Performance Evaluation 69 (2012) 601–622

Contents lists available at SciVerse ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Power-aware speed scaling in processor sharing systems: Optimality and robustness

Adam Wierman[a], Lachlan L.H. Andrew[b,*], Ao Tang[c]

[a] Computer Science Department, California Institute of Technology, United States
[b] Centre for Advanced Internet Architectures, Swinburne University of Technology, Australia
[c] School of ECE, Cornell University, United States

A R T I C L E I N F O

Article history:
Received 9 July 2010
Received in revised form 11 July 2012
Accepted 16 July 2012
Available online 25 July 2012

Keywords:
Energy efficiency
Variable service speed

A B S T R A C T

Adapting the speed of a processor is an effective method to reduce energy consumption. This paper studies the optimal way to scale speed to balance response time and energy consumption under processor sharing scheduling. It is shown that using a static rate while the system is busy provides nearly optimal performance, but having a wider range of available speeds increases robustness to different traffic loads. In particular, the dynamic speed scaling optimal for Poisson arrivals is also constant-competitive in the worst case. The scheme that equates power consumption with queue occupancy is shown to be 10-competitive when power is cubic in speed.

1. Introduction

Power management is increasingly important in computer systems. Not only is the energy consumption of the computing technology 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 often uses less energy per operation. This is now common in many chip designs [9,10] as well as in other domains such as wireless and wired communication links [11,12], and therefore a thorough understanding of its benefits and limitations is required.

Related work. There are many previous analytical studies of speed scaling designs. Beginning with Yao et al. [13], the focus has been on either (i) minimizing the total energy used in order to complete arriving jobs by their deadlines, e.g., [14,15,13], or (ii) 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., [16–18].

Many settings have neither job completion deadlines nor fixed energy budgets. In these cases, 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 energy expended on that job, and β' controls the relative cost of delay. This metric is a practical choice when both delay and energy incur a financial cost.

[*] Corresponding author.
E-mail addresses: l.andrew@ieee.org, landrew@swin.edu.au (L.L.H. Andrew).

0166-5316/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2012.07.002

This performance metric has attracted attention recently, e.g., [19–22]. The related analytical work falls into two categories: worst-case analyses and stochastic analyses. The former tend to provide (i) specific, simple speed scaling designs guaranteed to be within a constant factor of the optimal performance regardless of the workload, e.g., [19–21], or (ii) fundamental limits of the worst-case performance of optimal speed scaling algorithms, typically coupled with optimal schedulers such as shortest remaining processing time first (SRPT) [19,23,21,20]. However, SRPT can often not be implemented because it requires knowledge of the size of a job before the job completes. 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 [24,25,22,26]. One such approach [22] is reviewed in Section 3.3. Unfortunately, the structural insight obtained from stochastic models has been limited.

Contributions of this paper. The current paper builds on the analytical results described above in two key ways. Firstly, it studies speed scaling designs coupled with a practical scheduler, processor sharing (PS), which serves all jobs currently in the system at equal rates. PS is a tractable model for schedulers currently used in operating systems and many other applications. Secondly, it studies both the worst-case performance and the expected performance in a stochastic setting, and provides results that bridge the two models. More specifically, our work combines the stochastic and worst case approaches by studying the structure of an optimal speed scaler for an M/GI/1 PS queue and comparing it with the speed scalers with good worst-case performance under PS. This yields three main contributions, as follows.

We provide upper and lower bounds on the optimal performance of an M/GI/1 PS system with variable speeds, and upper and lower bounds on the optimal speeds. Surprisingly, these bounds show that, when the arrival process is Poisson of a known rate, 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 3.5). Counter-intuitively, these bounds also show that the power-optimized response time remains bounded as the load grows.

We use these bounds to prove that this optimal speed scaler has a finite competitive ratio in the worst case, when the workload need not be Poisson. That is, the performance is within a constant factor of the optimal performance achievable by an off-line algorithm. This demonstrates that the main benefit of dynamic speed scaling is improved robustness.

We provide a tighter upper bound on the best competitive ratio achievable by a non-clairvoyant scheduler, showing in Section 4.2.3 that a competitive ratio of 10 is achievable for typical parameters. We also provide the first competitive analysis of a non-clairvoyant system with unbounded speed in which the scheduler does not require knowledge of the power function and can thus be decoupled from the speed scaling mechanism.

This remainder of the paper is organized as follows. The analytical model is introduced in Section 2. Section 3 introduces both static and dynamic speed scaling schemes, and additionally derives the optimal speeds in each framework. Next, upper and lower bounds on the optimal dynamic scaling are provided in Section 3.4. They show that the optimal dynamic speed scaling does not provide significantly improved performance compared to the optimal static speed. However, in Section 3.5 we use numerical examples to illustrate the tightness of the bounds we prove and contrast the performance of static and dynamic speed scaling schemes. Then, Section 4 demonstrates that dynamic scaling provides significantly improved robustness to workload mis-estimation and bursty traffic. In particular, Section 4 considers the worst-case performance, when the assumption of a stochastic (Poisson) workload is relaxed. Finally, we conclude in Section 5.

2. Model, notation and discussion of assumptions

Let us now introduce the model of the server studied in this paper. The workload models will be different for the stochastic and worst-case analyses, and so will be discussed in their respective sections. We consider a variable speed processor sharing system. When there are $n \geq 0$ jobs in the system, the processor runs at speed s_n, and if $n > 0$ then each job is served at rate s_n/n. The only context in which the speed is allowed to vary for a given n is the adversarial strategy used in the competitive analysis of Section 4.2.

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. In the stochastic setting, $\mathbb{E}[\cdot]$ denotes the ensemble average of the stationary distribution, and in the worst-case setting it denotes the average over a specific (finite) instance. In the stochastic context, 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

$$z = \mathbb{E}[N] + \mathbb{E}[P(s_N)]/\beta', \tag{1}$$

where N is the number of jobs in the system and $P(s)$ denotes the power used when running at speed s.

Next, consider the form of $P(s)$. Prior literature, with the notable exception of [20], has typically assumed that P is convex, and often, that P is a monomial, specifically a cubic. That is because the dynamic power of CMOS is proportional to V^2f, where V is the supply voltage and f is the clock frequency [3]. 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 P, we consider data from real 90 nm chips in Fig. 1. The voltage versus speed data comes from the Intel PXA [27], and Pentium M 770 processor [28], and the TCP offload engine studied in [29] (specifically the NBB trace at 75 °C in Fig. 8.4.5).

Interestingly, the dynamic power use of real chips has the form s^α, but α is much less than 3. In fact, it is closer to quadratic, indicating that the voltage is scaled down sub-linearly with frequency. The reason for this is that physics of the transistors

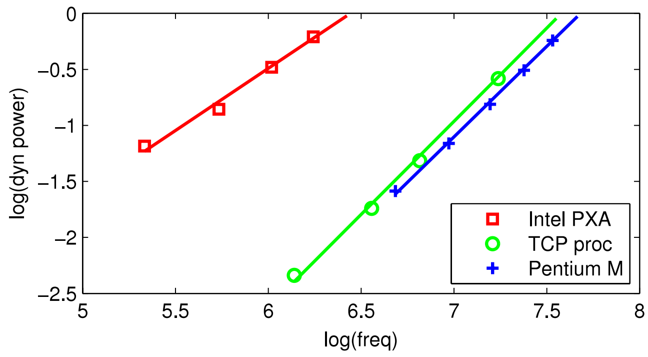


Fig. 1. Dynamic power for an Intel PXA 270, a TCP offload engine, and a Pentium M 770. The slopes of the fitted lines are 1.11, 1.66, and 1.62 respectively.

places lower bounds on the voltage, even when the frequency is reduced; see [30] for a discussion of challenges imposed by operating at very low voltages.

We model the power used by running at speed s by

$$\frac{P(s)}{\beta'} = \frac{s^\alpha}{\beta} \tag{2}$$

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

$$z = \mathbb{E}[N] + \frac{\mathbb{E}[s_N^\alpha]}{\beta} \tag{3}$$

where s_N is the (random) speed. We often focus on the case of $\alpha = 2$ to provide intuition. Note that if a constant were added so that $P(0) > 0$ then the competitive ratios would be reduced. However, in this case it is better to model a case in which servers can "sleep" and incur a cost for switching between sleeping and active modes; see [31] and references therein. That broad field is beyond the scope of this paper.

Since this model is a PS queue with a service rate that increases with the occupancy, it can be viewed as either a multiserver or single server system with variable speed, depending on the form of $P(s)$. A single server running at speed s_n will consume a different amount of power from n servers running at speed s_n/n. The model here corresponds to a single server. Systems with multiple servers have additional complications; see [32] for related analysis in that more difficult case.

Though we focus in this paper on dynamic power (power due to switching of states within the processor or transmission power in a wireless system), note that for CMOS chips, an increasing fraction of power is due to leakage, which is largely independent of processing speed. It represents 20%–30% of the power use of current and near-future chips [3]. However, analytical models for leakage are much less understood, and so including leakage in our analysis is beyond the scope of this paper.

Note that very different tradeoffs arise with other technologies. For example, transmission over a wireless "additive white Gaussian noise" channel requires an exponential increase in power as speed increases, while interference-limited communication channels require unbounded power even to approach a finite maximum capacity. Such applications are again beyond the scope of this paper.

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. Mathematically this includes the previous two approaches as special cases.

In addition to the overall polynomial shape of the power curve, this model makes three assumptions which at first sight may seem restrictive: (i) speed can vary continuously, (ii) the range of speeds is infinite, and (iii) the speed is allowed to depend only on the number of jobs in the system, regardless of the amount of work. These will be discussed in turn.

The assumption that speeds are continuously variable over-estimates the performance benefit obtainable from dynamic scaling. Removing this assumption would strengthen the conclusion that allowing dynamic speeds provides little benefit over a static speed optimally chosen (from a continuous set). Moreover, the optimal scheme only requires discrete speeds, one for each occupancy, and so if β is known in advance then the system need not be designed with continuously variable speeds. Allowing continuously variable speeds over-estimates the robustness results, but numerical results in the context of a fixed maximum speed [33] demonstrate that even having a small number of speeds available can significantly improve robustness.

A model which allows unbounded speeds suggests that dynamic scaling is robust to arbitrary workloads, whereas a static speed would result in instability for high loads. In practice there is a limit on the maximum speed achievable, which limits the set of workloads that can be stable. However, even in that context, dynamic speed selection improves robustness over all stable workloads.

