Energy–Performance Trade-Offs via the EP Queue

PETER G. HARRISON, Imperial College London
NARESH M. PATEL, NetApp Inc, Sunnyvale, California
WILLIAM J. KNOTTENBELT, Imperial College London

We introduce the EP queue – a significant generalization of the $M^B/G/1$ queue that has state-dependent service time probability distributions and incorporates power-up for first arrivals and power-down for idle periods. We derive exact results for the busy-time and response-time distributions. From these, we derive power consumption metrics during nonidle periods and overall response time metrics, which together provide a single measure of the trade-off between energy and performance. We illustrate these trade-offs for some policies and show how numerical results can provide insights into system behavior. The EP queue has application to storage systems, especially hard disks, and other data-center components such as compute servers, networking, and even hyperconverged infrastructure.

CCS Concepts: • **Mathematics of computing → Queueing theory**; • **Information systems → Storage power management**; • **Computing methodologies → Modeling methodologies**

Additional Key Words and Phrases: Energy-performance trade-off, power management, response times, modified M/G/1 queue, storage systems

ACM Reference Format:
Peter G. Harrison, Naresh M. Patel, and William J. Knottenbelt. 2016. Energy–performance trade-offs via the EP queue. ACM Trans. Model. Perform. Eval. Comput. Syst. 1, 2, Article 6 (June 2016), 31 pages.
DOI: http://dx.doi.org/10.1145/2818726

1. INTRODUCTION

1.1. Trading Off Power and Response Time In Modern Computing Infrastructures

Modern software-defined data centers are typically organized in a hierarchical fashion, with multiple tiers of cache present within and across components ranging from processors to network and storage components. Advances in caching technologies (e.g., host-side flash caches) mean that much read traffic is being absorbed before it reaches the lower tiers [Meng et al. 2013; Patel 2012]. Thus, the workloads of lower tiers are increasingly dominated by writes that can be potentially buffered in higher tiers. Nevertheless, occasional read bursts still occur due to the reading of as-yet-uncached data. This burstiness of access creates the possibility of using components that can be turned off during idle periods and restarted when new work arrives, thus saving energy while delivering acceptable access latencies. Efficient usage of compute, network, and storage resources is important in light of the fact that data centers are collectively storing about 35%–50% more data per year [Beath et al. 2012] and consuming more than 1% of global electricity [Koomey 2011]. Much progress has been made on the server side

Authors' addresses: P. G. Harrison and W. J. Knottenbelt, Department of Computing, Imperial College London, South Kensington Campus, London SW7 2AZ, UK; emails: {p.harrison, w.knottenbelt}@imperial.ac.uk; N. M. Patel, 495 East Java Drive, Sunnyvale, CA 94089, USA; email: naresh@netapp.com.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
2016 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM. 2376-3639/2016/06-ART6 $15.00
DOI: http://dx.doi.org/10.1145/2818726



This work is licensed under a Creative Commons Attribution International 4.0 License.

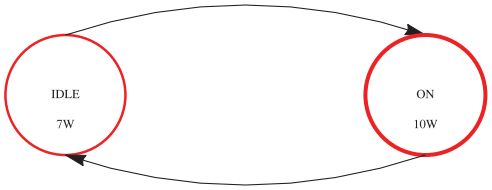


Fig. 1. Power usage for ON+IDLE sequence.

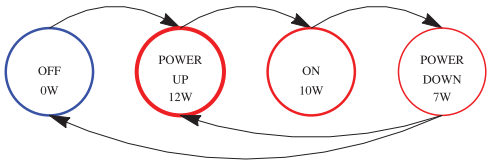


Fig. 2. Power usage for ON+OFF sequence.

with application consolidation via server virtualization and shared networked storage. However, further energy optimizations will be required as long as idle subcomponents consume energy.

In a system with components that shut down to save power, we need to understand the expected busy-period durations to determine power savings. Similarly, we need to incorporate startup (POWER UP) and shutdown (POWER DOWN) times as random variables to model the impact on both the power and response time characteristics. As one would expect, the first arrival to an idle server will incur a significant delay, but subsequent arrivals will likely fare much better. We would like to characterize the impact of this kind of response-time behavior on host-side applications.

It is in this context that we introduce and analyze the Energy–Performance (EP) queue – a significant generalization of the $M^B/G/1$ queue that has state-dependent service time probability distributions and incorporates power-up for first arrivals and power-down for idle periods. The analytical model developed compares two basic policies: the two-state *ON+IDLE* policy shown in Figure 1 and a four-state *ON+OFF* policy shown in Figure 2. The ON+IDLE policy is the typical case today in which lower-tier components oscillate between being fully loaded (the ON state, which operates at high power) and idle state (the IDLE state, which operates at lower power). In practice, in the case of disk drives, the IDLE state does not save significantly large amounts of power because disks continue to spin at a fixed rate even when no load is applied. Lower arm movement does save some power, however. Lower power states (e.g., so-called "Deep IDLE," in which new arrivals incur some additional wait) are also possible, ultimately leading to the zero power state (the OFF state). In this article, we will compare the baseline two-state ON+IDLE policy shown in Figure 1 with a four-state ON+OFF policy shown in Figure 2.

1.2. Metrics

It is important to consider appropriate metrics for power and performance. Power consumption is relatively straightforward to measure and can be modeled as the fraction of time the system component spends in each state and the corresponding power usage in Watts. For a *higher-is-better* metric, we use the following:

$$\text{EnergyEfficiency} = \frac{1}{\mathbb{E}[\text{Watts}]}$$

where

$$\mathbb{E}[\text{Watts}] = \frac{\text{Watts(Busy Period)}\mathbb{E}[\text{Busy Period}] + \text{Watts(Idle Period)}\mathbb{E}[\text{Idle Period}]}{\mathbb{E}[\text{Busy Period}] + \mathbb{E}[\text{Idle Period}]}.$$

Similarly, for the performance metric, we use a higher-is-better metric based on the reciprocal of latency:

$$\text{PerformanceMetric} = \frac{1}{\mathbb{E}[\text{Response Time}]}.$$

The higher-is-better metric for combined performance and energy may now be taken to be the product of the performance and energy-efficiency metrics:

$$\text{PE_Metric} = \frac{1}{\mathbb{E}[\text{Response Time}]\mathbb{E}[\text{Watts}]}.$$

In effect, this values the relative increase in the performance metric the same as the relative increase in the energy-efficiency metric: a 1% increase in latency affects the metric the same as a 1% increase in power usage. In practice, the value of these metrics becomes more intense when they approach boundaries of exhaustion such as running out of power capacity or inability to satisfy latency service-level objectives for many applications.

The PE_Metric is a higher-is-better measure for a given policy and at a given throughput level. By way of a running example, we will assume that energy is consumed at a constant rate in each state, as shown in Figures 1 and 2. This will be described by a vector $\vec{w}_2 = (7, 10)$ for the power usage in the states of the ON+IDLE sequence and, similarly, $\vec{w}_4 = (0, 12, 10, 7)$ for the four states in the ON+OFF sequence. As we have seen, $\mathbb{E}$[Watts] is computed by taking the dot product of $\vec{w}_i$ and $\vec{p}$, where p_j is the steady-state probability of the system being in state j. These steady-state probabilities can be computed from the average time spent in each state, which, for the idle period, is $1/\lambda$, and which requires the mean time spent in the POWER UP, ON, and POWER DOWN states. These follow from the busy-period analysis covered in Section 3.

1.3. Applications

In data centers, there is a class of applications that requires the storage to have good interactive performance during access periods, but otherwise spend a significant amount of time in the idle state. For example, disk drives storing old mail data have a very low IOPS per gigabyte ratio and are rarely accessed, but still require reasonable response times when an interactive user wants to read old mail. After the first request, there is a higher likelihood of more frequent subsequent accesses that generate a burst of activity. For archive mail, capacity-optimized hard disk drives continue to provide the lowest acquisition dollars-per-terabyte, but it seems wasteful to keep them spinning all the time. An open question is: When does it make economic sense to spin down these drives to optimize for a performance per watt metric?

A simple archive drive has a fixed total write capability during its five-year lifetime. The economics are such that the five-year life of a disk drive will require about 10W of power throughout its lifetime (with a typical ON+IDLE policy). In big-data terms, a storage pool of almost 1PB (ten 24-drive shelves of 4TB) will require about $18,000 of electricity in California, about half that in North Carolina, and more than double that in Germany[1]. The five-year electricity cost is a large percentage of the initial acquisition cost. Some form of lower-power mode for enterprise hard drives is needed to reduce the overall cost relative to potential competing technologies such as Quad-Level Cell (QLC) SSDs. Given the high costs, it may be insightful to compare the ON+OFF and ON+IDLE policies in terms of the energy–performance metric.

Figure 3 shows the type of trade-off we would like to quantify using the models described in later sections. The area of the red dots represents the power (watts) for the

[1]http://www.iea.org.

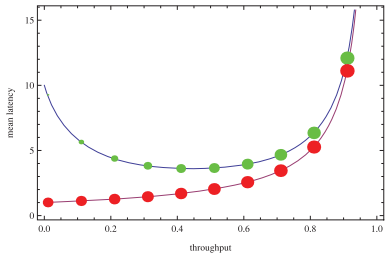


Fig. 3. Latency versus power trade-off.

ON+IDLE policy. The red dots are roughly the same size because the power usage for the IDLE state is similar to the ON state when the device is serving requests. Similarly, the green dots represent the power usage for the ON+OFF policy. At low loads, the power usage is very low, but it increases to the same levels as the ON+IDLE policy at high loads. The penalty for the ON+OFF policy is that latency increases at low loads because the first arrival has to wait for the server to power up. In this simple example, the power-up time is nine times the service time. At these low-to-medium load levels, we have a trade-off between the policies for which ON+OFF saves more power than ON+IDLE but incurs higher latencies. The PE_Metric is defined for each policy at a given throughput (or utilization) level. The ratio PE_Metric(ON+OFF)/PE_Metric(ON+IDLE) then shows the relative value of powering down the server. In our simple example, we see that the ON+IDLE policy has better metric values for most of the throughput range, but at very low throughput levels, ON+OFF is better.

1.4. Contributions and Outline

The main contributions of this article are:

—A new $M^B/GI/1$ queue with Poisson batch-arrivals and service times that are state-dependent, which can also incorporate power-up for first arrivals and power-down for idle periods.
—Exact results for the queue length distribution at departure instants when jobs arrive in batches with arbitrary batch size distribution.
—Exact results for the (Laplace-Stieltjes transforms of the) response-time distribution and busy-period distribution under certain simplifying assumptions, such as Poisson arrivals, but with general service-time distributions.
—Power consumption metrics during nonidle periods, mean response time (or latency) and mean nonidle periods, which together provide a single measure of goodness for energy–performance that is used to evaluate trade-offs quantitatively.

The remainder of this article is organized as follows. Sections 2 and 3 detail our analytical models for the response time and powered-up period distributions of a modified $M/G/1$ queue. This queue features state-dependent service rates (up to some maximum queue length n) and incorporates power-up for first arrivals and power-down for idle periods. Section 4 presents numerical results and Section 5 discusses related work. We present our conclusions in Section 6.

2. RESPONSE TIMES IN AN EP QUEUE

We consider an $M^B/GI/1$ queue with Poisson batch arrivals and service times that are state-dependent when the queue length i (including the task in service, if any) at the start of a service period is less than some threshold $n \geq 1$. Specifically, the service time random variable is S_i when $1 \leq i < n$ and S when $i \geq n$. We assume that

the service time is independent of the number of arrivals occurring during it, that is, that S_i depends only on i. The Poisson batch arrivals have rate λ, the batch size is an integer random variable B with probability generating function $G(z) = \sum_{i=1}^{\infty} b_i z^i$, and the service discipline is first come first served (FCFS). In addition, the server has to be powered up when an arrival occurs in its idle state (queue length 0) and powered down when a departure leaves the queue empty. Power-up and power-down times are independent random variables, denoted by U and D, respectively. If an arrival occurs during power-down, the power-down continues unaffected and is immediately followed by a power-up period, immediately after which the first task to have arrived commences service. Thus the nonidle, (partially or fully) powered-up period is elongated over the regular busy period, which is simply a maximal time period throughout which there is at least one task in the queue. We call this $\mathrm{M^B/GI/1}$ queue, with state-dependent service time distributions below a threshold n and startup and shutdown times, an EP queue or, more specifically, an EP_n^G *queue*. Note that when $U = D = 0$ with probability 1, $G(z) = z$, $n = 1$ and the EP queue reduces to a standard M/GI/1 queue.

We use the following notation regarding random variables. The probability distribution function of a continuous random variable X is denoted by $X(t) = \mathbb{P}(X \le t)$ and the Laplace-Stieltjes transform (LST) of this distribution by $X^*(\theta) = \mathbb{E}[e^{-\theta X}]$.[2] The density function of $X(t)$ is $x(t) = X'(t)$, the derivative of the distribution function, with Laplace transform $X^*(\theta)$. The mth moment of X is written $X_{[m]} = \mathbb{E}[X^m] = (-1)^m X^{*(m)}(0)$, where the parenthesized superscript denotes differentiation m times with respect to θ. Correspondingly, the probability-generating function (pgf) of a discrete random variable Y is written $G_Y(z) = \mathbb{E}[z^Y]$ (so that $G(z)$ is an abbreviation for $G_B(z)$) and its mth *factorial moment* $Y_{\langle m \rangle} = \mathbb{E}[Y(Y-1)\ldots(Y-m+1)] = G_Y^{(m)}(1)$.

2.1. Departure Time Instants in an EP Queue and Response Time Statistics

In a fixed-rate FCFS queue, that is, one with threshold $n = 1$, we consider the response time of a tagged task that is last in an arriving batch. This task's response time may be investigated using the observation that the queue existing just after that task's departure instant comprises precisely those tasks that arrived during its sojourn. Therefore, at equilibrium, when this exists, the pgf of the queue length, L, at the departure instant is that of the number of arrivals during the task's response time W, that is,

$$\begin{aligned}\Pi^W(z) &= \mathbb{E}[z^L] = \mathbb{E}[\mathbb{E}[z^{B_1+\cdots+B_{N_W}} \mid W]] = \mathbb{E}[\mathbb{E}[G(z)^{N_W} \mid W]] \\ &= \mathbb{E}[e^{-\lambda W(1-G(z))}] = W^*(\lambda(1-G(z))),\end{aligned}$$

where N_W is the number of arriving batches in a time period of length W and B_i is the number of tasks in the i^{th} batch ($1 \le i \le N_W$). This argument applies equally to the EP queue when service times are i.i.d., that is, S_i is distributed as the random variable S for all $i \ge 1$. We compute $\Pi^W(z)$ via the embedded Markov chain (EMC) of the queue at departure instants, hence obtain $W^*(\theta) = \Pi^W(1-\theta/\lambda)$.

