Speed scaling: An algorithmic perspective

Adam Wierman* Lachlan L. H. Andrew† Minghong Lin‡

Abstract

Speed scaling has long been used as a power-saving mechanism at a chip level. However, in recent years, speed scaling has begun to be used as an approach for trading off energy usage and performance throughout all levels of computer systems. This wide-spread use of speed scaling has motivated significant research on the topic, but many fundamental questions about speed scaling are only beginning to be understood. In this chapter, we focus on a simple, but general, model of speed scaling and provide an algorithmic perspective on four fundamental questions: (i) What is the structure of the optimal speed scaling algorithm? (ii) How does speed scaling interact with scheduling? (iii) What is the impact of the sophistication of speed scaling algorithms? and (iv) Does speed scaling have any unintended consequences? For each question we provide a summary of insights from recent work on the topic in both worst-case and stochastic analysis as well as a discussion of interesting open questions that remain.

1 Introduction

Computer systems must make a fundamental tradeoff between performance and energy usage. The days of "faster is better" are gone — energy usage can no longer be ignored in designs, from chips to mobile devices to data centers. This has led speed scaling, a technique once only applied in microprocessors, to become an important technique at all levels of systems. At this point, speed scaling is quickly being adopted across systems from chips [1] to disks [2] and data centers [3] to wireless devices [4].

Speed scaling works by adapting the "speed" of the system to balance energy and performance. It can be highly sophisticated — adapting the speed at all times to the current state (*dynamic speed scaling*) — or very simple — running at a static speed chosen *a priori* to balance energy and performance, except when idle (*gated-static speed scaling*) as used in current disk drives [5].

Dynamic speed scaling leads to many challenging algorithmic problems. Fundamentally, when implementing speed scaling, an algorithm must make two decisions at each time: (i) a *scheduling policy* decides which job(s) to serve, and (ii) a *speed scaler* decides how fast to run. Further, there is an unavoidable interaction between these decisions.

The growing adoption of speed scalers and the inherent complexity of their design has led to a significant, and still growing analytic literature studying the design and performance of speed scaling algorithms. The analytic study of the speed scaling problem began with the seminal work of [6] in 1995. Since [6], three main performance objectives have emerged: (i) the total energy used in order to meet job deadlines, e.g., [7, 8] (ii) the average response time given an energy/power budget, e.g., [9, 10], and (iii) a linear combination of expected response time and energy usage per job [11, 12].

The goal of this chapter *is not* to provide a complete survey of analytic work on speed scaling. For such purposes we recommend the reader consult the recent survey [13] in addition to the current chapter. Instead, the goal of the current chapter *is* to provide examples of the practical insights about fundamental issues in speed scaling that come from the analytic work. Consequently, to limit the scope of the chapter, we focus on only the third performance objective listed above, a linear combination of response time and energy usage, which captures how much reduction in response time justifies using one extra joule and applies where there is a known monetary cost to extra delay (e.g., many web applications). Note that (iii) is related to (ii) by duality. Further, we additionally simplify the setting considered in this chapter by studying only a simple, single resource model, which is described in detail in Section 2.

*Computer and Mathematical Sciences, California Institute of Technology, adamw@caltech.edu

†Centre for Advanced Internet Architectures, Swinburne University of Technology, landrew@swin.edu.au

‡Computer and Mathematical Sciences, California Institute of Technology, mhlin@caltech.edu

Through the simplifications described above, we avoid complexity in the model and can focus on the insights provided by the results we survey. More specifically, we focus on four fundamental issues in speed scaling design for which recent analytic results have been able to provide new, practical insights.

I *What structure do (near-)optimal algorithms have? Can an online speed scaling algorithm be optimal?*

II *How does speed scaling interact with scheduling?*

III *How important is the sophistication of the speed scaler? What benefits come from using dynamic versus static speed scaling?*

IV *What are the drawbacks of speed scaling?*

To provide insight into these issues, we provide an overview of recent work from two analytic frameworks – worst-case analysis and stochastic analysis. For each question we provide a summary of insights from both worst-case and stochastic analysis as well as a discussion of interesting open questions that remain. To conclude the introduction we provide a brief, informal overview of the insights into Issues I-IV that follow from recent results in the analytic community. The remainder of the chapter will then provide a more formal description of the results and insights referred to briefly below.

Issue I: The analytic results we survey prove that *"energy-proportional" speed scaling provides near-optimal performance*. Specifically, by using s_n, the speed to run at given n jobs, such that $P(s_n) = n\beta$ (where $P(s)$ is the power needed to run at speed s and $1/\beta$ is the price of energy) it is possible to be 2-competitive for general P; that is, to incur a total cost at most twice the optimum obtainable by a system will full prior knowledge. This provides analytic justification for a common heuristic applied by system designers, e.g., [14]. However, note that this algorithm does not match the offline optimal performance. In fact, no "natural" speed scaling algorithm (see Definition 1) can be better than 2-competitive; hence no online energy-proportional speed scaler matches the offline optimal. Thus, optimizing for energy and delay is, in some sense, fundamentally harder than optimizing for delay alone.[1]

Issue II: The analytic results we survey related to Issue II uncover two useful insights. First, at least among the policies analyzed so far, *speed scaling can be decoupled from scheduling* in that energy-proportional speed scaling provides nearly optimal performance for three common scheduling policies. Second, scheduling is not as important once energy is considered. Specifically, scheduling policies that are not constant competitive for delay alone become $O(1)$-competitive for the linear combination of energy and response time once speed scaling is considered. These insights allow designers to deal individually with two seemingly coupled design decisions.

Issue III: The analytic results we survey highlight that *increased sophistication provides minimal performance improvements in speed scaling designs.* More specifically, the optimal gated-static speed scaling performs nearly as well as the optimal dynamic speed scaler. However, increased sophistication does provide improved robustness. These insights have the practical implication that it may be better to design "optimally robust" speeds instead of "optimal" speeds, since robustness is the main benefit of dynamic scaling.

Issue IV: The analytic results we survey show that, unfortunately, *dynamic speed scaling can magnify unfairness*. This is perhaps surprising at first, but follows intuitive from the fact that dynamic speed scaling selects faster speeds when there are more jobs in the system: If some property of a job is correlated with the occupancy while it is in service, then dynamic speed scaling gives an unfairly high service rate to jobs with that property. In contrast to the conclusion for Issue II, these results highlight that designers should be wary about the interaction between the scheduler and speed scaler when considering fairness.

2 Model and notation

To highlight the insights that follow from analytic results, we focus on a simple, single server model and ignore many practical issues such as deadlines and the costs of switching speeds. It is important to emphasize however that there is a broad literature of analytic work that has begun to study the impact of such issues, and we do our best to give pointers to this work.

Throughout this chapter we consider a single server with two control decisions: speed scaling and scheduling. Further, we take as the performance objective the minimization of a linear combination of expected response time

[1] A simple, greedy policy (Shortest Remaining Processing Time) optimizes mean delay in a single server system of fixed rate [15].

(also called sojourn time or flow time), denoted by T, and energy usage per job, $\mathcal{E}$:

$$z = \mathbb{E}[T] + \mathbb{E}[\mathcal{E}]/\beta. \tag{1}$$

Using Little's law, this may be more conveniently expressed as

$$\lambda z = \mathbb{E}[N] + \mathbb{E}[P]/\beta \tag{2}$$

where N is the number of jobs in the system and $P = \lambda\mathcal{E}$ is the power expended.

Note that though we take a linear combination of energy and response time as our performance metric, there are a wide variety of other metrics that have been studied in the speed scaling literature. The first to be studied in an algorithmic context was minimizing the total energy consumption given deadlines for each job [6, 16, 17]. In this setting, the energy reduction incurs no performance penalty, since it is assumed that there is no benefit to finishing jobs before the deadline. Another objective is to minimize the makespan (time to finish all jobs) given a constraint on the total energy [17, 18]. This is called the "laptop problem", since the constrained energy consumption matches the fixed capacity of a portable device's battery. Circuit designers are often interested in the product of power and delay [19], which represents the energy per operation. Many other formulations have been considered, such as minimizing delay given a trickle-charged energy storage [20], and cases in which jobs need not be completed but can provide revenue dependent on their delay [21].

Further, though we focus only on a single server, there is recent work that studies speed scaling in multiserver environments as well: in the context of jobs with deadlines [22, 23], the laptop problem [17], or with the above objective [23–26].

2.1 Modeling energy usage

To model the energy usage, first let $n(t)$ denote the number of jobs in the system at time t and $s(t)$ denote the speed that the system is running at time t. Further, define $P(s)$ as the power needed to run at speed s. Then, the energy used by time t is $\mathcal{E}(t) = \int_0^t P(s(\tau))d\tau$.

In many applications $P(s) \approx ks^\alpha$ with $\alpha \in (1, 3)$. For example, for dynamic power in CMOS chips $\alpha \approx 1.8$ [27]. However, wireless communication has an exponential P [28] or even unbounded power at finite rate [29]. Some of the results we discuss assume a polynomial form, and simplify specifically for $\alpha = 2$. Others hold for general, even non-convex and discontinuous, power functions. Several results are limited to *regular* power functions [12], which are differentiable on $[0, \infty)$, strictly convex, non-negative, and $P(0) = 0$.