The constraint that the speed depend only on the number of jobs in the system is a natural consequence of the choice of metric. For other metrics, such as average energy plus fractional flow time [21], the optimal speeds may depend instead on the remaining work. To illustrate why little is lost in our context by forcing the speed to depend only on the number of jobs, consider the following two examples. In the first, the workload consists of a single batch of n jobs of sizes $(x_i)_{i=1}^n$ arriving at time 0. For arbitrary differentiable power functions P and objective z of (1), Jensen's inequality implies that it is optimal to serve the jobs at a constant rate $s_i(x)$ between departure instants. The cost is

$$\sum_{i=1}^{n} \frac{x_i}{s_i(x)} (i + P(s_i(x))).$$

Hence it is optimal to run at the speed s_n which satisfies

$$\beta' n = s_n P'(s_n) - P(s_n) \tag{4}$$

whenever there are n jobs in the system independent of the job sizes x; this optimality does not assume *a priori* that the speed depends only on n. The second example is to assume that for a fixed occupancy n, the speed s instead increases without bound for large remaining work. Such a scheme cannot have a constant worst-case competitive ratio. In particular, the cost per unit work of processing a single large job would be unbounded, whereas the optimal solution is given by (4), and results in bounded cost per unit work.

This suggests that choosing speed to depend only on n is reasonable, and we are now ready to investigate performance using this model.

3. Stochastic optimality

The user experience of a computer system is arguably dominated by the average-case performance. For each level of server flexibility, we first study the speed scaling design which gives the best average case performance. 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 $\Lambda = \lambda/\mu$ and recall that the service rate is $s\mu$, so that this Λ is not the fraction of time the server is busy.

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

First, we derive expressions for the optimal static speeds in cases (i) and (ii). For the dynamic case (iii), we 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. Our contribution is to provide such insight, by deriving bounds on the optimal speeds and the resulting performance, and then describing approximations for those quantities. The results in this section are refinements of those in [34].

3.1. 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 (3) as

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

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

3.2. 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, Λ/s. The cost per unit time (3) then becomes

$$z = \frac{\Lambda}{s-\Lambda} + \Lambda\frac{s^{\alpha-1}}{\beta}. \tag{5}$$

The optimum occurs when $s > \Lambda$ and

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

which is solved when

$$(\alpha-1)s^{\alpha-2}(s-\Lambda)^2 = \beta. \tag{6}$$

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

$$\begin{aligned} &G(\gamma;\alpha) = \sigma \quad \text{s.t. } \sigma > \gamma \\ &(\alpha-1)\sigma^{\alpha}(1-\gamma/\sigma)^2 = 1. \end{aligned} \tag{7}$$

That is, σ is the unique root of $(\alpha-1)\sigma^{\alpha}(1-\gamma/\sigma)^2 = 1$ such that $\sigma > \gamma$. This exists and is unique since the left hand side increases monotonically from 0 at $\sigma = \gamma$ to ∞ at $\sigma = \infty$. 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 $\alpha \le 2$,*

$$\gamma + \sqrt{\frac{\gamma^{2-\alpha}}{\alpha-1}} \le G(\gamma;\alpha) \le (\alpha-1)^{-1/\alpha} + \frac{2}{\alpha}\gamma \tag{8}$$

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 $G(\gamma;2) = \gamma + 1$.

3.3. 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 [35]. 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 [35]. 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 [36,22,37]. 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 provide an overview of this numerical approach. The core of this approach forms the basis of our derivation of bounds on the optimal speeds in Section 3.4.

We now describe the fixed point of [22] specialized to the case considered in this paper, where the holding cost in state n is simply n. This description is generalized to allow arbitrary arrival rates, λ. Let z^* be the minimal cost per unit time, including both the occupancy cost and the energy cost. As in [22,37,38], the minimum expected excess cost (above the mean z^* per unit time) of returning from state n to the empty system is given by the dynamic program whose relative value function is

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

where A is the set of available speeds. We usually assume $A = [0, \infty)$. With the substitution $u_n = \lambda(v_n - v_{n-1})$, this can be written as [22,38]

$$u_{n+1} = \sup_{s \in A} \left\{ z^* - n - \lambda \frac{P(s)}{\beta'} + \frac{s u_n}{\Lambda} \right\}. \tag{9}$$

As in [22], two additional functions are defined. First,

$$\phi(u) = \sup_{x \in A} \{ ux/\Lambda - \lambda P(x)/\beta' \} \tag{10}$$

is a scaled form of the convex conjugate of P. Second, the minimum value of x (i.e., speed) which achieves this supremum is

$$\psi(u) = \min\{x : ux/\Lambda - \lambda P(x)/\beta' = \phi(u)\}. \tag{11}$$

Note that under (2),

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

More generally, ϕ and ψ both satisfy the technical conditions of being finite whenever P is convex with unbounded subgradient, and non-zero for $u \neq 0$ whenever $P'(0) = 0$.

As shown in [22], the u_n satisfy

$$u_1 = z^* \tag{12a}$$

$$u_{n+1} = \phi(u_n) - n + z^*. \tag{12b}$$

The optimal value z^* can be found as the minimum value such that $(u_n)_{n=1}^{\infty}$ is an increasing sequence [22]. This allows z^* to be found by an efficient binary search, after which u_n can in principle be found recursively. Note that $z^* > 0$ since $\phi(0) = 0$ and $u_2 \geq u_1$, and z^* is finite provided $P(s)$ is finite for all s, whence u_n is finite and non-zero for all n by induction.

The optimal speed in state n is then given by

$$\frac{s_n^*}{\beta^{1/\alpha}} = \frac{\psi(u)}{\beta^{1/\alpha}} \in (0, \infty). \tag{13}$$

This form highlights the fact that, under (2) when $P(s) \propto s^\alpha$, the appropriate scaling of the workload information is $\gamma = \Lambda/\beta^{1/\alpha}$, because the cost z, normalized speed $s\beta^{-1/\alpha}$ and variables u_n depend on λ, μ and β only through γ.

As an aside, note that this "forward" recursive 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 the following sections.

3.4. *Bounds on stochastically optimal scaling*

So far, we have presented the optimal designs for the cases of static, gated static and dynamic speed scaling. In the first two cases, the optimal speeds were more-or-less explicit; however in the third case only a recursive numerical algorithm for determining the optimal dynamic speed scaling is available. Although this 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 1. The bounds on z^* for arbitrary α are essentially tight (i.e., agree to leading order) in the limits of small or large γ. Because the upper bound in (14) is fairly obscure, the result for $\alpha = 2$ is also provided in (16). Similarly, the upper bound on speed in (15) is only an asymptotic upper bound, and so an explicit bound is given in (17) for $\alpha = 2$. Although the bounds in (15) are only asymptotic, they illustrate a connection between the optimal stochastic policy and policies analyzed in the worst-case model. In particular [21] showed that, when nothing is known about future arrivals, a policy that gives speeds of the form $s_n = n^{1/\alpha}$ is constant-competitive, i.e., in the worst case the total cost is within a constant of optimal. In [23], this was shown to have competitive ratio exactly 2, which is the best bound that is achievable in general. This matches the bounds for $\alpha = 2$ up to leading order for large n.

3.4.1. *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 3.4.2 and 3.4.3).

Theorem 2. *For convex, strictly increasing $P : \mathbb{R} \to \mathbb{R}$ with $P(0) = 0$, the optimal cost per unit time for a* G/G/1 *queue satisfies*

$$\max \left(\frac{P(\Lambda)}{\beta}, \ \Lambda \inf_{s > 0} \left(\frac{1}{s} + \frac{P(s)}{\beta s} \right) \right) \leq z^* \leq z_{gs},$$

Table 1
Bounds on total costs and speed as a function of the number $n \geq 1$ of jobs in the system.

For any α,		
$\max\left(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}\right) \leq z^* \leq \frac{\gamma}{G(\gamma;\alpha)-\gamma} + \gamma G(\gamma;\alpha)^{\alpha-1}$	Theorem 2	(14)
$\left(\frac{n}{\alpha-1}\right)^{1/\alpha} \leq \frac{s_n^*}{\beta^{1/\alpha}} \leq \left(\frac{n}{\alpha-1}\right)^{1/\alpha} + \gamma\frac{1}{\alpha-1} + O(n^{-1/\alpha})$	Lemma 7 and Theorem 4	(15)
For $\alpha = 2$,		
$\max\left(\gamma^2, 2\gamma\right) \leq z^* \leq \gamma^2 + 2\gamma$	Corollary 3	(16)
$\sqrt{n-2\gamma} + \gamma \leq \frac{s_n^*}{\sqrt{\beta}} \leq \sqrt{n} + \gamma + \min\left(\frac{\gamma}{2n}, \frac{3}{2}\left(\frac{\gamma}{4}\right)^{1/3}\right)$	Corollaries 5 and 9	(17)
For $\alpha = 2$ and $n < 2\gamma$, $s_n^*/\sqrt{\beta} \geq \sqrt{n}$; a further lower bound on s_n^* results from linear interpolation between $\max(\gamma/2, 1)$ at $n = 1$ and γ at $n = 2\gamma$, or more precisely, $\gamma + \sqrt{\lceil 2\gamma\rceil - 2\gamma}$ at $n = \lceil 2\gamma\rceil$.		

where z_{gs} is the total cost per unit time of the optimal gated static policy for that queue. In particular, for $P(s) = s^\alpha$, the optimal cost per unit time for an M/GI/1 *queue satisfies*

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

Proof. The optimal cost z^* is bounded above by the cost of the gated static policy. For an M/GI/1 queue with $P(s) = s^\alpha$, by (5) this is

$$z_{gs} = \frac{\gamma}{G(\gamma;\alpha)-\gamma} + \gamma G(\gamma;\alpha)^{\alpha-1}. \tag{18}$$

Two lower bounds can be obtained as follows.

Let $\langle\cdot\rangle$ denote the time average operator. In order to maintain stability, the time-average speed must satisfy $\langle s\rangle \geq \Lambda$.[1] But $z^* > \langle P(s)\rangle/\beta \geq P(\langle s\rangle)/\beta$ by Jensen's inequality and the convexity of P. Thus, since P is increasing,

$$z^* > \frac{P(\langle s\rangle)}{\beta} \geq \frac{P(\Lambda)}{\beta}. \tag{19}$$

For $P(s) = s^\alpha$, the right hand side is $\Lambda^\alpha/\beta = \gamma^\alpha$.

For small loads, the bound of (19) is quite loose. An additional lower 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 [13]. Thus

$$\frac{z^*}{\lambda} \geq \mathbb{E}_X\left[\inf_{s>0}\left(\frac{X}{s} + \frac{P(s)}{\beta}\frac{X}{s}\right)\right].$$

Since the X factors out, and $\mathbb{E}_X[X]\lambda = \Lambda$, this becomes

$$z^* \geq \Lambda \inf_{s>0}\left(\frac{1}{s} + \frac{P(s)}{\beta}\frac{1}{s}\right).$$

For $P(s) = s^\alpha$, the infimum occurs for $s = (\beta/(\alpha-1))^{1/\alpha}$, whence $z^* \geq \Lambda\beta^{-1/\alpha}\alpha(\alpha-1)^{(1/\alpha)-1}$. Thus

$$z^* \geq \max\left(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}\right). \quad \square \tag{20}$$

Remark 1. The lower bound (20) applies to the expected performance of any speed-scaling schedulers, including off-line schedulers, schedulers that do not impose the restriction that speed be a function of n, and schedulers that do not use PS.

