} by

$$z_{gs} - z_{lin} = \frac{\epsilon\rho}{\beta} + \frac{\rho}{\sqrt{\beta} + \epsilon} - \frac{\rho}{\sqrt{\beta}}$$
$$= \frac{\epsilon\rho}{\beta} - \frac{\epsilon\rho}{\sqrt{\beta}(\sqrt{\beta} + \epsilon)}$$

from which (39) follows. ∎

This insensitivity to design load mirrors worst-case analysis. The Optimum Available scaling [14], which designs for $\rho = 0$, is $O(1)$ worst-case competitive [5]. However, Fig 5 suggests that linear scaling is much better than designing for $\rho = 0$. Tighter bounds are known for $s_n = n^{1/\alpha}$ [16], [20], but those are still looser than Theorem 11.

VII. CONCLUDING REMARKS

Speed scaling is an important method for reducing energy consumption in computer communication systems. Intrinsically, it trades off the mean response time and the mean energy consumption, and this paper provides insight into this tradeoff using a stochastic analysis.

Specifically, in the M/GI/1 PS model, both bounds and asymptotics for the optimal speed scaling scheme are provided. These bounds are tight for small and large γ and provide a number of insights, e.g., that the mean response time is bounded as the load grows under the optimal dynamic speed scaling and that the optimal dynamic speeds in the stochastic model match (for large n) dynamic speed scalings that have been shown to have good worst-case performance.

Surprisingly, the bounds also illustrate that a simple scheme which gates the clock when the system is idle and uses a static rate otherwise provides performance within a factor of 2 of the optimal dynamic speed scaling. However, the value of dynamic speed scaling is also illustrated — dynamic speed scaling schemes provide significantly improved robustness to bursty traffic and mis-estimation of workload parameters. The dynamic scheme that optimizes the mean cost is no longer optimal when robustness is considered: a scheme that scales speeds linearly with n provides significantly improved robustness while increasing cost only slightly.

There are a number of related directions in which to extend this work. For example, we have only considered dynamic power consumption, which can be modeled as a polynomial of the speed. However, the contribution of leakage power is

growing and an important extension is to develop models of total power use that can be used for analysis. Also, it will be very interesting to extend the analysis to scheduling policies beyond PS. For example, given that the speed can be reduced if there are fewer jobs in the system, it is natural to suggest scheduling according to Shortest Remaining Processing Time first (SRPT), which is known to minimize the number of jobs in the system [29].

VIII. ACKNOWLEDGEMENT

REFERENCES

[1] J. Baliga, R. Ayre, W. Sorin, K. Hinton, and R. Tucker, "Energy consumption in access networks," in *IEEE Conf. Optical Fiber communication (OFC)*, Feb. 2008, pp. 1–3.
[2] O. S. Unsal and I. Koren, "System-level power-aware deisgn techniques in real-time systems," *Proc. IEEE*, vol. 91, no. 7, pp. 1055–1069, 2003.
[3] S. Irani and K. R. Pruhs, "Algorithmic problems in power management," *SIGACT News*, vol. 36, no. 2, pp. 63–76, 2005.
[4] S. Kaxiras and M. Martonosi, *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool, 2008.
[5] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, pp. 1–39, Mar. 2007.
[6] Y. Zhu and F. Mueller, "Feedback EDF scheduling of real-time tasks exploiting dynamic voltage scaling," *Real Time Systems*, vol. 31, pp. 33–63, Dec. 2005.
[7] L. Yuan and G. Qu, "Analysis of energy reduction on dynamic voltage scaling-enabled systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 12, pp. 1827–1837, Dec. 2005.
[8] S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Proc. ISLPED*, 2007, p. 6.
[9] "Intel Xscale." [Online]. Available: www.intel.com/design/intelxscale
[10] "IBM PowerPC." [Online]. Available: http://www-03.ibm.com/technology/power/powerpc.html
[11] L. Mastroleon, D. O'Neill, B. Yolken, and N. Bambos, "Power aware management of packet switches," in *Proc. High-Perf. Interconn.*, 2007.
[12] S. Narendra *et al.*, "Ultra-low voltage circuits and processor in 180 nm to 90 nm technologies with a swapped-body biasing technique," in *Proc. IEEE Int. Solid-State Circuits Conf*, 2004, p. 8.4.
[13] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," in *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008, pp. 135–146.
[14] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, 1995, pp. 374–382.
[15] K. Pruhs, P. Uthaisombut, and G. Woeginger, "Getting the best response for your erg," in *Scandinavian Worksh. Alg. Theory*, 2004.
[16] S. Albers and H. Fujiwara, "Energy-efficient algorithms for flow time minimization," in *Lecture Notes in Computer Science (STACS)*, vol. 3884, 2006, pp. 621–633.
[17] K. Pruhs, R. van Stee, and P. Uthaisombut, "Speed scaling of tasks with precedence constraints," in *Proc. Worksh. Approx. Online Alg.*, 2005.
[18] D. P. Bunde, "Power-aware scheduling for makespan and flow," in *Proc. ACM Symp. Parallel Alg. and Arch.*, 2006.
[19] S. Zhang and K. S. Catha, "Approximation algorithm for the temperature-aware scheduling problem," in *Proc. IEEE Int. Conf. Comp. Aided Design*, Nov. 2007, pp. 281–288.
[20] N. Bansal, H.-L. Chan, and K. Pruhs, "Speed scaling with an arbitrary power function," in *Proc. ACM-SIAM SODA*, 2009.
[21] N. Bansal, K. Pruhs, and C. Stein, "Speed scaling for weighted flow times," in *Proc. ACM-SIAM SODA*, 2007, pp. 805–813.
[22] J. M. George and J. M. Harrison, "Dynamic control of a queue with adjustable service rate," *Oper. Res.*, vol. 49, no. 5, pp. 720–731, 2001.
[23] Intel Corp., "Intel PXA270 processor: Electrical, mechanical, and thermal specification." 2005.
[24] M. Telgarsky, J. C. Hoe, and J. M. F. Moura, "SPIRAL: Joint runtime and energy optimization of linear transforms," in *Proc. ICASSP*, 2006.
[25] F. P. Kelly, *Reversibility and Stochastic Networks*. Wiley, 1979.
[26] B. Ata and S. Shneorson, "Dynamic control of an M/M/1 service system with adjustable arrival and service rates," *Management Science*, vol. 51, no. 11, pp. 1778–1791, Nov. 2006.
[27] D. Low, "Optimal pricing policies for an M/M/s queue," *Oper. Res.*, vol. 22, pp. 545–561, 1974.
[28] J. Wijngaard and S. Stidham, "Forward recursion of Markov decision processes with skip-free-to-the-right transitions, part I: Theory and algorithm," *Math. Oper. Res.*, vol. 11, no. 2, pp. 295–308, May 1986.
[29] L. E. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Oper. Res.*, vol. 16, pp. 678–690, 1968.