Note that implicit in the model above is that all jobs have the same $P(s)$. In practice, different classes of jobs may have different mixes of requirements for CPU, disk, IO, etc. This leads to each class of jobs having a different $P(s)$ function characterizing its power-speed tradeoff curve.

2.2 Modeling speed scaling and scheduling

A speed scaling algorithm $\mathcal{A} = (\pi, \Sigma)$ consists of a scheduling discipline π that defines the order in which jobs are processed, and a speed scaler Σ that defines the speed as a function of system state, in terms of the power function, P. Throughout, we consider the speed to depend only on the number of jobs in the system, i.e., s_n is the speed when the occupancy is n.[2]

We consider online preempt-resume schedulers, which are not aware of a job j until it arrives at time $r(j)$, at which point they learn its size, x_j, and which may interrupt serving a job and later restart it from the point it was interrupted without overhead. We focus on Shortest Remaining Processing Time (SRPT), which preemptively serves the job with the least remaining work, and Processor Sharing (PS), which serves all jobs in the system at equal rate. SRPT is important due to its optimality properties: in a classical, constant speed system, SRPT minimizes the expected response time [15]. PS is important because it is a simple model of scheduling that is commonly applied for, e.g., operating systems and web servers. A third policy that we discuss occasionally is First Come First Served (FCFS), which serves jobs in order of arrival.

The speed scaling rules, s_n, we consider can be *gated-static*, which runs at a constant speed while the system is non-empty and sleeps while the system is empty, i.e., $s_n = s_{gs}1_{n\neq 0}$; or more generally *dynamic* $s_n = g(n)$ for

[2]This suits objective (1); e.g., it is optimal for an isolated batch arrival, and the optimal s is constant between arrival/departures. For other objectives, it is better to base the speed on the unfinished work instead [30].

some function $g : \mathbb{N} \cup \{0\} \to [0, \infty)$. To be explicit, we occasionally write s_n^π as the speed under policy π when the occupancy is n. The queue is single-server in the sense that the full speed s_n can be devoted to a single job.

2.3 Analytic framework

Analytic research studying speed scaling has used two distinct approaches: worst-case analysis and stochastic analysis. We discuss results from both settings in this paper, and so we briefly introduce the settings and notation for each in the following.

Notation for the worst-case model

In the worst-case model we consider finite, arbitrary (maybe adversarial) deterministic instances of ν arriving jobs. Let $\mathcal{E}(I)$ 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 $r(j)$. The analog of (1) is to replace the ensemble average by the sample average, giving the cost of an instance I under a given algorithm $\mathcal{A}$ as

$$z^{\mathcal{A}}(I) = \frac{1}{\nu}\left(\sum\nolimits_j T_j + \frac{1}{\beta}\mathcal{E}(I)\right). \tag{3}$$

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

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

where $z^O(I)$ is the optimal cost achievable on I. A scheme is "c-competitive" if its competitive ratio is at most c.

Notation for the stochastic model

In the stochastic model, we consider an M/GI/1 (or sometimes GI/GI/1) queue with arrival rate λ. Let X denote a random job size with cumulative distribution function (c.d.f.) $F(x)$, complementary c.d.f. (c.c.d.f.) $\bar{F}(x) = 1 - F(x)$, and continuous probability density function (p.d.f.) $f(x) = dF(x)/dx$. Let $\rho = \lambda\mathbb{E}[X] \in [0, \infty)$ denote the load of arriving jobs. Note that ρ is not the utilization of the system and that many dynamic speed scaling algorithms are stable for all ρ. When the power function is $P(s) = s^\alpha$, it is natural to use a scaled load, $\gamma := \rho/\beta^{1/\alpha}$ (see [27]).

Denote the response time of a job of size x by $T(x)$. We consider the performance metric (1) where the expectations are averages per job. In this model the goal is to optimize this cost for a specific workload, ρ. Let z^O be the average cost of the optimal offline algorithm. and define the competitive ratio in the M/GI/1 model as

$$CR = \sup_{F,\lambda} z^{\mathcal{A}}/z^O.$$

3 Optimal speed scaling

The most natural starting point for discussing speed scaling algorithms is to understand:

What structure do (near-)optimal algorithms have? Can a speed scaling algorithm be optimal?

In this section, we discuss the stream of research that has sought to answer these questions.

To begin, recall that a speed scaling algorithm depends on two choices, the scheduler and the speed scaler. However, when determining the optimal speed scaling algorithm, the optimality properties of SRPT in the static speed setting [15] make it a natural choice for the scheduler. In fact, it is easy to argue that any speed scaling algorithm which does not use SRPT can be improved by switching the scheduler to SRPT and leaving the speeds $s(t)$ unchanged [12].

Thus, in this section we focus on speed scaling algorithms that use SRPT scheduling and seek to understand the structure of the (near-)optimal speeds. The study of optimal speed scaling algorithms has primarily been performed in the worst-case analytic framework. The reason for this is simple: SRPT is difficult to analyze in the stochastic model even in the case when the speed is fixed. So, when dynamic, controllable speeds are considered the analysis is, to this point, seemingly intractable.

However, in the worst-case framework considerable progress has been made. The focus of this research has primarily been on one particularly promising algorithm: (SRPT, $P^{-1}(n)$), which uses policy $\pi =$ SRPT and speed scaler Σ which sets $s_n = P^{-1}(n)$, and there has been a stream of upper bounds on its competitive ratio for objective (1): for unit-size jobs in [11, 30] and for general jobs with $P(s) = s^\alpha$ in [31, 32]. A major breakthrough was made in [12], which shows the 3-competitiveness of $(SRPT, P^{-1}(n+1))$ for general P. Following this breakthrough, [33] was able to use similar techniques to reduce the competitive ratio slightly and prove a matching lower bound. In particular, [33] establishes the following:

Theorem 1. *For any regular power function P, (SRPT, $P^{-1}(n\beta)$) has a competitive ratio of exactly 2.*

Thus, (SRPT, $P^{-1}(n)$) is not optimal, in that an offline algorithm could do better. However, it is highly robust — guaranteeing to provide cost within a factor of two of optimal.

We will not dwell on the analytic approaches in this chapter, however it is useful to give some feel for the proof techniques for the major results. In this case, the proof uses amortized local competitiveness arguments with the potential function

$$\Phi(t) = \int_0^\infty \sum_{i=1}^{n[q;t]} \frac{1+\eta}{\beta} P'\left(P^{-1}\left(i\beta\right)\right)\, dq, \tag{4}$$

where $n[q;t] = \max(0, n^{\mathcal{A}}[q;t] - n^O[q;t])$ with $n^{\mathcal{A}}[q;t]$ and $n^O[q;t]$ the number of unfinished jobs at time t with remaining size at least q under (SRPT,$P^{-1}(n\beta)$) (algorithm $\mathcal{A}$) and the optimal (offline) algorithm O, respectively. Specifically, it first shows that

$$z^{\mathcal{A}}(t) + \frac{d\Phi}{dt} \le (1+\eta) z^O(t), \tag{5}$$

where $z^{\mathcal{A}}(t)$ and $z^O(t)$ are the instantaneous costs of $\mathcal{A}$ and O respectively, given by $n(t) + P(s(t))/\beta$. It then shows that integrating this, with $\eta = 1$, gives the above theorem.

The following more general form can be proven analogously.

Corollary 2. *Let $\eta \ge 1$ and Φ be given by (4). Any scheme $\mathcal{A} = (SRPT, s_n)$ with $s_n \in [P^{-1}(n\beta), P^{-1}(\eta n\beta)]$ is $(1+\eta)$-competitive.*

The above results motivate the question:

Are there other speed scaling algorithms with lower competitive ratios?

A recent result [33] suggests that the answer is "No". In particular, no "natural" speed scaling algorithm can be better than 2-competitive.

Definition 1. *A speed scaling algorithm $\mathcal{A}$ is defined to be **natural** if it runs at speed s_n when it has n unfinished jobs, and for convex P, one of the following holds:*

(a) the scheduler is work-conserving and works on a single job between arrival/departure events; or

(b) $g(s) + P(s)/\beta$ is convex, for some g with $g(s_n) = n$; or

(c) the speeds s_n satisfy $P(s_n) = \omega(n)$; or

(d) the speeds s_n satisfy $P(s_n) = o(n)$.

Theorem 3. *For any $\varepsilon > 0$ there is a regular power function P_ε such that any natural algorithm $\mathcal{A}$ on P_ε has competitive ratio larger than $2 - \varepsilon$.*

Note that natural algorithms include SRPT scheduling for any s_n, energy-proportional speeds for all schedulers and any speeds consistently faster or slower than energy-proportional. We conjecture that Theorem 3 holds for all speed scaling algorithms, which would imply that the competitive ratio of (SRPT, $P^{-1}(n\beta)$) is minimal.