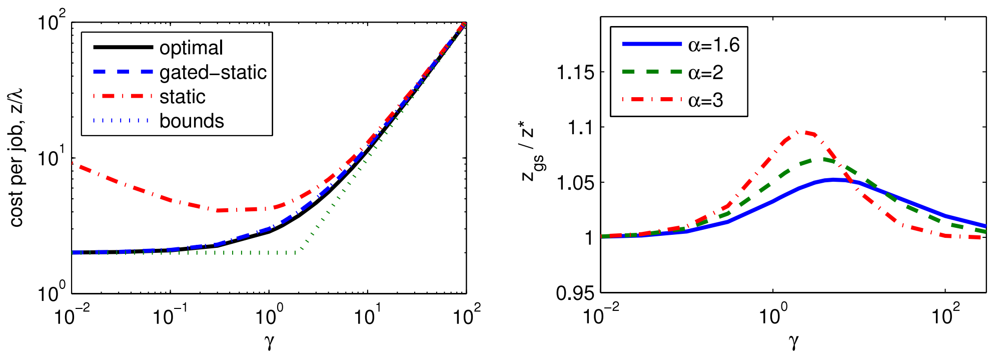
Remark 2. Theorem 2 implies $z^* \in (0, \infty)$ for $P(s) = s^\alpha$. Since the sequence (u_n) is increasing and $u_1 = z^*$, this implies $u_n > 0$ for all $n \geq 1$. Similarly u_n is finite for all n, since each term in (12b) is finite by induction. By (13), this implies $s_n \in (0, \infty)$ for all $n \geq 1$.

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,*

$$\max(\gamma^2, 2\gamma) \leq z^* \leq z_{gs} = \gamma^2 + 2\gamma. \tag{21}$$

[1] This holds with equality iff $s_0 = 0$.



(a) Absolute costs, $\alpha = 2$.

(b) Ratio of cost for gated static to optimal, z_{gs}/z^*.

Fig. 2. Cost z^* vs energy-aware-load γ.

Proof. For $\alpha = 2$, $G(\gamma; 2) = \gamma + 1$. Hence (18) gives

$$z_{gs} = \frac{\gamma}{(\gamma+1)-\gamma} + \gamma(\gamma+1) = \gamma^2 + 2\gamma, \tag{22}$$

which establishes the upper bound.

The lower bound follows from substituting $\alpha = 2$ into (20):

$$z^* \geq \max(\gamma^2, 2\gamma). \tag{23}$$

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. □

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, as shown in Fig. 2. 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) shows that the gated static (i.e., the upper bound) has cost very close to the optimum. Specifically, the maximum ratios for typical α are below 1.1. This suggests that there is little to be gained by dynamic scaling in terms of *mean cost*. However, Section 4 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 (19),

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

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

3.4.2. Upper bounds on the optimal dynamic speeds

We now provide upper bounds on the optimal dynamic speed scaling scheme.

Theorem 4. *For all n and α,*

$$u_n \leq \gamma\frac{n+\sigma^\alpha-\gamma^\alpha}{\sigma-\gamma} + \frac{\gamma^2}{(\sigma-\gamma)^2} \tag{24}$$

for all $\sigma > \gamma$, whence

$$\frac{s_n^*}{\beta^{1/\alpha}} \leq \left(\frac{1}{\alpha}\min_{\sigma>\gamma}\left(\frac{n+\sigma^\alpha-\gamma^\alpha}{\sigma-\gamma} + \frac{\gamma}{(\sigma-\gamma)^2}\right)\right)^{1/(\alpha-1)} \tag{25}$$

$$\leq \left(\frac{n}{\alpha-1}\right)^{1/\alpha} + \gamma\frac{1}{\alpha-1} + O(n^{-1/\alpha}) \tag{26}$$

where $O(n^{-1/\alpha})$ denotes terms that tend to zero no slower than $n^{-1/\alpha}$ as $n \to \infty$, and the notation $A(n) \leq B(n) + O(f(n))$ means that there exists a function $g(n)$ such that $g(n) = O(f(n))$ and $A(n) \leq B(n) + g(n)$.

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

$$u_n \le n^{(\alpha-1)/\alpha} \gamma \; (1 + (1+\gamma)^\alpha) + \gamma^2 \tag{27}$$

which is concave in n.

Proof. As explained in [38], (9) can be rewritten as

$$u_n = \Lambda \min_{s_n} \left[\frac{s_n^\alpha/\beta + n + u_{n+1} - z^*}{s_n} \right]. \tag{28}$$

Unrolling the dynamic program (28) gives a joint minimization over all s_n

$$\begin{aligned} u_n &= \Lambda \min_{s_n} \frac{1}{s_n} \left[s_n^\alpha/\beta + n - z^* + \Lambda \min_{s_{n+1}} \frac{1}{s_{n+1}} \left[s_{n+1}^\alpha/\beta + (n+1) - z^* + u_{n+2} \right] \right] \\ &= \min_{s_i, i \ge n} \sum_{i=n}^{\infty} \left(\prod_{j=n}^{i} \frac{\Lambda}{s_j} \right) \left(s_i^\alpha/\beta + i - z^* \right). \end{aligned} \tag{29}$$

Note that these s_i are the optimal speeds, which satisfy $s_i \in (0, \infty)$ as discussed in Remark 2. As an aside, note that (29) provides an interpretation for u_n, as the stationary expected excess cost, above the minimum cost z^*, accrued in states with occupancy at least n, divided by the probability that $N = n$ given $N \ge n$.

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

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

Since $z^* \ge \gamma^\alpha$ from (20), Eq. (24) follows. With (13), this establishes (25).

The minimum in (25) cannot be greater than the argument for a given $\sigma > \gamma$. Take $\sigma = (n/(\alpha-1))^{1/\alpha} + \gamma$. Then

$$\begin{aligned} \left(\frac{1}{\alpha} \left(\frac{n + \sigma^\alpha - \gamma^\alpha}{\sigma - \gamma} + \frac{\gamma}{(\sigma-\gamma)^2} \right) \right)^{1/(\alpha-1)} &= \left(\frac{1}{\alpha} \left(\frac{n + \sigma^\alpha - \gamma^\alpha + O(n^{-1/\alpha})}{\sigma - \gamma} \right) \right)^{1/(\alpha-1)} \\ = \left(\frac{1}{\alpha} \left(\frac{n + \left(\frac{n}{\alpha-1} + \alpha \left(\frac{n}{\alpha-1} \right)^{1-1/\alpha} \gamma + O(n^{1-2/\alpha}) + \gamma^\alpha \right) - \gamma^\alpha}{(n/(\alpha-1))^{1/\alpha}} \right) \right)^{1/(\alpha-1)} & \\ = \left(\left(\frac{n}{\alpha-1} \right)^{1-1/\alpha} + \gamma \left(\frac{n}{\alpha-1} \right)^{1-2/\alpha} + O(n^{1-3/\alpha}) \right)^{1/(\alpha-1)} & \\ = \left(\frac{n}{\alpha-1} \right)^{1/\alpha} \left(1 + \gamma \left(\frac{n}{\alpha-1} \right)^{-1/\alpha} + O(n^{-2/\alpha}) \right)^{1/(\alpha-1)} & \\ = \left(\frac{n}{\alpha-1} \right)^{1/\alpha} + \frac{\gamma}{\alpha-1} + O(n^{-1/\alpha}) & \end{aligned}$$

which establishes (26).

For $n = 0$, (27) holds since $u_0 = 0$. Otherwise, it follows from the inequality $\sigma^\alpha = n(1 + \gamma n^{-1/\alpha})^\alpha \le n(1+\gamma)^\alpha$ and the fact that $n^{-2/\alpha} \le 1$. □

Although (26) has only been proven up to $O(n^{-1/\alpha})$ terms, numerical evidence suggests that these terms are negative, which would make the first two terms of (26) an upper bound on $s^n/\beta^{1/\alpha}$.

Theorem 4 has some natural interpretations. It is to be expected that, when the occupancy n is very much larger than average, the speed is approximately the optimal speed for batch processing when there are n in the system, given by (4). It is also not surprising that the offset increases with γ, and decreases in α since the cost of additional speed is more expensive for larger α. It is more noteworthy that the offset tends to a constant, $\gamma/(\alpha-1)$.

By specializing to the case when $\alpha = 2$, we can provide explicit bounds on the $O(n^{-1/\alpha})$ terms.

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

$$\frac{s_n^*}{\beta^{1/\alpha}} \le \sqrt{n} + \gamma + \min\left(\frac{\gamma}{2n}, \frac{3}{2}\left(\frac{\gamma}{4}\right)^{1/3}\right). \tag{30}$$

Proof. Factoring the difference of squares in the first term of (24) and canceling with the denominator yields

$$u_n \le \frac{\gamma n}{\sigma - \gamma} + \left[2\gamma^2 + \gamma(\sigma - \gamma)\right] + \frac{\gamma^2}{(\sigma - \gamma)^2}. \tag{31}$$

One term of (31) 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

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

By (13), this gives a bound on the optimal speeds of

$$\frac{s_n^*}{\sqrt{\beta}} \le \sqrt{n} + \gamma + \frac{\gamma}{2n}. \tag{32}$$

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

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

which, upon division by 2γ, gives

$$\frac{s_n^*}{\sqrt{\beta}} \le \frac{n}{2}\left(\frac{1}{2\gamma}\right)^{1/3} + \gamma + \frac{3}{2}\left(\frac{\gamma}{4}\right)^{1/3}. \tag{33}$$

The minimum of the right hand sides of (32) and (33) is a bound on s_n.

The result then follows from the fact that

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

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

3.4.3. 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 (12a).

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

$$\max\left(\frac{\gamma}{2}, 1\right) \le \frac{s_1^*}{\sqrt{\beta}} \le \frac{\gamma}{2} + 1. \tag{34}$$

Observe that the bounds in (34), like those in Corollary 3, are essentially tight for both large and small γ, but loose for γ near 1, especially the lower bound.

We now state a simple bound on s_n^*.