More generally, the pgf of the number of task arrivals N_X from a batch Poisson process with batch-arrival rate λ and batch size pgf $G(z)$ during a random time X is $G_X(z) = \mathbb{E}[z^X] = X^*(\lambda(1-G(z)))$. Thus, the pgf of the number of task arrivals R_i during a task's service time is $G_{R_i}(z) = \mathbb{E}[z^{R_i}] = S_i^*(\lambda(1-G(z)))$ if the queue length was i when service commenced.

For nonnegative random variable X with probability distribution function having suitably smooth LST X^*, and for suitably differentiable function $g(z)$, we also define,

[2] $\mathbb{E}[\cdot]$ and $\mathbb{E}[\cdot \mid \cdot]$ denote the expectation and conditional expectation operators, respectively.

for $i \geq 0$,

$$a_j(X, g, c) = \frac{1}{j!} \left. \frac{\partial^j X^*(\lambda(1-g(z)))}{\partial z^j} \right|_{z=c},$$

where λ is a constant. Thus, $a_j(X, G, 0)$ is the probability that there are j task arrivals during a random time X from a Poisson batch-arrival process with rate λ and batch size pgf G(z); this follows since it is the coefficient of z^j in the pgf.

Let the one-step transition probability matrix of the EMC defined at service completion instants be $Q^c = [q^c_{ij} \mid i, j \geq 0]$ so that, for $i > 0$, $q^c_{ij} = \mathbb{P}(j-i+1$ arrivals in service time $S_i) = r_{i,j-i+1}$, say. When $i = 0$, the next service time does not start until after both a power-down and a power-up period have been completed, together with a wait for the next arrival in the event that no arrival occurred during the power-down.

At equilibrium (assuming that this exists), just after a task-departure (not necessarily the last task in a batch), let the probability that the queue length (i.e., the state of the chain) is j be p_j, so that the pgf of the equilibrium state is $\Pi^W(z) = \sum_{j=0}^{\infty} p_j z^j$. Then, the equilibrium probabilities p_i for $i \geq 0$ may be computed iteratively in terms of p_0 via the recurrence (for $p_{i+1}, i = 0, 1, \ldots$),

$$p_i = p_0 q^c_{0i} + \sum_{j=1}^{i+1} p_j r_{j,i-j+1}, \tag{1}$$

where $r_{j,k}$ is the coefficient of z^k in $G_{R_j}(z)$, that is, the probability that there are k arrivals during the service period S_j $(k \geq 0, j \geq 1)$. Computation of q^c_{0i} is deferred to after Proposition 2.2, but its pgf is given by the following result, which is proved in the appendix.

LEMMA 2.1. *Let $S_0 = D + U + S_{B_1+\cdots+B_{N_D+N_U}+\boldsymbol{I}_{N_D=0}B_0}$, where B_0 is a batch-size random variable, and $\sigma(z) = \mathbb{E}[\mathbb{E}[z^{N_{S_0}+\boldsymbol{I}_{N_D=0}B_0} \mid U, D]]$. Then, the pgf of the probability mass function $q^c_{0\cdot}$ is $\sigma(z)/z$, where*

$$\begin{aligned}\sigma(z) \;=\; & S^*_n U^*[D^* - (1-G(z))D^*(\lambda)] + \sum_{k=1}^{n-1}(S^*_k - S^*_n) z^k \sum_{j=1}^{k} a_{k-j}(U, G, 0) a_j(D, G, 0) \\ & + D^*(\lambda) \sum_{k=0}^{\infty} z^k a_k(U, G, 0) \sum_{\ell=1}^{n-k-1} \frac{1}{\ell!} G^{(\ell)}(0) z^\ell (S^*_{k+\ell} - S^*_n),\end{aligned}$$

where the argument of an LST is taken to be $\lambda(1-G(z))$ when unspecified.

This gives the following result for the pgf of the queue length probabilities just after a departure instant in an EP queue at equilibrium. The proof is also given in the appendix.

PROPOSITION 2.2.

$$\Pi^W(z) = \frac{p_0[S^*_n(\lambda(1-G(z))) - \sigma(z)] + \sum_{j=1}^{n-1}[S^*_n(\lambda(1-G(z))) - S^*_j(\lambda(1-G(z)))]p_j z^j}{S^*_n(\lambda(1-G(z))) - z}.$$

The unknown quantities $p_0, \ldots, p_{n-1}$ are determined from Equation (1) for $i = 0, \ldots, n-2$ together with $\Pi^W(1) = 1$, which reduces to, after application of l'Hopital's

rule,

$$p_0(\lambda G'(1)S_n^{*\prime}(0)+\sigma'(1))+\lambda G'(1)\sum_{j=1}^{n-1}[S_n^{*\prime}(0)-S_j^{*\prime}(0)]p_j=\lambda G'(1)S_n^{*\prime}(0)+1,$$

where the prime denotes differentiation with respect to z. It remains to calculate the probabilities q_{0i}^c for $i=1,\ldots,n-1$. These are given by the derivatives of the pgf $\sigma(z)/z$ at $z=0$, that is,

$$q_{0i}^c=\frac{1}{i!}\frac{\partial^i(\sigma(z)/z)}{\partial z^i}\bigg|_{z=0}.$$

For example, in the case of unit-sized batches (for simplicity),

$$\begin{aligned}q_{00}^c &= S_n^*(\lambda)U^*(\lambda)\lim_{z\to 0}\frac{D^*-(1-z)D^*(\lambda)}{z}+(S_1^*(\lambda)-S_n^*(\lambda))(\lambda a_0(U,\lambda)a_1(D,\lambda)+D^*(\lambda)a_0(U,\lambda))\\ &= S_n^*(\lambda)U^*(\lambda)(-\lambda D^{*\prime}(\lambda)+D^*(\lambda))+(S_1^*(\lambda)-S_n^*(\lambda))U^*(\lambda)(-\lambda D^{*\prime}(\lambda)+D^*(\lambda))\\ &= S_1^*(\lambda)U^*(\lambda)(-\lambda D^{*\prime}(\lambda)+D^*(\lambda)),\end{aligned}$$

which is the probability that there are either 0 or 1 arrivals during a power-down, none during a power-up, and none during a service period that commences with only one task in the queue, as required.

This gives the mean response time using the argument that the tasks in the queue at a departure instant are those that arrived from a Poisson process during the departing task's sojourn in the system (queuing plus being served), together with those tasks that arrived in the same batch behind it. However, this argument does not extend, in general, to the probability distribution of the response time, nor even its higher moments, because of subtle independence properties. Specifically, the queue is not overtake-free in the sense that the service time of a task may be influenced by later arrivals if they cause the queue length existing at the start of the task's service to be different.

Nevertheless, it does yield the correct mean value by Little's result. The mean queue length is $\Pi^{W\prime}(1)$ and the mean number of tasks behind a randomly selected task in a batch is $G''(1)/2G'(1)$, where the prime denotes differentiation.[3] Therefore, we calculate the mean response time of a task, $\overline{W}$, via $\lambda G'(1)\overline{W}+G''(1)/2G'(1)=\Pi^{W\prime}(1)$.

2.2. The EP Queue with Threshold $n>1$

When a task is being served, we refer to the set of other tasks waiting to commence service as the *waiting queue* or "queue only." Suppose that all service times are i.i.d. when the waiting queue is nonempty at the start of service. However, when a task-entering service is the only one in the system, it may have a different service-time distribution. In this scenario, the time spent *in the waiting queue* by a tagged task *is* independent of later arrivals. The preceding argument can then be applied correctly to the queuing (only) time. We can then add on the service time, which will be dependent on the queuing time, to obtain the response-time distribution, provided that arrivals are single [Bertsimas and Nakazato 1995; Haji and Newell 1971]. We next obtain the pgf of the waiting-queue length that exists immediately after a task enters service in the case that service times are i.i.d. when they start at queue lengths at or above any

[3]This follows from the renewal theory result that the joint probability that there are $i\geq 0$ tasks in front and $j\geq 0$ tasks behind a task in a batch with size having probability mass function $p_i(i\geq 1)$ is Cp_{i+j+1}, where C is a constant. The corresponding joint pgf is then routinely shown to be $J(x,y)=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}Cp_{i+j+1}x^iy^j=\frac{G(x)-G(y)}{G'(1)(x-y)}$.

given threshold value $n \geq 1$. The response-time distribution then follows, when the threshold $n = 2$ (nonempty waiting queue), by adding the queuing time and dependent service time of a tagged task.

Analogously to the previous section, let the one-step transition probability matrix of the EMC defined at start-of-service instants be $Q^s = [q_{ij} \mid i, j \geq 0]$. At equilibrium (assuming that this exists), just after a task enters service, let the probability that the number of tasks in the queue (*excluding* the one entering service) is j be π_j, with pgf $\Pi^Q(z) = \sum_{j=0}^{\infty} \pi_j z^j$. The terms q_{ij} are defined similarly to the corresponding q^c_{ij}, but q_{0j} is the probability that there are $j + 1$ arrivals (including the tagged task) in a period that begins when the chain enters state 0 (at the start of a service period) and ends when the next task enters service. Then, we have the following lemma and proposition, with proofs in the appendix.

LEMMA 2.3. *The number of task arrivals in a period that begins when the chain enters state 0 and ends when the next task enters service has pgf*

$$\beta(z) = S_1^*(\lambda(1 - G(z))) - S_1^*(\lambda)(1 - U^*(\lambda(1 - G(z)))(D^*(\lambda(1 - G(z))) - (1 - G(z))D^*(\lambda))).$$

This gives the following result for the pgf of the queue length random variable (excluding the task in service) at the start of a service period in an EP queue at equilibrium. A recurrence relation for the moments of Q^s at equilibrium is given in the appendix.

PROPOSITION 2.4. *The probability generating function of the number of tasks waiting in the queue just after the start of a service period in an equilibrium EP queue with threshold $n \geq 1$ is*

$$\Pi^Q(z) = \frac{\sum_{j=1}^{n-2} \pi_j z^j (S_n^*(\lambda(1 - G(z))) - S_{j+1}^*(\lambda(1 - G(z)))) + \pi_0 (S_n^*(\lambda(1 - G(z))) - \beta(z))}{S_n^*(\lambda(1 - G(z))) - z}$$

or, in alternate form, omitting the arguments of the Laplace transforms when they are $\lambda(1 - G(z))$,

$$\Pi^Q(z) = \frac{\sum_{j=0}^{n-2} \pi_j z^j (S_n^* - S_{j+1}^*) + \pi_0 S_1^*(\lambda)\big(1 - (D^* - (1 - G(z))D^*(\lambda))U^*\big)}{S_n^* - z}$$

where the probabilities $\pi_1, \ldots, \pi_{n-2}$ are given by the equations

$$\pi_i = \pi_0 q_{0i} + \sum_{j=1}^{i+1} \pi_j r_{j+1,i-j+1} \tag{2}$$

for $i = 0, \ldots, n - 3$ and the normalization condition,

$$\pi_0(\lambda G'(1) S_n^{*\prime}(0) + \beta'(1)) + \lambda G'(1) \sum_{j=1}^{n-2} \pi_j (S_n^{*\prime}(0) - S_{j+1}^{*\prime}(0)) = \lambda G'(1) S_n^{*\prime}(0) + 1.$$

The term $q_{0i} = \beta^{(i+1)}(0)$, where $\beta(z) = S_1^* - S_1^*(\lambda)(1 - U^*(D^* - (1 - G(z))D^*(\lambda)))$.

When $n = 2$ and arrivals are unbatched Poisson (batch size is one w.p. 1), or even a renewal process, at equilibrium we can use the argument that the pgf of the number of tasks in the queue (excluding any in service) at the instant a task enters service is the same as the pgf of the number of tasks that arrive during a queuing time [Bertsimas and Nakazato 1995]. This yields the queuing time distribution's LST via $Q^*(\lambda(1 - z)) = \Pi^Q(z)$, so that $Q^*(\phi) = \Pi^Q(1 - \phi/\lambda)$.

When the batch size is not one w.p. 1, we can consider whole batches, of B tasks, as the "customers" in the queue, with superthreshold batch-service time $Y = S_{21} + \cdots + S_{2B}$ having probability distribution with LST

$$Y^*(\theta) = \mathbb{E}\big[\mathbb{E}\big[e^{-\theta(S_{21}+\cdots+S_{2B})} \mid B\big]\big] = G(S_2^*(\theta))$$

provided that there has been a new arrival before the commencement of service of the last task in the batch. For an energy-saving device, this will often not be the case, since the workload is scheduled deliberately to create frequent idle periods at the end of batch servicing; we return to this in the proof of Proposition 2.5. Therefore, we have to modify Equation (6) of the appendix to find the pgf of the number of batches J'_m queuing at the beginning of service of the first task in batch m. Writing $Y' = S_{21} + \cdots + S_{2,B-1}$ as an abbreviation,

$$J'_m = J'_{m-1} - 1 + K_Y + \boldsymbol{I}_{J'_{m-1}=0}\boldsymbol{I}_{K_{Y'=0}}(K_{S_1} + \boldsymbol{I}_{K_{S_1}=0}(\boldsymbol{I}_{K_D=0} + K_D + K_U) - K_Y), \tag{3}$$

where, here, B is the number of tasks in a batch that started service with no more batches in the queue and K_X denotes the number of batches that arrive in time X. This adaptation, which we call a batched EP_2^G queue, leads to the following modified version of Proposition 2.4 in which arrivals are single "super-tasks" with service time distribution having LST $G(S_2^*(\theta))$ when the last task in the batch is served above the threshold $n = 2$.

PROPOSITION 2.5. *The probability generating function of the number of batches waiting in the queue just after the start of a batch-service period in an equilibrium batched EP_2^G queue with a threshold of n tasks is*

$$\Pi^{\tilde{Q}}(z) = C \cdot \frac{S_2^*(\lambda(1-z)) - S_1^*(\lambda(1-z)) + S_1^*(\lambda)\big(1 - U^*(\lambda(1-z))(D^*(\lambda(1-z)) - (1-z)D^*(\lambda))\big)}{G(S_2^*(\lambda(1-z)) - z}$$

where $C = \frac{1+\lambda G'(1)S_2^{*\prime}(0)}{\lambda S_2^{*\prime}(0) - \lambda S_1^{*\prime}(0) - S_1^*(\lambda)(\lambda U^{*\prime}(0) + \lambda D^{*\prime}(0) - D^*(\lambda))}$.

