International Journal of Systems Science
Vol. 00, No. 00, 00 Month 20xx, 1–15

RESEARCH ARTICLE

Architecture and robustness tradeoffs in speed-scaled queues with application to energy management

Tuan V. Dinh[a1], Lachlan L. H. Andrew[a2] and Yoni Nazarathy[b3]

[a]*Swinburne University of Technology, Australia*; [b]*The University of Queensland, Australia*

(*Received 00 Month 20xx; final version received 00 Month 20xx*)

We consider single-pass, lossless, queueing systems at steady-state subject to Poisson job arrivals at an unknown rate. Service rates are allowed to depend on the number of jobs in the system, up to a fixed maximum, and power consumption is an increasing function of speed. The goal is to control the state dependent service rates such that both energy consumption and delay are kept low. We consider a linear combination of the mean job delay and energy consumption as the performance measure.

We examine both the "architecture" of the system, which we define as a specification of the number of speeds that the system can choose from, and the "design" of the system, which we define as the actual speeds available. Previous work has illustrated, that when the arrival rate is precisely known, there is little benefit in introducing complex (multi-speed) architectures, yet in view of parameter uncertainty, allowing a variable number of speeds improves robustness.

We quantify the tradeoffs of architecture specification with respect to robustness, analysing both global robustness and a newly defined measure which we call local robustness.

Keywords: Parameter uncertainty; robust design; controlled single server queue; speed scaling

1. Introduction

Performance analysis, design and control by means of stochastic queueing models (cf. Wolff 1989) has affected a variety of fields, including not only telecommunications and computing systems but also service engineering, manufacturing, logistics, health-care, road traffic and biological modelling. A typical queueing model abstracts unknown job arrival and service requirements by means of stochastic processes and distributions. The resulting dynamics of queue-length, workload or other performance processes are analysed yielding performance measures that ultimately allow for better design and control of the system at hand. *Design* of the system often refers to an off-line specification of parameters whereas *control* of the system typically refers to an on-line decision making based on state measurements (e.g. setting service speeds). In this paper we shall use a third term, *architecture selection*, referring to the action of deciding what are the design and control parameters that are available to process.

Almost all of the queueing theoretic, performance analysis, design, control and architecture selection literature is based on the underlying assumption that the probability laws of arrival and service processes are precisely known. A few exceptions to this rule are mentioned later in this section. In practice, this assumption is often too strong, especially due to the fact that obtaining precise a-priori parameter estimates is not possible in many settings. Our contribution in this paper is in quantifying the effect of architecture selection on robustness. Here the property of *robustness* refers to the ability of the system to operate in a

[1]Corresponding author. Email: tdinh@swin.edu.au

[2]Email: landrew@swin.edu.au

[3]Email: y.nazarathy@uq.edu.au

ISSN: 0020-7721 print/ISSN 1464-5319 online

DOI: 10.1080/00207721.20xx.CATSid
http://www.informaworld.com

near-optimal manner even when estimates of parameter values are not precise, or even grossly incorrect. As this is generally a vague concept, one of the contributions of this paper is in proposing measures of robustness.

Our analysis focuses on a model that is applicable to computing systems operating in an energy aware speed-scaling environment. The model we consider is an M/G/1-PS queue with state dependent service rates. A Poisson stream of jobs arriving at rate λ is served by a processor sharing (PS) regime that operates as follows: When there are n jobs in the system, each job is served at a rate s_n/n, where $s_0 = 0$. The objective of design and control is to minimise the operating cost, defined as a linear combination of mean delay and mean energy consumption. High service rates generally imply low job delay yet typically incur higher computing energy costs due to the fact that power consumption of devices is often a strictly convex, increasing function of the processing speed.

Wierman et al. (2009) and Andrew et al. (2010), in their studies of similar models and cost objectives, showed that when λ is known, a single speed architecture ($s_1 = s_2 = \ldots$) yields comparable performance to an optimally tailored sequence of speeds. In that sense, a simple architecture can be sufficient. The pitfall mentioned in those studies is that in the more realistic setting in which λ is unknown, multi-speed architectures are generally more robust. More precisely, fix some design arrival rate, λ_d, then a multi-speed architecture where the speeds are optimised for λ_d greatly outperforms a single-speed architecture also optimised for λ_d in cases where the actual arrival rate λ_a differs from λ_d. Andrew et al. (2010) also discussed the service fairness along with optimality and robustness, and argued that only two out of three objectives can be achieved at the same time, although progress towards achieving all three has since been made by Elahi et al. (2012).

The robust multi-speed architecture in Andrew et al. (2010) generally allows each system occupancy, n, to have an arbitrary speed s_n which is not subjected to an overall maximum bound. Such an architecture generally does not come without additional costs of manufacturing, device-footprint, control complexity and other application specific issues. The question then remains: *How many speeds are required in order to allow for robust speed-scaled systems?* Or equivalently: *How does architecture selection affect the robustness of the system to parameter uncertainty?*

In specifying an architecture, one aspect is the number of available speeds, and another is the ability of the control to adapt to the actual load. We consider two regimes: *Fixed Allocation* (FA) and *Adaptive Allocation* (AA). In both regimes, the set of available speeds is fixed at design time, yet the way states are mapped to speeds varies:

- **Fixed Allocation (FA)**: There is a fixed (design-time) mapping, setting s_n to be one of the available speeds. In this case there is no run-time control calculation.
- **Adaptive Allocation (AA)**: It is assumed that the true arrival rate λ_a is accurately estimated at run-time hence allowing s_n to be mapped to one of the available speeds in a way that optimises performance for the given λ_a.

It is clear that adaptive allocation provides greater robustness than fixed allocation, yet in many computing scenarios, this is not without additional design complexity. Note that our adaptive allocation scheme assumes that λ_a is estimated perfectly and that the resulting system is in steady state with that λ_a. One may also consider adaptive control in the sense of estimating λ_a and optimising the control in a time-varying environment, yet this is not the focus of our current work.

In this paper, we examine optimal designs for architectures with a finite available number of speeds subject to a fixed maximum. We compare robustness measures between the AA and FA regimes. We introduce two robustness measures: *global* robustness, applicable when nothing is known about λ_a, and *local robustness* applicable when $\lambda_a \approx \lambda_d$. These measures of system architecture robustness are important in their own right and may be applied to similar models. We also show numerically an interesting counter-intuitive result: Having more speeds under the FA regime can be less globally robust as λ_a becomes sufficiently large.

Related work: There has been extensive work on optimal control of birth-death processes and related models. Low (1974) considered a similar queue but used price policies to control the unknown arrival rate to maximise the long run average expected reward per unit time. George and Harrison (2001) considered the case where the arrival rate is known with state-dependent service rates and developed a numerical technique that imposes no upper bound on the service rate. Ata and Shneorson (2006) considered controlling both arrival and service rates to maximise the system objective, which is the average welfare in their model. Efrosinin and Semenova (2009) looked at a slightly different system where server reliability is uncertain. Control for an M/M/s system was discussed by Serfozo (1981) assuming arrival and service rates can be chosen upon arrival based on the current system occupancy. Jain et al. (2005) investigated a queue-dependent multiprocessor service system which also adopts dynamic service rates. The trade-off between delay and energy was also studied in Goseling et al. (2009) in the context of a system with multiple queues. None of these papers deals with robustness properties in depth.

Research on speed scaling often studies the worst-case performance, rather than average performance. In this context, Chan et al. (2007), Lam et al. (2008) and Bansal et al. (2008) have considered the linear combination of delay and energy with an upper bound on the speed. Unlike the present paper, they make the usual assumption that all speeds below some upper bound are permissible and do not consider the case of constrained architectures to a small finite number of speeds as we do.