Interestingly, the proof considers only two classes of workloads: one consisting of an isolated burst of jobs and the other consisting of a long train of uniformly spaced jobs. Any "natural" scheduler which processes the long train fast enough to avoid excessive delay costs must process the burst so fast that it incurs excessive energy costs; thus optimizing over these two specific workloads is already impossible for a "natural" speed scaler.

3.1 Open questions

The results above highlight that that energy-proportional speed scaling ($P(s_n) = n\beta$) is nearly optimal, which provides analytic justification of a common design heuristic, e.g., [14]. Further, they highlight that optimizing for the combination of energy and delay is fundamentally harder than optimizing for delay alone (where SRPT is optimal, i.e., 1-competitive). However, they also leave a number of interesting and important questions unanswered. Three such questions that are particularly interesting are the following.

First, the fact that Theorem 3 holds only for "natural" algorithms is bothersome and likely only a technical restriction. It would be very nice to remove this restriction and provide a true lower bound on the achievable competitive ratio.

Additionally, the fact that (SRPT,$P^{-1}(n)$) is exactly 2-competitive for every power function highlights that it may be possible to outperform this algorithm for specific power functions. In particular, for the realistic case of $P(s) = s^\alpha$ with $\alpha \in (1, 3)$ it is possible to derive other speed scalers besides $P^{-1}(n)$ that have competitive ratios smaller than 2. In the extreme case that P is concave, it is optimal to run at the maximum possible speed whenever there is work in the system, and be idle otherwise. It would be interesting to understand in general what the lower bound on the achievable competitive ratio for a fixed $P(s)$ is, and what speed scaler can achieve it.

Finally, a third open problem is to understand the optimal speeds for SRPT scheduling in the stochastic model, i.e., in an M/GI/1 queue. This problem is completely open, and seemingly difficult, but would provide interesting new insight.

These three open questions all relate to the simple model which is the focus of this chapter; however the issues discussed in this section are clearly also interesting (and open) when the model is made more complex. For example, by allowing class-dependent $P(s)$ functions.

4 The interaction of speed scaling and scheduling

In the prior section, we focused on optimal speed scaling algorithms, and thus limited our attention to SRPT. However, in practice, SRPT may not be appropriate, for a variety of reasons. In many situations, other scheduling policies are desirable (or required) and in particular, policies such as PS are used. Once a different scheduling policy is considered it is not clear how the optimal speeds should be adjusted. In particular, there is a clear coupling between the speed scaling and the scheduling. So, the focus of this section is to understand:

How does speed scaling interact with scheduling?

To shed light on this question we focus primarily on PS in this section, due to its importance in application such as operating systems and web servers. However, the insights we describe certainly hold more generally.

There is work relating to the interaction of scheduling and speed scaling from both the worst-case and stochastic frameworks, so we will discuss each briefly. Interestingly, the two distinct sets of results highlight similar insights. In particular they highlight that (i) scheduling and speed scaling can be largely decoupled with minimal performance loss and (ii) scheduling is much less important when considering speed scaling than it is in standard models with constant speeds.

4.1 Worst-case analysis

In contrast to the long stream of research focusing on SRPT, until recently there was no worst-case analysis of speed scaling under PS. The first result relating to PS is from [33], which shows that (PS, $P^{-1}(n\beta)$) is $O(1)$-competitive for $P(s) = s^\alpha$ with fixed α. In particular:

Theorem 4. *If $P(s) = s^\alpha$ then (PS, $P^{-1}(n\beta)$) is $\max(4\alpha - 2, 2(2 - 1/\alpha)^\alpha)$-competitive.*

As with SRPT, this admits a more general corollary

Corollary 5. *For s_n given in Corollary 2, the scheme (PS, s_n) is $O(1)$-competitive in the size of the instance.*

Notice that the competitive ratio in Theorem 4 reduces to $(4\alpha - 2)$-competitive for the α typical in CPUs, i.e., $\alpha \in (1, 3]$.

The proof of Theorem 4 in [33] builds on the analysis in [34] of LAPS, another blind policy. (LAPS, $P^{-1}(n\beta)$) is also $O(1)$-competitive in this case, and actually has better asymptotic performance for large α. However, as $\alpha \in (1, 3]$ in most computer sub-systems, the performance for small α is more important than asymptotics in α, and PS outperforms LAPS in that case. In both the case of PS and LAPS the proof uses the potential function

$$\Phi = (1+\eta)(2\alpha-1)(\mathsf{H}/\beta)^{1/\alpha} \sum_{i=1}^{n^{\mathcal{A}}(t)} i^{1-1/\alpha} \max(0, q^{\mathcal{A}}(j_i;t) - q^{O}(j_i;t)) \tag{6}$$

where $q^{\pi}(j;t)$ is the remaining work on job j at time t under scheme $\pi \in \{\mathcal{A}, O\}$, and $\{j_i\}_{i=1}^{n^{\mathcal{A}}(t)}$ is an ordering of the jobs in increasing order of release time: $r(j_1) \leq r(j_2) \leq \cdots \leq r(j_{n^{\mathcal{A}}(t)})$.

This result has been generalized in [35] to show that weighted PS is again $O(1)$-competitive in the case when jobs have differing weights. If the processor has a maximum speed, then the result still holds under a resource augmentation assumption.

There are a number of remarks we would like to highlight about Theorem 4. First, notice that this scheme uses the same energy-proportional speeds $s_n = P^{-1}(n\beta)$ that achieve the optimal competitive ratio under SRPT. Further, the same speeds are also used for LAPS in [34]. This suggests that the two design decisions of speed scaling and scheduling are not as closely coupled as may be first thought.

A second remark about Theorem 4 is that, although this competitive ratio of around 10 is larger than may be hoped, it is very much smaller than the competitive ratio of PS on a fixed-speed processor. On a fixed-speed processor, PS has an unbounded competitive ratio: PS is $\Omega(\nu^{1/3})$-competitive for instances with ν jobs [36]. This is because PS is "blind" to the job sizes, which can cause it to make a poor decision early on, from which it can never recover. With unbounded speeds and dynamic speed scaling, the processor can always run a little faster to recover quickly, and thus continue with a modest additional cost. Note however, that difference in cost across scheduling policies grows if there is a bounded maximum speed.

To explain the limited impact of scheduling, one can view speed scaling as a physical realization of the analytical technique of resource augmentation [37]. The latter shows that many algorithms with high competitive ratios could achieve almost the same cost as the optimal algorithm, if they were given the advantage of a slightly faster processor.

Finally, a third remark about Theorem 4 is that it is much weaker than Theorem 1 for SRPT: Not only is the competitive ratio larger, but the result is only valid for power functions of the form $P(s) = s^{\alpha}$, and even depends strongly on the exponent α. This is not simply because the result is incomplete, but because the problem is fundamentally less well approximable. In fact, all policies blind to job sizes have unbounded competitive ratio as $\alpha \to \infty$ [34], which shows that the form of power function must unavoidably enter any competitive ratio result.

4.2 Stochastic analysis

In contrast to the limited amount of research studying PS in the worst-case framework, there has been a substantial amount of work applicable to PS in the stochastic framework, due to its analytic tractability. In particular, in the M/GI/1 model, [38–41][3] study speed scaling in the context of generic "operating costs".

This work formulates the determination of the optimal speeds as a dynamic programming (DP) problem and provides numerical techniques for determining the optimal speeds. Further, it has been proven that the optimal speeds are monotonic in the queue length [43] and that the optimal speeds can be bounded as follows [27]. Recall that $\gamma = \rho/\beta^{1/\alpha}$.

Proposition 6. *Consider an M/GI/1 PS queue with controllable service rates s_n. Let $P(s) = s^{\alpha}$. The optimal dynamic speeds are concave and satisfy the dynamic program given in [27]. For $\alpha = 2$ and any $n \geq 2\gamma$, they satisfy*

$$\gamma + \sqrt{n - 2\gamma} \leq \frac{s_n^{DP}}{\sqrt{\beta}} \leq \gamma + \sqrt{n} + \min\left(\frac{\gamma}{2n}, \gamma^{1/3}\right). \tag{7}$$

[3] Since M/GI/1 PS with controllable service rates is symmetric [42], it has the same occupancy distribution and mean delay as M/M/1 FCFS studied in the references.

For general $\alpha > 1$, they satisfy

$$\frac{s_n^{DP}}{\beta^{1/\alpha}} \leq \left(\frac{n}{\alpha-1}\right)^{1/\alpha} + \gamma\frac{1}{\alpha-1} + O(n^{-1/\alpha}) \tag{8}$$

$$\frac{s_n^{DP}}{\beta^{1/\alpha}} \geq \left(\frac{n}{\alpha-1}\right)^{1/\alpha}. \tag{9}$$

Interestingly, the bounds in Proposition 6 are tight for large n and have a form similar to the form of the worst-case speeds for SRPT and PS in Theorems 1 and 4.