PROOF. Following the proof steps of Proposition 2.4, the pgf of the queue length at equilibrium, when this exists, is $\Pi^{\tilde{Q}}(z)$, where

$$\begin{aligned} z\Pi^{\tilde{Q}}(z) &= \sum_{j=1}^{\infty} \pi_j z^j Y^* + \pi_0 \mathbb{E}\big[z^{K_Y + \boldsymbol{I}_{K_{Y'=0}}(K_{S_1} + \boldsymbol{I}_{K_{S_1}=0}(\boldsymbol{I}_{K_D=0} + K_D + K_U) - K_{S_2})}\big] \\ &= \Pi^{\tilde{Q}}(z)Y^* - \pi_0 Y^* + \pi_0 \mathbb{E}\big[z^{K_2}(z^{K_{Y'}} - 1 + z^{\boldsymbol{I}_{K_{Y'=0}}(K_1 + \boldsymbol{I}_{K_1=0}(\boldsymbol{I}_{K_D=0} + K_D + K_U) - K_2)})\big], \end{aligned}$$

again omitting the arguments $(\lambda(1-z))$, where we have abbreviated K_{S_1}, K_{S_2} by K_1, K_2, respectively, and used the result that $z^{X+\boldsymbol{I}_{X=0}Y} = z^X + z^{\boldsymbol{I}_{X=0}Y} - 1$. Noting that $\mathbb{P}(Y' = 0) = (S_2^*(\lambda))^{B-1}$, two further applications of this result yield (omitting explicit applications of the conditional expectation tower rule)

$$\begin{aligned} (Y^* - z)\Pi^{\tilde{Q}}(z) &= \pi_0 Y^* - \pi_0 \mathbb{E}\Big[z^{K_Y} - z^{K_2} + \big(S_2^*(\lambda)\big)^{B-1} z^{K_1 + \boldsymbol{I}_{K_1=0}(\boldsymbol{I}_{K_D=0} + K_D + K_U)} + (1 - \big(S_2^*(\lambda)\big)^{B-1})z^{K_2}\Big] \\ &= -\pi_0\big(Y^*(\lambda)/S_2^*(\lambda)\big)\big(S_1^* - S_2^* - S_1^*(\lambda)\big(1 - U^*(D^* - (1-z)D^*(\lambda))\big)\big). \end{aligned}$$

Normalization uses l'Hopital's rule to impose $\Pi^{\tilde{Q}}(z) = 1$, giving the value of the constant C. □

The response time W of a given tagged task is composed of three components: the queuing time $\tilde{Q}$ of the batch containing the tagged task, the sum of the service times ahead of the tagged task, and the service time of the tagged task itself. The first two components are independent, but the third depends on whether or not the tagged task

is last in its batch (which depends on the number of tasks ahead of it in the batch, hence on the second component) and, if so, whether there have been any arrivals during the tagged task's sojourn up to the point that it enters service. Taking these dependencies into account, we have the following main result of this article.

THEOREM 2.6. *The response time W of a task at a random position in a batch in an EP queue with threshold 2 has a probability distribution with LST*

$$W^*(\theta) = \frac{\tilde{Q}^*(\theta+\lambda)Y^*(\theta+\lambda)(S_1^*(\theta)-S_2^*(\theta))}{m_b S_2^*(\theta+\lambda)} + \frac{\tilde{Q}^*(\theta)S_2^*(\theta)(1-Y^*(\theta))}{m_b(1-S_2^*(\theta))}$$

where $\tilde{Q}^*(\theta) = \Pi^{\tilde{Q}}(1-\theta/\lambda)$, $Y^*(\phi) = G(S_2^*(\phi))$, *and* $m_b = \frac{dG}{dz}\big|_{z=1}$ *is the mean batch size.*

PROOF.

$$W^*(\theta) = \mathbb{E}\big[e^{-\theta(\tilde{Q}+S_{21}+\cdots+S_{2I}+S)} \mid B\big],$$

where I is the number of tasks ahead of the tagged task in its batch and S is the service time of the tagged task – either S_1 or S_2. Denoting the number of tasks behind the tagged task by $J \geq 0$ and abbreviating $X = S_{21}+\cdots+S_{2I}$, we have that

$$\begin{aligned}
W^*(\theta) &= \mathbb{E}\big[\mathbb{E}\big[e^{-\theta\tilde{Q}}e^{-\theta X}e^{-\theta S_2}e^{-\theta(S_1-S_2)\boldsymbol{I}_{N_{\tilde{Q}}=0}\boldsymbol{I}_{N_X=0}\boldsymbol{I}_{J=0}} \mid \tilde{Q}, X, I, J\big]\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[e^{-\theta\tilde{Q}}S_2^*(\theta)^{I+1} + e^{-(\lambda+\theta)\tilde{Q}}e^{-\theta X}e^{-\theta S_2}\big(e^{-\theta(S_1-S_2)\boldsymbol{I}_{N_X=0}\boldsymbol{I}_{J=0}}-1\big) \mid X, I, J\big]\big] \\
&= \mathbb{E}\big[\mathbb{E}\big[\tilde{Q}^*(\theta)S_2^*(\theta)^{I+1} + \tilde{Q}^*(\theta+\lambda)e^{-(\theta+\lambda)X}e^{-\theta S_2}\big(e^{-\theta(S_1-S_2)\boldsymbol{I}_{J=0}}-1\big) \mid I, J\big]\big] \\
&= \mathbb{E}\big[\tilde{Q}^*(\theta)S_2^*(\theta)^{I+1} + \tilde{Q}^*(\theta+\lambda)e^{-(\theta+\lambda)X}\big(e^{-\theta S_1}-e^{-\theta S_2}\big)\boldsymbol{I}_{J=0}\big] \\
&= \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\big(\tilde{Q}^*(\theta)S_2^*(\theta)^{i+1} + \tilde{Q}^*(\theta+\lambda)S_2^*(\theta+\lambda)^i\big(S_1^*(\theta)-S_2^*(\theta)\big)\delta_{j0}\big)b_{i+j+1}/m_b,
\end{aligned}$$

where $\mathbb{P}(I=i, J=j) = b_{i+j+1}/m_b$ is the joint forward-backward recurrence formula in basic renewal theory in the discrete case, b_k being the probability that a batch contains k tasks. This sums to the required result. □

COROLLARY 2.7. *The p^{th} moment $W_{[p]}$ of response time in the earlier batch-EP$_2^G$ queue, $p = 0, 1, 2, \ldots,$ is*

$$W_{[p]} = \frac{\sum_{k=0}^{p-1}\binom{p+1}{k}(-1)^{p+1-k}\gamma_1^{(p+1-k)}(0)W_{[k]} + (-1)^p\sum_{k=0}^{p}\binom{p+1}{k}\left[\gamma_2^{(p+1-k)}(0)\tilde{Q}^{*(k)}(\lambda) + \gamma_3^{(p+1-k)}(0)\tilde{Q}^{*(k)}(0)\right]}{(p+1)\gamma_1^{(1)}(0)}$$

where

$$\begin{aligned}
\gamma_1(\theta) &= m_b S_2^*(\theta+\lambda)(1-S_2^*(\theta)) \\
\gamma_2(\theta) &= Y^*(\theta+\lambda)(S_1^*(\theta)-S_2^*(\theta))(1-S_2^*(\theta)) \\
\gamma_3(\theta) &= S_2^*(\theta)S_2^*(\theta+\lambda)(1-Y^*(\theta))
\end{aligned}$$

and $\tilde{Q}^{*(i)}(0)$ *and* $\tilde{Q}^{*(i)}(\lambda)$ *are given by the following recurrence relations:*

$$\begin{aligned}
\tilde{Q}^{*(p+1)}(0) &= \frac{\nu^{(p+2)}(0) - \sum_{k=0}^{p} \binom{p+2}{k} \lambda Y^{*(p+2-k)}(0) \tilde{Q}^{*(k)}(0)}{(p+2)(1+\lambda G'(1) S_2^{*\prime}(0))} \\
\tilde{Q}^{*(0)}(0) = \tilde{Q}^{*}(0) &= 1 \\
\tilde{Q}^{*(p+1)}(\lambda) &= \frac{\nu^{(p+1)}(\lambda) - \sum_{k=0}^{p} \binom{p+1}{k} (\lambda Y^{*(p+1-k)}(\lambda) + \delta_{kp}) \tilde{Q}^{*(k)}(\lambda)}{\lambda Y^{*}(\lambda)} \\
\tilde{Q}^{*(0)}(\lambda) = \tilde{Q}^{*}(\lambda) &= \Pi^{\tilde{Q}}(0) = \pi_0
\end{aligned}$$

where $\nu(\theta) = C(\lambda(S_2^*(\theta) - S_1^*(\theta)) + S_1^*(\lambda)(\lambda - U^*(\theta)(\lambda D^*(\theta) - \theta D^*(\lambda))))$.

Proof. From the theorem, $\gamma_1(\theta) W^*(\theta) = \gamma_2(\theta) \tilde{Q}^*(\theta + \lambda) + \gamma_3(\theta) \tilde{Q}^*(\theta)$, where $\tilde{Q}^*(\theta) = \Pi^{\tilde{Q}}(1 - \theta/\lambda)$; thus, from Proposition 2.5, $[\lambda Y_2^*(\theta) - \lambda + \theta] \tilde{Q}^*(\theta) = \nu(\theta)$. Differentiating both sides $p+2$ times at $\theta = 0$ and $p+1$ times at $\theta = \lambda$ yields the required results for $Q^{*(p+1)}(0)$, $Q^{*(p+1)}(\lambda)$, respectively. Differentiating the equation for $W^*(\theta)$ $p+1$ times at $\theta = 0$ gives the recurrence for $W_{[p]}$. □

3. POWERED-UP PERIODS AND ENERGY CONSUMPTION

Let the time elapsed between the start of service of a task at queue length i and the first subsequent instant at which the server becomes fully powered down, or idle, be denoted by the random variable T_i, for $i \geq 1$. Further, let T_0 denote the time elapsed between an instant at which the queue becomes empty and the end of the current nonidle period. This is not just the power-down period D since it may be that new tasks will arrive during this period, starting a busy period after the power-down period has been completed[4].

Then, there is the nonidle, that is, partially or fully powered-up period, $H = U + T_{B+N_U}$, where B is the number of tasks in the batch that started the nonidle period and N_U is the number of task arrivals during the powering-up period U. Thus, we have the recurrence

$$\begin{aligned}
T_j &= S_j + T_{N_{S_j}+j-1} \qquad \text{for } j \geq 1 \\
T_0 &= D + \big(U + T_{N_U+N_D}\big) \boldsymbol{I}_{N_D>0}
\end{aligned} \tag{4}$$

The conventional busy period in the two-state model, which we denote by the random variable P, is simply $P = T_B$, where the random variables U and D are set to 0 with probability one.

3.1. LSTs of Powered-Up Period Distributions

Above the threshold, that is, for $i > n$, the situation is as for a standard $M^B/G/1$ queue and T_i is just the sum of T_n and $(n-i)$ busy periods. Below the threshold, we need a new recurrence, which is given in the following result.

Lemma 3.1. *For* $i \geq n-1$, $T_i^*(\theta) = (V^*(\theta))^{i-n+1} T_{n-1}^*(\theta)$, *where* V^* *is the fixed point of the equation*

$$V^*(\theta) = S_n^*(\theta + \lambda(1 - G(V^*(\theta))))$$

[4]We assume that, once started, a power-down period must complete fully before a new startup period can begin. Other modi operandi are possible, for example, the startup period could begin immediately; these would require the model to be adapted suitably.

and for $1 \le i \le n-2$,

$$T_i^*(\theta) = \left\{ S_i^*\big(\theta + \lambda(1 - G(V^*(\theta)))\big)\big(V^*(\theta)\big)^{i-n} - \sum_{j=0}^{n-i-1} s_{ij}(\theta)\big(V^*(\theta)\big)^{j+i-n} \right\} T_{n-1}^*(\theta) + \sum_{j=0}^{n-i-1} s_{ij}(\theta) T_{j+i-1}^*(\theta),$$

where $T_0^*(\theta)$ *is* assumed to be known *and* $s_{ij}(\theta) = \frac{1}{j!}\frac{\partial^j S_i^*(\theta+\lambda(1-G(z)))}{\partial z^j}|_{z=0} = a_j(S_i, \tilde{G}, 0)(1 \le i \le n, 0 \le j \le n)$, *where* $\tilde{G}(z) = G(z) - \theta/\lambda$.

Proof. Above the threshold, for $i \ge n$, the time elapsed, V, for the queue length to reduce by one after the start of a new service period is the same as the busy period in an $M^B/G/1$ queue with i.i.d. service-time random variables S_n at all queue lengths. This follows from the same recursive argument, in which K_S denotes the number of *batches* that arrive from a Poisson process with rate λ in time S:

$$V^*(\theta) = \mathbb{E}\big[e^{-\theta(S_n + V_{11} + \cdots + V_{1B_1} + V_{21} + \cdots + V_{2B_2} + \cdots + V_{K_{S_n}1} + \cdots + V_{K_{S_n}B_{K_{S_n}}})}\big]$$

where the V_{ij} are i.i.d. as V. Thus,

$$\begin{aligned} V^*(\theta) &= \mathbb{E}\big[e^{-\theta S_n}\mathbb{E}\big[V^*(\theta)^{B_1+\cdots+B_{K_{S_n}}} \mid S_n\big]\big] = \mathbb{E}\big[e^{-\theta S_n}\mathbb{E}\big[G(V^*(\theta))^{K_{S_n}} \mid S_n\big]\big] \\ &= \mathbb{E}\big[e^{-\theta S_n} e^{-\lambda S_n(1-G(V^*(\theta)))}\big] = S_n^*(\theta + \lambda(1 - G(V^*(\theta)))) \end{aligned}$$

and, inductively, $T_i^*(\theta) = (V^*(\theta))^{i-n+1} T_{n-1}^*(\theta)$.