Robustness, parameter uncertainty and adaptive control of queues: It appears that the field of performance analysis and control of queues in face of parameter uncertainty is almost unstudied. For illustration, observe the annotated bibliography, Nazarathy and Pollet (2011), containing a comprehensive list of papers in the literature dealing with parameter estimation in queues. There are under 250 such publications, and almost none of them explicitly deals with control in the view of uncertainty. An exception is Jain et al. (2010), dealing with robustness with respect to the probability laws of the underlying stochastic processes using advanced point process theory. A comprehensive survey of robust control methods in the context of operations research is in Lim et al. (2006), yet it appears that the robustness point of view has not yet been fully investigated in queues. Note though that one may view the general line of research of insensitivity (cf. Taylor 2011) as supplying robust results. Yet these are with respect to distributions and typically not with respect to unknown demand rates.

Our contribution is mainly conceptual and numerical, yet we believe it bears significant importance for computer system engineers as well as for future research on design and control of systems in view of parameter uncertainty. The remainder of the paper is organised as follows: Section 2 defines the model and objective function, and surveys related work. Section 3 presents the robustness measure results for both global and local robustness. The results are then summarised in Section 4 where further open questions are put forward.

2. Model and Design Framework

2.1. *Model and notations*

We consider an M/G/1-PS queue with state dependent service rates. Jobs arrive according to a Poisson process with rate $\lambda > 0$. Job sizes are finite mean i.i.d. random variables independent of the arrival process. Without loss of generality we assume the mean job size is 1. Let $Q(t)$ denote the number of jobs in the system at time t. The PS scheme is as follows: At time t if $Q(t) = n$, each job is served at a rate s_n/n, where the sequence of speeds, $0 = s_0 < s_1 \leq s_2 \leq \ldots$ is a result of the design and control of the system.

The insensitivity of the M/G/1-PS, even under speed scaling, (cf. Kelly 1979), allows us to ignore the actual shape of the job-size distribution, as it does not affect the law of the process $Q(t)$. In other words, the occupancy distribution of this queue is the same as that of an M/M/1 queue with the same arrival and state-dependent service rates. We therefore limit ourselves to performance objectives that depend only on the marginal occupancy distribution. The process $Q(t)$ is represented by an irreducible continuous time birth-death process on the state space $\{0, 1, \ldots\}$; see for example Norris (1997). We

assume $\lambda < \sup\{s_1, s_2, \ldots\}$ and hence $Q(t)$ is positive-recurrent with a unique stationary distribution, $(\pi_0, \pi_1, \ldots,)$, $\pi_i = \lim_{t\to\infty} P(Q(t) = i)$, satisfying the partial balance equations, $\lambda\pi_i = s_{i+1}\pi_{i+1}$ and $\sum_{i=0}^{\infty} \pi_i = 1$.

In this model, speeds are constrained to be within the set $[0, \mu_{\max}]$. The number of unique speeds is specified by the architecture parameter, $K \in \{0, 1, 2, \ldots\} \cup \infty$. For finite K, the available set of speeds is, $\mathcal{M} = \{0 = \mu_0, \mu_1, \ldots, \mu_K, \mu_{\max}\}$, with $\mu_i \le \mu_{i+1} \le \mu_{\max}$. Hence there are $K+2$ available speeds. If $K = \infty$, any speed within $[0, \mu_{\max}]$ is allowed. We refer to the latter case as *continuum speed* architecture which is equivalent to the multi-speed architecture discussed in Wierman et al. (2009), except that the speeds are now constrained by an upper bound. In this case, we denote s_n by μ_n for simplicity of the notation below.

Crabill (1972) showed that the optimal speeds are non-decreasing, hence the optimal policy would be a threshold policy characterised by thresholds $\theta_1, \ldots, \theta_K$. Thus, for finite K, our policies remain optimal if we assume the speeds are monotonically non-decreasing. The mapping of s_n to $\mathcal{M}$ can be then specified by a non-decreasing sequence of integer thresholds such that $0 = \theta_0 < \theta_1 \le \theta_2 \le \cdots \le \theta_K < \theta_{K+1} = \infty$. For a given queue occupancy $n > 0$, let $J(n) = \max\{k : \theta_k < n\}$ with $J(0) = -1$. Now the speed-scaling mapping is given by: $s_n = \mu_{J(n)+1}$. For example, if $K = 5$, $\theta_3 = 17$ and $\theta_4 = 20$ then $s_{18} = s_{19} = s_{20} = \mu_4$ while $s_{17} = \mu_3$ and $s_{21} = \mu_5$.

The performance metric we consider is the average running cost per unit time. The running cost of a single job consists of two parts: the sojourn time in the system (T_{waiting}) and the energy consumed by processing it (E). Let Z/λ denote the running cost for a single job. Then $Z/\lambda = \beta T_{\text{waiting}} + E$. The average running cost per job is then $\mathbb{E}[Z]/\lambda = \beta\mathbb{E}[T] + \mathbb{E}[E]$. The average running cost per unit time — which we will henceforth refer to as simply "cost" — is then achieved by multiplying both sides by λ and applying Little's law (Little 1961) to give

$$z = \mathbb{E}[Z] = \beta\mathbb{E}[N] + \mathbb{E}[P_N], \tag{1}$$

where P_n denotes the power consumption rate when the occupancy is n. This objective has been studied previously in both the stochastic context (George and Harrison 2001, Wierman et al. 2009) and in worst-case contexts (Pruhs et al. 2008). The parameter β indicates the relative cost of delay. This can be omitted by the appropriate choice of units, but we retain it to emphasise that the relative weights given to N and P_n are problem specific. Often P_n is a strictly convex non-decreasing function of the speed, and we assume that

$$P_n = s_n^{\alpha}, \quad \alpha > 1. \tag{2}$$

For a given architecture specification, the design variables can be cast as the vectors

$$\mu = (\mu_1, \ldots, \mu_K), \quad \theta = (\theta_1, \ldots, \theta_K),$$

which may be taken to be infinite vectors if $K = \infty$. Then with β and α fixed, the cost can be written as a function — denoted $z_K(\mu, \theta, \lambda)$ — of the architecture K, the design variables and the load. Let $\rho_i = \lambda/\mu_i$ for $i = 1, \ldots, K+1$, let $\rho_i^{\bullet} = \prod_{j=1}^{i} \rho_j^{\theta_j - \theta_{j-1}}$ for $i = 1, \ldots, K$, let $\mathcal{I} = \{i : i \in \{1, 2, \ldots, K+1\}, \rho_i \ne 1\}$ and let $\overline{\mathcal{I}} = \{i : i \in \{1, 2, \ldots, K+1\}, \rho_i = 1\}$. Then $z_K(\mu, \theta, \lambda)$ is

$$\pi_0 \left[\sum_{i\in\mathcal{I}} \rho_{i-1}^{\bullet} \Big(\beta \frac{(\theta_{i-1} - \theta_i \rho_i^{\theta_i - \theta_{i-1}})(1-\rho_i) + (\rho_i - \rho_i^{1+\theta_i-\theta_{i-1}})}{(1-\rho_i)^2} + \frac{\rho_i - \rho_i^{\theta_i - \theta_{i-1}+1}}{1-\rho_i} \mu_i^{\alpha} \Big) \right. \tag{3}$$

$$\left. + \sum_{i\in\overline{\mathcal{I}}} \rho_{i-1}^{\bullet} \Big(\frac{\theta_i(\theta_i - 1) - \theta_{i-1}(\theta_{i-1} - 1)}{2} + (\theta_i - \theta_{i-1})\mu_i^{\alpha} \Big) \right],$$

with the normalising constant

$$\pi_0 = \left[\sum_{i \in \mathcal{I}} \rho_{i-1}^{\bullet} \frac{1 - \rho_i^{\theta_i - \theta_{i-1}}}{1 - \rho_i} + \sum_{i \in \overline{\mathcal{I}}} \rho_{i-1}^{\bullet} (\theta_i - \theta_{i-1}) \right]^{-1}. \tag{4}$$

This follows from a straightforward (yet tedious) computations as detailed in Appendix A. Observe that in practical situations, the right hand summations (over $\overline{\mathcal{I}}$), in both (3) and (4), remain empty.