The asymptotic form of these results also matches the form of the optimal speed when there are n jobs in the system and no more jobs will arrive before the queue is emptied, as occurs when the instance is a single batch arrival. For both SRPT and PS, this batch speed is given by the solution s_n^* to

$$\beta n = s_n^* P'(s_n^*) - P(s_n^*). \tag{10}$$

This is not surprising, since when the backlog is large, the arrival rate will be negligible compared with the optimal speed. This insight is further supported by looking at the fluid model, e.g., [44], where the differences between scheduling policies are negligible.

As we have discussed previously, the analysis of the optimal dynamic speeds given SRPT scheduling is seemingly intractable. However, taking inspiration from the fact that the speeds given by the worst-case analysis of SRPT and PS match, combined with the fact that (10) is the optimal batch speed for both SRPT and PS and also matches the asymptotic form of the stochastic results for PS in Proposition 6, a natural proposal for the dynamic speeds under SRPT is to use the optimal PS speeds.

It turns out that this heuristic works quite well. In particular the simulation experiments in Fig. 1 compare the performance of this dynamic programming speed scaling algorithm ("DP") both to the form $P^{-1}(n\beta)$ motivated by the worst-case results ("INV") and to the following lower bound, which was proven by [27] and holds for all scheduling policies.

Proposition 7. *In a GI/GI/1 queue with $P(s) = s^\alpha$,*

$$z^O \geq \frac{1}{\lambda}\max(\gamma^\alpha, \gamma\alpha(\alpha-1)^{(1/\alpha)-1}).$$

The above was stated in [27] for M/GI/1 PS, but the same proof holds more generally.

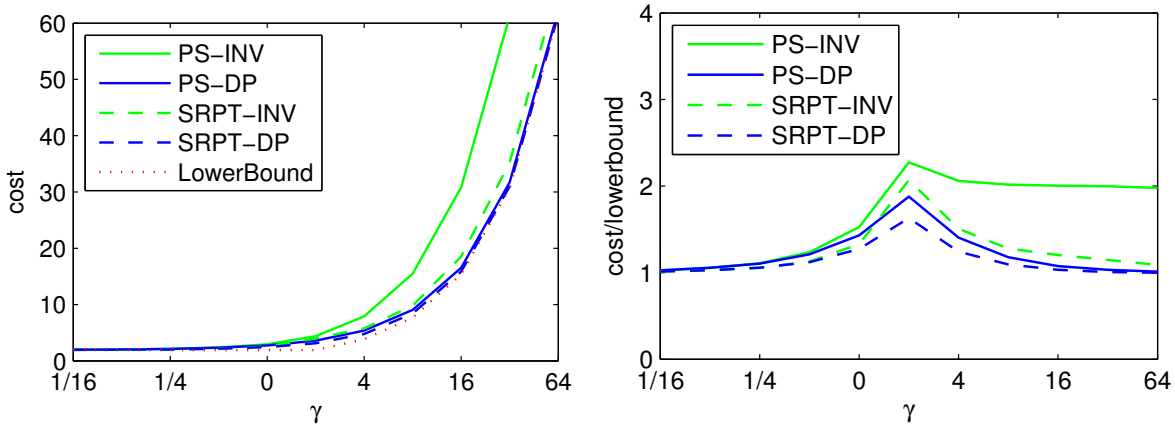
Simulation experiments also allow us to compare (i) the performance of the worst-case schemes for SRPT and PS with the stochastic schemes and (ii) the performance of SRPT and PS in the speed scaling model. In these experiments, the optimal speeds for PS in the stochastic model are found using the numerical algorithm described in [27, 40], and then these speeds are also used for SRPT.

Figure 2 shows that the optimal speeds from the DP have a similar form to the speeds $P^{-1}(n\beta)$ appearing in the worst-case results, differing by γ for high queue occupancies. Figure 1 shows how the total cost (1) depends on the choice of speeds and scheduler. At low loads, all schemes are indistinguishable. At higher loads, PS-INV degrades significantly, but SRPT-INV maintains good performance. Note though that if $P(s) = s^\alpha$ for $\alpha > 3$, SRPT-INV degrades significantly too. In contrast, the DP-based schemes benefit significantly from having the slightly higher speeds chosen to optimize (1) rather than minimize the competitive ratio. Finally, SRPT-DP performs nearly optimally, which justifies the heuristic of using the optimal speeds for PS in the case of SRPT. However, PS-DP performs nearly as well as SRPT-DP. Together, these observations suggest that it is important to optimize the speed scaler, but not necessarily the scheduler.

4.3 Open questions

The results in this section have highlighted a number of important insights regarding the decoupling of speed scaling and scheduling. In particular, it seems that energy proportional speed scaling ($s_n = P^{-1}(n)$) performs well for a variety of scheduling policies and it seems that performance differences between scheduling policies shrink dramatically when speed scaling is considered.

However, to this point these insights are derived by comparing a small number of common policies. We conjecture that these insights will apply much more broadly; however it is an open question to understand exactly how broadly



(a) (b)

Figure 1: Comparison of SRPT and PS under both $s_n = P^{-1}(n\beta)$ and speeds optimized for an M/GI/1 PS system, using Pareto(2.2) job sizes and $P(s) = s^2$.

they apply. More specifically, it would be interesting to characterize the class of scheduling policies for which $s_n = P^{-1}(n)$ guarantees O(1)-competitiveness. This would jointly highlight a decoupling between scheduling and speed scaling and show that scheduling has much less impact on performance. The above question proposes to study the decoupling of scheduling and speed scaling in the worst-case framework. In the stochastic framework however, the parallel question can be asked: for what class of policies will $s_n = P^{-1}(n)$ guarantee O(1)-competitiveness in the M/GI/1 model.

5 Impact of speed scaler sophistication

Up until this point we have focused on dynamic speed scaling, which allows the speed to change as a function of the number of jobs in the system. Dynamic speed scaling can perform nearly optimally; however its complexity may be prohibitive. In contrast, *gated-static* speed scaling, where $s_n = s_{gs}1_{n\neq 0}$ for some constant speed s_{gs}, requires minimal hardware to support; e.g., a CMOS chip may have a constant clock speed but AND it with the gating signal to set the speed to 0.

In this section our goal is to understand:

> *How important is the sophistication of the speed scaler? What benefits come from using dynamic versus static speed scaling?*

Note that we can immediately see that gated-static speed scaling can be arbitrarily bad in the worst case since jobs can arrive faster than s_{gs}. Thus, we study gated-static speed scaling only in the stochastic model, where the constant speed s_{gs} can depend on the load.

We focus our discussion on SRPT and PS, though the insights we highlight do not depend on the workings of these particular policies and should hold more generally.

Overall, there are three main insights that have emerged from the literature regarding the questions above. First, and perhaps surprisingly, the simplest policy (gated-static) provides nearly the same expected cost as the most sophisticated policy (optimal dynamic scaling). Second, the performance of gated-static under PS and SRPT are not too different, thus scheduling is less important to optimize than in systems in which the speed is fixed in advance. This aligns with the observations discussed in the previous section for the case of dynamic speed scaling. Third, though dynamic speed scaling does not provide significantly improved cost, it does provide significantly improved robustness (e.g., to time-varying and/or uncertain workloads).

5.1 Optimal gated-static speeds

To begin our discussion, we must first discuss the optimal speeds to use for gated-static designs under both PS and SRPT. In the case of PS, this turns out to be straightforward; however in the case of SRPT it is not.

Before stating the results, note that since the power cost is constant at $P(s_{gs})$ whenever the server is running, the optimal speed is

$$s_{gs} = \arg\min_s \beta\mathbb{E}[T] + \frac{1}{\lambda}P(s)\Pr(N \neq 0). \tag{11}$$

In the second term $\Pr(N \neq 0) = \rho/s$, and so multiplying by λ and setting the derivative to 0 gives that the optimal gated-static speed satisfies

$$\beta\frac{d\mathbb{E}[N]}{ds} + r\frac{P^*(s)}{s} = 0, \tag{12}$$

where $r = \rho/s$ is the utilization and $P^*(s) \equiv sP'(s) - P(s)$. Note that if P is convex then P^* is increasing and if P'' is bounded away from 0 then P^* is unbounded.

Under PS, $\mathbb{E}[N] = \rho/(s-\rho)$, and so $d\mathbb{E}[N]/ds = \mathbb{E}[N]/(\rho - s)$. By (12), the optimal speeds satisfy [27]

$$\beta\mathbb{E}[N] = (1-r)rP^*(s). \tag{13}$$

For SRPT, things are not as easy. For $s = 1$, we have [45]

$$\mathbb{E}[T] = \int_{x=0}^{\infty}\int_{t=0}^{x} \frac{dt}{1-\lambda\int_0^t \tau\, dF(\tau)} + \frac{\lambda\int_0^x \tau^2\, dF(\tau) + x^2\bar{F}(x)}{2(1-\lambda\int_0^x \tau dF(\tau))^2}\, dF(x)$$

The complexity of this equation rules out calculating the speeds analytically. So, instead [33] derives simpler forms for $\mathbb{E}[N]$ that are exact in asymptotically heavy or light traffic. Note that using a heavy-traffic approximation is natural for this setting because, if energy is important (β small), than the speed will be provisioned so that the server is in "high" load.