For $1 \le i \le n-1$, or any $i \ge 1$, for that matter,

$$\begin{aligned} T_i^*(\theta) &= \mathbb{E}\big[\mathbb{E}\big[e^{-\theta(S_i + T_{N_{S_i}+i-1})} \mid S_i\big]\big] \\ &= \mathbb{E}\left[\sum_{j=0}^{\infty} \frac{1}{j!} \left.\frac{\partial^j e^{-\lambda S_i(1-G(z))}}{\partial z^j}\right|_{z=0} e^{-\theta(S_i + T_{j+i-1})}\right] \\ &= \mathbb{E}\left[\sum_{j=0}^{\infty} \frac{1}{j!} \left.\frac{\partial^j e^{-S_i(\theta+\lambda(1-G(z)))}}{\partial z^j}\right|_{z=0} e^{-\theta T_{j+i-1}}\right] \end{aligned}$$

since $\frac{1}{j!}\frac{\partial^j e^{-\lambda S_i(1-G(z))}}{\partial z^j}|_{z=0}$ is the probability that there are j task arrivals during a service time S_i. Thus,

$$\begin{aligned} T_i^*(\theta) &= \sum_{j=0}^{\infty} s_{ij} T_{j+i-1}^*(\theta) = \sum_{j=0}^{n-i-1} s_{ij} T_{j+i-1}^*(\theta) + \sum_{j=n-i}^{\infty} s_{ij}(V^*(\theta))^{j+i-n} T_{n-1}^*(\theta) \\ &= \sum_{j=0}^{\infty} s_{ij}(V^*(\theta))^{j+i-n} T_{n-1}^*(\theta) + \sum_{j=0}^{n-i-1} s_{ij}\big(T_{j+i-1}^*(\theta) - (V^*(\theta))^{j+i-n} T_{n-1}^*(\theta)\big) \\ &= S_i^*\big(\theta + \lambda(1 - G(V^*(\theta)))\big)\big(\big(V^*(\theta)\big)^{i-n} T_{n-1}^*(\theta) + \sum_{j=0}^{n-i-1} s_{ij}\big(T_{j+i-1}^*(\theta) - (V^*(\theta))^{j+i-n} T_{n-1}^*(\theta)\big) \end{aligned}$$

and the result follows. □

LEMMA 3.2.

$$T_0^*(\theta) = D^*(\lambda+\theta) + T_n^*(\theta)U^*(\theta+\lambda(1-G(V^*(\theta))))V^*(\theta)^{-n}\left[D^*(\theta+\lambda(1-G(V^*(\theta))))\right.$$
$$\left.-D^*(\lambda+\theta)\right] + \sum_{k=1}^{n-1}\left(T_k^*(\theta) - T_n^*(\theta)V^*(\theta)^{k-n}\right)\sum_{j=1}^{k} u_{k-j}(\theta)d_j(\theta),$$

where $u_j(\theta) = \frac{1}{j!}\frac{\partial^j U^*(\theta+\lambda(1-G(z)))}{\partial z^j}|_{z=0}$ *and* $d_j(\theta) = \frac{1}{j!}\frac{\partial^j D^*(\theta+\lambda(1-G(z)))}{\partial z^j}|_{z=0}$.

PROOF. The proof is by direct simplification of the expression $e^{-\theta T_0}$, where $T_0 = D + (U + T_{N_U+N_D})\boldsymbol{I}_{N_D>0}$, similar to previous proofs. □

The following result now follows immediately.

PROPOSITION 3.3. *For each* θ *and* $i \geq 1$, $T_i^*(\theta)$ *may be computed by the following recurrence for* $\tau_i(\theta)$*:*

$$\tau_{n-1}(\theta) = 1;$$
$$\tau_i(\theta) = \tau_{n-1}(\theta)V^*(\theta)^{i-n+1} \quad \textit{for } i \geq n;$$
$$\tau_i(\theta) = \frac{1}{s_{i+1,0}(\theta)}\left[\tau_{i+1}(\theta) - \sum_{j=1}^{n-i-2} s_{i+1,j}(\theta)\tau_{i+j}(\theta)\right.$$
$$\left. - \tau_{n-1}(\theta)\left(V^*(\theta)^{i-n+1}S_{i+1}^*(\theta+\lambda(1-G(V^*(\theta)))) - \sum_{j=0}^{n-i-2} s_{i+1,j}(\theta)V^*(\theta)^{i+j-n+1}\right)\right]$$
$$\textit{for } i < n-1.$$

Then, for $i \geq 1$, $T_i^*(\theta) = T_0^*(\theta)\tau_i(\theta)/\tau_0(\theta)$*, where, omitting the argument* $(\theta + \lambda(1 - G(V^*(\theta))))$ *from* U^* *and* D^**,*

$$T_0^*(\theta) = \frac{D^*(\lambda+\theta)\tau_0(\theta)}{\tau_0 - \tau_n(\theta)(V^*(\theta))^{-n}U^*[D^* - D^*(\lambda+\theta)] - \sum_{k=1}^{n-1}\left(\tau_k(\theta) - \tau_n(\theta)(V^*(\theta))^{k-n}\right)\sum_{j=1}^{k} u_{k-j}(\theta)d_j(\theta)}.$$

Finally, the required LSTs of the probability distributions of the busy period and the (partially or fully) powered-up time are given by the following:

THEOREM 3.4. *The LSTs of the probability distribution functions of the (partially or fully) powered-up time,* H*, in the four-state model, and the busy period (when a task is being served),* P*, in the two-state model, are*

$$H^*(\theta) = T_{n-1}^*(\theta)U^*(\theta+\lambda(1-G(V^*(\theta))))\left(V^*(\theta)\right)^{-n+1}G(V^*(\theta))$$
$$+\sum_{i=1}^{n-2} b_i \sum_{j=0}^{n-i-2} u_j(\theta)\left(T_{i+j}^*(\theta) - (V^*(\theta))^{i+j-n+1}T_{n-1}^*(\theta)\right)$$

and

$$P^*(\theta) = T_{n-1}^*(\theta)(V^*(\theta))^{-n+1}G(V^*(\theta)) + \sum_{i=1}^{n-2} b_i\left(T_i^*(\theta) - \left(V^*(\theta)\right)^{i-n+1}T_{n-1}^*(\theta)\right),$$

which is calculated by the same recurrence as for T_i *in Equation (4), but with* $U = D = 0$ *with probability 1, so that* $T_0^*(\theta) = 1$.

PROOF.

$$H^*(\theta) = \mathbb{E}\big[\mathbb{E}\big[e^{-\theta(U+T_{B+N_U})} \mid U\big]\big]$$
$$= \mathbb{E}\left[\sum_{i=1}^{\infty}\sum_{j=0}^{\infty} b_i \frac{1}{j!} \left.\frac{\partial^j e^{-\lambda U(1-G(z))}}{\partial z^j}\right|_{z=0} e^{-\theta(U+T_{i+j})}\right]$$
$$= \mathbb{E}\left[\sum_{i=1}^{\infty}\sum_{j=0}^{\infty} b_i \frac{1}{j!} \left.\frac{\partial^j e^{-U(\theta+\lambda(1-G(z)))}}{\partial z^j}\right|_{z=0} e^{-\theta T_{i+j}}\right]$$

since $\frac{1}{j!}\frac{\partial^j e^{-\lambda U(1-G(z))}}{\partial z^j}|_{z=0}$ is the probability that there are j task arrivals during time U. The rest of the proof for $H^*(\theta)$ is now similar to that of Lemma 3.1. The result for P^* is similar but somewhat simpler, beginning with $P^*(\theta) = \sum_{i=1}^{\infty} b_i T_i^*(\theta)$. □

3.2. Moments

The moments of H and the T_j depend on the moments of V, which has its Laplace transform defined as a fixed point. A complex recurrence for $T_{j[p]}$ could be derived in terms of the moments of V, which themselves satisfy a recurrence relation. More simply, we approximate the function $V^*(\theta) = 1 + \sum_{i=1}^{p} v_i\theta^i$ to order p, when we know *a priori* that only p moments are required. The coefficient $v_i = V^{*(i)}(0)/i! = (-1)^i V_{[i]}/i!$ in terms of moments. It is routine to compute any finite number of these symbolically, using standard mathematical software. To illustrate, the first two moments of V, $-v_1$ and $2v_2$, are obtained by direct differentiation as

$$v_1 = -\frac{S_n'(0)}{1+\lambda G'(1)S_n'(0)}$$
$$v_2 = \frac{S_n''(0)(1-\lambda G'(1)v_1)^2 - \lambda S_n'(0)G''(1)v_1^2}{2(1+\lambda G'(1)S_n'(0))}$$

The moments of T_j and H then follow by direct differentiation, using the algorithm of Proposition 3.3 and Theorem 3.4.

4. NUMERICAL RESULTS

We now apply the results to understand energy–performance trade-offs in a storage system by using a combination of measurements and estimated data for inputs that were not readily available. Sensitivity analysis of selected parameters was performed to quantify the impact of future storage-device characteristics. To specify an EP queue, we need the following input data: the power usage (watts) in various power states; a characterization of the incoming workload batch-arrival process (e.g., interarrival time and batch size probability distributions); the service-time distribution at each queue length; and the power-up and power-down time distributions. For power usage, we will use the consumption shown in Figures 1 and 2 – namely, 0W for the OFF state, 7W for the IDLE state, 10W for the ON state, 12W for the POWER UP state, and 7W for the POWER DOWN state. For the incoming workload, we assume that the arrival-point processes are Poisson with rate λ and that each arrival comprises a batch of new customers at the EP queue of a number that is geometrically distributed with mean 10; we chose geometric arbitrarily from well-known distributions with infinite support, but any other choice is equally feasible, for example, constant-sized batches. For simplicity, we assume that power consumption is constant in any given state, although we acknowledge that power surges may occur. The POWER UP time distribution is assumed to be Erlang-4 with mean 60ms or 600ms (thus its LST is $(1/(1+4s/0.06)^4)$ in

the former case) and the POWER DOWN time distribution is assumed to be Erlang-4 with mean 30ms or 300ms (thus LST $(1/(1 + 4s/0.03)^4)$ in the former case).

We use a gamma distribution for the service times; thus, when the mean is m and the variance is v, the LST is $\beta^\alpha/(s + \beta)^\alpha$, where $\alpha = m^2/v$ is the shape parameter and $\beta = m/v$ is the rate parameter. The FCFS queue discipline is approximate, but we note that driver software will order requests to allow most IOs to be serviced in order under load even though the drive implements a shortest position time first policy. The drive service time varies with queue length because of optimizations that can be made by the drive firmware.

In an experimental setup using hybrid storage (mixture of SSDs and HDDs), we observed that the SSDs absorbed most of the reads and writes when exposed to an OLTP storage workload[5]. In many systems with performance monitoring of their storage devices, only the first moment is measured, but a few systems also measure the second moment for some statistics. In our case, we monitor both and estimate the variance as needed.

4.1. Mean Response Time (Latency)

Hard disk drives typically deliver lower service times as queue lengths increase because high-level software or the drive itself reorders requests. At high queue depths, drives can generally handle a limited number of requests per revolution. This has been our observation with both high-capacity (SATA) drives and high-performance enterprise drives. For example, the 7200 RPM SATA drives used in our tests were capable of completing two requests in one 8.33ms revolution. Thus, for high loads, the mean service time becomes 4.17ms. For low loads, random-read service times would be the sum of the seek, rotation, and transfer times. Drive service-time statistics were collected while a steady-state load was applied using an OLTP-like workload generator. The number of requests queued to the drive is limited to a small number, which can be reordered by the drive firmware for improved efficiency. The FCFS assumption is an approximation in that the front of the queue will see some reordering, but the rest of the queue is FCFS. The effect of the reordering at the front of the queue is captured by the queue-length-dependent service times; the rest of the queue is modeled correctly as FCFS.

If we need only the mean response time of storage accesses, we can allow state dependence at queue lengths greater than two, using Proposition 2.2. For our hybrid storage example, we found that it was reasonable to model the mean service times as follows:

$$\mathbb{E}[S_j] = 11.22 - 1.41j \text{ for } 1 \le j \le n,$$

where $n = 5$ and $\mathbb{E}[S] = 4.17$ms for all queue lengths 5 and above. The coefficient of variation of the service times generally decreases with increased queue length, starting at 0.8 for a queue length of 1 and dropping to 0.3 for a queue length 5 and greater. More precisely, we use a coefficient of variation $0.81 - 0.0098j$ for queue lengths $j < n$ and 0.30 for $j \ge 5$. These means and coefficients of variation yield 5 different gamma distributions, with corresponding parameters.

We first consider the performance of the system in terms of its latency as the load increases, characterized by the normalized throughput, which we also refer to as utilization. For the 4-state power-mode (OFF, UP, BUSY, DOWN), with 5 service-time distributions (i.e., with threshold 5) and mean power-up/down times of 60ms/30ms, Figure 4 shows the mean response time under increasing load for batch sizes between

[5]In a hybrid Flash-HDD storage system, we expect the Flash storage to be powered up all the time but allow parts of the HDD storage to power down. Each power state of the HDD is assumed to have a fixed mean power-usage level. Different types of HDD storage may indeed have different mean power-usage levels for each state; these would be model inputs for the EP queue.

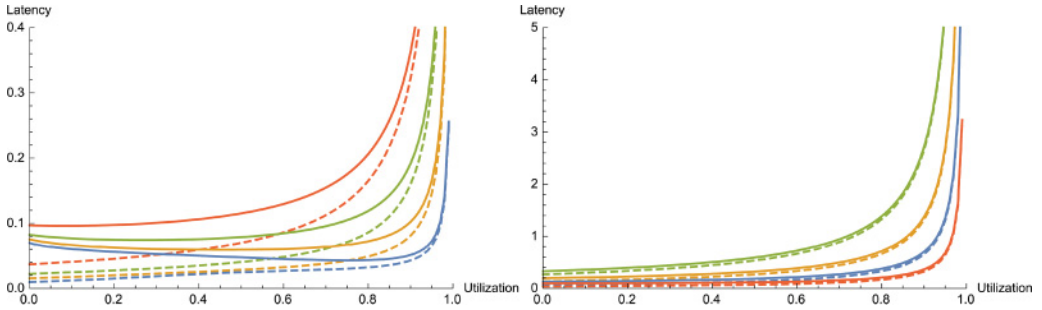


Fig. 4. Mean response time (in milliseconds) under increasing load in the 4-state model with threshold $n = 5$: the left-hand graphs for batch sizes 1, 2, 4, and 8; and the right-hand graphs for batch sizes 8, 16, 32, and 64, increasing from bottom to top. The dashed plots show the corresponding latency obtained by the 2-state, ON/IDLE policy (mean power-up/power-down times 60ms/30ms).

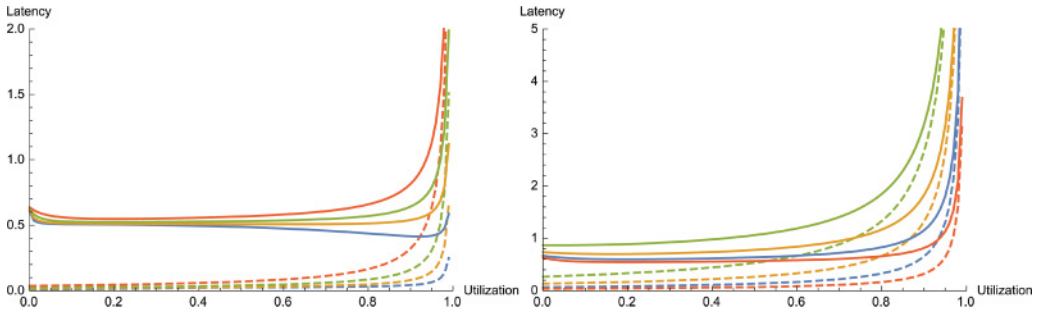


Fig. 5. Mean response time (in milliseconds) under increasing load in the 4-state model with threshold $n = 5$: the left-hand graphs for batch sizes 1, 2, 4, and 8; and the right-hand graphs for batch sizes 8, 16, 32, and 64, increasing from bottom to top. The dashed plots show the corresponding latency obtained by the 2-state, ON/IDLE policy (mean power-up power-down times 600ms/300ms).