2.2. *Design Framework*

In our framework the design variables are optimised for a pre-determined arrival rate, $\lambda_d < \mu_{\max}$ ("d" stands for design), yet at runtime there is often another arrival rate $\lambda_a < \mu_{\max}$ ("a" stands for actual), where typically $\lambda_d \neq \lambda_a$. For $K < \infty$, in the *Fixed Allocation* (FA) case, let $\mu^*_{FA}(\lambda_d)$ and $\theta^*_{FA}(\lambda_d)$ denote the optimising design variables μ and θ of

$$\min_{\mu,\theta} \; z_K(\mu, \theta, \lambda_d),$$

subject to the coordinates of μ and θ being ordered. Hence given a *design assumption* of λ_d, the optimal design would be $\left(\mu^*_{FA}(\lambda_d), \theta^*_{FA}(\lambda_d)\right)$.

In the *Adaptive Allocation* (AA) case, use the fixed component $\mu^*_{FA}(\lambda_d)$ as above and consider the optimisation

$$\min_{\theta} \; z_K(\mu^*_{FA}(\lambda_d), \theta, \lambda_a).$$

Denote the optimiser as $\theta^*_{AA}(\lambda_d, \lambda_a)$. Hence given a *design assumption* of λ_d, the optimal design remains $\mu^*_{FA}(\lambda_d)$ as above, and further based on actual measurements of λ_a, the optimal control is $\theta^*_{AA}(\lambda_d, \lambda_a)$.

For a given architecture, solving the fixed allocation design problem or the adaptive allocation control problem involves optimisation of $z_K(\cdot)$. For $K < \infty$ we have implemented the optimisation using a Gauss-Seidel method with a local-search refinement. In case of $K = \infty$ we use dynamic-programming. More details are in Appendix B.

Note that in the continuum speed case ($K = \infty$) under Fixed Allocation, the thresholds are $\theta_i = i$, which allows (3)–(4) to be represented as

$$z_K(\mu, \theta, \lambda) = \sum_{n=0}^{\infty} (\beta n + \mu_n^{\alpha}) \pi_n = \sum_{n=0}^{\infty} \frac{(\beta n + \mu_n^{\alpha}) \lambda^n}{\prod_{i=1}^{n} \mu_i} \Bigg/ \left(1 + \sum_{n=1}^{\infty} \frac{\lambda^n}{\prod_{i=1}^{n} \mu_i} \right). \tag{5}$$

2.3. *Practical Implication*

The foregoing model was motivated by the study of a multitasking operating system on a processor with typical power saving features. Multitasking operating systems allow multiple jobs to be run in parallel. A simple but reasonable model is to treat the sharing policy as processor sharing. Hence, we model the dynamics as an M/G/1 processor sharing queue with a constant arrival rate λ and occupancy-dependent service rate s_n in the state n. As explained in Wierman et al. (2009), processor sharing is insensitive to the job size distribution even when the service rate depends on the occupancy. This greatly simplifies the evaluation of any performance metric depending only on this distribution, as considered here.

We are specifically interested in the case that speed variation is achieved using dynamic voltage and frequency scaling (DVFS) in a CMOS (complementary metal-oxide-semiconductor) integrated circuit. The processing speed of such systems is often assumed to be proportional to the clock frequency f,

although issues such as cache performance have an increasing influence. The power consumption is often modelled as proportional to the cube of the clock frequency. This is because the dynamic power in CMOS is proportional to $V^2 f$ where V is the scalable voltage, which was historically scaled proportional to clock frequency, as mentioned in Chandrakasan et al. (1992). However, some CMOS devices do not scale V this way, as power is closer to quadratic in speed f (Wierman et al. 2009).

DVFS is becoming less important because Dennard's scaling law (Dennard et al. 1974) is no longer used, resulting in less flexibility in scaling voltages. However, future architectures may still result in power that is super-linear in speed. Multiple processing core architectures are increasingly adopted by current processor designs, replacing DVFS (Kumar et al. 2005). In multicore architectures, speed scaling of paralleling workloads can be achieved by turning cores on and off. If cores are heterogeneous, this can also result in power P_n being a convex function of the speed as considered here, according to Hofstee (2005).

Prior work such as George and Harrison (2001) or Wierman et al. (2009) assumed no upper limit of system speed. However, as technology limits always place an upper bound on the processing speed, we constrain all speeds s_n to be in the range $[0, \mu_{\max}]$ for some given constant $\mu_{\max}$. Since our interest is to see the effect of the number of speeds, we constrain the total number of speeds to be $K + 2$, with speeds from $\mathcal{M}$. Our decision variables include the K speeds μ_1 to μ_K.

In the CMOS situation we are modelling, different decision variables are decided on at different phases in the design process. We assume the μ_i are fixed properties of a given piece of hardware. We assume they must be chosen when that chip is designed, before it is known what load it will be subjected to. The thresholds θ_i are typically implemented in software in the operating system, and can be determined later based on a real-time estimate of the load, or of β. In contrast, K might determine the size of a software-visible register that stores the current speed, or might determine the number of external pins required to signal this information; for compatibility reasons, this information may need to be held constant over an entire family of chips sharing the same instruction set architecture (ISA). For that reason, we concern ourselves more with robustness resulting from the choice of K as opposed to robustness of the individual μ_i.

Note that this objective places no cost on switching between speeds, as this switching cost is relatively small in comparison with the energy cost or the delay cost. This assumption is made due to the fact that the typical switching time between speeds is in the order of 10^{-4} seconds as shown in an experiment by Hartman (2008), while the job service time is often in the order of seconds.

3. Quantifying Robustness

In this section we present the main contribution of our paper: quantifying the effects of parameter uncertainty due to discrepancies between λ_d and λ_a for various architecture specifications. We consider the *error metrics* $\Delta_{FA}(\cdot)$ and $\Delta_{AA}(\cdot)$ for fixed allocation and adaptive allocation respectively:

$$\Delta_{FA}(\lambda_a, \lambda_d, K) = z_K(\mu^*_{FA}(\lambda_d), \theta^*_{FA}(\lambda_d), \lambda_a) \quad - z_\infty(\mu^*_{FA}(\lambda_a), \theta^*_{FA}(\lambda_a), \lambda_a). \tag{6}$$

$$\Delta_{AA}(\lambda_a, \lambda_d, K) = z_K(\mu^*_{FA}(\lambda_d), \theta^*_{AA}(\lambda_d, \lambda_a), \lambda_a) - z_\infty(\mu^*_{FA}(\lambda_a), \theta^*_{FA}(\lambda_a), \lambda_a). \tag{7}$$

Both metrics compare the cost under a given architecture specification and design for a given λ_d to the minimal possible cost, $z_\infty(\mu^*_{FA}(\lambda_a), \theta^*_{FA}(\lambda_a), \lambda_a)$, for this λ_d, attained with $K = \infty$ and $\lambda_d = \lambda_a$. The metrics capture the distance to the optimal cost indicating the effect of parameter uncertainty on performance: a low Δ value implies "more robustness".