We state the heavy-traffic results for distributions whose c.c.d.f. $\bar{F}$ has lower and upper Matuszewska indices [46] of m and M. Intuitively this means that $C_1x^m \lesssim \bar{F}(x) \lesssim C_2x^M$ as $x \to \infty$ for some C_1, C_2, so the Matuszewska index can be thought of as a "moment index." Further, let $G(x) = \int_0^x tf(t)\, dt/\mathbb{E}[X]$ be the fraction of work coming from jobs of size at most x, and define $h(r) = (G^{-1})'(r)/G^{-1}(r)$. The following was proven in [47].

Proposition 8 ([47]). *For an $M/GI/1$ under SRPT with speed 1, $\mathbb{E}[N] = \Theta(H(\rho))$ as $\rho \to 1$, where*

$$H(\rho) = \begin{cases} E[X^2]/((1-\rho)G^{-1}(\rho)) & \text{if } M < -2 \\ E[X]\log(1/(1-\rho)) & \text{if } m > -2. \end{cases} \tag{14}$$

For the case of unit speed, this suggests the heavy traffic approximation

$$\mathbb{E}[N] \approx CH(\rho) \tag{15}$$

where the constant C depends on the distribution of job sizes.

Theorem 9. *Under the assumption that (15) holds with equality:*

(i) If $M < -2$, then for the optimal gated-static speed,

$$\beta\mathbb{E}[N]\left(\frac{2-r}{1-r} - rh(r)\right) = rP^*(s). \tag{16a}$$

(ii) If $m > -2$, then for the optimal gated-static speed,

$$\beta\mathbb{E}[N]\left(\frac{1}{(1-r)\log(1/(1-r))}\right) = P^*(s). \tag{16b}$$

Moreover, if $P^*(s)$ is unbounded as $s \to \infty$ and $-2 \notin [m, M]$ then as $\rho \to \infty$, (16) induces the heavy-traffic regime, $\rho/s \to 1$.

Although the heavy traffic results hold for large ρ, for small ρ the performance of the optimal GS speed for PS gives better performance. To combine these, [33] proposes using the speed

$$s_{gs}^{SRPT} = \min(s_{gs}^{PS}, s_{gs}^{SRPT(HT)}), \tag{17}$$

where s_{gs}^{PS} satisfies (13), and $s_{gs}^{SRPT(HT)}$ is given by (16) with $\mathbb{E}[N]$ estimated by (15).

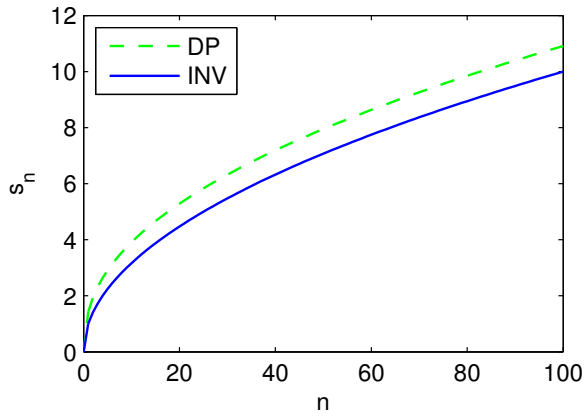


Figure 2: Comparison of $s_n = P^{-1}(n\beta)$ with speeds "DP" optimized for an M/GI/1 system with $\gamma = 1$ and $P(s) = s^2$.

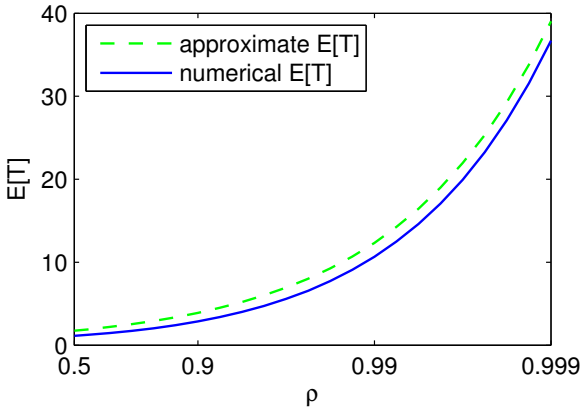


Figure 3: Validation of the heavy-traffic approximation (15) by simulation using Pareto(3) job sizes with $E[X] = 1$.

5.2 Dynamic vs. gated static: cost comparison

Given the optimal gated static speeds derived in the previous section, we can now contrast the cost of gated-static, the simplest scheme, with that of dynamic speed scaling, the most sophisticated under both SRPT and PS.

As Fig. 4 shows, the performance of a well-tuned gated-static system is almost indistinguishable from that of the optimal dynamic speeds. Moreover, there is little difference between the cost under PS-GATED and SRPT-GATED, again highlighting that scheduling becomes less important when speed can be scaled than in traditional models.

Additionally, [27] has recently provided analytic support for the empirical observation that gated-static schemes are near optimal. Specifically, in [27] it was proven that PS-GATED is within a factor of 2 of PS-DP when $P(s) = s^2$. With the results of Section 3, this implies:

Corollary 10. *If $P(s) = s^2$ then the optimal PS and SRPT gated-static designs are $O(1)$-competitive in an M/GI/1 queue with load ρ.*

5.3 Dynamic vs. gated static: robustness comparison

The previous section shows that near-optimal performance is obtained by running at a static speed when not idle. *Why then do CPUs have multiple speeds?* The reason is that the optimal gated-static design depends intimately on the load ρ. This cannot be exactly known in advance, especially since workloads vary with time. So, an important property of a speed scaling design is *robustness* to uncertainty in the workload (ρ and F) and to model inaccuracies.

Figure 5 shows that the performance of gated-static degrades dramatically when ρ is mispredicted. If the load is lower than expected, excess energy will be used; if the system has static speed s and $\rho \geq s$ then the cost is unbounded. In contrast, Fig. 5 also shows that dynamic speed scaling (SRPT-DP) is significantly more robust to misprediction of the workload.

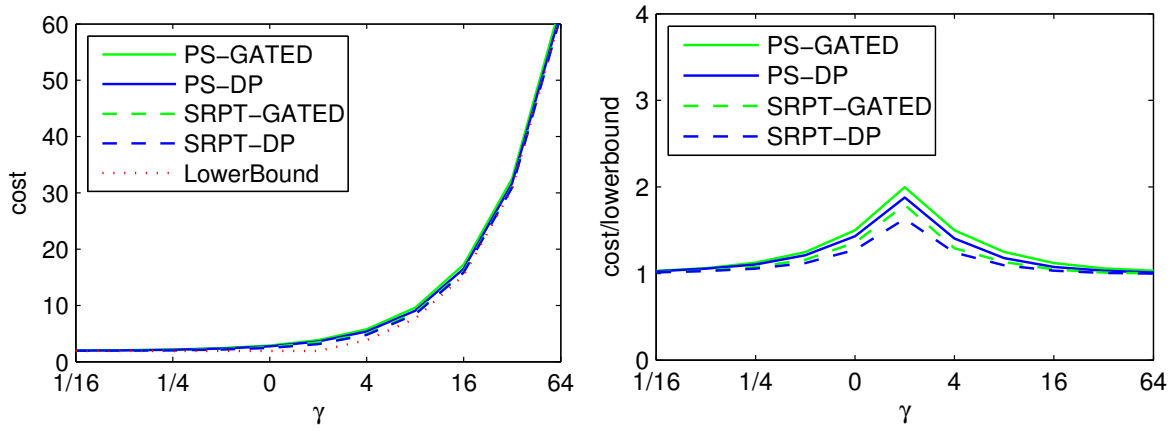
Recently, [33] proved the robustness of SRPT-DP and PS-DP by providing worst-case guarantees for SRPT-DP and PS-DP showing that they are both constant competitive. These results are nearly unique in that they provide worst-case guarantees for stochastic control policies. In the following, let s_n^{DP} denote the speeds used for SRPT-DP and PS-DP.

Corollary 11. *Consider $P(s) = s^\alpha$ with $\alpha \in (1, 2]$ and algorithm $\mathcal{A}$ which chooses speeds s_n^{DP} optimal for PS in an M/GI/1 queue with load ρ. If $\mathcal{A}$ uses either PS or SRPT, then $\mathcal{A}$ is $O(1)$-competitive in the worst-case.*

The proof of the result follows from combining (8) and (9) with Corollaries 2 and 5.