1 and 8 (left) and for batch sizes between 8 (repeated) and 64 (right). Corresponding graphs for the slower power-up/down times of 600ms/300ms are shown in Figure 5. It is well known that performance degrades when an input process becomes more bursty while keeping the task-arrival rate fixed, and this is evident here. In particular, we see in the former figure the interesting effect that, at mean batch sizes of about 4 and below, the latency actually decreases as utilization increases up to a certain point (about 0.8 for unit batches, 0.3 for a mean batch size of 4). The same qualitative observation applies in the latter figure, pertaining to slower power-up/down, but here, all mean batch sizes show a decreasing latency at lower utilizations. This effect is exhibited especially at batch size 1, for which latency continues to decrease up to high utilizations. It is unsurprising that the smoothest input (Poisson) performs the best, but the whole rationale of our work is to take into account the energy consumption penalty, which is clearly the greatest with a "smooth" arrival process that rarely admits a low power mode of operation.

The 2-state, ON/IDLE policy achieves a much smaller latency at low utilizations, as shown by the dashed curves. Especially for the faster power-up/down times, the difference is less pronounced at high utilizations. We will see quantitatively in Section 4.5 that it is at low utilizations that the 2-state policy is most costly in terms of energy; qualitatively, an obvious observation since the idle-power (at least) is needed at all times. This is the area in which the energy–performance trade-off is most apparent.

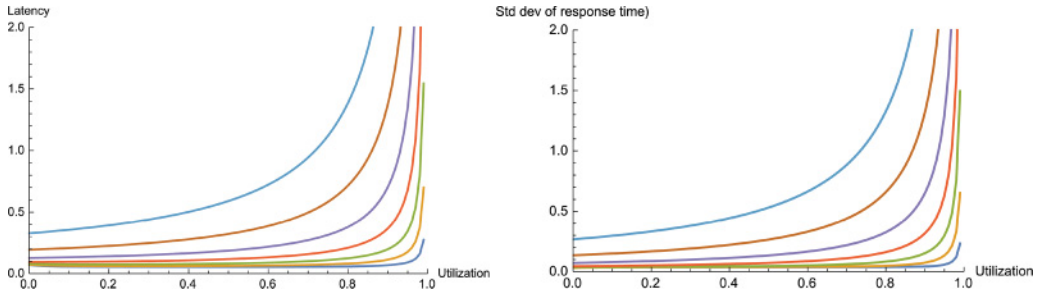


Fig. 6. Mean (left) and standard deviation (right) of response time (in milliseconds) under increasing load in the 4-state model with threshold $n = 2$, for batch sizes 1, 2, 4, 8, 16, 32, and 64, increasing from bottom to top (mean power-up/power-down times 60ms/30 ms).

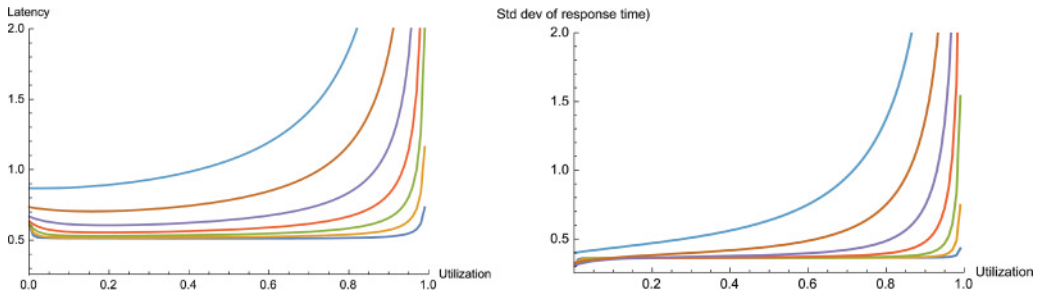


Fig. 7. Mean (left) and standard deviation (right) of response time (in milliseconds) under increasing load in the 4-state model with threshold $n = 2$, for batch sizes 1, 2, 4, 8, 16, 32, and 64, increasing from bottom to top (mean power-up/power-down times 600ms/300ms).

4.2. Higher Moments of Response Time

To compute the (Laplace transform of the) distribution of response time, we require that the threshold be 2, that is, service times have the same distribution for queue lengths greater than 1 at the start of service. For the high-capacity disk drive, the mean service time is 9.8ms and the standard deviation is 7.8ms when the queue length is one, but as the queue length increases, the disk scheduling algorithms have more opportunity to reorder requests to improve efficiency. At higher queue lengths (two or more), we assume that the mean service time is 4.2ms (about half a rotation) and the standard deviation is 1.3ms. The restriction to just two distinct service-time distributions (queue lengths of one and two or more) allows us to obtain the higher moments of the EP-queue response time via Corollary 2.7 to Theorem 2.6.

Qualitatively, the graphs of both the mean and standard deviation of response time are similar to each other and to the latency (mean response time) graphs in the threshold-5 case, although the decreases are less pronounced at lower utilizations. This may be because the decrease in mean service time stops above queue length 2. In the case of higher power-up/down times (Figure 7), the effects at low utilization are more dramatic, essentially following the characteristics of the dominating power-up time distribution.

4.3. Response-Time Densities and Quantiles

Probability density functions and quantiles of response time were computed by numerical inversion of the Laplace transforms $W^*(\theta)$, given by Theorem 2.6; necessarily, we require the threshold to be $n = 2$. The density functions shown in Figure 8 use mean power-up and power-down times of 60ms and 30ms and a mean batch size of 32 in

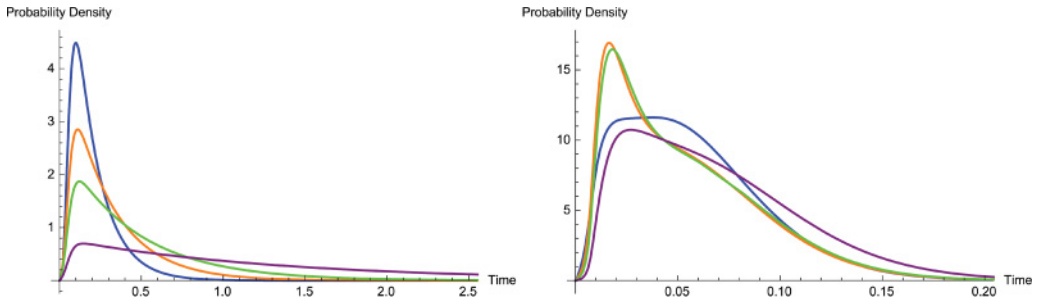


Fig. 8. PDF of response time (horizontal axis in milliseconds) for batch sizes 32 (left) and 1 (right) at utilizations 0.1, 0.5, 0.7, and 0.9, colored blue, orange, green, and purple, respectively (mean power-up/power-down times 60ms/30ms).

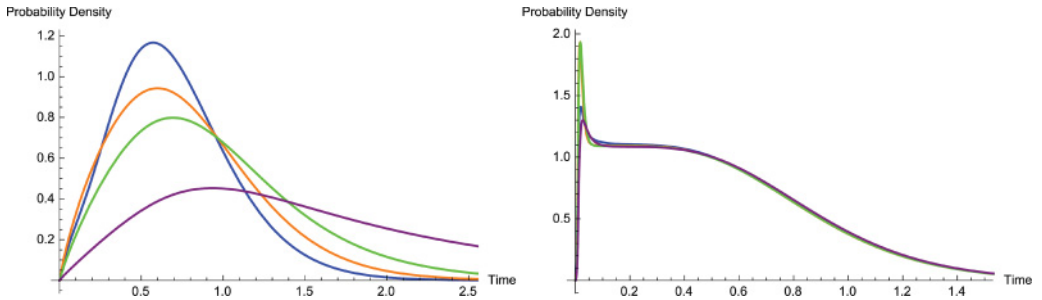


Fig. 9. PDF of response time (horizontal axis in milliseconds) for batch sizes 32 (left) and 1 (right) at utilizations 0.1, 0.5, 0.7, and 0.9, colored blue, orange, green, and purple, respectively (mean power-up/power-down times 600ms/300ms).

the left-hand figure, and 1 in the right-hand figure. This shows the big influence that powering up and down has on performance at low batch sizes. We therefore plotted similar graphs for the case in which the power-up and power-down times are 10 times greater (Figure 9).

For batch size 32, there is little qualitative difference; in both cases, the densities are unimodal with peak (maximum value) decreasing as utilization and mean value increase. For batch size 1, however, the curves appear unusual at lowest (0.1) and highest (0.9) utilizations (at least). More significantly, the ranking of the curves with respect to their peaks is not monotonic with regard to utilization; the curves at utilization 0.5 and 0.7 have the highest peaks at lower power-up/down times. This is likely because, in order to benefit from a low response time, a task must arrive when there is at least one other task present; otherwise, a power-up time will be necessary. At very low utilizations, this eventuality is unlikely. The observation is also consistent with the decreasing latency seen for batch size 1 in Figures 4 and 5.

The left of the wide peak at utilization 0.1 and batch size 1 (Figure 9, right) corresponds to tasks that arrive near the end of a power-up period (just before or just after, while a task is being served). The right of the peak corresponds to tasks that arrive near the beginning of a power-up period. These two scenarios would generate their own (conditional) PDFs; the result that we see is their superposition. Hence, if their conditional PDFs were more sharply peaked, that is, with smaller variances, a bimodal density ought to result.

To investigate this, we decreased the variance of the up period by increasing the number of Erlang stages (keeping the mean the same) from 4 to 40, as shown in Figure 10. We do indeed see bimodal distributions at lower variances of the up period.

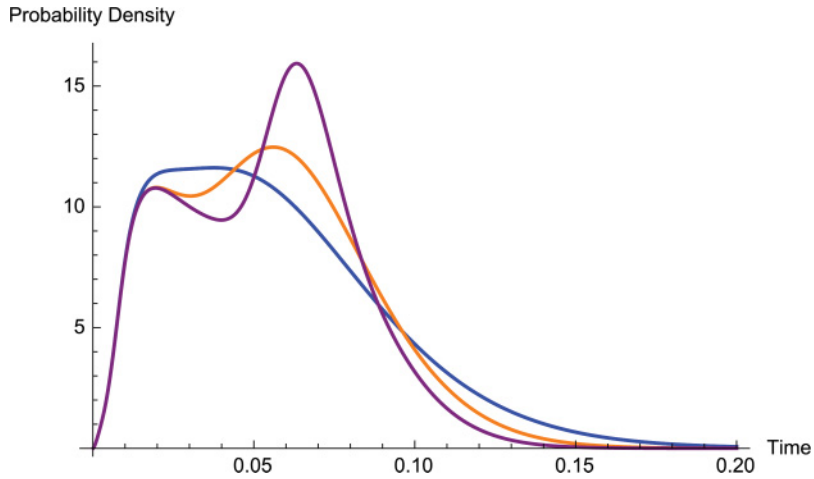


Fig. 10. PDF of response time (horizontal axis in milliseconds) for decreasing variance of the up period. The numbers of stages in its Erlang distribution are 4, 10, and 40, with the corresponding graphs having increasingly higher peaks (mean power-up/power-down times 60ms/30ms, batch size 1).

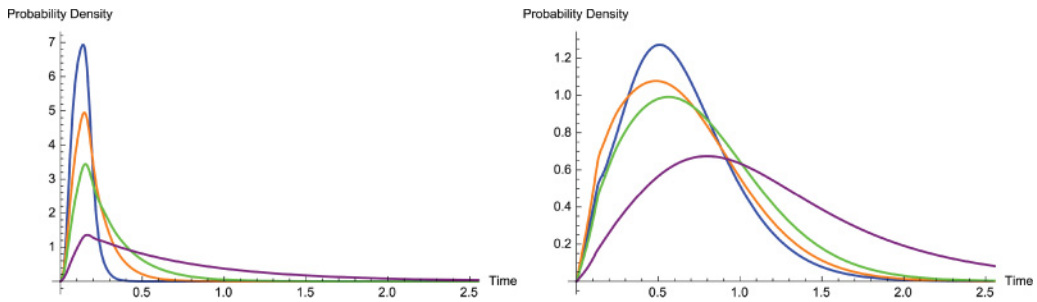


Fig. 11. PDF of response time (horizontal axis in milliseconds) for the discrete batch size distribution with mean 32 at utilizations 0.1, 0.5, 0.7, and 0.9, colored blue, orange, green, and purple, respectively (mean power-up/power-down times 60ms/30ms on the left, 600ms/300ms on the right).

Curiously, the right-hand peak not only heightens but also moves to the right as the variance decreases. It can be seen that, essentially, the flatter tails caused by the dominant up period "pull in" the curves from the right to preserve the area underneath.

In the preceding, we chose to use geometric batch sizes to represent a discrete probability mass function with infinite support, but this was an arbitrary choice since our results allow general batch-size distributions. For completeness, we recomputed the left-hand graphs of Figures 8 and 9, replacing the geometric batch size with mean 32 by one having sizes 31 or 33 with equal probability 0.5 – that is, same mean of 32. This produced the graphs of Figure 11, which show little difference qualitatively. The peaks are higher and the tails flatter, as would be expected with the less variable batch sizes. The response time distribution seems largely insensitive to the distribution of batch size, depending primarily on its mean value, as we saw for the geometric case.

Finally, we investigated the effect of batch size and power-up/down times on response time in terms of the mean, standard deviation, median, and lower and upper quartiles at various normalized throughputs (which we also call utilizations). These results are shown in Table I. Similar insight may be gained as from the preceding graphs, but it is instructive to see the numerical values of the quantiles in particular, which are vital in time-critical applications.