We have calculated the error metrics for a variety of combinations of λ_a, λ_d and architecture specifications K for "FA" and "AA" cases. In all cases we used $\mu_{\max} = 1$, and $\alpha = 3$. Our results are best presented by fixing λ_d at one of several values: 0.3 (design for light load), 0.5 (design for medium load) or 0.7 (design for moderately heavy load). We considered two values of the relative cost of delay, β: 0.01 and 0.1. The former places more importance on energy savings than the latter. We then varied λ_a over a fine

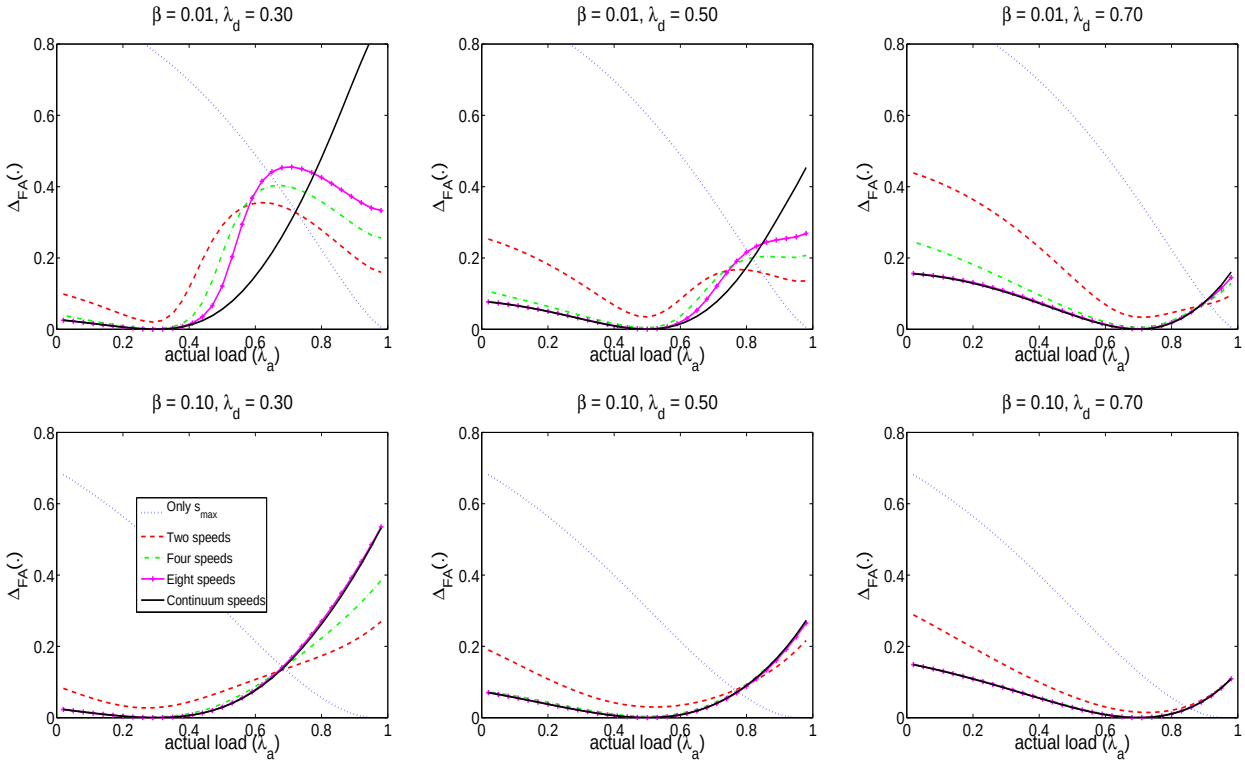

Figure 1. Fixed Allocation (FA): The error metric, or the distance to the optimal cost using policy optimised for load λ_d with the reality of load λ_a and fixed thresholds.

grid within $(0, 1)$ and evaluated the error metrics for each value. Note that each such evaluation requires carrying out two optimisations, one for $z_K(\cdot)$ and one for $z_\infty(\cdot)$, as outlined in Appendix B.

Since we are looking at the error metric over the whole possible range $\lambda_a \in (0, \mu_{\max})$, assessment of Δ over that range may be viewed as measuring *global robustness*. This is in contrast to the *local robustness* defined later in this section.

3.1. *Global Robustness in Fixed Allocation and Adaptive Allocation Designs*

The Fixed Allocation (FA) case: Figure 1 shows the metric $\Delta_{FA}(\lambda_a, \lambda_d, K)$ for increasing architectures K from $K = 0$ ("Only $\mu_{\max}$") and $K = 1$ ("Two speeds") up to a continuum of speeds, for a system that uses both the speeds and the thresholds optimised for λ_d. Note that all examples include $\mu_{\max}$ as one (the largest) of the available speeds; this is in contrast to the single-speed "gated static" policy studied in Wierman et al. (2009), in which the single speed was optimised for the design load, and the "continuum" case could use arbitrarily high speeds as the occupancy increase.

Recall that Wierman et al. (2009) observed substantially greater robustness when using a system designed with no constraints on the number of speeds than when using a system with a single optimally chosen speed. This was explained by the fact that, if the actual load is much higher than the design load, then the mean occupancy will be higher; since the speed is an increasing function of the occupancy, this increased occupancy causes the average speed used to be higher, as is appropriate for a higher load.

It is natural to expect that a similar conclusion would apply to the model studied here, in which the system is forced to include the maximum speed $\mu_{\max}$ as one available speed, and that increasing the number of available speeds will monotonically increase the robustness. When the actual load is low, this is indeed the case. In fact, having even a single additional speed ("Two speeds") yields most of the benefit obtained from having a continuum of speeds.

However, as the actual load approaches $\mu_{\max}$, designs with more available speeds actually become

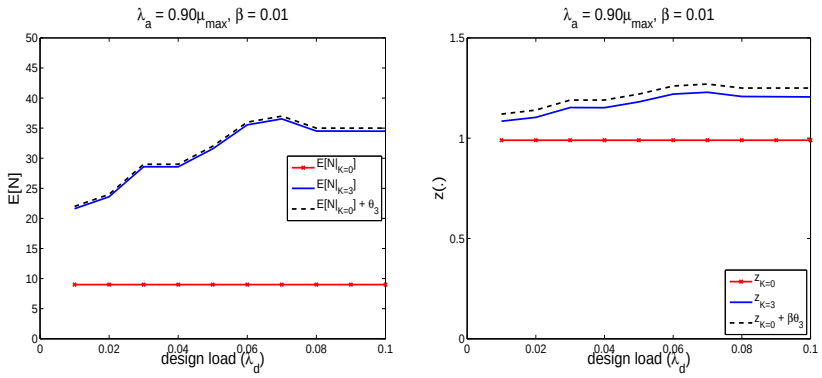

Figure 2. System occupancies and general costs when actual load is very close to μ_{max} ($\lambda_a = 0.9\mu_{max}$). This figure compares costs for $K = 0$ and $K = 3$ with low design loads

monotonically *less* robust, in the sense that the penalty $\Delta_{FA}(\lambda_a, \lambda_d, K)$ for mis-estimating the load increases. This is most apparent when the design load is low, and little weight (β) is given to delay.

This paradox is explained by noting that fixed thresholds were used. When the load is almost $\mu_{\max}$, the average processing speed must also be almost $\mu_{\max}$. Thus the system in which $\mu_{\max}$ is the only speed available is optimal. If more speeds are available, then the system will need to maintain a higher occupancy N to cause speed $\mu_{\max}$ to be used.

If the second highest speed μ_{K-1} is much less than μ_K, then the occupancy will be increased by almost θ_K, increasing the cost by almost $\beta\theta_K$, as illustrated in Figure 2. This figure shows the system occupancy and cost at a very high λ_a ($\lambda_a = 0.98\mu_{max}$) for small λ_d so that the second highest speeds are relatively small. The mean occupancies and costs are computed for two cases: (a) only use μ_{max} ($K = 0$) and (b) optimal settings for designed load λ_d for a four speed system ($K = 3$). When only running with μ_{max}, the costs are constant regardless of λ_d. However, for multiple speed system, the occupancy is increased by roughly θ_K and the cost is increased by roughly $\beta\theta_K$.