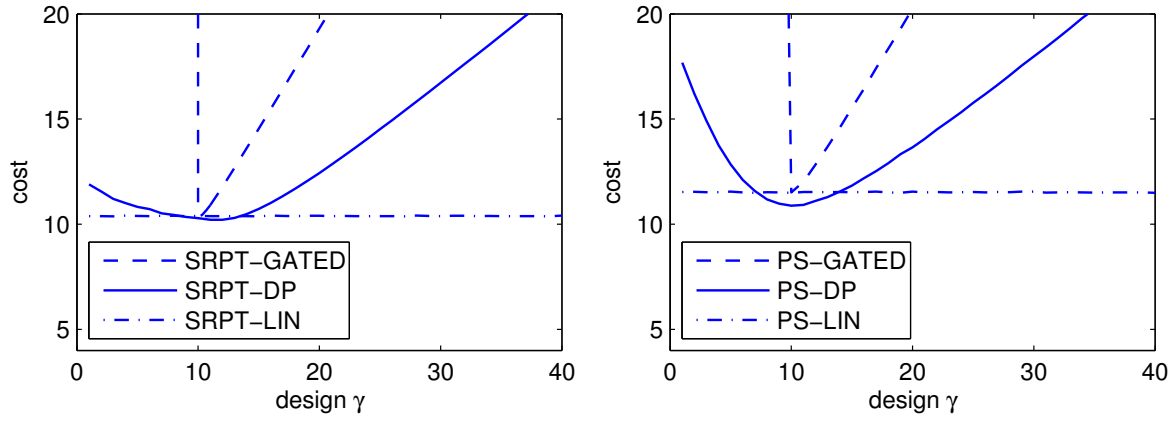
Notice that for $\alpha = 2$, Proposition 6 implies $s_n^{DP} \leq (2\gamma + 1)P^{-1}(n\beta)$, whence (SRPT, s_n^{DP}) is $(4\gamma^2 + 4\gamma + 2)$-competitive.

Finally, though Corollary 11 shows that s_n^{DP} designed for a given ρ leads to a "robust" speed scaler, note that the cost still degrades significantly when ρ is mispredicted badly (see Fig. 5). To address this issue, we consider a second,



(a) (b)

Figure 4: Comparison of PS and SRPT with gated-static speeds (13) and (17), versus the dynamic speeds optimal for an M/GI/1 PS. Job sizes are distributed as Pareto(2.2) and $P(s) = s^2$.



(a) SRPT (b) PS

Figure 5: Effect of misestimating γ under PS and SRPT: cost when $\gamma = 10$, but s_n are optimal for a different "design γ". Pareto(2.2) job sizes; $P(s) = s^2$.

weaker form of robustness: robustness only to misestimation of ρ, not to the underlying stochastic model. In particular, in this model the arrivals are known to be Poisson, but ρ is unknown. In this setting, it was shown in [27] that using "linear" speeds, $s_n = n\sqrt{\beta}$, gives near-optimal performance when $P(s) = s^2$ and PS scheduling is used. Specifically, this scheme ("LIN") out-performs the other load-independent scaling, $s_n = P^{-1}(n\beta)$, as shown in Figure 5. Further, the decoupling of scheduling and speed scaling suggested in Section 4 motivates using LIN also for SRPT. Figure 6 also shows that SRPT-LIN is again nearly optimal when the exact load is known. However, it is important to note that LIN is not robust to model inaccuracies: it is not $O(1)$-competitive for non-Poisson workloads.

5.4 Open questions

This section has highlighted that the increased sophistication of dynamic speed scaling as compared with gated-static speed scaling does not provide much reduction in cost. However, it does provide at least one important benefit – improved robustness. This message of "sophistication improves robustness" is supported by results that combine worst-case and stochastic analysis to provide worst-cast guarantees for optimal stochastic control.

One key implication of these insights is that dynamic speed scalers should be designed with robustness as the goal, not cost minimization (since cost can be minimized without dynamic speeds). Thus, an open question is how to formulate and solve a stochastic control problem for optimizing robustness of speed scalers.

A second set of open problem relates to Corollary 11, which provides worst-case guarantees for the optimal stochastic control policy. This is the first result of its kind in this area, and thus the guarantees it provides are likely loose and should be able to be improved, either via a tighter analysis or by finding a new algorithm with a better tradeoff between optimality in the stochastic model and robustness in the worst-case model.

Finally, and perhaps most importantly, throughout this chapter we are ignoring the costs (both in terms of energy and delay) of switching the speed of the server and we are allowing arbitrary speeds to be chosen. In practice these simplifications are significant, and certainly should affect the design of the speed scaler. In particular, they are important to consider when comparing dynamic and gated static speed scaling. Some analytic work has begun to take these issues into consideration [48], however there are many unanswered questions that remain.

6 Drawbacks of speed scaling

We have seen that speed scaling has many benefits; however it is also important to consider potential drawbacks of speed scaling.

Are there unintended drawbacks to speed scaling?

In this section we discuss one unintended, undesirable consequence of speed scaling — magnifying unfairness.

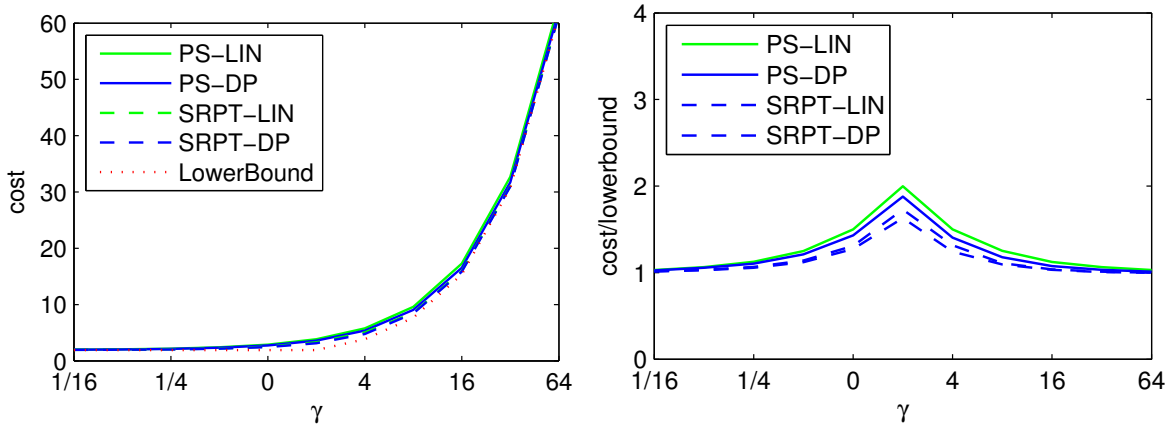
Fairness is important in many applications, and as a result there is a large literature of work studying the fairness of scheduling policies. However, the interaction of fairness and energy efficiency has received little attention so far, e.g., [33, 49]. In the domain of speed scaling, [33] was the first to identify that speed scaling can have unintended negative consequences with respect to fairness. Intuitively, speed scaling can create unfairness as follows: If there is some job type that is always served when the queue length is long/short it will receive better/worse performance than it would have in a system with a static speed. To see that this magnifies unfairness, note that the scheduler has greatest flexibility in job selection when the queue is long, and so jobs served at that time are likely to be those that already get better service.

In fact, it is possible to prove that the intuition above is correct and that the service rate differential can lead to unfairness in a rigorous sense under SRPT and non-preemptive policies such as First-Come-First-Serve (FCFS). However, under PS, speed scaling does not lead to unfairness because of the inherent fairness of the policy.

6.1 Defining fairness

The fairness of scheduling policies has been studied extensively, leading to a variety of fairness measures, e.g., [50–52], and the analysis of nearly all common scheduling policies, e.g., [52–54]. See also the survey [55].

We compare fairness not between individual jobs, but between classes of jobs, consisting of all jobs of a given size. Since this chapter focuses on delay, we compare $\mathbb{E}[T(x)]$ across x. where $\mathbb{E}[T(x)]$ denotes the expected response time for a job of size x. Fairness when $s = 1$ has been previously defined as [55]:



(a) (b) c

Figure 6: Comparison of PS and SRPT with linear speeds, $s_n = n\sqrt{\beta}$, and with dynamic speeds optimal for PS. Job sizes are Pareto(2.2) and $P(s) = s^2$.

Definition 2. *A policy π is fair if for all x*

$$\frac{\mathbb{E}[T^{\pi}(x)]}{x} \leq \frac{\mathbb{E}[T^{PS}(x)]}{x}.$$

This metric is motivated by the fact that (i) PS is intuitively fair since it shares the server evenly among all jobs at all times; (ii) for $s = 1$, the slowdown ("stretch") of PS is constant, i.e., $\mathbb{E}[T(x)]/x = 1/(1-\rho)$; (iii) $\mathbb{E}[T(x)] = \Theta(x)$ [56], so normalizing by x when comparing the performance of different job sizes is appropriate. Additional support is provided by the fact that $\min_{\pi} \max_{x} \mathbb{E}[T^{\pi}(x)]/x = 1/(1-\rho)$ [52].

By this definition, the class of large jobs is treated fairly under all work-conserving policies, i.e., $\lim_{x\to\infty} \mathbb{E}[T(x)]/x \leq 1/(1-\rho)$ [56] — even policies such as SRPT that seem biased against large jobs. In contrast, all non-preemptive policies, e.g., FCFS have been shown to be unfair to small jobs [52].

The foregoing applies only when $s = 1$. However, the following Proposition shows that PS still maintains a constant slowdown for arbitrary speeds, and so Definition 2 is still a natural notion of fairness.