Lemma 7. *For all $n \ge 1$, $\alpha \ge 1$ and $\beta > 0$,*

$$\frac{s_n^*}{\beta^{1/\alpha}} \ge \left(\frac{n}{\alpha - 1}\right)^{1/\alpha}. \tag{35}$$

Proof. Since u_n is non-decreasing [22], we have $u_{n+1} \ge u_1 = z^*$ by (12a). Thus (12b) becomes

$$\frac{u_n}{\alpha\gamma} = \left(\frac{n - z^* + u_{n+1}}{\alpha - 1}\right)^{(\alpha-1)/\alpha} \ge \left(\frac{n}{\alpha - 1}\right)^{(\alpha-1)/\alpha}. \tag{36}$$

Applying $s_n^*/\beta^{1/\alpha} = (u_n/(\alpha\gamma))^{1/(\alpha-1)}$ from (13) gives (35). □

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*

$$\sigma_n^{\alpha-1}\big((\alpha-1)\sigma_n - \alpha\gamma\big) \geq n - \frac{\gamma}{G(\gamma;\alpha)-\gamma} - \gamma G(\gamma;\alpha)^{\alpha-1}. \tag{37}$$

Proof. Note that $u_n \leq u_{n+1}$ [22]. Thus by (12b)

$$u_n \leq \frac{\alpha-1}{(\alpha\gamma)^{\alpha/(\alpha-1)}} u_n^{\alpha/(\alpha-1)} - n + z^*. \tag{38}$$

By (13), 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 (37) follows from (18) 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 $n \geq 2\gamma$,*

$$\frac{s_n^*}{\beta^{1/\alpha}} \geq \gamma + \sqrt{n-2\gamma}. \tag{39}$$

Proof. For $\alpha = 2$, (38) 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 (13),

$$\frac{s_n^*}{\beta^{1/\alpha}} \geq \gamma + \sqrt{(n-z^*)+\gamma^2} \tag{40}$$

and substituting $z^* \leq 2\gamma + \gamma^2$ from (21) gives the result. □

There are two important observations about the above corollary. First, the corollary only applies when $s^* \geq \Lambda$, 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 (32) and combine it with (39) to obtain:

$$\sqrt{n-2\gamma} + \gamma \leq \frac{s_n^*}{\sqrt{\beta}} \leq \sqrt{n} + \gamma + \frac{1}{4}. \tag{41}$$

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

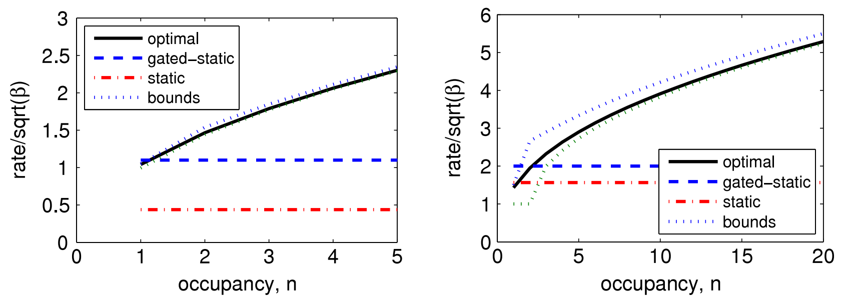
Finally, when $n < 2\gamma$, Corollary 9 does not apply and Lemma 7 may be loose. This leaves the other loose bound $s_n^* \geq s_1^* \geq \sqrt{\beta}\max(\gamma/2, 1)$, which follows from Corollary 6 and the fact that s_n^* is increasing in n [22]. The following lemma proves that an improved lower bound can be attained by interpolating linearly between $\max(\gamma/2, 1)$ for $n = 1$ and γ for $n = 2\gamma$.

Lemma 10. *The sequence u_n is strictly concave increasing. Consequently, s_n^* is strictly concave increasing for $\alpha \geq 2$.*

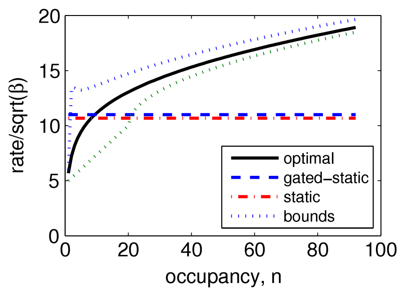
Proof. Let $\mathsf{P}(n)$ be the proposition

$$u_{n+1} - u_n \geq u_n - u_{n-1}. \tag{42}$$

Strict concavity of (u_n) is equivalent to there being no n for which $\mathsf{P}(n)$ holds. First, we claim that it is sufficient to show that $\mathsf{P}(n)$ implies $\mathsf{P}(n+1)$. To see this, note that if that implication holds, then if there is an i such that $u_.$ is not strictly concave at i, then $u_.$ would be convex for all $n > i$. This implies that either u_n is constant for all $n > i$, or there exists a lower bound on u_n for $n > i$ of the form $u_n \geq k_1 + k_2 n$ with $k_2 > 0$. It is impossible to have u_n constant for all $n > i$, since if (12b) holds for some $n > i$ then it would be violated for $n+1$. Similarly, having a bound with $k_2 > 0$ implies u_n would eventually violate



(a) $\gamma = 0.1$. (b) $\gamma = 1$.



(c) $\gamma = 10$.

Fig. 3. Rate vs n, for $\alpha = 2$ and different energy-aware-load, γ.

the upper bound (27). Thus, if $\mathsf{P}(u)$ implies $\mathsf{P}(u+1)$ then the sequence (u_n) must be strictly concave. This establishes the claim. It remains to show that $\mathsf{P}(n)$ implies $\mathsf{P}(n+1)$.

By (12b), $u_{n+1} - u_n = \phi(u_n) - \phi(u_{n-1}) - 1$. With this identity, $\mathsf{P}(n)$ 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 hence $\mathsf{P}(n)$ is equivalent to

$$\left(\frac{\phi(u_n) - \phi(u_{n-1})}{u_n - u_{n-1}} - 1\right)(u_n - u_{n-1}) \geq 1. \tag{43}$$

Note that $\mathsf{P}(n)$ implies 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. By the definition of the subgradient,

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

and so the first factor of (43) is increasing in n. Since (42) implies that the second factor of (43) also increases when going from $\mathsf{P}(n)$ to $\mathsf{P}(n+1)$, and since the product of two positive increasing functions is increasing, this establishes that $\mathsf{P}(n)$ implies $\mathsf{P}(n+1)$. Hence (u_n) is strictly concave. Since it is also non-decreasing [22], the result about u_n follows. Since $\psi(\cdot)$ is concave increasing for $\alpha \geq 2$, the result about s_n^* follows by (13). □

Numerical evidence suggests that s_n^* is also concave for $\alpha \in (0, 2)$, although the above proof does not apply.

3.5. Comparing static and dynamic schemes

To this point, we have focused on analytical results. We now use numerical experiments to contrast static and dynamic schemes.

Although Fig. 2 showed that the optimal cost of gated-static and dynamic speed scaling are similar, the actual speeds are quite different. Fig. 3 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. 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 [3].

To interpret these numerical results, it is important to know what ranges of the normalized load γ are realistic. The following reasoning suggests that the range $\gamma \in [0.1, 10]$ is probably typical. If $\gamma \ll 1$ then the utilization of the server becomes very small. Because each server incurs a capital cost, operators are likely to consolidate load onto fewer servers if γ is below around 0.1. Conversely, if $\gamma \gg 1$, then the number of jobs waiting will be large. For a constant speed server,

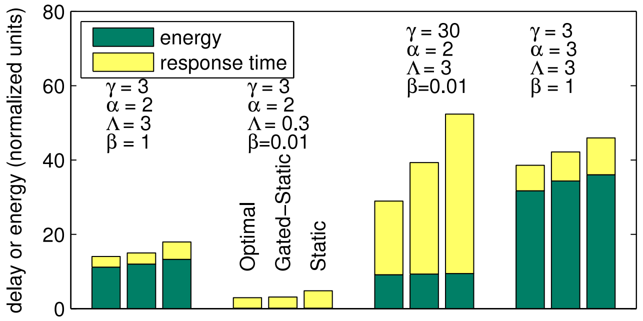


Fig. 4. Breakdown of $\mathbb{E}[T]$ and $\mathbb{E}[s_N^\alpha]$, for several scenarios.

this would correspond to the load being almost 1, which is a regime to be avoided if there is any uncertainty about the load. However, for variable-speed servers, it may be reasonable to have a significant queue but still have significant speed in reserve.

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

4. 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 then raises the question of whether dynamic scaling is worth the complexity. This section illustrates one reason that it is: *robustness*. Specifically, dynamic schemes provide significantly better performance than static speed designs in the face of mis-estimation of workload (Section 4.1) and bursty traffic (Section 4.2).

4.1. Robustness to uncertain mean load

We focus first on robustness with respect to the load, Λ, within the setting of Poisson arrivals. The optimal speeds are sensitive to Λ, but in reality this parameter must be estimated, and will be time-varying. We will investigate the performance of three speed scaling schemes when Λ is inaccurate.

Gated static speed. It is easy to see the problems mis-estimation of Λ 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 $\Lambda \in [\underline{\Lambda}, \bar{\Lambda}]$. Let $z(\Lambda_1|\Lambda_2)$ denote the expected cost per unit time if the arrival rate is Λ_1, but the speed was optimized for Λ_2. Then, the robust design problem is to select the speed Λ' to solve

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

The optimal design is to provision for the highest foreseen load, i.e., $\max_{\Lambda\in[\underline{\Lambda},\bar{\Lambda}]} z(\Lambda|\Lambda') = z(\bar{\Lambda}|\Lambda')$. However, this is wasteful in the typical case that the load is less than $\bar{\Lambda}$; the energy cost will be much higher than necessary. Since we cannot design to maximize robustness, let us consider the robustness of the static design of (6).

The performance of (6) is shown as the dashed line in Fig. 5. (The other curves will be explained later.) This graph shows the average cost under a load of $\Lambda = 10$ of several different speed scaling schemes. Each curve in this graph represents many different designs, each optimized for a different "design Λ". The gated static design has a very high cost if the actual load is slightly above the design load; in fact, for sufficiently small design Λ, the chosen speed s_{gs} is less than the actual load, which results in an unstable queue with infinite delay cost. If the speed scalar is designed for a load that is higher than the actual load, then the selected speed is higher than necessary, resulting in an excessive energy cost. In either case, the cost is very sensitive to the nominal load.

Optimal dynamic speeds. The performance of optimal dynamic speed scaling satisfying (13) is considerably more robust. The speed s_n in each state n still depends on the design Λ, and if the design Λ is too small, then each speed s_n will be lower than the optimal value. However, this does not cause an unstable queue. Note that, for any design Λ, s_n is unbounded for large n; as the queue occupancy rises, the speed rises to match the actual Λ. This improves the performance when the design Λ is far from the actual Λ, as seen in the solid line in Fig. 5.

Linear scaling. Although dynamic scaling is more robust than static scaling, its performance still depends on the estimate of Λ at design time. The ultimate in robustness would be to have performance independent of this estimate. This can be achieved,

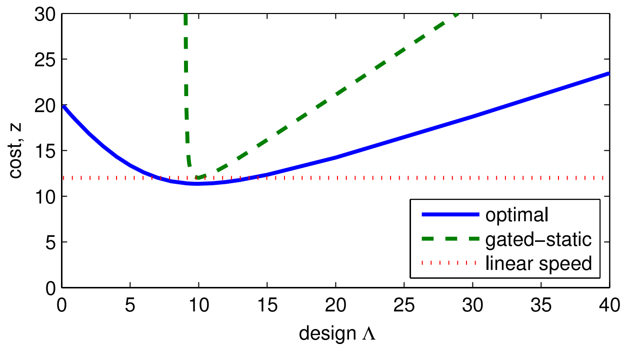


Fig. 5. Cost at load $\Lambda = 10$, when speeds are designed for "design Λ", using $\beta = 1$, $\alpha = 2$. Note that the curve "optimal" uses the optimal speeds for the specified design Λ, which are not necessarily optimal for the actual load of $\Lambda = 10$.

with minimal degradation to performance in the lucky case that the load actually is known in advance, by the following scheme which we term "linear". This scheme scales the server speed 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. The dotted line in Fig. 5 shows that linear scaling provides significantly better robustness than the optimal dynamic scheme; indeed, in the case that the true $\Lambda = 10$, if the speeds are optimized for Λ outside the range [7, 14] then the design is worse than that of linear scaling, and even within that range the optimized design is only slightly better. 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 show that if Λ is known, the cost of the linear scheme, denoted z_{lin}, 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 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 Λ. In contrast, the cost of gated static grows linearly in this difference, as seen in Fig. 5.