Table I. Latencies for Various Parameterizations of the 4-State Model with Threshold $n = 2$: Means, Standard Deviations, and 50%, 75%, and 95% Quantiles Against Mean Batch Size, Mean Power-Up and Power-Down Times at Utilizations 50% and 75%

Normalized throughput	Mean batch size	Power-up/down (milliseconds)	Mean (milliseconds)	Std Dev (milliseconds)	50% (milliseconds)	75% (milliseconds)	95% (milliseconds)
50%	1	60/30	53.24	36.03	45.41	74.86	122.1
50%	1	100/50	86.19	60.98	75.21	123.4	201.1
50%	1	600/300	511.6	364.4	454.1	732.4	1191.
50%	2	60/30	63.25	38.06	56.85	86.17	134.6
50%	2	100/50	96.41	62.01	86.28	134.2	212.3
50%	2	600/300	522.2	363.9	464.7	742.5	1201.
50%	8	60/30	117.3	71.88	102.9	151.6	254.2
50%	8	100/50	150.6	86.51	137.1	198.1	311.3
50%	8	600/300	574.2	368.4	517.7	797.3	1259.
50%	64	60/30	590.8	531.6	428.9	796.7	1651.
50%	64	100/50	628.4	533.3	468.3	836.1	1690.
50%	64	600/300	1065.	635.7	948.7	1384.	2265.
75%	1	60/30	55.93	36.94	47.76	78.13	126.6
75%	1	100/50	88.62	61.80	77.30	126.5	205.1
75%	1	600/300	513.5	365.3	456.1	735.2	1195.
75%	2	60/30	74.94	44.23	67.84	100.9	157.5
75%	2	100/50	108.2	66.13	98.10	148.4	231.1
75%	2	600/300	534.0	364.9	476.7	755.1	1214.
75%	8	60/30	178.7	130.8	144.7	232.8	435.5
75%	8	100/50	211.9	139.5	182.0	276.0	479.8
75%	8	600/300	636.7	384.5	581.3	869.7	1348.
75%	64	60/30	1121.	1062.	795.5	1531.	3239.
75%	64	100/50	1157.	1063.	833.2	1569.	3277.
75%	64	600/300	1587.	1119.	1311.	2062.	3770.

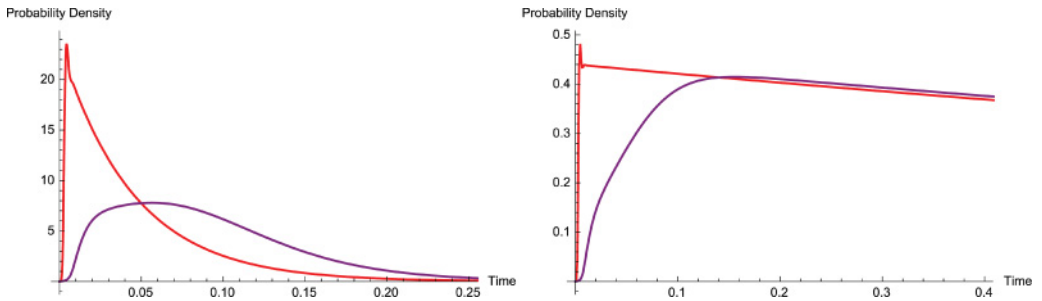


Fig. 12. Comparison of PDFs of response time (horizontal axis in milliseconds) for batch size 1 at utilizations 0.95 (left) and 0.999 (right). The red, peaky plot is for the 4-state policy; the purple plot is for standard M/G/1 (mean power-up/power-down times 60ms /30ms).

4.4. Asymptotic Behavior

As the utilization approaches 1, there will be few times at which the queue is empty for any batch size. We illustrate the asymptotic behavior by comparing, at high utilizations, an EP_2^G queue with unit batch size (which will exhibit the greatest difference) with a simple $M/G/1$ queue that has a fixed service time distribution equal to that of the EP_2^G queue above the threshold. The results are shown in Figure 12.

As expected, the tails of the distributions converge, quickly at the higher utilization of 0.999. More interesting is the difference in the density functions at small response times. The probabilities in this region are, of course, tiny since the queues are near saturation, but the relative difference is large, reflecting the large overhead of powering

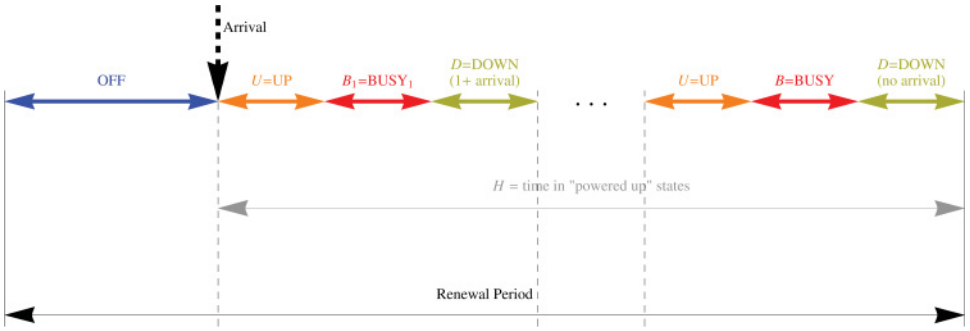


Fig. 13. Components of the renewal period in the 4-power-state case.

up and down. The results for mean power-up/down times of 600ms/300ms are similar but the convergence occurs further to the right, reflecting the greater impact of the 10x overhead.

4.5. Power Usage

In the 2-state case (IDLE and ON), the system cycles through a series of idle and busy periods that together form a renewal process. The power usage is nonzero in each state but is lower in the IDLE state. The length of the idle period is negative exponential with mean $1/\lambda$ and the busy (powered-up) period random variable is, say, P; the mean renewal period is then $1/\lambda + \mathbb{E}[P]$. The average power usage over all renewal periods can be readily computed from the dot product of $\vec{w}$ (the vector of the watts used in each state of the power strategy, $\vec{w}_2$ or $\vec{w}_4$ for 2 or 4 states, respectively) and the probability vector of being in those states, as explained in Section 1.2:

$$\mathbb{E}(\text{Watts2}) = \frac{\vec{w}_2 \cdot \left(\frac{1}{\lambda}, \mathbb{E}(\text{P})\right)}{\frac{1}{\lambda} + \mathbb{E}(\text{P})}.$$

In the 4-state case (OFF, UP, BUSY, DOWN), the power usage varies during the powered-up states, whereas in the 2-state case, the power usage is constant during the busy period. It also forms a renewal process but with an OFF state followed by a series of UP, BUSY, and DOWN sequences (see Figure 13). The number of UP-BUSY-DOWN sequences N during a renewal period is a geometric random variable with parameter $D^*(\lambda)$, which is the probability of no Poisson arrival during a DOWN period; thus, $\mathbb{E}[N] = 1/D^*(\lambda)$ is the average number of DOWN periods in a renewal period. The UP times and the DOWN times are i.i.d.; all the BUSY periods after the first one in the renewal period are also i.i.d. However, the first BUSY time in a renewal period is special because it starts with one initial batch arrival, the one that initiates the first UP period, and the arrivals during that UP period. In contrast, subsequent ones depend on the number of arrivals during the preceding DOWN time, conditioned on there being at least one arrival to start another UP period, as well as the arrivals during the said UP period.

The average power consumption during the powered-up period can be found by elementary methods involving only mean values, noting that it is zero during the OFF period in the 4-state model. The average overall busy period duration $\mathbb{E}[H]$ can be computed from the Laplace transform $H^*(\cdot)$ (specifically its first derivative at the origin), together with the average UP time $\mathbb{E}[U]$ and the average DOWN time $\mathbb{E}[D]$, which are given by their specified distributions. Using Wald's Equation, we can compute

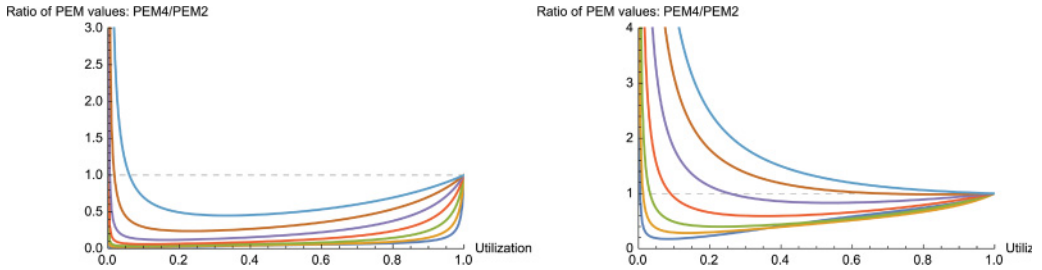


Fig. 14. The ratio PE_Metric-4 (OFF, UP, BUSY, DOWN) / PE_Metric-2 (IDLE, ON) under increasing load and for various batch sizes (batch size increasing from bottom to top as 1, 2, 4, 8, 16, 32, 64). Mean power-up / power-down times: 600/300 milliseconds in the left hand graph, 60/30 milliseconds in the right hand graph. Threshold, $n = 5$.

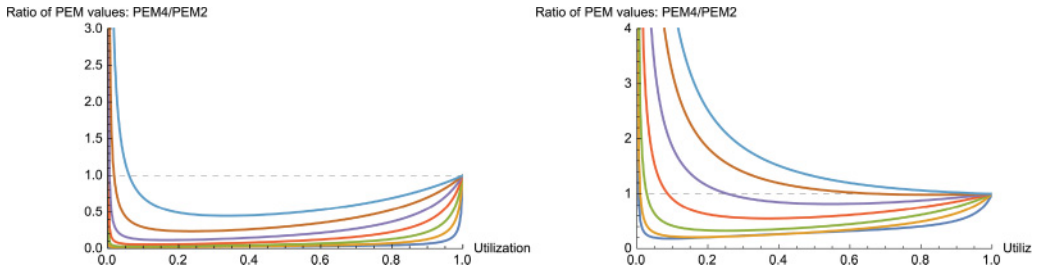


Fig. 15. The ratio PE_Metric-4 (OFF, UP, BUSY, DOWN)/PE_Metric-2 (IDLE, ON) under increasing load and for various batch sizes (batch size increasing from bottom to top as 1, 2, 4, 8, 16, 32, 64). Mean power-up/power-down times: 600ms/300ms in the left-hand graph, 60ms/30ms in the right-hand graph. Threshold, $n = 2$.

the fraction of the powered-up period spent in the UP and DOWN states as

$$q_U = \frac{\mathbb{E}(U)\mathbb{E}(N)}{\mathbb{E}(H)}, \quad q_D = \frac{\mathbb{E}(D)\mathbb{E}(N)}{\mathbb{E}(H)}.$$

The remaining portion of the powered-up period is spent in the full-power state; thus,

$$\mathbb{E}(\text{Watts4}) = \frac{\mathbb{E}(H)\vec{w}_4 \cdot (0, q_U, q_B, q_D)}{\frac{1}{\lambda} + \mathbb{E}(H)},$$
$$\text{where } q_B = 1 - q_U - q_D.$$

For various load levels, we compare the ratio of the PE metric achieved by the 4-state (OFF, UP, BUSY, DOWN) and 2-state (IDLE, ON) policies for various batch sizes under increasing load, as shown in Figures 14 and 15 for threshold values of 5 and 2, respectively. Where the ratio (vertical axis value) is above 1, the 4-state policy is beneficial, delivering a superior performance/energy efficiency.

We observe, at both low and high power-up/down times, that the ratio increases with increasing batch size (with the exception of unit batches, to follow), but always tends to 1 at high utilizations, irrespective of batch size. This accords with intuition, since the proportion of time spent in the UP or DOWN states in the 4-state policy will become vanishingly small as utilization approaches 1. It can be seen that, with our chosen PE metric, there is no benefit in using the 4-state policy at utilizations above about 0.075, even for mean batch sizes of 64; this number is even lower at lower mean batch sizes. The reason is that the overhead placed by the power-up time, in particular, is too great to merit the switching off of a device; the idle periods are just too short. This motivates an investigation as to how small should mean power-up/down time be to warrant the

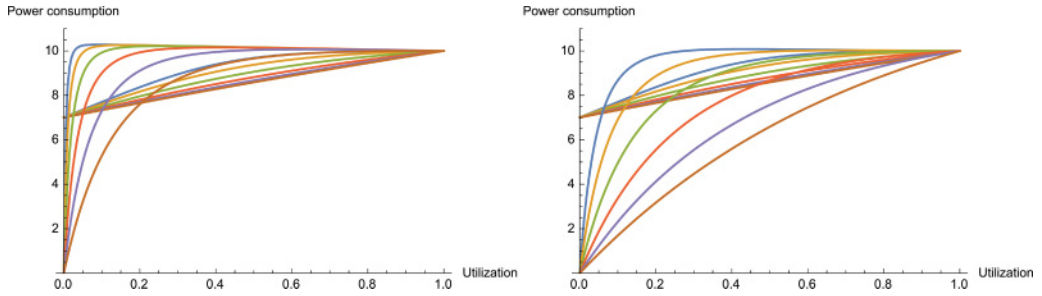


Fig. 16. Power consumption (Watts) for the (OFF, UP, BUSY, DOWN) policy (plots starting from the origin) and for the (IDLE, ON) policy (plots starting from 7W on the vertical axis) under increasing load and for various batch sizes (batch size increasing from bottom to top as 1, 2, 4, 8, 16, and 32). Mean power-up/power-down times: 600ms/300ms in the left-hand graph, 60ms/30ms in the right-hand graph. Threshold, $n = 5$.

4-state policy. To this end, we plotted the corresponding graphs (with thresholds 5 and 2) for power-up/down time 60ms/30ms to the right of those for 600ms/300ms in each figure. This shows that the 4-state policy is advantageous at much higher utilizations, as expected with the 10 times decreased overhead.

Note that, at a mean power-up/down time of 60ms/30ms, batch size 1 gives a higher ratio than for batch sizes 2 and 4 at utilizations above about 0.4, but definitely not below; in other words, the curves cross. This interesting behavior is probably due to the more rapid decrease in latency observed for unit batches in Section 4.1, Figures 4 and 5.

However, it must be noted that our choice of PE metric, although exhibiting the right qualitative behavior for any one policy as load or other parameters (e.g., batch size) vary, does not necessarily provide a good quantitative comparison between different policies: for example, the metric for each policy could be scaled by arbitrary different factors to preserve their own consistency with respect to various parameters but be meaningless as a quantitative comparison with another policy. For example, we could modify our PE metric to become

$$\text{PE_Metric} = \frac{1}{\mathbb{E}[\text{Response Time}]^{\alpha}\,\mathbb{E}[\text{Watts}]^{\beta}}$$

for any positive constants α, β, without losing any of its quality of representation. It would be interesting to experiment with choices for these constants to compare policies more faithfully. In fact, we could consider the logarithm of the PE metric instead, so that

$$\log \text{PE_Metric} = \alpha \log \frac{1}{\mathbb{E}[\text{Response Time}]} + \beta \log \frac{1}{\mathbb{E}[\text{Watts}]}.$$

This is a simple weighted linear mixture of (logarithms of) standard measures, and choosing values for such weights follows a well-trodden path.

Finally, we compared the average (raw) power consumption of the 4-state (OFF, UP, BUSY, DOWN) and 2-state (IDLE, ON) policies for various batch sizes under increasing load in Figure 16. This represents the extreme case of a weighting with $\alpha = 0$ in the PE metric, for which we are interested only in energy saving, not in performance. The mean power-up/down times were 600ms/300ms on the left and 60ms/30ms on the right. The threshold value was $n = 5$ in both cases.