Proposition 12. *Consider an M/GI/1 queue with a symmetric scheduling discipline, e.g., PS with controllable service rates. Then,* $\mathbb{E}[T(x)] = x\left(\mathbb{E}[T]/\mathbb{E}[X]\right)$.

6.2 Speed scaling magnifies unfairness

Using the above criterion for fairness, we can now illustrate that speed scaling creates/magnifies unfairness under SRPT and non-preemptive policies such as FCFS.

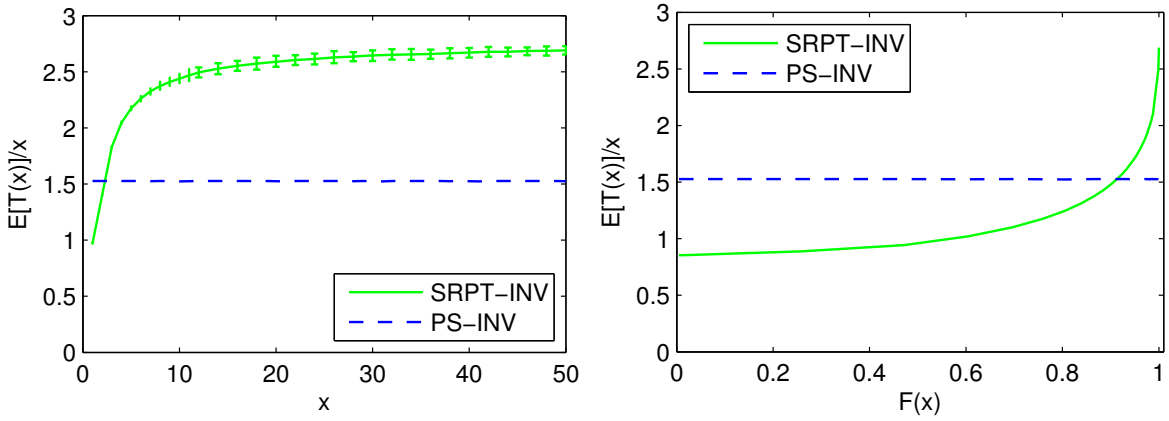
6.2.1 SRPT

We first show that SRPT treats the largest jobs unfairly in a speed scaling system. Recall that the largest jobs are always treated fairly in the case of a static speed ($s = 1$).

Let $\bar{s}^{\pi}$ be the time average speed under policy π, and let $\pi + 1$ denote running policy π on a system with a permanent customer in addition to the stochastic load (e.g., $\bar{s}^{PS+1}$).

Theorem 13. *Consider a GI/GI/1 queue with controllable service rates and unbounded inter-arrival times. Let $s_n^{SRPT} \leq s_n^{PS}$ be weakly monotone increasing and satisfy $\bar{s}^{PS+1} > \rho$ and $\bar{s}^{SRPT+1} > \rho$. Then*

$$\lim_{x\to\infty} \frac{T^{PS}(x)}{x} <_{a.s.} \lim_{x\to\infty} \frac{T^{SRPT}(x)}{x}.$$



(a) vs job size

(b) vs CDF of job size

Figure 7: Slowdown of large jobs under PS and SRPT under Pareto(2.2) job sizes, $\gamma = 1$, $s_n = P^{-1}(n)$, and $P(s) = s^2$. Note the fairness of PS.

The intuition behind Theorem 13 is the following. An infinitely sized job under SRPT will receive almost all of its service while the system is empty of smaller jobs. Thus it receives service during the idle periods of the rest of the system. Further, if $s_n^{SRPT} \leq s_n^{PS}$ then the busy periods will be longer under SRPT and so the slowdown of the largest job will be strictly greater under SRPT. This intuition also provides an outline of the proof.

Figure 7 shows that, under SRPT, large jobs suffer a significant increase in slowdown as compared to PS, although only 10% of the jobs are worse off. Since this setting has a moderate load, SRPT with static speeds would be fair to all job sizes. Figure 7(a) shows 90% confidence intervals.

Theorem 13 proves that SRPT cannot use dynamic speeds and provide fairness to large jobs; however, by using gated-static speed scaling SRPT can provide fairness, e.g., [52]. Further, as Fig. 4 illustrates, gated-static speed scaling provides nearly optimal cost. So, it is possible to be fair and near-optimal using SRPT scheduling but, to be fair, robustness must be sacrificed.

6.2.2 Non-preemptive policies

The magnification of unfairness by speed scaling also occurs for all non-preemptive policies.

For a fixed speed, all non-preemptive policies are unfair to small jobs [52] since the response time must include at least the residual of the job size distribution if the server is busy:

$$\mathbb{E}[T(x)]/x \geq 1 + \rho\mathbb{E}[X^2]/(2\mathbb{E}[X]x),$$

which grows unboundedly as $x \to 0$. However, if we condition on the arrival of a job to an empty system (i.e., the work in system at arrival $W = 0$), then non-preemptive policies are "fair", in the sense that the slowdown is constant: $T(x|W = 0)/x = 1$. Speed scaling magnifies unfairness under non-preemptive policies in the sense that it allows $T(x|W = 0)/x$ to differ dramatically across job sizes.

Proposition 14. *Consider a non-preemptive GI/GI/1 speed scaling queue with mean inter-arrival time $1/\lambda$ and speeds s_n monotonically approaching $s_\infty \in (0, \infty]$ as $n \to \infty$. Then, with probability 1,*

$$\lim_{x\to 0} \frac{T(x|W=0)}{x} = \frac{1}{s_1} \quad \textit{and} \quad \lim_{x\to\infty} \frac{T(x|W=0)}{x} = \frac{1}{s_\infty}.$$

The intuition for this result is that small jobs receive their whole service while alone in the system; whereas large jobs have a large queue build up behind them, and therefore get served at a faster speed. Thus, the service rates of large jobs exceeds that of small jobs, magnifying the unfairness of non-preemptive policies.

6.3 Open questions

This section has highlighted that dynamic speed scaling has an important drawback – it magnifies unfairness. However, the results summarized here represent only a first step toward understanding this phenomenon. In particular, they only highlight the existence of the phenomenon and say little else.

There are many remaining questions that should be answered. For example, what is the magnitude of the unfairness created? How does this depend on the workload? the speed scaler? and the scheduler? Further, the results summarized above only focus on the largest or smallest jobs. Is it possible to characterize the fairness experienced by intermediate jobs?

Additionally, note that the discussion of fairness in this section has focused on one particular measure of fairness. There are a wide variety of other fairness measures that have been proposed and studied in the classic, static speed model. It would be quite interesting whether the results summarized in this section are corroborated by results for other fairness metrics or not.

Finally, note that this section has focused on only one particular drawback of dynamic speed scaling – unfairness. There are certainly other possible drawbacks, such as increased switching costs (both in terms of added delay and energy), which warrant study. The particular case of switching costs becomes very important when considering scaling the "speed" of a data center by turning servers on and off. This has been the focus of some work, e.g., [57–60]; however there is much yet to be understood.

7 Concluding remarks

In this chapter we have reviewed recent results on algorithms for speed scaling. Our focus has not been on providing a complete overview of the literature, which is large and growing quickly. Rather, our goal has been to illustrate the insight provided by recent analytic results for four fundamental issues in speed scaler design.

We have highlighted that significant progress has been made in understanding speed scaling design and, specifically, in providing insight into these four issues. However, we have also illustrated throughout that there are many remaining open questions that provide interesting directions for future work.

To end, it is interesting to tie together the four issues we have discussed throughout the chapter to highlight one final important direction for future work. The results we have surveyed demonstrate that there is a tension in the design of speed scaling algorithms between three desirable objectives: near-optimal performance, robustness, and fairness. As we have seen, SRPT with dynamic speed scaling is robust and nearly optimal, but is unfair. However, SRPT can be fair and still nearly optimal if gated-static speed scaling is used, but this is not robust. On the other hand, dynamic speed scaling with PS can be fair and robust but, in the worst case, pays a significant performance penalty compared to using SRPT. Thus, among the policies considered in this chapter, *it is possible to achieve any two of near-optimal, fair, and robust — but not all three.* So, the question remains for future research: is it possible for a speed scaling algorithm to be near-optimal, robust, and fair or alternatively, is there a fundamental limitation that prevents this?

References

[1] S. Kaxiras and M. Martonosi, *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool, 2008.

[2] S. W. Son and M. Kandemir, "Energy-aware data prefetching for multi-speed disks," in *Proc. Conference on Computing Frontiers*, 2006.

[3] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase, "Balance of power: dynamic thermal management for internet data centers," *IEEE Internet Computing*, vol. 9, no. 1, pp. 42 – 49, 2005.

[4] S.-L. Kim, Z. Rosberg, and J. Zander, "Combined power control and transmission rate selection in cellular networks," in *Proc. Vehicular Technology Conference*, 1999.

[5] "Intellipower." [Online]. Available: http://www.storagereview.com/1000.sr

[6] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, 1995, pp. 374–382.