The Adaptive Allocation (AA) case: Results for adaptive allocation are in Figure 3. Note that here the robustness monotonically increases as the number of available speeds increases, as intuition suggests. As expected, the penalty for mis-estimating the load is small near the design load, λ_d. However, it also tends to 0 when the actual load approaches $\mu_{\max}$, even though this is far from λ_d. This is again because at such high loads, the system must nearly always operate at speed $\mu_{\max}$, which is implemented in all designs regardless of λ_d. The resulting bi-modality raises the interesting question of what speeds are "maximally robust" in the sense of minimising the maximum over λ_a of $\Delta_{AA}(\lambda_a, \lambda_d, K)$. This optimum depends on β, but Figure 3 suggests that it corresponds roughly to designing the system to work at a load of about $\mu_{\max}/3$. This is in contrast to the common practice of designing CPUs for desktop computers with a second speed that is very close to the maximum speed.

Comparison of FA and AA: Adjusting the thresholds θ_i occurs on a much slower time-scale than adjusting the actual speeds. Implementing dynamic thresholds incurs a non-negligible additional cost to software development. To decide whether or not to implement dynamic thresholds, it is important to quantify how much benefit it provides over using dynamic (state-dependent) speeds with static thresholds. Figure 4 shows this improvement for several parameter combinations. This suggests that dynamically adjusting the thresholds may not be justified unless the design load is well below the maximum load for which the system can be stable.

3.2. *Quantitative measure of "Local" Robustness*

When the load is known approximately but not perfectly, the designer may still be interested in the sensitivity of costs to minor deviations of λ_a from λ_d. For this we introduce the notion of local robustness defined in terms of the curvature of the cost penalty Δ. More precisely, define the *local robustness* of the

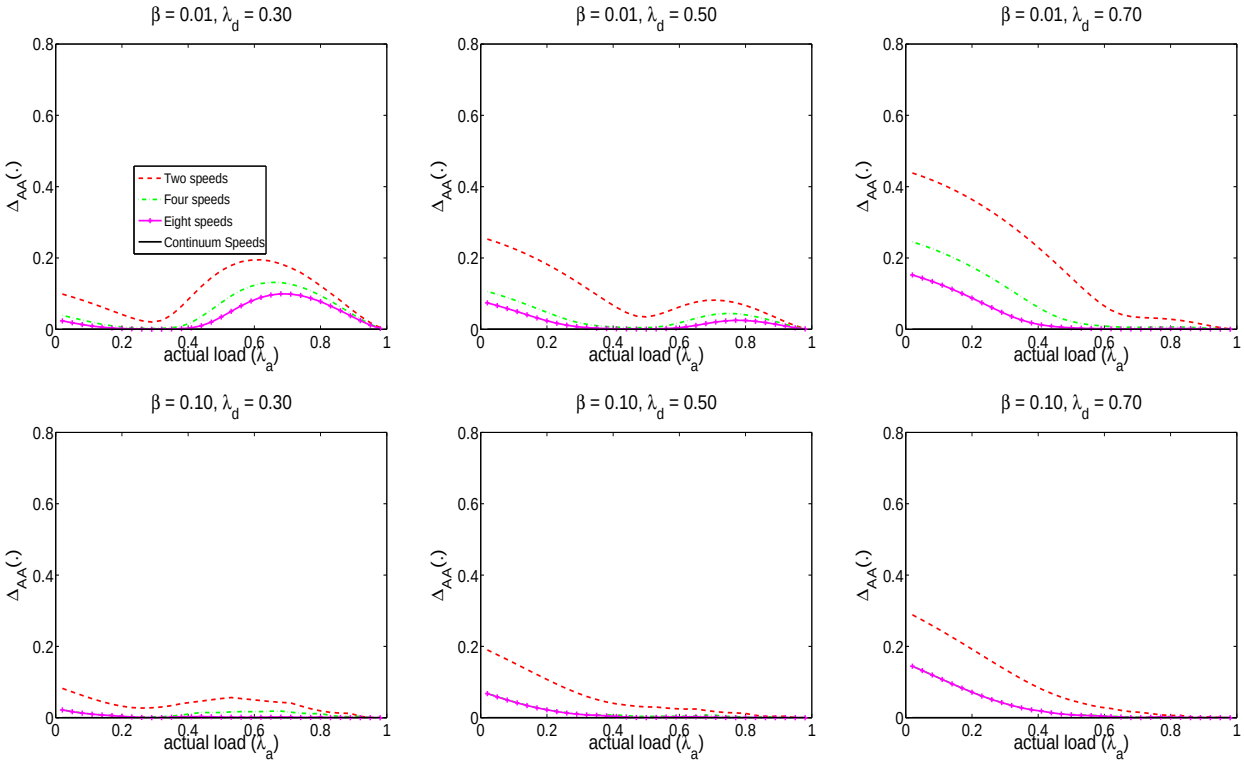

Figure 3. Adaptive Allocation (AA): Distance to the optimal cost using policy optimised for load λ_d with the reality of load λ_a and adjustable thresholds.

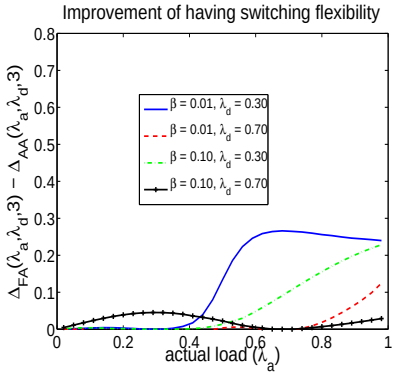

Figure 4. Benefits in cost reduction of having adjustable thresholds at runtime.

system with $K+2$ speeds optimised for a load of λ_d as

$$R(\lambda_d, K) = \left[\frac{d^2 \Delta_{FA}(\lambda_a, \lambda_d, K)}{d\lambda_a^2} \bigg|_{\lambda_a = \lambda_d} \right]^{-1}. \tag{8}$$

It is sensible to use the second derivative here instead of the first, since the first derivative is approximately zero at the design load; if Δ_{FA} were defined as $z_K(\mu^*_{FA}(\lambda_d), \theta^*_{FA}(\lambda_d), \lambda_a) - z_K(\mu^*_{FA}(\lambda_a), \theta^*_{FA}(\lambda_a), \lambda_a)$, it would be exactly zero.

For a given setting, we estimate R_{K,λ_d} by taking a least square fit (Boyd and Vandenberghe 2004) of a parabola to the difference in costs over 40 equally spaced points in the interval $[\lambda_d - \mu_{max}/50, \lambda_d + \mu_{max}/50]$. The value of $\mu_{max}/50$ was chosen to be large enough to avoid numerical errors, yet small