As expected, at low utilizations, the 4-state policy uses less power since devices are frequently switched off, using no power, whereas the 2-state policy always uses at least the Idle-power, that is, 7W, in our case. On the other hand, a highly utilized device is almost continually in use, thus requires 10W of power. Moreover, at higher utilizations, the 2-state policy uses less power because of the energy-cost of occasional power-ups

in the 4-state policy, which use 12W. The crossover regions for each batch size can be clearly seen in the graphs. Clearly, if energy efficiency were the only consideration, the best policy would be the 4-state one with a mean batch size of at least 32. However, this would incur a performance penalty, as shown in Figures 5 to 7.

5. RELATED WORK

Recent years have seen a renewed interest in the literature not only in novel mechanisms and application areas for energy/performance trade-offs but also for developing novel queuing theory concepts that can be helpful in their modeling. In addition, there is a growing interest from government regulators and major industry players in benchmarks for energy–performance trade-offs.

Readers are referred to Buzbee et al. [2013] for a recent survey of a wide range of proposed mechanisms for power savings and trade-offs in storage systems. This includes, for example, the PARAID "gear-shifting" RAID system that powers down some disks when idle or under light workload (migrating data between disks as appropriate) [Weddle et al. 2007], and the Hibernator system, which features several tiers of variable RPM drives; within each tier, all disks run at the same speed, which is adjusted according to workload [Zhu et al. 2005].

Gandhi et al. [2010] investigated energy–performance trade-offs in the context of server farms, focusing on an Energy–Response Time product (ERP) metric and four different scheduling policies. For any given scheduling policy, the ERP metric is defined as the product of the long-run average power consumed and the mean customer response time under that policy. Sleep states were investigated in the context of data centers [Gandhi et al. 2012], suggesting that a normalized performance-per-watt metric can benefit greatly from intelligent sleep-state deployment, and that the benefit increases with data-center scale. Yin et al. [2009] carried out a study of energy-saving/performance trade-offs in the context of grid resource management by applying and extending the theory of $M/G/1$ queues with vacations. Factors influencing energy and performance in data-center storage systems (including deployed file systems) were studied in Sehgal et al. [2010] while the disruptive impact of SSD technology on the energy and performance required to process database queries is considered in Schall et al. [2010]. It was concluded that it is necessary to completely reassess traditional rules of thumb related to circumstances under which it is appropriate to use different kinds of table scans (i.e., index-based or full-table scans).

The utility of the $M/G/1$ queuing model for energy–performance modeling is aptly demonstrated in Tudor and Teo [2013], in which the objective is to derive the energy consumption and response times of server workloads on low-power multicore systems. The model is based on the submodeling of CPU cores, memory, and I/O execution, and is parameterized from traces. Results for multiserver systems have proved more elusive, although some progress on the analysis of M/M/k systems with vacations was made by Levy and Yechiali [1976]. Gandhi et al. [2013] recently presented an exact closed-form analytical solution for the M/M/k/Setup model, in which there is a penalty for turning servers on, the M/K/k/Setup/DelayedOff model, in which idle servers turn off after a finite time, and the M/M/k/Setup/Sleep model, in which idle servers either turn off or enter a low-power state.

Matrix-analytic methods for M/G/1-like Markov processes, in which the generator matrix has the requisite form (block upper-Hessenberg), are also based on an embedded Markov chain and may provide an alternate approach to the generating function method for analyzing the EP queue [Neuts 1981, 1989]. In general, the matrix-analytic numeric approach is algorithmic in nature and would require truncation in our model, giving approximate results. However, closed-form results can also be obtained in certain cases, such as in the matrix-geometric method when the sought matrices (typically

written G or R) can be expressed in closed form. Furthermore, the so-called Invariant Subspace Method avoids truncation when the generating function of the generator matrix can be written as a rational matrix-polynomial [Sohraby and Akar 1997].

In fact, as discussed at length in Daniels [1999], the matrix-analytic approach is "essentially equivalent" to the functional equation approach that solves for generating functions, when applied to batched M/G/1 queues – more specifically, to DBMAP/G/1 queues (with discrete-time, batched Markovian arrivals). The only difficulty would appear to be the dependence between the queueing and service times in an EP queue with threshold $n = 2$, but an appropriate modification may be possible to obtain results equivalent to ours. Computing higher moments of the response time distribution would lead to a corresponding matrix-algorithm, based on differentiating Laplace transforms, which would require further manipulation to obtain a scalar iteration such as ours. Nevertheless, this analysis might be feasible.

6. CONCLUSION

We have investigated a queue – the EP Queue – which is a significant variant of the $M/G/1$ queue that has state-dependent service-time distributions, and incorporates power-up for first arrivals and power-down for idle periods. We derived exact results for the busy-time and response-time distributions (under certain simplifying assumptions such as Poisson arrivals but general-service times). From these, we obtained both power-consumption and response-time metrics, which together provide a single measure of goodness for energy–performance. We illustrated some of the energy–performance trade-offs numerically for storage systems operating with the (IDLE, ON) and (OFF, UP, BUSY, DOWN) energy-saving policies.

Although we have focused on storage-system performance, the EP queue is a multipurpose tool with a wide range of applications – for modeling any resource, or class of resources, that is required to provide high-quality service while conserving energy. The idea is that energy savings can be made by switching off devices when they are idle, rather than continuing to consume power, albeit at a reduced rate. The price to pay is that energy consumption during powering up is typically greater than that in normal operating mode, and the power-up period may be relatively long compared to service times. In our numerical comparisons of storage-system energy-saving policies, we confirmed that increasing the batch size of the offered load causes response time to increase but allows more opportunities for energy-saving – an obvious trade-off. Quantitatively, we were able to establish (given the assumptions of the model):

(1) For response times,
   —The performance penalty on latency of the (OFF, UP, BUSY, DOWN) energy-saving policy compared to the (IDLE, ON) policy. The magnitude of this penalty can be quite large at low utilizations (throughputs), but it decreases at higher utilizations and with mean batch size (see Figures 4 and 5). Of course, conversely, it is at low utilizations that the (IDLE, ON) policy has the biggest penalty with respect to energy consumption, highlighting the trade-off issue.
   —With small batches, the latency of the (OFF, UP, BUSY, DOWN) policy decreases as utilization increases up to a "knee," after which the queue starts to saturate (see Figures 4 and 5). The knee decreases as the batch size increases, and for a mean batch size above 8, it disappears – that is, the latency increases monotonically with utilization. The effect becomes less pronounced as power-up/down times increase.
   —Table I gives a more detailed picture of the effect of varying the batch size and power-up/down time at different utilizations. The magnitude of the increase in both the mean response time and its standard deviation can be seen as the mean

batch size and power-up/down time increase. The standard deviation of response time is smaller than the mean, partly reflecting the small variability in the power-up/down times (approximated by Erlang-4 distributions) as well as the Central Limit Theorem at high utilizations. Table I also summarizes properties of the tail of the response time distribution, in particular, how the quantiles move to the right as the utilization and power-up/down times increase. Some of these conclusions are displayed graphically in Figures 8, 9, and 11.

(2) For energy consumption,
   —At low utilizations, considerable energy is saved by powering off devices.
   —The threshold region of utilization, above which the simpler (IDLE, ON) policy saves more energy because of the power-ups in the (OFF, UP, BUSY, DOWN) policy, is clearly identified. This is near only about 12% when the mean batch size is 16 with current technology (mean power-up/down time of 600ms/300ms), so that a mean batch size of at least 16 is required. However, if the power-up/down time could be reduced by a factor of 10, the threshold would be about 40% for mean batch size 8 and almost 100% for mean batch size 16 and above.

(3) For the energy–performance trade-off,
   —A joint energy–performance metric, which combines power usage and latency into one "bigger is better" metric, so that a single number can be used to compare quantitatively the overall efficacy of scheduling policies to optimize response time subject to energy constraints. Of course, the relative importance of high performance versus low energy usage will vary according to application, installation objectives, and legal requirements, but our metric is parameterizable to reflect this.
   —The graphs that we obtained in Figures 14 and 15, which show clearly the critical points for choosing appropriate parameters: when the mean power-up/down time is 600ms/300ms, utilization must be very low for the 4-state policy to have any benefit – below 6% even at batch size 64. If the mean power-up/down time were reduced by a factor of 10, these critical points become highly significant: for example, for mean batch size 16, the 4-state policy wins at utilizations below 25%; for mean batch size 32, the threshold becomes about 70%; for mean batch size 64, the 4-state policy is always better.

Energy usage in data centers is increasingly recognized as a problem by industry and government agencies. In this context, the EPA (Environmental Protection Agency in the United States) has recently created a benchmark to evaluate the performance and power consumption of data-center storage systems. In recent years, international efforts have also been ongoing in the Standard Performance Evaluation Corporation (SPEC) community to develop a comprehensive power and performance benchmarking methodology Standard Performance Evaluation Corporation [2011]; in fact, a recent SPEC benchmark (SPECvirt_sc2013) is targeted at aspects of performance/power in virtualized data centers, including storage.

Realistically, in the case of contemporary storage systems, we conclude that it is rarely worthwhile to save energy, while keeping good performance, by switching devices off. However, if power-up/down times can be reduced, the opposite applies. Moreover, in other systems in which shutting down devices can save energy – for example, in networks of solar-powered sensors – these overheads may be significantly less, rendering our model of immediate use.

Much work remains to be done in this exciting area of energy–performance trade-offs as new technologies such as storage class memories are encroaching on traditional storage devices, which themselves are also evolving in order to compete. Storage class memories, in which low-power modes are easier to apply without significant latency penalties, are an important area for future investigation based on the present work.

APPENDIX: PROOFS OMITTED FROM MAIN TEXT

LEMMA 2.1. *Let* $S_0 = D + U + S_{B_1+\cdots+B_{N_D+N_U}+\boldsymbol{I}_{N_D=0}B_0}$ *and* $\sigma(z) = \mathbb{E}[\mathbb{E}[z^{N_{S_0}+\boldsymbol{I}_{N_D=0}B_0} \mid U, D]]$. *Then, the pgf of the probability mass function* $q^c_{0\cdot}$ *is* $\sigma(z)/z$, *where*

$$\sigma(z) = S^*_n U^*[D^* - (1 - G(z))D^*(\lambda)] + \sum_{k=1}^{n-1}(S^*_k - S^*_n)z^k \sum_{j=1}^{k} a_{k-j}(U, G, 0)a_j(D, G, 0)$$
$$+ D^*(\lambda)\sum_{k=0}^{\infty} z^k a_k(U, G, 0) \sum_{\ell=1}^{n-k-1} \frac{1}{\ell!}G^{(\ell)}(0)z^\ell(S^*_{k+\ell} - S^*_n),$$

where the argument of an LST is taken to be $\lambda(1 - G(z))$ *when unspecified.*

PROOF.

$$\begin{aligned}\sigma(z) &= \mathbb{E}\big[\mathbb{E}\big[z^{N_{S_0}+\boldsymbol{I}_{N_D=0}B_0} \mid U, D\big]\big] \\ &= \mathbb{E}\big[\mathbb{E}\big[z^{B_1+\cdots+B_{N_D+N_U}+N_{S_{B_1+\cdots+B_{N_D+N_U}+\boldsymbol{I}_{N_D=0}B_0}+\boldsymbol{I}_{N_D=0}B_0}} \mid U, D\big]\big] \\ &= \mathbb{E}\big[\mathbb{E}\big[z^{B_1+\cdots+B_{N_D+N_U}+\boldsymbol{I}_{N_D=0}B_0} S^*_{B_1+\cdots+B_{N_D+N_U}+\boldsymbol{I}_{N_D=0}B_0}(\lambda(1 - G(z))) \mid U, D\big]\big],\end{aligned}$$

where B_0 is a batch size random variable and the B_k are the mutually independent sizes of the k^{th} batch ($1 \le k \le N_U + N_D$). Let there be i and j task arrivals in times U and D with probabilities $p_i(U) = \frac{1}{i!}\frac{\partial^i}{\partial x^i}e^{-\lambda U(1-G(x))}|_{x=0}$ and $p_j(D) = \frac{1}{j!}\frac{\partial^j}{\partial x^j}e^{-\lambda D(1-G(x))}|_{x=0}$, respectively. Using k to denote $i + j$, we have, since D and U are independent, and interchanging the summation and differentiation operations,

$$\begin{aligned}\sigma(z) &= \sum_{\ell=1}^{\infty}\sum_{k=0}^{\infty}\sum_{j=0}^{k} z^{k+\ell\boldsymbol{I}_{j=0}} S^*_{k+\ell\boldsymbol{I}_{j=0}} \frac{1}{\ell!(k-j)!j!}G^{(\ell)}(0)\frac{\partial^{k-j}}{\partial x^{k-j}}\frac{\partial^j}{\partial y^j}\mathbb{E}\big[e^{-\lambda U(1-G(x))}e^{-\lambda D(1-G(y))}\big]\big|_{x=0,y=0} \\ &= \sum_{k=0}^{\infty} z^k a_k(U, G, 0)D^*(\lambda)\sum_{\ell=1}^{\infty}\frac{1}{\ell!}G^{(\ell)}(0)z^\ell S^*_{k+\ell} \\ &\quad + \sum_{k=0}^{\infty}\sum_{j=1}^{k} z^k a_{k-j}(U, G, 0)a_j(D, G, 0)\sum_{\ell=1}^{\infty}\frac{1}{\ell!}G^{(\ell)}(0)S^*_k \\ &= \sum_{k=0}^{\infty} z^k a_k(U, G, 0)D^*(\lambda)\left[\sum_{\ell=1}^{n-k-1}\frac{1}{\ell!}G^{(\ell)}(0)z^\ell(S^*_{k+\ell} - S^*_n) + S^*_n G(z)\right] \\ &\quad + \sum_{k=1}^{n-1}(S^*_k - S^*_n)\sum_{j=1}^{k} z^k a_{k-j}(U, G, 0)a_j(D, G, 0) + \sum_{k=1}^{\infty}\sum_{j=1}^{k} z^k a_{k-j}(U, G, 0)a_j(D, G, 0)S^*_n \\ &= U^*D^*(\lambda)S^*_n G(z) + D^*(\lambda)\sum_{k=0}^{\infty} z^k a_k(U, G, 0)\sum_{\ell=1}^{n-k-1}\frac{1}{\ell!}G^{(\ell)}(0)z^\ell(S^*_{k+\ell} - S^*_n) \\ &\quad + \sum_{k=1}^{n-1}(S^*_k - S^*_n)\sum_{j=1}^{k} z^k a_{k-j}(U, G, 0)a_j(D, G, 0) + S^*_n\sum_{j=1}^{\infty}\sum_{k=0}^{\infty} a_k(U, G, 0)z^k a_j(D, G, 0)z^j.\end{aligned}$$