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

Proof. If the speed in state n is kn then

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

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

$$\begin{aligned} z_{\text{lin}} &= \mathbb{E}[N] + \frac{\mathbb{E}[s_N^2]}{\beta} = \frac{\Lambda}{\sqrt{\beta}} + \left(\frac{\Lambda}{\sqrt{\beta}} + \frac{\Lambda^2}{\beta}\right) \\ &= \gamma^2 + 2\gamma, \end{aligned}$$

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 Λ' that is operating at load $\Lambda = \Lambda' - \epsilon$. When $\alpha = 2$ and $\epsilon > -\sqrt{\beta}$,*

$$z_{\text{lin}} = \frac{\Lambda^2}{\beta} + 2\frac{\Lambda}{\sqrt{\beta}} \tag{44}$$

$$z_{gs} = z_{\text{lin}} + \frac{\Lambda}{\beta}\left(\frac{\epsilon^2}{\sqrt{\beta} + \epsilon}\right). \tag{45}$$

Proof. The optimal rates for the linear policy are $s_n = n\sqrt{\beta}$, independent of Λ'. Thus its cost is always (44).

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

$$\mathbb{E}[N] = \frac{\Lambda}{\sqrt{\beta} + \Lambda' - \Lambda} \quad \frac{\mathbb{E}[s_N^2]}{\beta} = \frac{\Lambda\Lambda'}{\beta} + \frac{\Lambda}{\sqrt{\beta}}$$

and

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

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

$$\begin{aligned} z_{gs} - z_{\text{lin}} &= \frac{\epsilon \Lambda}{\beta} + \frac{\Lambda}{\sqrt{\beta}+\epsilon} - \frac{\Lambda}{\sqrt{\beta}} \\ &= \frac{\epsilon \Lambda}{\beta} - \frac{\epsilon \Lambda}{\sqrt{\beta}(\sqrt{\beta}+\epsilon)} \end{aligned}$$

from which (45) follows. □

This simplicity of this result is specific to $\alpha = 2$. It is not clear whether similarly efficient load-independent scaling schemes exist for $\alpha \neq 2$. However, numerical results suggest that scaling proportional to $n^{2/\alpha}$ performs well in general, with the optimal constant of proportionality becoming large as $\alpha \downarrow 1$, and approaching $3/4$ as $\alpha \to \infty$.

4.2. Worst-case analysis: competitive ratio

Although the linear scheme works well for arbitrary Poisson loads, workloads can be significantly more or less bursty than Poisson. We now consider a more general workload model: finite, arbitrary (maybe adversarial) instances of arriving jobs. That is, in this section the workload is not stochastic. A problem instance consists of J jobs, with the jth job having arrival time (release time) $r(j)$ and size (work) x_j. Our objective is again a linear combination of response time and energy usage. Let $E(I)$ be the total energy used to complete instance I, and T_j be the response time of job j, the completion time minus the release time. The analog of (3) is to replace the ensemble average by the sample average and consider the cost of the entire finite instance rather than the cost per unit time, giving the cost of an instance I under a given algorithm $\mathcal{A}$ as

$$z^{\mathcal{A}}(I) = \sum_{j=1}^{J} T_j + \frac{1}{\beta'} E(I) = \int \left(n(t) + \frac{s(t)^\alpha}{\beta} \right) dt. \tag{46}$$

In this model, we compare the cost of speed scaling algorithms to the cost of the optimal offline algorithm, OPT. In particular, we study the competitive ratio, defined as

$$CR = \sup_I z^{\mathcal{A}}(I)/z^O(I), \tag{47}$$

where $z^O(I)$ is the optimal cost achievable on I. If a scheme has a competitive ratio at most c then it is called c-competitive.

This concept of robustness is very different from the robustness considered in the previous subsection. Although "linear" scaling is robust against uncertainty in the rate of a Poisson workload, it has a poor competitive ratio. In particular, it incurs a high cost when processing infrequent large batches. Consider batches of N jobs each of size 1, occurring with period $\sqrt{n}$, with $\beta = 1$ and $\alpha = 2$. The linear scheme $s_n = n$ will finish all jobs after time 1 and incur a cost of $z^{\text{lin}}(\text{batch}) = N + N^2 = \Theta(N^2)$ to process each batch. The optimal scheme can be no worse than the scheme $s_n = \sqrt{n}$, which finishes all jobs after $\sqrt{N}$, which is still before the next batch arrives, and incurs a cost of $(N+N)\sqrt{N} = \Theta(N^{3/2})$ to process each batch. Since the set of instances in (47) includes unboundedly large N, the CR of the linear scheme is infinite. In contrast, we will soon see that there are schemes with finite competitive ratios.

4.2.1. Background and notation

The analysis in this section, foreshadowed in [23], is the first worst-case analysis of speed scaling under processor sharing with unbounded speeds, for any objective. Like the analysis for speed-bounded processors in [39], it uses tools developed in Bansal et al. [40], which in turn builds on the earlier work [41], for the analysis of a different scheduling algorithm. The aim of [40,41] was to find a non-clairvoyant schedule (i.e., one which does not know the size of a job until it finishes) with good scaling in the limit of large α. They showed that a competitive ratio of $O(\alpha^3)$ is achieved by a limited form of PS, called Latest Arrivals Processor Sharing (LAPS), which shares the processor among a fixed fraction of the active flows, dependent on α. Their analysis can also be used to derive a scaling for pure PS which has worse asymptotic performance for large α (namely $O(2^\alpha)$), but has better performance for typical values of α in the range 2–3.

Consider a PS scheduler running at speed $s^A(t) = k(n^A(t))^{1/\alpha}$ at time t when there are $n^A(t)$ jobs in the system, for some $k > 0$. Let the amount of unfinished work in job j be $q^A(j,t)$. This will be compared with an optimal adversary, denoted OPT, with speed, occupancy and unfinished work $s^O(t)$, $n^O(t)$ and $q^O(j,t)$. The argument t is omitted when there is no risk of confusion.

4.2.2. Main lemma

The results of this section are proven using the following important lemma.

Lemma 13. *Let $\mathsf{H} > 0$, $\eta \geq 1$, and A be the discipline (PS, s_n) with $s_n \in [(n\beta/\mathsf{H})^{1/\alpha}, (n\beta\eta)^{1/\alpha}]$. If $P(s) = s^\alpha$ then A is c-competitive, where $c = (1+\eta)\max((2\alpha-1), \mathsf{H}(2-1/\alpha)^\alpha)$.*

Proof. The proof uses a technique termed *amortized local competitive analysis* [42,43], which works as follows.

To show that an algorithm $\mathcal{A}$ is c-competitive with an optimal algorithm *OPT* for a performance metric $z = \int z(t)dt$ it is sufficient to find a *potential function* $\Phi : \mathbb{R} \to \mathbb{R}$ such that, for any instance of the problem:

1. *Boundary condition.* $\Phi = 0$ before the first job is released, and $\Phi \geq 0$ after the last job is finished;
2. *Jump condition.* At any point where Φ is not differentiable, it does not increase;
3. *Running condition.* When Φ is differentiable,

$$z^{\mathcal{A}}(t) + \frac{d\Phi}{dt} \leq cz^{O}(t), \tag{48}$$

where $z^{\mathcal{A}}(t)$ and $z^{O}(t)$ are the cost $z(t)$ under $\mathcal{A}$ and *OPT* respectively.

Given these conditions, the competitiveness follows from integrating (48), which gives

$$z^{\mathcal{A}} \leq z^{\mathcal{A}} + \Phi(\infty) - \Phi(-\infty) \leq cz^{O}.$$

To prove the result, let $\mathsf{H} > 0$, $\eta \geq 1$, and $\Gamma = (1+\eta)(2\alpha - 1)(\mathsf{H}/\beta)^{1/\alpha}$ and define the potential function

$$\Phi = \Gamma \sum_{i=1}^{n^A(t)} i^{1-1/\alpha} \max(0, q^A(j_i; t) - q^O(j_i; t)) \tag{49}$$

where $q^{\pi}(j; t)$ is the remaining work on job j at time t under scheme π, and $\{j_i\}_{i=1}^{n^A(t)}$ is an ordering of the jobs in increasing order of release time: $r(j_1) \leq r(j_2) \leq \cdots \leq r(j_{n^A(t)})$. Note that this is a scaling of the potential function that was used in [41] to analyze LAPS. Hence, the proof that the boundary and jump conditions are satisfied is the same as that in [41]. All that remains is the running condition, which follows from the technical Lemma 14. □

The following lemma used in the proof of Lemma 13 is provided in Appendix C.

Lemma 14. *Let Φ be given by (49) and A be the discipline (PS, s_n) with $s_n \in [(n\beta/\mathsf{H})^{1/\alpha}, (n\beta\eta)^{1/\alpha}]$. Then under A, at points where Φ is differentiable,*

$$n^A + (s^A)^{\alpha}/\beta + \frac{d\Phi}{dt} \leq c(n^O + (s^O)^{\alpha}/\beta) \tag{50}$$

where $c = (1+\eta)\max((2\alpha - 1), \mathsf{H}(2 - 1/\alpha)^{\alpha})$.

4.2.3. Worst-case bounds on stochastic optimal design

Section 4.1 described a means to improve robustness to uncertainty in the mean load, provided that arrivals are Poisson. However, we now show that the stochastically optimal speeds given in (13) actually yield a system which is very robust to non-Poisson arrivals.

Let s_n^* denote the stochastically optimal speed, given by (13). The following is a consequence of Lemma 13 and Theorem 8.

Theorem 15. *Consider $P(s) = s^{\alpha}$ with $\alpha > 1$ and algorithm A which uses PS scheduling and chooses speeds s_n^* optimal for an* M/GI/1 *queue with load Λ. Then A is $O(1)$-competitive in the worst-case model.*

Proof. The proof applies Lemma 13 from the worst-case model to the speeds from the stochastic model.

By (35) of Theorem 8, $s_n \geq (n\beta/\mathsf{H})^{1/\alpha}$ with $\mathsf{H} = \alpha - 1$. Further, (26) implies that $s_n^* = O(n^{1/\alpha})$ for any fixed Λ and β, and s_n^* is bounded for finite n.

Hence the speeds s_n^* are of the form given in Lemma 13 for some finite $\eta \geq 1$ and $\mathsf{H} > 0$ (which may depend on the constant Λ), from which it follows that A is constant competitive, for fixed α, β and (design parameter) Λ. □

Note that the above corollary is distinctive in that it provides worst-case guarantees for a stochastic control policy.