APPENDIX A
BOUNDS ON $G(\gamma; \alpha)$

Proof of Lemma 1: Let k_1 satisfy

$$\sigma = G(\gamma; \alpha) = (\alpha - 1)^{-1/\alpha} + k_1\gamma. \qquad (40)$$

Substituting the identity $(a + b)^\alpha = a^\alpha(1 + b/[(a + b) - b])^\alpha$ and (40) into (4) gives

$$1 = (\alpha - 1)(\alpha - 1)^{-\alpha/\alpha}\left(1 + \frac{k_1\gamma}{\sigma - k_1\gamma}\right)^\alpha\left(1 - \frac{\gamma}{\sigma}\right)^2,$$

solved by $(1 - k_1\gamma/\sigma)^{\alpha/2} = 1 - \gamma/\sigma$. Thus, for $\alpha \geq 2$,

$$1 - \frac{\alpha k_1}{2}\frac{\gamma}{s} \leq 1 - \frac{\gamma}{s},$$

with the inequality reversed for $\alpha \leq 2$. For small γ, this inequality tends to equality. Hence $k_1 \geq 2/\alpha$ for $\alpha \geq 2$, and $k_1 \leq 2/\alpha$ for $\alpha \leq 2$ and the second inequality in (5) is accurate to leading order in γ.

Similarly, substituting $G(\gamma; \alpha) = \gamma + k_2$ into (4) gives[2]

$$1 = (\alpha - 1)(\gamma + k_2)^\alpha\left(1 - \frac{\gamma}{\gamma + k_2}\right)^2$$
$$= (\alpha - 1)(\gamma + k_2)^{\alpha-2}k_2^2.$$

This is solved for

$$k_2 = \sqrt{\frac{\gamma^{2-\alpha}}{\alpha - 1}} - \epsilon_2.$$

For $\alpha \geq 2$, $0 \leq \epsilon_2 \to 0$ as $k_2/\gamma \to 0$, which shows that the first inequality of (5) is an upper bound. For $\alpha \leq 2$, $0 \geq \epsilon_2 \to 0$ as $k_2/\gamma \to 0$, which shows that the first inequality of (5) is a lower bound. The requirement $k_2 \ll \gamma$ is then $\gamma \gg \sqrt{\gamma^{2-\alpha}/(\alpha - 1)}$ or equivalently $\gamma^\alpha \gg 1/(\alpha - 1)$. ∎

APPENDIX B
NUMERICAL CONSIDERATIONS OF OPTIMAL SCALING

Let $\hat{u}_n$ and $\hat{z}$ be numerical estimates of u_n and z, with errors $\Delta_n = \hat{u}_n - u_n$ and $\delta = \hat{z} - z$, and consider how errors propagate under (9b). If z is known exactly, then $\Delta_{n+1} = \phi(\hat{u}_n) - \phi(u_n)$ giving

$$|\Delta_{n+1}| > \phi'(\min(u_n, \hat{u}_n))|\Delta_n|$$

since ϕ is convex. If $\phi'(u) = \alpha(u/(\alpha\gamma))^{1/(\alpha-1)} > 1$, then the error grows exponentially if $\hat{y}_{n+1}$ is calculated from $\hat{y}_n$, but decreases exponentially if calculation instead starts from a large n and works backwards using (24). Working backwards requires an initial condition to replace (9a). It is sufficient to choose an initial estimate such as (33).

---

[2]Corrected since the version presented at INFOCOM'09.