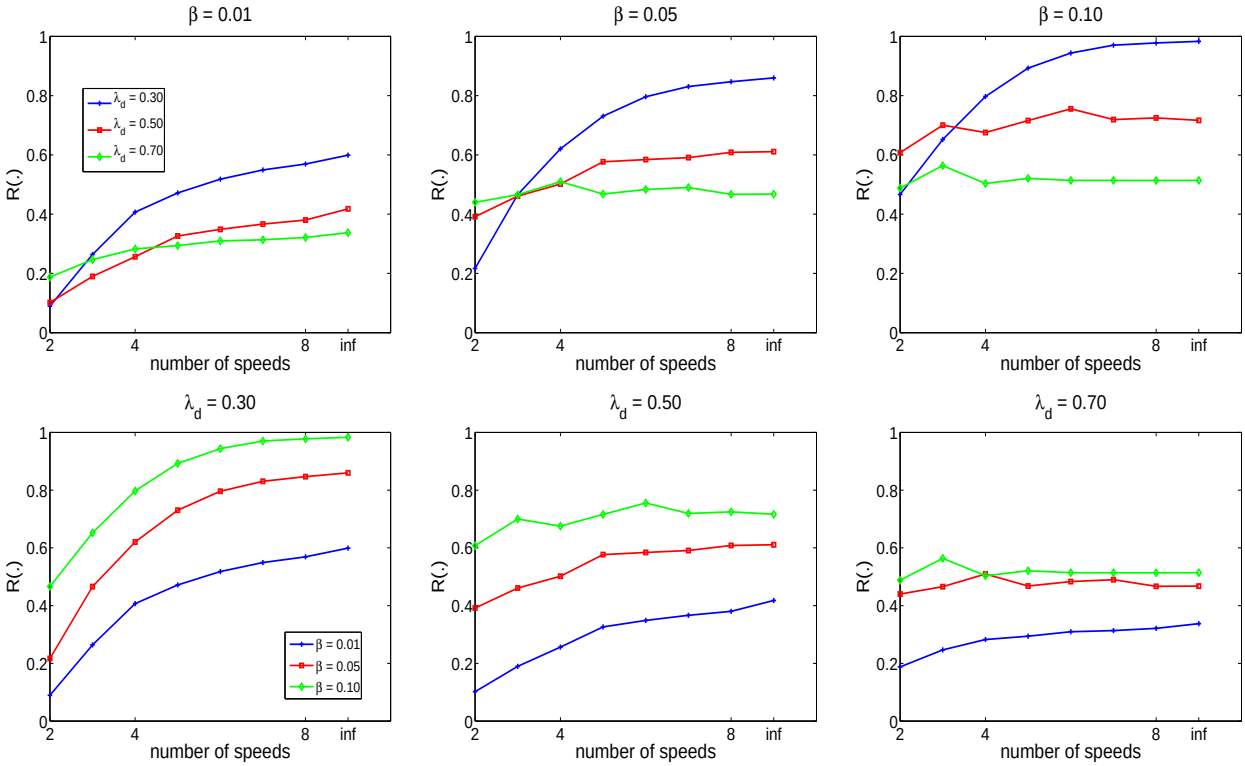

Figure 5. Local robustness of a fixed allocation (FA) as function of the number of speeds: R(.)

enough allowing an accurate estimate of the second derivative. Note that such numerical evaluation of the local robustness is computationally intensive, since each point used for the least squares fit requires solving an optimisation.

As in the global robustness cases described above, we consider light load, medium load or moderately heavy load. We also consider three values of β: 0.01 and 0.1 as analysed for global robustness as well as an intermediate value of 0.05. We generally expect that allowing more speeds in the architecture specification will make the system more "locally robust". The question is how quickly local robustness improves with the complexity of the architecture. Answers are in Figure 5.

To identify how many speeds are enough, we seek the "knee" of the curve of $R(\cdot)$ against the number of speeds $K + 1$. In the curve we label the axes with powers of 2, since this determines the number of bits required by the instruction set architecture (ISA). Note that the final point on the axis, labelled "inf", is the continuum case when there is no constraint on the speeds. The results show that the knee is most pronounced for the smallest λ_d and appears to be at around six speeds ($K = 5$). Indeed, eight speed ($K = 7$) gives robustness almost indistinguishable from that of the continuum case. This suggests that it is sufficient that the ISA allocate three bits to the register specifying the current speed.

In general, it appears that the local robustness is increasing in λ_d when there are few speeds, and decreasing in λ_d when there are many speeds. Similarly, the local robustness tends to increase as β increases.

4. Conclusion and Outlook

This paper has identified and explained an anomaly in the performance of queues with speed scaling optimised for an inaccurate estimate of the load. This effect causes the performance of such a speed scaler operating at high load to degrade as the number of available speeds increases. Specifically, this is due to inappropriate choice of speed at runtime, making it difficult to quantify the improvement in

robustness due to such an increase.

However, when the error in the load estimate is low, the performance does monotonically improve as the number of levels increases. This suggests that it may be possible to define a meaningful measure of "local robustness" which could be used to determine the number of speeds required, as we have proposed in this paper.

This work assumed very simplified models. For example, the speed scaling function P_n only considers active power in CMOS circuits, whereas leakage power is becoming an increasing fraction of total power consumption (Kaxiras and Martonosi 2008). The form of P_n is also suitable for cases where the speed of processing is proportional to the clock speed, whereas current processors are heavily influenced by delays due to memory access. Nevertheless, we expect the qualitative insight to hold more generally: If the speed selection algorithm is based on inaccurate parameter estimates, then having a wider range of speeds available may be counter-productive.

5. Acknowledgement

This work was funded in part by Australian Research Council (ARC) grant FT0991594.

Appendix A. Objective Cost

The analysis in this appendix follows the notation and assumptions of Section 2.

Proposition A.1: *For $K < \infty$, when it exists, the stationary distribution $\left(\pi_0, \pi_1, ...\right)$ of $Q(t)$ is*

$$\pi_n = \pi_0 \rho^{\bullet}_{J(n)} \rho^{n-\theta_{J(n)}}_{J(n)+1}, \tag{A1}$$

for $n \geq 1$ and,

$$\pi_0 = \left[\sum_{i \in \mathcal{I}} \rho^{\bullet}_{i-1} \frac{1-\rho_i^{\theta_i - \theta_{i-1}}}{1-\rho_i} + \sum_{i \in \bar{\mathcal{I}}} \rho^{\bullet}_{i-1} (\theta_i - \theta_{i-1}) \right]^{-1}. \tag{A2}$$

Further the expected cost $z = \mathbb{E}[Z]$ is represented by equation (3).

Proof: From the partial balance equations, the stationary probability of being in state n is

$$\pi_n = \pi_{\theta_{i-1}} \rho_i^{n-\theta_i} \qquad n \in \{\theta_{i-1}+1, \theta_i\}$$

Induction on n yields (A1). Now π_0 is obtained by $1 = \sum_{i=0}^{\infty} \pi_i$:

$$\begin{aligned} \pi_0 &= \left[\sum_{n=0}^{\theta_1} \rho_1^n + \sum_{n=\theta_1+1}^{\theta_2} \rho_1^{\theta_1} \rho_2^{n-\theta_1} + \sum_{n=\theta_2+1}^{\theta_3} \rho_1^{\theta_1} \rho_2^{\theta_2-\theta_1} \rho_3^{n-\theta_2} + \cdots + \sum_{n=\theta_K+1}^{\theta_{K+1}} \rho_1^{\theta_1} \rho_2^{\theta_2-\theta_1} \ldots \rho_{K+1}^{n-\theta_K} \right]^{-1} \\ &= \left[\sum_{n=0}^{\theta_1-1} \rho_1^n + \rho_1^{\bullet} \sum_{n=\theta_1}^{\theta_2-1} \rho_2^{n-\theta_1} + \rho_2^{\bullet} \sum_{n=\theta_2}^{\theta_3-1} \rho_3^{n-\theta_2} + \cdots + \rho_K^{\bullet} \sum_{n=\theta_K}^{\theta_{K+1}-1} \rho_{K+1}^{n-\theta_K} \right]^{-1}. \end{aligned}$$