Some insight into the performance of (PS,s_n^*) on non-Poisson arrivals can be obtained by considering its response to bursty loads. Consider a load consisting of batches of simultaneous arrivals of unit size, spaced far enough apart that the optimal speed scaler finishes one batch before the next arrives. Fig. 6 compares the costs for (i) the dynamic programming solution (PS,s_n^*) designed for a Poisson load with $\gamma = 5$, (ii) the optimal speed scaler for this bursty workload constrained to use PS, and (iii) the unconstrained optimal speed scaler, which uses SRPT. For small batches, the constraint of using PS is small and so the penalty for using suboptimal speeds is a significant fraction of the suboptimality. However, for very bursty workloads, the penalty for using the speeds designed for Poisson traffic is small compared with the penalty for being constrained to use PS.

Although the stochastic optimal speeds s^* for PS are $O(1)$-competitive, the cost may be many times higher than optimal. Other speed scalings can give tighter bounds, at the expense of suboptimality in the case of Poisson arrivals with known rate.

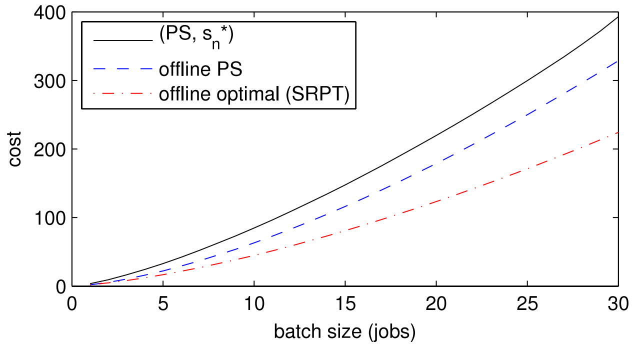


Fig. 6. Costs of the scheme (PS, s_n^*) designed for $\gamma = 5$, PS with the best offline speeds, and the optimal offline speed scaler (SRPT with the best offline speeds). In each case, $\alpha = 2$.

When shortest remaining processing time (SRPT) scheduling is used, good performance is often obtained by the scaling $s_n = P^{-1}(\beta n)$, where $P^{-1}(\cdot)$ denotes the inverse of P. This sets the cost of energy use $P(s_n)/\beta$ to balance the holding cost n exactly. It was shown in [14] that the optimal speed at which to run a job is proportional to the number of jobs delayed by that job, which prompted the use of $s_n = P^{-1}(\beta n)$ in [19]. This scaling was shown in [23] to give the best possible competitive ratio for the objective (46). This can be viewed as a theoretical justification for the useful heuristic of making devices "power proportional" [44]. When not all jobs have equal weight and $P(s) = s^\alpha$, $O(1)$-competitiveness is achieved by the related scaling $s = P^{-1}(w)$, where w is the sum of the job weights times the fraction of unfinished work on each [21].

In the present context of PS, $s_n = P^{-1}(\beta n)$ also performs well in the worst case. It sets $\eta = \mathsf{H} = 1$ in Lemma 13, which gives

Theorem 16. *If $P(s) = s^\alpha$ then (PS, $P^{-1}(\beta n)$) is c-competitive, where $c = 2\max(2\alpha - 1, (2 - 1/\alpha)^\alpha)$.*

In particular, (PS, $P^{-1}(\beta n)$) is $(4\alpha - 2)$-competitive for α in the typical range of $(1, 3]$. Note that Theorem 16 is tighter than the $O(\alpha 3^\alpha)$ result of [39], although the latter result is for the more challenging case of serving weighted jobs on bounded speed servers.

It was shown in [41] that it is impossible for any speed scaler using a non-clairvoyant scheduler, such as PS, to have a competitive ratio which remains bounded for large α. In particular, the competitive ratio is $\Omega(\alpha^{1/3} - \epsilon)$ for all $\epsilon > 0$. The LAPS scheduler [41] achieves good scaling in α, being $O([\alpha/\log(\alpha)]^2)$-competitive [40]. To achieve this scaling, LAPS adapts its scheduling discipline to α. In particular, instead of sharing the processor among all active jobs as PS does, it shares the processor equally among a fixed fraction of jobs. This fraction depends only on α, subject to rounding to an integer. As a result, not only the speeds but also the choice of which jobs to serve depends on the system's estimate of α.

The $O(2^\alpha)$ result in Theorem 16 is very much weaker than these for large α. However, in practical cases where power is well approximated by s^α, the value of α is typically small. For CMOS processors, Dennard's law [45] limits α to at most about 3 and, as indicated in Section 2, it is typically much less than this in real processors. For that reason, we can often consider α to be fixed.

For fixed α, Theorem 16 shows that (PS, $P^{-1}(\beta n)$) is $O(1)$-competitive in the number of jobs, as LAPS is. Moreover, for practical values such as $\alpha = 3$, Theorem 16 shows that PS with $\beta = 1$ has a competitive ratio of slightly below 10, compared with the best known bound of 231 for LAPS [40]. For $\alpha = 2$, PS has a competitive ratio of at most $2\max(3, 9/4) = 6$.

Note also that the PS scheduler itself does not depend on either the power function P or the choice of speeds. This is in contrast to schedulers such as LAPS, in which the selection of which job to run depends on α. This decoupling shows again how robustness can be increased with a slight reduction in performance in some cases.

The asymptotic form for large α may provide insight into likely performance in cases where P is not polynomial but grows very rapidly, such as the exponential power required for transmission over a noise-limited communication channel [46], or the $1/(1 - s)$ power required for transmission over an interference-limited channel. In this case, LAPS [41] outperforms PS, but a designer may be required to choose PS for reasons such as its fairness [23], and so its optimal performance is of interest.

Although (PS, $P^{-1}(\beta n)$) performs well, the bound can be tightened further. The value of $\eta = 1/\mathsf{H}$ which minimizes the bound in Lemma 13 is $\eta = (2 - 1/\alpha)^{\alpha-1}/\alpha$. This gives

Theorem 17. *Let $\eta = (2 - 1/\alpha)^{\alpha-1}/\alpha$. If $P(s) = s^\alpha$ then (PS, $P^{-1}(\beta n\eta)$) is c-competitive, where $c = 2\alpha - 1 + (2 - 1/\alpha)^\alpha$.*

For large α, the speed $s_n = P^{-1}(n\beta\eta) = P^{-1}(n\beta)(2-1/\alpha)^{1-1/\alpha}\alpha^{-1/\alpha} \sim 2P^{-1}(n\beta)$. This can be compared with the resource augmentation result that PS is $(2+\epsilon)$-speed $O(1+1/\epsilon)$-competitive for fixed-speed servers [42]. The similarity is that, with or without speed scaling, it seems appropriate to run PS around twice as fast as SRPT. A possible explanation for the similarity in the form of the results is as follows. As α becomes large, the power penalty approaches a barrier function, with no penalty for speeds less than $s = 1$ and infinite penalty for larger speed, which corresponds to a fixed-speed server.

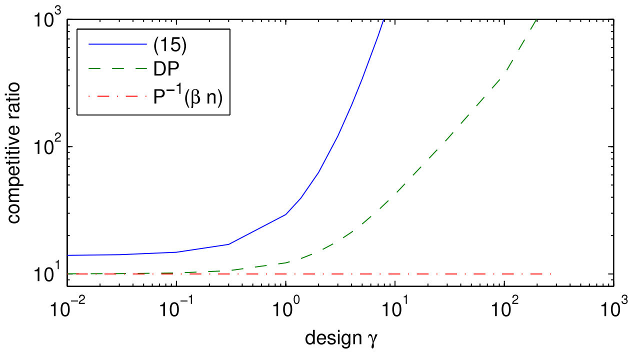


Fig. 7. Competitive ratios vs. normalized load γ for $\alpha = 3$: bounds for (PS,s_n^*) obtained by using (15) in Lemma 13; bounds for (PS,s_n^*) with η and H calculated from s_n^* obtained by solving the dynamic program (DP) numerically; bounds for (PS,$P^{-1}(\beta n)$).

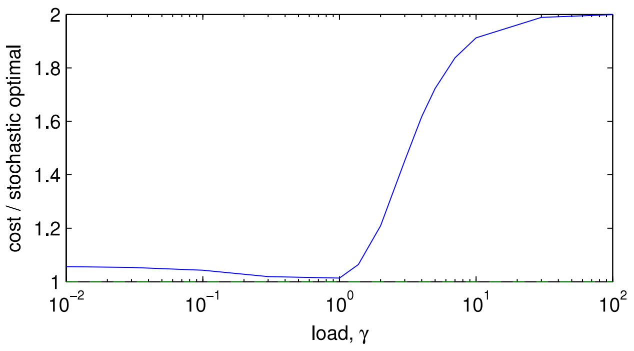


Fig. 8. Stochastic performance vs. normalized load γ for $\alpha = 3$: numerical results for (PS, $P^{-1}(\beta n)$), obtained by solving the Markov chain numerically, normalized by the optimal cost z obtained by solving the dynamic program (DP) numerically.

Small $\alpha \in (1, 3)$ gives the paradoxical result that $\eta < 1$ in Theorem 17. In particular, for $\alpha = 2$ (which gives $\eta = 3/4$) this implies that the tightest bound is for speeds *slower* than the batch speeds given by (35). This is clearly an artifact of the proof technique, which gives hope of finding a tighter competitive ratio for these cases.

4.3. Numerical comparisons

To quantify the trade-off between robustness and performance, let us consider the competitive ratio penalty incurred by using the stochastic-optimal speeds s_n^* and the penalty in expected cost under Poisson arrivals for using $P^{-1}(\beta n)$. The foregoing analytical results focus on $\alpha = 2$, but since the asymptotic form of s_n^* for $\alpha = 2$ is very similar to $P^{-1}(\beta n)$, the penalties may be unrepresentatively low in that case. Hence we consider $\alpha = 3$, and perform the comparisons numerically.

Fig. 7 compares the bounds obtained from Lemma 13 for the competitive ratio for (PS, $P^{-1}(\beta n)$), for the stochastic optimal scheme (PS, s_n^*) and for analytical bounds on s_n derived in Section 3.4. When the speeds are optimized for small loads, the bounds are very similar. As γ increases, the bound from Lemma 13 becomes very large. Since the bounds become increasingly loose as γ increases, it is not possible from this to see how quickly the true competitive ratio of the optimal scheme degrades. The bounds based on Section 3.4 are significantly much looser, even for small γ when the bounds on the actual s_n^* are comparatively tight.

Fig. 8 considers the complementary performance measure: expected performance under Poisson arrivals. The penalty for using the robust speeds $P^{-1}(\beta n)$ approaches 2 as the load γ increases. This can be understood as follows. Both schemes must run on average fast enough to maintain the speed $\mathbb{E}[s_N]/\beta^{1/\alpha} > \gamma$. The stochastically optimal scheme can select the per-state speeds to achieve this with negligible total queueing, while the robust scheme's queueing cost is always equal to its energy cost. For low loads, the robust scheme again incurs a small penalty, since it processes a single job at speed $(\beta n)^{1/\alpha}$, whereas the optimal scheme runs only slightly faster than the speed $(\beta n/(\alpha - 1))^{1/\alpha}$ which is optimal when the current job finishes before the next arrives. Note that the performance ratio for $\gamma = 10$, which we argued is a realistic value, is close to the worst case value of 2.