The result then follows. □

PROPOSITION 2.2.

$$\Pi^W(z) = \frac{p_0[S^*_n(\lambda(1 - G(z))) - \sigma(z)] + \sum_{j=1}^{n-1}[S^*_n(\lambda(1 - G(z))) - S^*_j(\lambda(1 - G(z)))]p_j z^j}{S^*_n(\lambda(1 - G(z))) - z}.$$

PROOF. Let the queue length random variable just after transition $m \geq 0$ (i.e., the m^{th} departure) be denoted L_m, with $L_0 = 0$ and S_0 as defined in Lemma 2.1. Then, for $m \geq 1$,

$$L_m = L_{m-1} + N_{S_{L_{m-1}}} - 1 + \boldsymbol{I}_{L_{m-1}=0}\boldsymbol{I}_{N_D=0}B_0,$$

where $\boldsymbol{I}$ is the indicator function and B_0 is a batch size random variable. Hence, the pgf of the queue length just after the m^{th} transition is

$$\begin{aligned}\Pi_m^W(z) &= \mathbb{E}[z^{L_m}] = \mathbb{E}\big[\mathbb{E}\big[\mathbb{E}\big[z^{L_{m-1}+N_{S_{L_{m-1}}}-1+\boldsymbol{I}_{L_{m-1}=0}\boldsymbol{I}_{N_D=0}B_0} \mid L_{m-1}, U, D\big] \mid L_{m-1}\big]\big] \\ &= z^{-1}\mathbb{E}\big[z^{L_{m-1}}\mathbb{E}\big[\mathbb{E}\big[z^{N_{S_{L_{m-1}}}+\boldsymbol{I}_{L_{m-1}=0}\boldsymbol{I}_{N_D=0}B_0} \mid L_{m-1}, U, D\big] \mid L_{m-1}\big]\big].\end{aligned}$$

Then, letting $m \to \infty$, at equilibrium,

$$\begin{aligned}z\Pi^W(z) &= \sum_{j=1}^{\infty} p_j z^j \mathbb{E}\big[z^{N_{S_j}}\big] + p_0\mathbb{E}\big[\mathbb{E}\big[z^{N_{S_0}+\boldsymbol{I}_{N_D=0}B_0} \mid U, D\big]\big] \\ &= \sum_{j=1}^{\infty} p_j z^j S_j^*(\lambda(1-G(z))) + p_0\mathbb{E}\big[\mathbb{E}\big[z^{N_{S_0}+\boldsymbol{I}_{N_D=0}B_0} \mid U, D\big]\big] \\ &= \sum_{j=0}^{\infty} p_j z^j S_n^* + \sum_{j=1}^{n-1} p_j z^j (S_j^* - S_n^*) - p_0 S_n^* + p_0\sigma(z) \\ &= \Pi^W(z)S_n^* + \sum_{j=1}^{n-1} p_j z^j (S_j^* - S_n^*) - p_0 S_n^* + p_0\sigma(z),\end{aligned}$$

omitting the LSTs' arguments $(\lambda(1-G(z)))$ for brevity. □

LEMMA 2.3. *The number of task arrivals in a period that begins when the chain enters state 0 and ends when the next task enters service has pgf*

$$\beta(z) = S_1^*(\lambda(1-G(z))) - S_1^*(\lambda)(1-U^*(\lambda(1-G(z)))(D^*(\lambda(1-G(z))) - (1-G(z))D^*(\lambda))).$$

PROOF. The number of task arrivals in the said period is N_{S_1} if this is positive or else the number of arrivals in the ensuing power-down and power-up periods, D and U, together with one additional batch if there are no arrivals during the power-down period. Thus,

$$\begin{aligned}\beta(z) &= \mathbb{E}\big[z^{N_{S_1}+\boldsymbol{I}_{N_{S_1}=0}(\boldsymbol{I}_{N_D=0}B_0+N_D+N_U)}\big] \\ &= S_1^* - 1 + \mathbb{E}\big[z^{\boldsymbol{I}_{N_{S_1}=0}(\boldsymbol{I}_{N_D=0}B_0+N_D+N_U)}\big],\end{aligned}$$

recalling that B_0 is a batch-size random variable, using the result that, for any X, Y and constant z, $z^{X+\boldsymbol{I}_{X=0}Y} = z^X + z^{\boldsymbol{I}_{X=0}Y} - 1$, and, again, omitting the LSTs' arguments $(\lambda(1-G(z)))$ for brevity. Therefore,

$$\begin{aligned}&= S_1^* - 1 + \mathbb{E}\big[\mathbb{E}\big[z^{\boldsymbol{I}_{N_{S_1}=0}(\boldsymbol{I}_{N_D=0}B_0+N_D+N_U)} \mid N_{S_1}\big]\big] \\ &= S_1^* - 1 + S_1^*(\lambda)\mathbb{E}\big[z^{\boldsymbol{I}_{N_D=0}B_0+N_D+N_U}\big] + (1-S_1^*(\lambda)) \\ &= S_1^* - S_1^*(\lambda)(1-U^*(D^* - 1 + G(z)D^*(\lambda) + 1 - D^*(\lambda))) \quad \square\end{aligned}$$

This gives the following result for the pgf of the queue length probabilities at the start of a service period in an EP queue at equilibrium.

PROPOSITION 2.4. *The probability-generating function of the queuing time in an equilibrium EP queue with threshold n is*

$$\Pi^Q(z) = \frac{\sum_{j=1}^{n-2} \pi_j z^j (S_n^*(\lambda(1-G(z))) - S_{j+1}^*(\lambda(1-G(z)))) + \pi_0(S_n^*(\lambda(1-G(z))) - \beta(z))}{S_n^*(\lambda(1-G(z))) - z}$$

or, in alternate form, omitting the arguments of the Laplace transforms when they are $\lambda(1-G(z))$*,*

$$\Pi^Q(z) = \frac{\sum_{j=0}^{n-2} \pi_j z^j (S_n^* - S_{j+1}^*) + \pi_0 S_1^*(\lambda)\big(1-(D^* - (1-G(z))D^*(\lambda))U^*\big)}{S_n^* - z},$$

where the probabilities $\pi_1, \ldots, \pi_{n-2}$ *are given by the equations*

$$\pi_i = \pi_0 q_{0i} + \sum_{j=1}^{i+1} \pi_j r_{j+1,i-j+1} \tag{5}$$

for $i = 0, \ldots, n-3$ *and the normalization condition,*

$$\pi_0(\lambda G'(1) S_n^{*\prime}(0) + \beta'(1)) + \lambda G'(1) \sum_{j=1}^{n-2} \pi_j (S_n^{*\prime}(0) - S_{j+1}^{*\prime}(0)) = \lambda G'(1) S_n^{*\prime}(0) + 1.$$

The term $q_{0i} = \beta^{(i+1)}(0)$*, where* $\beta(z) = S_1^* - S_1^*(\lambda)(1 - U^*(D^* - (1-G(z))D^*(\lambda)))$*.*

PROOF. Let the queue length random variable just after the m^{th} task enters service be denoted J_m, with $J_0 = 0$. Then, for $m \geq 1$,

$$J_m = J_{m-1} - 1 + N_{S_{J_{m-1}+1}} + \boldsymbol{I}_{J_{m-1}=0} \boldsymbol{I}_{N_{S_{J_{m-1}+1}}=0} (\boldsymbol{I}_{N_D=0} B_0 + N_D + N_U). \tag{6}$$

Hence, the pgf of the queue length just after the m^{th} task enters service is

$$\begin{aligned}\Pi_m^Q(z) &= \mathbb{E}[z^{J_m}] \\ &= z^{-1}\mathbb{E}\left[z^{J_{m-1}}\mathbb{E}\left[\mathbb{E}\left[z^{N_{S_{J_{m-1}+1}} + \boldsymbol{I}_{J_{m-1}=0}\boldsymbol{I}_{N_{S_{J_{m-1}+1}}=0}(\boldsymbol{I}_{N_D=0}B_0 + N_D + N_U)} \mid J_{m-1}, U, D\right] \mid J_{m-1}\right]\right].\end{aligned}$$

As $m \to \infty$, at equilibrium, when this exists, $\Pi_m^Q(z) \to \Pi^Q(z)$ and

$$\begin{aligned} z\Pi^Q(z) &= \sum_{j=1}^{\infty} \pi_j z^j S_{j+1}^* + \pi_0 \mathbb{E}\left[z^{N_{S_1} + \boldsymbol{I}_{N_{S_1}=0}(\boldsymbol{I}_{N_D=0}B_0 + N_D + N_U)}\right] \\ &= \Pi^Q(z) S_n^* + \sum_{j=1}^{n-2} \pi_j z^j (S_{j+1}^* - S_n^*) + \pi_0(-S_n^* + \beta(z)) \end{aligned}$$

so that

$$(S_n^* - z)\Pi^Q(z) = \sum_{j=1}^{n-2} \pi_j z^j (S_n^* - S_{j+1}^*) + \pi_0(S_n^* - \beta(z)) \tag{7}$$

$$= \sum_{j=0}^{n-2} \pi_j z^j (S_n^* - S_{j+1}^*) + \pi_0 S_1^*(\lambda)(1 - U^*(D^* - (1-G(z))D^*(\lambda))). \tag{8}$$

Equations (5) are obvious apart from the term q_{0i}, which is the probability that there are $i+1$ arrivals before the next task enters service after the chain is in state 0, that is, $q_{0i} = \beta^{(i+1)}(0)$, by Lemma 2.3.

Finally, normalization requires that $\Pi^Q(z) = 1$ whereby, using l'Hopital's rule in Equation (7),

$$\lambda G'(1)S_n^{*\prime}(0) + 1 = \pi_0(\lambda G'(1)S_n^{*\prime}(0) + \beta'(1)) + \lambda G'(1)\sum_{j=1}^{n-2}\pi_j(S_n^{*\prime}(0) - S_{j+1}^{*\prime}(0)) \quad \square$$

COROLLARY. *The m^{th} factorial moment $Q_{\langle m\rangle}$ of the queuing time, $m = 0, 1, 2, \ldots$, is given by the following recurrence:*

$$\begin{aligned} Q_{\langle m+1\rangle} &= \frac{\sum_{k=0}^{m}\left((m+2)!/k!\right)a_{m-k+2}(S_n, G, 1)Q_{\langle k\rangle} - \alpha^{(m+2)}(1)}{(m+2)(1+\lambda G'(1)S_n^{*\prime}(0))} \\ Q_{\langle 0\rangle} &= 1, \end{aligned}$$

where $\alpha(z) = \sum_{j=1}^{n-2}\pi_j z^j(S_n^*(\lambda(1-G(z))) - S_{j+1}^*(\lambda(1-G(z)))) + \pi_0(S_n^*(\lambda(1-G(z))) - \beta(z))$.

PROOF. From the proposition, $[S_n^*(\lambda(1-G(z))) - z]\Pi^Q(z) = \alpha(z)$. Differentiating both sides $m+2$ times and setting $z = 1$ yields the required result. $\square$

REFERENCES

C. Beath, I. Becerra-Fernandez, J. Ross, and J. Short. 2012. Finding value in the information explosion. *MIT Sloan Management Review* 53, 18–20.

D. Bertsimas and D. Nakazato. 1995. The distributional Little's law and its applications. *Operations Research* 43, 298–310.

B. Buzbee, W. Wang, and A. A. Wang. 2013. Power-saving approaches and trade-offs for storage systems. Technical Report TR-130627, Department of Computer Science, Florida State University, Tallahassee, FL.

T. Daniels. 1999. *Asymptotic Behaviour of Queuing Systems*. PhD dissertation. University of Antwerp, Antwerp, Belgium.

A. Gandhi, V. Gupta, M. Harchol-Balter, and M. A. Kozuch. 2010. Optimality analysis of energy–performance trade-off for server farm management. *Performance Evaluation* 67, 11, 1155–1171.

A. Gandhi, M. Harchol-Balter, and M. Kozuch. 2012. Are sleep states effective in data centers? In *Proceedings of the International Conference on Green Computing (IGCC'12)* San Jose, CA.

A. Gandhi, S. Doroudi, M. Harchol-Balter, and A. Scheller-Wolf. 2013. Exact analysis of the M/M/K/Setup class of Markov chains via recursive renewal reward. *SIGMETRICS Performance Evaluation Review* 41, 1 153–166.

R. Haji and G. Newell. 1971. A relation between stationary queue and waiting time distributions. *Journal of Applied Probability* 8, 617–620.

J. Koomey. 2011. Growth in data center electricity use 2005 to 2010. Analytics Press, Berkeley, CA.

Y. Levy and U. Yechiali. 1976. An m/m/s queue with servers' vacations. *INFOR* 14, 153–163.

F. Meng, L. Zhou, S. Uttamchandani, and X. Ma. 2013. Redefining ESXi IO multipathing in the flash era. *VMWare Technical Journal* 2, 13–18.

K. Sohraby and N. Akar. 1997. An invariant subspace approach in m/g/l and g/m/l type Markov chains. *Stochastic Models* 13, 381–416.

M. F. Neuts. 1981. *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*. Johns Hopkins University Press, Baltimore, MD.

M. F. Neuts. 1989. *Structured Stochastic Matrices of M/G/1 Type and their Applications*. Marcel Dekker, New York, NY.

N. M. Patel. 2012. Performance implications of flash and storage class memories. In *Proceedings of ACM SIGMETRICS 2012*, London, UK.

D. Schall, V. Hudlet, and T. Härder. 2010. Enhancing energy efficiency of database applications using SSDS. In *Proceedings of the 3rd C* Conference on Computer Science and Software Engineering, (C3S2E'10)*, ACM, New York, NY, 1–9.

P. Sehgal, V. Tarasov, and E. Zadok. 2010. Optimizing energy and performance for server-class file system workloads. *ACM Transactions on Storage* 6, 3, 10:1–10:31.

Standard Performance Evaluation Corporation. 2011. SPEC power and performance benchmark methodology v2.1.

B. M. Tudor and Y. M. Teo. 2013. On understanding the energy consumption of arm-based multicore servers. *SIGMETRICS Performance Evaluation Review* 41, 1, 267–278.

C. Weddle et al. 2007. PARAID: A gear-shifting power-aware RAID. *ACM Transactions on Storage* 3, 3.

F. Yin, C. Jiang, R. Deng, and J. Yuan. 2009. Grid resource management policies for load-balancing and energy-saving by vacation queuing theory. *Computer Electronics Engineering* 35, 6, 966–979.

Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. 2005. Hibernator: Helping disk arrays sleep through the winter. *SIGOPS Operating Systems Review* 39, 5, 177–190.

Received November 2014; revised August 2015; accepted August 2015