The summations above simplify due to

$$\sum_{n=\theta_{i-1}}^{\theta_i-1} \rho_i^{n-\theta_{i-1}} = \begin{cases} \frac{1-\rho_i^{\theta_i-\theta_{i-1}}}{1-\rho_i} & \rho_i \neq 1, \\ \theta_i - \theta_{i-1}, & \rho_i = 1 \end{cases} \tag{A3}$$

yielding (A2). The expected cost is $z = \beta\mathbb{E}[N] + \mathbb{E}[P_N] = \sum_{i=0}^{\infty}(\beta n + P_n)\pi_n$, in which

$$\mathbb{E}[N] = \pi_0 \sum_{i=1}^{K+1} \rho_{i-1}^{\bullet} \sum_{n=\theta_{i-1}}^{\theta_i-1} n\rho_i^{n-\theta_{i-1}},$$

and,

$$\mathbb{E}[P_N] = \pi_0 \sum_{i=1}^{K+1} \rho_{i-1}^{\bullet} \mu_i^{\alpha} \sum_{n=\theta_{i-1}+1}^{\theta_i} \rho_i^{n-\theta_{i-1}}.$$

Note that

$$\sum_{n=\theta_{i-1}}^{\theta_i-1} n\rho_i^{n-\theta_i} = \begin{cases} \frac{(\theta_{i-1} - \theta_i\rho_i^{\theta_i-\theta_{i-1}})(1-\rho_i) + (\rho_i - \rho_i^{1+\theta_i-\theta_{i-1}})}{(1-\rho_i)^2}, & \rho_i \neq 1, \\ \frac{1}{2}\left[\theta_i(\theta_i-1) - \theta_{i-1}(\theta_{i-1}-1)\right], & \rho_i = 1. \end{cases} \tag{A4}$$

Equation (3) can be obtained by substituting (A3) and (A4) into the expression for $\mathbb{E}[N]$ and $\mathbb{E}[P_N]$. □

Appendix B. Numerical Optimization Algorithms

B.1. $K = 0$

When $K = 0$, the system runs only with the maximum service rate μ_{max} for all s_i except for $i = 0$. In this case, the system is simply an M/M/1 queue with arrival rate λ and service rate μ_{max} and the cost is deterministic for a given λ:

$$z = \beta\mathbb{E}[N] + \mathbb{E}[P_N] = \beta\sum_{n=0}^{\infty} n\pi_n + \sum_{n=1}^{\infty} \pi_n\mu_{max} = \beta\frac{\lambda}{\mu_{max}-\lambda} + (1-\pi_0)\mu_{max}^{\alpha} = \beta\frac{\lambda}{\mu_{max}-\lambda} + \lambda\mu_{max}^{\alpha-1}. \tag{B1}$$

B.2. $1 \leq K < \infty$

Recall that we want to solve:

$$\min_{\mu,\theta} z_K(\mu, \theta, \lambda_d, \beta, \alpha).$$

It is known that the optimal policy is of threshold type as proved in Crabill (1972) and mentioned in George and Harrison (2001). Thus, we enforce that μ is monotonically non-decreasing. We can then use the Gauss-Seidel method (or block coordinate decent, described in (Bertsekas 1999, section 2.7)) to numerically find the optimal settings (speeds and thresholds).

In general, the search starts with an initial feasible guess (μ^0, θ^0). The search was performed in $2K$ dimensions, each dimension being an element of speed vector μ or threshold vector θ. Each iteration

consists of the search for new values for all $2K$ variables, and is divided into $2K$ phases. At each phase, it is a single dimension optimisation, holding the most recent computed values of other variables. For each iteration, searches were performed for all speeds first and then thresholds. The iterations continue until the maximum change between (μ^k,θ^k) and (μ^{k+1},θ^{k+1}) is less than $\epsilon = 10^{-10}$ indicating that (μ^k,θ^k) has converged to (μ^*,θ^*). The searches for speeds and thresholds were performed by different techniques.

In particular, phases $i = 1,\ldots,K$ of iteration k calculate

$$\mu_i^k = \underset{\mu_{i-1}^k \le \mu_i^k \le \mu_{i+1}^{k-1}}{\operatorname{argmin}} z_K(\mu_1^k,\ldots,\mu_{i-1}^k,\mu_i^k,\mu_{i+1}^{k-1},\ldots,\mu_K^{k-1};\theta_1^{k-1},\ldots,\theta_K^{k-1}) \tag{B2}$$

This phase uses the new values that were found for μ_j^k $(i < j)$ in this iteration while it retains μ_j^{k-1} $(i > j)$ from the previous iteration. Using the assumption that the cost function is uni-modal (as numerical evidence suggests it is), golden section search was used to calculate μ_i^k. The search range was bounded between the most recent adjacent speeds $[\mu_{i-1}^k,\mu_{i+1}^{k-1}]$, with the convention that $\mu_{K+1} = \mu_{max}$ and $\mu_0 = 0$.

Phases $i + K = K + 1,\ldots,2K$ calculate

$$\theta_i^k = \underset{\theta_{i-1}^k < \theta_i^k < \theta_{i+1}^{k-1}}{\operatorname{argmin}} z_K(\mu_1^k,\ldots,\mu_K^k;\theta_1^k,\ldots,\theta_{i-1}^k,\theta_i^k,\theta_{i+1}^{k-1},\ldots,\theta_K^{k-1}). \tag{B3}$$

Finding θ_i^k cannot use Golden Section search due to the shape of cost as a function of θ_i^k. Numerical errors sometimes cause it to enter an incorrect search range that does not include the global optimum. Since thresholds are naturally discrete, an exhaustive search was instead performed over a truncated state space for such cases. Again, the search range was bounded between $[\theta_{i-1}^k,\theta_{i+1}^{k-1}]$. For the search of θ_K^k, a sufficiently large θ_{K+1}^k to be considered infinite was chosen.

Unfortunately, the discrete nature of θ_i^k sometimes causes the convergence to a local optimum. To overcome this problem, we introduced a heuristic to refine the results. When Gauss-Seidel iterations returns $(\overline{\mu},\overline{\theta})$ and an estimate $\overline{z}$ of the optimal cost, an exhaustive search was then performed for all the neighbour points of $\overline{\theta}$. We used each point as an initial guess for new Gauss-Seidel iterations. If any of these searches returned a cost that is less than $\overline{z}$, the process would be repeated with this new point at the centre of the search region. In all of these searches, $\overline{\mu}$ was kept unchanged. This refinement process was repeated until it found a point with a lower of $\overline{z}$ than all its neighbours. The final $(\overline{\mu},\overline{\theta})$ and $\overline{z}$ were taken as (μ^*,θ^*) and z^*.

An experiment was performed to test this heuristic. About a thousand combinations of λ, β and K were chosen. A procedure was performed to each combination with a thousand initial feasible points which were generated randomly. If the heuristic worked then for each combination, it would return the same optimal settings for all the initial guesses. This occurred in 92.8% of cases, which suggests that this search is seldom stuck in poor local optima. For some extreme cases (specifically for λ is very small), there were differences among the optimal settings; however, in these cases the optimal costs were approximately the same, and on average 78.8% of initial feasible points converged to the lowest cost point.

B.3. $K = \infty$

The case $K = \infty$ refers to our *continuum* architecture representing the case where no restriction in number of speeds is applied. Speeds can be chosen freely within $[0,\mu_{max}]$ for each state of system occupancy. A feasible policy $\mu = [\mu_1,\mu_2,\ldots]$ maps the speed $\mu_i \in [0,\mu_{max}]$ to system state i, where system state refers to the number of jobs in the system. The optimal policy μ^* is a feasible policy that minimises the average cost per unit time z. It is a classic infinite horizon control for state dependent M/M/1 queue problem which was numerically solved using dynamic programming with relative value iteration and uniformisation of the service rate.