5. Concluding remarks

Speed scaling is an important method for reducing energy consumption in computer systems. Intrinsically, it trades off the mean response time and the mean energy consumption, and this paper provides insight into this tradeoff by comparing stochastic and worst-case analyses for processor sharing systems.

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 (except for the lower bound on s_n^* with general α) 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 for $\alpha = 2$ the optimal dynamic speeds in the stochastic model match (for large n) dynamic speed scalings that 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 mean performance for a Poisson workload 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. With a Poisson workload, a scheme that scales speeds linearly with n provides significantly improved robustness while increasing cost only slightly, while for arbitrary workloads, a tighter competitive ratio can be achieved by a scheme that sets speeds independently of the expected load.

There are a number of related directions in which to extend this work. For example, numerical results suggest that, for $\alpha \le 2$, significantly tighter lower bounds on the stochastic speeds can be obtained by linearizing the speeds around the low load case, $\gamma \to 0$. More importantly, we have only considered dynamic power consumption, which can be modeled as a polynomial of the speed. However, the contribution of leakage power in CMOS chips is growing and an important extension is to develop models of total power use that can be used for analysis. It will also be interesting to extend the analysis to more detailed models of schedulers used in current operating systems, such as multi-level feedback schedulers [47].

Acknowledgments

This work was supported by grants from the NSF CCF 0830511, CCS 0835706 and CNS 0435520, Microsoft Research, IBM Faculty Award, the Lee Center for Advanced Networking and the Australian Research Council grant FT0991594.

Appendix A. Bounds on $G(\gamma;\alpha)$

Proof of Lemma 1. Let k_1 satisfy

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

Substituting the identity $(a+b)^\alpha = a^\alpha(1+b/[(a+b)-b])^\alpha$ and (A.1) into (7) gives

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

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

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

with the inequality reversed for $\alpha \le 2$. For small γ, this inequality tends to equality. Hence $k_1 \ge 2/\alpha$ for $\alpha \ge 2$, and $k_1 \le 2/\alpha$ for $\alpha \le 2$ and the second inequality in (8) is accurate to leading order in γ.

Similarly, substituting $G(\gamma;\alpha) = \gamma + k_2$. into (7) gives

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

This is solved for

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

For $\alpha \ge 2, 0 \le \epsilon_2 \to 0$ as $k_2/\Lambda \to 0$, which shows that the first inequality of (8) is an upper bound. For $\alpha \le 2, 0 \ge \epsilon_2 \to 0$ as $k_2/\Lambda \to 0$, which shows that the first inequality of (8) is a lower bound. The requirement $k_2 \ll \gamma$ is then $\gamma \gg \sqrt{\gamma^{2-\alpha}/(\alpha-1)}$ or equivalently $\gamma^\alpha \gg 1/(\alpha-1)$. $\square$

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 $\delta = \hat{z} - z$. The following proposition quantifies the growth in the error in $\hat{u}_n$, or equivalently the speed. If the error is positive, it grows exponentially. If it is negative then the absolute error is again unbounded and the relative error becomes arbitrarily close to 100%; specifically, the estimates either are bounded above (while the actual speed is unbounded) or oscillate with unbounded magnitude.

Proposition 18. *Let $\hat{u}_n$ be the value obtained by applying the iteration (12a), and (12b), with $z = \hat{z} \neq z^*$. There exists a $\kappa > 0$ such that, if $\hat{z} \geq z^*$ then for all $n \geq 1$, $\Delta_n - \delta \geq 2^n(\hat{z} - z^*)\kappa$. Moreover, there exists a $\underline{u} > 0$ such that if $\hat{z} < z^*$ then for all $N > 0$ there exists an $n > N$ such that $\hat{u}_n \notin [\underline{u}, \infty)$.*

Proof. By (12b), $\Delta_{n+1} = \phi(\hat{u}_n) - \phi(u_n) + \delta$ whence, if $\hat{u}_n > 0$,

$$\Delta_{n+1} = \nu\Delta_n + \delta \tag{B.1}$$

where $\nu = \phi'(x)$ for some x in the closed interval between $\hat{u}_n$ and u_n. If $\hat{u}_n > 0$ then

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

since ϕ is convex increasing and real for positive arguments. Moreover, since ϕ is convex and has unbounded derivative, there is a $\underline{u}$ such that $\phi'(u) \geq 2$ for all $u > \underline{u}$.

If $\hat{z} = z^*$ then $\Delta_n = \delta = 0$ for all n, and the proposition is trivially true.

If $\hat{z} > z^*$, then $\Delta_n > 0$ for all $n \geq 1$, whence $\nu \geq \phi'(u_n)$. Let $\underline{n}$ be the smallest n such that $u_n \geq \underline{u}$, which exists since u_n is unbounded by (36). By induction, the claim is true with $\kappa = \min_{n\in[1,\underline{n}]} 2^{-n}(\Delta_n - \delta)/\delta$, which is positive since $\Delta_n > \delta$ for all $n \geq 1$ by (B.1).

To prove the claim in the case that $\hat{z} < z^*$, assume instead that there exists an N such that $\hat{u}_n \in [\underline{u}, \infty)$ for all $n > N$. Similarly to the case $\hat{z} > z^*$, that implies that Δ_{n+1} grows exponentially in magnitude and remains negative. Since $\phi(u_n)$ is concave in n by Lemma 10, this implies $\phi(\hat{u}_n)$ must be unbounded below, which is a contradiction. □

Remark 3. The growth in the error is actually super-exponential; the same proof applies if "2" in the statement of the theorem and proof is replaced by any value greater than 1.

Remark 4. The proof does not require the specific form of ϕ used in this paper; any ϕ with unbounded derivative is sufficient. Recall that the error grows exponentially for $n > \underline{n}$; for the ϕ of this paper, (36) implies that

$$\underline{n} \leq \left\lceil (\alpha - 1)\left(\frac{\alpha - 1}{\alpha\gamma^\alpha}\right)^{\alpha/(\alpha-1)} \right\rceil$$

which is close to 1 for the typical case that α is near 2 and γ is near 1.

Conversely, if calculations are performed backwards, with $\hat{u}_n$ calculated from $\hat{u}_{n+1}$ starting from $\hat{u}_N > u_N$ for some large N, then the error will decrease towards the interval $\pm 2|\delta|$ as long as $\min(u_n, \hat{u}_n) > \underline{u}$. When z^* is not known exactly, the errors are minimized by using a hybrid approach of starting calculation from both $n = 1$ and from a large N, and working towards the n for which $\phi'(\hat{u}_n) \approx 1$.

Appendix C. Proof of running condition, Lemma 14

Proof of Lemma 14. First note that if $n^A = 0$ then the left hand side of (50) is 0, and the inequality holds. Henceforth, consider the case $n^A \geq 1$.

The rate of change of Φ caused by running OPT is at most $\Gamma(n^A)^{1-1/\alpha}s^O$, which occurs when all of the speed is allocated to the job with the largest weight in (49).

Let $l \geq 0$ be the number of zero terms in the sum (49), corresponding to jobs on which PS is leading OPT. The sum in (49) contains $n^A - l$ non-zero terms, each decreasing due to PS at some rate $i^{1-1/\alpha}dq^A/dt = i^{1-1/\alpha}s^A/n^A$. The sum is minimized (in magnitude) if these are terms $i = 1, \ldots, n^A - l$. Thus, the change in Φ due to PS is at least as negative as

$$\begin{aligned} -\Gamma\sum_{i=1}^{n^A-l} i^{1-1/\alpha}\frac{s^A}{n^A} &\leq -\Gamma\int_0^{n^A-l} i^{1-1/\alpha}\frac{s^A}{n^A}\,di \\ &\leq -\Gamma\frac{\alpha}{2\alpha-1}(n^A - l)^{2-1/\alpha}(\beta/\mathsf{H})^{1/\alpha}(n^A)^{(1/\alpha)-1} \end{aligned} \tag{C.1}$$

since $s^A \geq (n^A\beta/\mathsf{H})^{1/\alpha}$. This gives

$$\frac{d\Phi}{dt} \leq \Gamma(n^A)^{1-1/\alpha}s^O - \Gamma\frac{\alpha(\beta/\mathsf{H})^{1/\alpha}}{2\alpha-1}(n^A)^{(1/\alpha)-1}(n^A - l)^{2-1/\alpha}.$$

Moreover, since $(s^A)^\alpha/\beta \leq \eta n^A$ and $l \leq n^O$, we have $n^A + (s^A)^\alpha/\beta \leq (1+\eta)n^A$ and $n^O + (s^O)^\alpha/\beta \geq l + (s^O)^\alpha/\beta$. To show (50), it is sufficient to show that

$$(1+\eta)n^A + \Gamma(n^A)^{1-1/\alpha}s^O - \Gamma\frac{\alpha(\beta/\mathsf{H})^{1/\alpha}(n^A)^{(1/\alpha)-1}(n^A - l)^{2-1/\alpha}}{2\alpha-1} \leq c(l + (s^O)^\alpha/\beta).$$

Since $n^A > 0$, dividing by n^A gives the sufficient condition

$$0 \le c(s^O)^\alpha/(\beta n^A) - \Gamma s^O/(n^A)^{1/\alpha} + cl/n^A + (1+\eta)\alpha(1-l/n^A)^{2-1/\alpha} - (1+\eta) \tag{C.2}$$

since $\Gamma = (1+\eta)(2\alpha-1)(H/\beta)^{1/\alpha}$. To find a sufficient condition on c, we take the minimum of the right hand side with respect to s^O, l and n^A. Following [40], note that the minimum of the first two terms with respect to s^O occurs for $s^O = (\frac{\beta\Gamma}{c\alpha})^{1/(\alpha-1)}(n^A)^{1/\alpha}$, at which point the first two terms become

$$-\left(\frac{\alpha-1}{\alpha}\right)\left(\frac{\beta\Gamma^\alpha}{c\alpha}\right)^{1/(\alpha-1)}. \tag{C.3}$$

Now consider a lower bound on the sum of the terms in l. Setting the derivative with respect to l to 0 gives $c = (1+\eta)(2\alpha-1)(1-l/n^A)^{1-1/\alpha}$. Hence the minimum for $l \ge 0$ is for $l/n^A = 1-\min(1, ((1+\eta)(2\alpha-1)/c)^{\alpha/(1-\alpha)})$. For $c \ge (1+\eta)(2\alpha-1)$, the sum of the terms in l achieves a minimum (with respect to l) of $(1+\eta)\alpha$ at $l = 0$, for all n^A. In this case, it is sufficient that

$$0 \le -\left(\frac{\alpha-1}{\alpha}\right)\left(\frac{\beta\Gamma^\alpha}{c\alpha}\right)^{1/(\alpha-1)} + (1+\eta)\alpha - (1+\eta).$$