[7] N. Bansal, H.-L. Chan, K. Pruhs, and D. Katz, "Improved bounds for speed scaling in devices obeying the cube-root rule," in *Automata, Languages and Programming*, 2009, pp. 144–155.

[8] K. Pruhs, P. Uthaisombut, and G. Woeginger, "Getting the best response for your erg," in *Scandinavian Worksh. Alg. Theory*, 2004.

[9] D. P. Bunde, "Power-aware scheduling for makespan and flow," in *Proc. ACM Symp. Parallel Alg. and Arch.*, 2006.

[10] S. Zhang and K. S. Catha, "Approximation algorithm for the temperature-aware scheduling problem," in *Proc. IEEE Int. Conf. Comp. Aided Design*, Nov. 2007, pp. 281–288.

[11] S. Albers and H. Fujiwara, "Energy-efficient algorithms for flow time minimization," in *Lecture Notes in Computer Science (STACS)*, vol. 3884, 2006, pp. 621–633.

[12] N. Bansal, H.-L. Chan, and K. Pruhs, "Speed scaling with an arbitrary power function," in *Proc. SODA*, 2009.

[13] S. Albers, "Energy-efficient algorithms," *Comm. of the ACM*, vol. 53, no. 5, pp. 86–96, 2010.

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

[15] L. E. Schrage, "A proof of the optimality of the shortest remaining processing time discipline," *Oper. Res.*, vol. 16, pp. 678–690, 1968.

[16] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, pp. 1–39, Mar. 2007.

[17] K. Pruhs, R. van Stee, and P. Uthaisombut, "Speed scaling of tasks with precedence constraints," *Theory of Computing Systems*, vol. 43, no. 1, pp. 67–80, 2008.

[18] D. P. Bunde, "Power-aware scheduling for makespan and flow," *Journal of Scheduling*, vol. 12, no. 5, pp. 489–500, 2009.

[19] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473 – 484, Apr. 1992.

[20] N. Bansal, H.-L. Chan, and K. Pruhs, "Speed scaling with a solar cell," *Theoretical Computer Science*, vol. 410, no. 45, pp. 4580–4587, 2009.

[21] K. Pruhs and C. Stein, "How to schedule when you have to buy your energy," in *Approximation, Randomization, and Combinatorial Optimization (LNCS)*, 2010.

[22] S. Albers, F. Müller, and S. Schmelzer, "Speed scaling on parallel processors," in *Proc. ACM Symp. Parallel Algorithms and Architectures (SPAA)*, 2007.

[23] G. Greiner, T. Nonner, and A. Souza, "The bell is ringing in speed-scaled multiprocessor scheduling," in *Proc. ACM Symp. Parallel Algorithms and Architectures (SPAA)*, 2009.

[24] T.-W. Lam, L.-K. Lee, I. K. K. To, and P. W. H. Wong, "Competitive non-migratory scheduling for flow time and energy," in *Proc. ACM Symp. Parallel Algorithms and Architectures (SPAA)*, 2008.

[25] ——, "Nonmigratory multiprocessor scheduling for response time and energy," *IEEE Trans. Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1527 – 1539, Nov. 2008.

[26] H. Sun, Y. Cao, and W.-J. Hsu, "Non-clairvoyant speed scaling for batched parallel jobs on multiprocessors," in *ACM conference on Computing Frontiers*, 2009.

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

[28] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.

[29] S. V. Hanly, "Congestion measures in DS-CDMA networks," *IEEE Trans. Commun.*, vol. 47, no. 3, pp. 426–437, Mar. 1999.

[30] N. Bansal, K. Pruhs, and C. Stein, "Speed scaling for weighted flow times," in *Proc. SODA*, 2007, pp. 805–813.

[31] N. Bansal, H.-L. Chan, T.-W. Lam, and L.-K. Lee, "Scheduling for speed bounded processors," in *Proc. Int. Colloq. Automata, Languages and Programming*, 2008, pp. 409–420.

[32] T.-W. Lam, L.-K. Lee, I. K. K. To, and P. W. H. Wong, "Speed scaling functions for flow time scheduling based on active job count," in *Proc. Euro. Symp. Alg.*, 2008.

[33] L. H. Andrew, M. Lin, and A. Wierman, "Optimality, fairness, and robustness in speed scaling designs," in *Proc. of ACM Sigmetrics*, 2010.

[34] H.-L. Chan, J. Edmonds, T.-W. Lam, L.-K. Lee, A. Marchetti-Spaccamela, and K. Pruhs, "Nonclairvoyant speed scaling for flow and energy," in *Proc. STACS*, 2009.

[35] S.-H. Chan, T.-W. Lam, L.-K. Lee, H.-F. Ting, and P. Zhang, "Non-clairvoyant scheduling for weighted flow time and energy on speed bounded processors," in *Proceedings of Computing: the Australasian Theory Symposium (CATS)*, 2010.

[36] R. Motwani, S. Phillips, and E. Torng, "Nonclairvoyant scheduling," *Theoret. Comput. Sci.*, vol. 130, no. 1, pp. 17–47, 1994.

[37] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance," *J. ACM (JACM)*, vol. 47, no. 4, Jul. 2000.

[38] J. R. Bradley, "Optimal control of a dual service rate M/M/1 production-inventory model," *European Journal of Operations Research*, vol. 161, no. 3, pp. 812–837, 2005.

[39] T. B. Crabill, "Optimal control of a service facility with variable exponential service times and constant arrival rate," *Management Science*, vol. 18, no. 9, pp. 560–566, 1972.

[40] J. M. George and J. M. Harrison, "Dynamic control of a queue with adjustable service rate," *Oper. Res.*, vol. 49, no. 5, pp. 720–731, 2001.

[41] R. Weber and S. Stidham, "Optimal control of service rates in networks of queues," *Adv. Appl. Prob.*, vol. 19, pp. 202–218, 1987.

[42] F. P. Kelly, *Reversibility and Stochastic Networks*, 1979.

[43] S. Stidham and R. R. Weber, "Monotonic and insensitive policies for control of queues," *Operations Research*, vol. 87, no. 4, pp. 611–625, 1989.

[44] W. Chen, D. Huang, A. Kulkarni, J. Unnikrishnan, Q. Zhu, P. G. Mehta, S. Meyn, and A. Wierman, "Approximate dynamic programming using fluid and diffusion approximations with applications to power management," in *Proc. of CDC*, 2009.

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

[46] N. Bingham, C. Goldie, and J. Teugels, *Regular Variation*. Cambridge University Press, 1987.

[47] M. Lin, A. Wierman, and B. Zwart, "Heavy-traffic analysis of mean response time under shortest remaining processing time," Preprint, 2009. [Online]. Available: http://www.cs.caltech.edu/∼adamw/papers/heavy_traffic_srpt.pdf

[48] F. Dabiri, A. Vahdatpour, M. Potkonjak, and M. Sarrafzadeh, "Energy minimization for real-time systems with non-convex and discrete operation modes," in *Design, Automation and Test in Europe (DATE)*, 2009, pp. 1416 – 1421.

[49] P. Tsiaflakis, Y. Yi, M. Chiang, and M. Moonen, "Fair greening for DSL broadband access," in *GreenMetrics*, 2009.

[50] B. Avi-Itzhak, H. Levy, and D. Raz, "A resource allocation fairness measure: properties and bounds," *Queueing Systems Theory and Applications*, vol. 56, no. 2, pp. 65–71, 2007.

[51] W. Sandmann, "A discrimination frequency based queueing fairness measure with regard to job seniority and service requirement," in *Proc. of Euro NGI Conf. on Next Generation Int. Nets*, 2005.

[52] A. Wierman and M. Harchol-Balter, "Classifying scheduling policies with respect to unfairness in an M/GI/1," in *Proc. of ACM Sigmetrics*, 2003.

[53] A. A. Kherani and R. N. nez Queija, "TCP as an implementation of age-based scheduling: fairness and performance," in *IEEE INFOCOM*, 2006.

[54] I. A. Rai, G. Urvoy-Keller, and E. Biersack, "Analysis of FB scheduling for job size distributions with high variance," in *Proc. of ACM Sigmetrics*, 2003.

[55] A. Wierman, "Fairness and classifications," *Perf. Eval. Rev.*, vol. 34, no. 4, pp. 4–12, 2007.

[56] M. Harchol-Balter, K. Sigman, and A. Wierman, "Asymptotic convergence of scheduling policies with respect to slowdown," *Perf. Eval.*, vol. 49, no. 1-4, pp. 241–256, Sep. 2002.

[57] A. Gandhi, V. Gupta, M. Harchol-Balter, and M. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," in *Proc. of IFIP Performance*, 2010.

[58] S. Irani, S. Shukla, and R. Gupta, "Algorithms for power savings," *ACM Transactions on Algorithms*, vol. 3, no. 4, Nov. 2007.

[59] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE INFOCOM*, 10-15 Apr 2011.

[60] Z. Liu, M. Lin, A. Wierman, L. H. Andrew, and S. Low, "Greening geographical load balancing," Under submission.