At iteration k, the *relative value function* for state i as the number of jobs in the system is, according

to Bertsekas (2005),

$$h_k(i) = \min_{\mu\in[0,\mu_{max}]}\left[\frac{z(i,\mu)}{\lambda+\mu_{max}} + \frac{\mu}{\lambda+\mu_{max}}h_{k-1}(i-1) + \frac{\mu_{max}-\mu}{\lambda+\mu_{max}}h_{k-1}(i) + \frac{\lambda}{\lambda+\mu_{max}}h_{k-1}(i+1)\right]$$
$$- \min_{\mu\in[0,\mu_{max}]}\left[\frac{z(S,\mu)}{\lambda+\mu_{max}} + \frac{\mu}{\lambda+\mu_{max}}h_{k-1}(S-1) + \frac{\mu_{max}-\mu}{\lambda+\mu_{max}}h_{k-1}(S) + \frac{\lambda}{\lambda+\mu_{max}}h_{k-1}(S+1)\right]$$

where $z(i,\mu)$ is the uniformised cost at state i with speed μ, given by $z(i,\mu) = \dfrac{\beta i+\mu^{\alpha}}{\lambda+\mu_{max}}$, and $S = 0$ is a reference state. As $k\to\infty$, h_k converges to the h^* satisfying

$$z^* + h^*(i) = \min_{\mu\in[0,\mu_{max}]}\left[\frac{z(i,\mu)}{\lambda+\mu_{max}} + \frac{\mu}{\lambda+\mu_{max}}h^*(i-1) + \frac{\mu_{max}-\mu}{\lambda+\mu_{max}}h^*(i) + \frac{\lambda}{\lambda+\mu_{max}}h^*(i+1)\right],$$

where the optimal cost is

$$z^* = \min_{\mu\in[0,\mu_{max}]}\left[\frac{z(S,\mu)}{\lambda+\mu_{max}} + \frac{\mu}{\lambda+\mu_{max}}h_{k-1}(S-1) + \frac{\mu_{max}-\mu}{\lambda+\mu_{max}}h_{k-1}(S) + \frac{\lambda}{\lambda+\mu_{max}}h_{k-1}(S+1)\right].$$

With $S = 0$, the optimal cost can be evaluated as

$$z^* = \frac{\mu_{max}}{\lambda+\mu_{max}}h^*(0) + \frac{\lambda}{\lambda+\mu_{max}}h^*(1) \tag{B4}$$

and the optimal speed in state i is the "argmax" of the latest optimisation over μ in calculating $h_k(i)$. Thus, every state has its own speed. We started the iteration with $\{h_0(i)\} = 0$ and performed the search for a truncated state space at each iteration. The iteration stopped when $|h_k - h_{k-1}| < \epsilon = 10^{-10}$.

References

Andrew, L.L.H., Lin, M., and Wierman, A. (2010), "Optimality, fairness and robustness in speed scaling designs," in *Proc. ACM SIGMETRICS*, 14-18 Jun, New York, NY.

Ata, B., and Shneorson, S. (2006), "Dynamic Control of an M/M/1 Service System with Adjustable Arrival and Service Rates," *Manage. Sci.*, 52, 1778–1791.

Bansal, N., Chan, H., Lam, T., and Lee, L. (2008), "Scheduling for speed bounded processors," *Automata, Languages and Programming*, pp. 409–420.

Bertsekas, D.P., *Nonlinear Programming*, Athena Scientific (1999).

Bertsekas, D.P., *Dynamic Programming and Optimal Control*, 3rd ed., Athena Scientific (2005).

Boyd, S., and Vandenberghe, L., *Convex Optimization*, Cambridge University Press (2004).

Chan, H., Chan, W., Lam, T., Lee, L., Mak, K., and Wong, P. (2007), "Energy efficient online deadline scheduling," in *Proc. ACM Symp On Discrete Algorithms*, pp. 795–804.

Chandrakasan, A., Sheng, S., and Brodersen, R. (1992), "Low-power CMOS digital design," *Solid-State Circuits, IEEE Journal of*, 27(4), 473–484.

Crabill, T.B. (1972), "Optilmal control of a service facility with variable exponential service times and constant arrival rate.," *Manage. Sci.*, 18(9), 560 – 566.

Dennard, R., Gaensslen, F., Rideout, V., Bassous, E., and LeBlanc, A. (1974), "Design of ion-implanted MOSFET's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, 9(5), 256–268.

Efrosinin, D., and Semenova, O. (2009), "Optimal control of M/M/1 queueing system with constant retrial rate and non-reliable removable server," in *Ultra Modern Telecommunications Workshops, 2009. ICUMT '09.*, Oct., pp. 1 –6.

Elahi, M., Williamson, C., and Woelfel, P. (2012), "Decoupled Speed Scaling: Analysis and Evaluation," in *Proc. Int. Conf. Quantitative Evaluation of SysTems (QEST)*.

George, J.M., and Harrison, J.M. (2001), "Dynamic Control of a Queue with Adjustable Service Rate," *Oper. Res.*, 49, 720–731.

Goseling, J., Boucherie, R., and van Ommeren, J.K. (2009), "Energy consumption in coded queues for wireless information exchange," in *Workshop on Network Coding, Theory, and Applications, 2009. NetCod '09.*, june, pp. 30 –35.

Hartman, M. (2008), "Processor Energy Savings Through Adaptive Voltage Scaling," *"http://lowpowerdesign.com/article_national_hartman.htm"*.

Hofstee, H.P. (2005), "Power Efficient Processor Architecture and The Cell Processor," in *Proc. High-Performance Computer Architecture*, HPCA '05, Washington, DC, USA: IEEE Computer Society, pp. 258–262.

Jain, A., Lim, A., and Shanthikumar, J. (2010), "On the optimality of threshold control in queues with model uncertainty," *Queueing Systems*, 65(2), 157–174.

Jain, M., Sharma, G.C., and Shekhar, C. (2005), "Processor-shared service systems with queue-dependent processors," *Computers & OR*, 32, 629–645.

Kaxiras, S., and Martonosi, M., *Computer Architecture Techniques for Power-Efficiency*, 1st ed., Morgan and Claypool Publishers (2008).

Kelly, F.P., *"Reversibility and Stochastic Networks"*, Vol. 40, Wiley New York (1979).

Kumar, R., Zyuban, V., and Tullsen, D. (2005), "Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling," in *Proc. Computer Architecture. ISCA '05*, june, pp. 408 – 419.

Lam, T., Lee, L., To, I., and Wong, P. (2008), "Speed scaling functions for flow time scheduling based on active job count," *Algorithms-ESA 2008*, pp. 647–659.

Lim, A., Shantikumar, J., and Shen, Z.M. (2006), "Model Uncertainty, Robust Optimization and Learning," *Tutorials in Operations Research*, pp. 66–94.

Little, J.D.C. (1961), "A Proof for the Queuing Formula: $L = \lambda W$," *Oper. Res.*, pp. 383–387.

Low, D. (1974), "Optimal dynamic pricing policies for an M/M/s queue," *Oper. Res.*, 22, 545–561.

Nazarathy, Y., and Pollet, P.K. (2011), "Parameter estimation in queues: A Bibliography," *"http://www.maths.uq.edu.au/~pkp/papers/Qest/Qest.html"*.

Norris, J., *Markov Chains*, Cambridge University Press (1997).

Pruhs, K., Uthaisombut, P., and Woeginger, G. (2008), "Getting the best response for your erg," *ACM Trans. Algorithms*, 4(3), 38.

Serfozo, R. (1981), "Optimal Control of Random Walks, Birth and Death Processes, and Queues," *Advances in Applied Probability*, 13(1), 61 –83.

Taylor, P. (2011), "Insensitivity in Stochastic Models," *Queueing Networks: Eds. R.J. Boucherie, N. M. van Dijk*, pp. 121–140.

Wierman, A., Andrew, L.L.H., and Tang, A. (2009), "Power-Aware Speed Scaling in Processor Sharing Systems," in *INFOCOM 2009, IEEE*, Apr., pp. 2007 –2015.

Wolff, R., *Stochastic Modeling and the Theory of Queues*, Prentice Hall (1989).