Rearranging shows that it is sufficient that both

$$c \ge (1+\eta)(2\alpha-1)$$

and

$$c \ge \beta\left(\frac{\Gamma}{\alpha}\right)^\alpha (1+\eta)^{1-\alpha} = \mathsf{H}(1+\eta)\left(\frac{2\alpha-1}{\alpha}\right)^\alpha$$

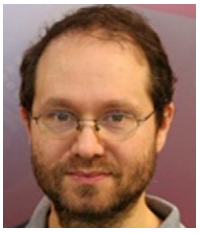
where the equality uses $\Gamma = (1+\eta)(2\alpha-1)(\mathsf{H}/\beta)^{1/\alpha}$. □

References

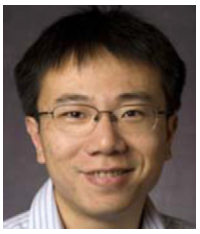
[1] J. Baliga, R. Ayre, W. Sorin, K. Hinton, R. Tucker, Energy consumption in access networks, in: IEEE Conf. Optical Fiber communication, OFC, 2008, pp. 1–3. http://dx.doi.org/10.1109/OFC.2008.4528538.
[2] S. Irani, K.R. Pruhs, Algorithmic problems in power management, SIGACT News (ISSN: 0163-5700) 36 (2) (2005) 63–76. http://doi.acm.org/10.1145/1067309.1067324.
[3] S. Kaxiras, M. Martonosi, Computer Architecture Techniques for Power-Efficiency, Morgan and Claypool, 2008.
[4] O.S. Unsal, I. Koren, System-level power-aware design techniques in real-time systems, Proc. IEEE 91 (7) (2003) 1055–1069.
[5] N. Bansal, T. Kimbrel, K. Pruhs, Speed scaling to manage energy and temperature, J. ACM 54 (1) (2007) 1–39.
[6] S. Herbert, D. Marculescu, Analysis of dynamic voltage/frequency scaling in chip-multiprocessors, in: Proc. ISLPED, vol. 6, 2007.
[7] L. Yuan, G. Qu, Analysis of energy reduction on dynamic voltage scaling-enabled systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 24 (12) (2005) 1827–1837.
[8] Y. Zhu, F. Mueller, Feedback EDF scheduling of real-time tasks exploiting dynamic voltage scaling, Real-Time Syst. 31 (2005) 33–63.
[9] IBM PowerPC. http://www-03.ibm.com/technology/power/powerpc.html.
[10] Intel Xscale. www.intel.com/design/intelxscale.
[11] S.V. Hanly, Congestion measures in DS–CDMA networks, IEEE Trans. Commun. 47 (3) (1999) 426–437.
[12] P. Tsiaflakis, Y. Yi, M. Chiang, M. Moonen, Fair greening of broadband access: spectrum management for energy-efficient DSL networks, EURASIP Journal on Wireless Communications and Networking 2011 (1) (2011) 1–17.
[13] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proc. IEEE Symp. Foundations of Computer Science, FOCS, 1995, pp. 374–382.
[14] K. Pruhs, P. Uthaisombut, G. Woeginger, Getting the best response for your erg, in: Scandinavian Worksh. Alg. Theory, 2004.
[15] K. Pruhs, P. Uthaisombut, G. Woeginger, Getting the best response for your erg, ACM Trans. Algorithms 4 (3) (2008) Article 38.
[16] D.P. Bunde, Power-aware scheduling for makespan and flow, J. Sched. 12 (5) (2009) 489–500.
[17] K. Pruhs, R. van Stee, P. Uthaisombut, Speed scaling of tasks with precedence constraints, Theory Comput. Syst. 43 (1) (2008) 67–80.
[18] S. Zhang, K.S. Catha, Approximation algorithm for the temperature-aware scheduling problem, in: Proc. IEEE Int. Conf. Comp. Aided Design, 2007, pp. 281–288.
[19] S. Albers, H. Fujiwara, Energy-efficient algorithms for flow time minimization, in: Lecture Notes in Computer Science (STACS), vol. 3884, 2006, pp. 621–633.
[20] N. Bansal, H.-L. Chan, K. Pruhs, Speed scaling with an arbitrary power function, in: Proc. ACM–SIAM Symp. Discrete Algorithms, SODA, 2009, pp. 693–701.
[21] N. Bansal, K. Pruhs, C. Stein, Speed scaling for weighted flow times, in: Proc. ACM–SIAM SODA, 2007, pp. 805–813.
[22] J.M. George, J.M. Harrison, Dynamic control of a queue with adjustable service rate, Oper. Res. 49 (5) (2001) 720–731.
[23] L.L.H. Andrew, M. Lin, A. Wierman, Optimality, fairness and robustness in speed scaling designs, in: Proc. ACM SIGMETRICS, 2010.
[24] J.R. Bradley, Optimal control of a dual service rate $M/M/1$ production–inventory model, European J. Oper. Res. 161 (3) (2005) 812–837.
[25] T.B. Crabill, Optimal control of a service facility with variable exponential service times and constant arrival rate, Manag. Sci. 18 (9) (1972) 560–566.
[26] R.R. Weber, S. Stidham, Optimal control of service rates in networks of queues, Adv. in Appl. Probab. 19 (1987) 202–218.
[27] Intel Corp., Intel PXA270 processor: electrical, mechanical, and thermal specification, 2005.
[28] M. Telgarsky, J.C. Hoe, J.M.F. Moura, SPIRAL: joint runtime and energy optimization of linear transforms, in: Proc. ICASSP, III-1048-III-1051, 2006.
[29] S. Narendra, et al., Ultra-low voltage circuits and processor in 180–90 nm technologies with a swapped-body biasing technique, in: Proc. IEEE Int. Solid-State Circuits Conf., 8.4, 2004.
[30] A.P. Chandrakasan, D.C. Daly, D.F. Finchelstein, J. Kwong, Y.K. Ramadass, M.E. Sinangil, V. Sze, N. Verma, Technologies for ultradynamic voltage scaling, Proc. IEEE 98 (2) (2010) 191–214.
[31] M. Lin, A. Wierman, L.L.H. Andrew, E. Thereska, Dynamic right-sizing for power-proportional data centers, in: Proc. INFOCOM, 2011.

[32] A. Gupta, R. Krishnaswamy, K. Pruhs, Nonclairvoyantly scheduling power-heterogeneous processors, Sustainable Computing: Informatics and Systems 1 (3) (2011) 248–255.
[33] T. Dinh, L.L.H. Andrew, Y. Nazarathy, Robust control of an $M/G/1$ processor sharing queue with applications to energy management, Tech. Rep. CAIA-TR-120226A, Swinburne University of Technology, 2012.
[34] A. Wierman, L.L.H. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems, in: Proc. IEEE INFOCOM, 2009, pp. 2007–2015.
[35] F.P. Kelly, Reversibility and Stochastic Networks, Wiley, 1979.
[36] B. Ata, S. Shneorson, Dynamic control of an $M/M/1$ service system with adjustable arrival and service rates, Manag. Sci. 51 (11) (2006) 1778–1791.
[37] D. Low, Optimal pricing policies for an $M/M/s$ queue, Oper. Res. 22 (1974) 545–561.
[38] J. Wijngaard, S. Stidham, Forward recursion of Markov decision processes with skip-free-to-the-right transitions, part I: theory and algorithm, Math. Oper. Res. 11 (2) (1986) 295–308.
[39] S. Chan, T. Lam, L. Lee, H. Ting, P. Zhang, Non-clairvoyant scheduling for weighted flow time and energy on speed bounded processors, in: Proc. Computing: The Australasian Theory Symposium, CATS, Australian Computer Society, Inc., 2010, pp. 3–10.
[40] N. Bansal, H.-L. Chan, J. Edmonds, K. Pruhs, Preprint, 2009. Available http://www.cs.pitt.edu/~kirk/postdoc/spaa.pdf.
[41] H.-L. Chan, J. Edmonds, T.-W. Lam, L.-K. Lee, A. Marchetti-Spaccamela, K. Pruhs, Nonclairvoyant speed scaling for flow and energy, in: Proc. STACS, 2009, pp. 255–264.
[42] J. Edmonds, Scheduling in the dark, in: Proc. ACM STOC, 1999, pp. 179–188.
[43] R.E. Tarjan, Amortized computational complexity, SIAM J. Algebr. Discrete Methods 6 (2) (1985) 306–318.
[44] L.A. Barroso, U. Hölzle, The case for energy-proportional computing, Computer 40 (12) (2007) 33–37.
[45] R. Dennard, F. Gaensslen, V. Rideout, E. Bassous, A. LeBlanc, Design of ion-implanted MOSFET's with very small physical dimensions, IEEE J. Solid-State Circuits 9 (5) (1974) 256–268.
[46] D.J.C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.
[47] F.J. Corbató, M.M. Daggett, R.C. Daley, An experimental time-sharing system, in: AFIPS Joint Computer Conference, 1962, pp. 335–344.

Adam Wierman is a Professor in the Department of Computing and Mathematical Sciences at the California Institute of Technology, where he is a member of the Rigorous Systems Research Group (RSRG). He received his Ph.D., M.Sc. and B.Sc. in Computer Science from Carnegie Mellon University in 2007, 2004, and 2001, respectively. His research interests center around resource allocation and scheduling decisions in computer systems and services. He received the ACM SIGMETRICS Rising Star award in 2011, and has also been co-recipient of best paper awards at ACM SIGMETRICS, IEEE INFOCOM, IFIP Performance, IEEE Green Computing Conference, and ACM GREENMETRICS. He was named a Seibel Scholar, received an Okawa Foundation grant, and received an NSF CAREER grant. He has also received multiple teaching awards, including the Associated Students of the California Institute of Technology (ASCIT) Teaching Award.



Lachlan L.H. Andrew received the B.Sc., B.E. and Ph.D. degrees in 1992, 1993, and 1997, from the University of Melbourne, Australia. Since 2008, he has been an associate professor at Swinburne University of Technology, Australia, and since 2010 he has been an ARC Future Fellow. From 2005 to 2008, he was a senior research engineer in the Department of Computer Science at Caltech. Prior to that, he was a senior research fellow at the University of Melbourne and a lecturer at RMIT, Australia. His research interests include energy-efficient networking and performance analysis of resource allocation algorithms. He was co-recipient of the best paper award at IEEE INFOCOM 2011 and IEEE MASS 2007. He is a senior member of the IEEE and a member of the ACM.



Ao Tang received the B.E. degree in electronics engineering from Tsinghua University, Beijing, China, and the Ph.D. degree in electrical engineering with a minor in applied and computational mathematics from the California Institute of Technology (Caltech), Pasadena, in 1999 and 2006, respectively. He is currently an Assistant Professor in the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, where his current research interests focus on control and optimization of large scale engineering networks. His recent awards include a Michael Tien'72 Excellence in Teaching Award from the college of engineering at Cornell in 2011, a AFOSR Young investigator award in 2012, and a Presidential Early Career Award for Scientists and Engineers (PECASE) in 2012.