2025 36th International Teletraffic Congress (ITC 36)

A Fast Algorithm for Multiserver Queueing Systems with Setup Times and Power-saving Modes

Thu Le-Anh
Graduate School of Science and Technology
University of Tsukuba
Tsukuba, Japan
0000-0002-1474-134X

Tuan Phung-Duc
Institute of Systems and Information Engineering
University of Tsukuba
Tsukuba, Japan
0000-0002-5002-4946

***Abstract*—Multi-server queueing models with setup times have been extensively investigated due to their applications in data centers, which have rapidly expanded in recent years. These models employ an ON-OFF policy to reduce energy consumption for idle servers, though this can increase delay due to setup processes. To minimize the drawback of the ON-OFF policy, a setup queueing model with a power-saving policy was recently proposed in [9], enabling servers in a power-saving mode to process incoming jobs while simultaneously setting up to a normal mode. Our paper considers the same model but adopts a generating function approach rather than the matrix geometric method as in [9]. Our approach provides exact expressions for the joint stationary queue length distribution, generating functions, and factorial moments of any order. By exploiting the special structure of the Markov chain, the generating function approach achieves significant reductions in computational complexity compared to existing methods. In addition, this paper presents numerical experiments to evaluate the performance-energy trade-off using an existing speed-based cost function.**

***Index Terms*—Setup, Power-saving, Data center**

I. Introduction

The rapid adoption of artificial intelligence is significantly driving the expansion of data centers. Large-scale data centers are critical for meeting the demands of modern digital life, but they require a massive amount of energy to operate. Data centers consumed 460 TWh in 2022, which could rise to over 1,000 TWh by 2026 in the worst-case scenario [1]. While some may consider this an unavoidable cost, the reality is that servers spend a substantial portion of their time idle. Indeed, the servers are typically utilized only 10–30% of the time, yet during idle periods, they consume around 60% of their peak [2]. To reduce idle power consumption, idle servers are often switched to an off state, which we refer to as an ON-OFF policy. However, servers that are turned off need some setup time to become active again, during which they consume energy without performing any tasks. Hence, choosing whether and when to switch off a server is far from trivial. Turning off idle servers poses a trade-off between energy conservation and system performance.

To mitigate the drawback of the ON-OFF policy, many policies have been proposed, such as s-staggered setup policy [3], [4], delayed-off policy [5], [6], and sleep policy [7], [8]. The s-staggered policy restricts the number of setup servers to s, leading to lower power consumption. To reduce delays caused by setup processes, the delayed-off policy allows idle time before turning off, while the sleep policy enables idle servers to either shut down or go to sleep modes. Although each of the above policies offers distinct benefits, they share the drawbacks of wasted energy and delays due to the server's inability to process jobs during setup processes. To address this issue, a setup queueing model with a power-saving policy, where a server in power-saving mode can process a job and setup simultaneously, was proposed in [9]. This policy allows an idle server to remain in the power-saving mode and be set up to its normal mode when a job arrives. While maintaining idle servers in the power-saving mode requires some energy rather than completely shutting them down, it reduces delay by enabling servers to process jobs without waiting for setup processes to complete.

Although queueing models with setup times have been extensively investigated in the literature, [9] is the first to consider such a power-saving policy in a multiserver setup queueing model. Some closely related models proposed working vacation (WV) policies that allow servers to process jobs during the vacations [10]–[12]. Servi and Finn [12] first introduced an M/M/1 queue with WV, where the server works at a lower service rate rather than completely stopping the service during a vacation. This model was generalized to an M/G/1 queue by Wu and Takagi [10]. In [11], a setup is considered in a GI/M/1 queue with a single WV and vacation interruption governed by the Bernoulli rule. Specifically, when the server is in an off state, an arriving job initiates a setup process to normal mode. Moreover, during a WV, if a job is present, the server may either interrupt the vacation and switch to normal mode with probability p or continue the vacation with probability $1 - p$. The model in [11] is similar to the delayed-off model, where the system is closed if there is no job when a vacation ends, whereas the power-saving model in [9] keeps servers in power-saving mode even when the system is empty. In WV models, a vacation starts when there is no waiting job and ends after the vacation period. In contrast, a setup process in the power-saving model occurs upon arrival, continuing until service completion or the server switches to normal mode. Hence, the vacation schedule is independent

978-3-948377-03-8/19/$31.00

of the queue length, whereas the setup rates depend on the queue length, which makes the model with setup times more complex.

Motivated by the applications of the power-saving modes in data centers, this paper considers the same model as in [9] but with a different perspective in terms of method for performance evaluation of large-scale data centers. While [9] uses the matrix geometric method to obtain the steady-state probabilities and system performance, this paper employs a generating function approach to derive the joint queue length distribution. The advantage of the generating function approach is that it provides exact expressions for the joint stationary queue length distribution, generating functions, and factorial moments of any order. The generating function approach allows utilizing the special structure of the non-homogeneous part of the Markov chain to have significant reductions of the computational complexity in comparison with existing methods in the literature [9], [13], [14].

The closest to our work in terms of the method is the paper of Phung-Duc [15]. The author considers a common multi-server queue with setup times, that is, M/M/k/Setup, and obtains exact solutions for the joint stationary queue length distribution of the model via the generating function approach and matrix analytic method. The M/M/k/Setup model and some variants are studied by Gandhi et al. [13] using the recursive renewal reward method to determine the Laplace transform of the response time and exact mean values. The authors point out that applying the generating function approach to the model, even for a simple case, the M/M/2/Setup, is incredibly complex. However, the use of the generating function approach is made possible by exploiting a special structure of the Markov chain in our model, which is even more complex than the M/M/k/Setup. Furthermore, the generating function approach yields solutions of a form similar to the form obtained by the Clear Analytic by Phases (CAP) in [16] and allows for a speedy computation compared to the matrix geometric method.

The rest of this paper is organized as follow. Section II describes the model in detail and Section III presents the analysis of the model via generating functions. Section IV is devoted to performance measures and numerical experiments are presented in Section V. Finally, concluding remarks are presented in Section VI.

II. Model

A. Model description

We consider an M/M/c/Setup queueing model with the power-saving policy defined in [9]. In this model, all idle servers are in power-saving modes instead of shutting down. Jobs arrive at the system according to a Poisson process with rate λ. When a job arrives, an idle server (if any) will process the job immediately and be set up to a normal mode simultaneously. A server needs some setup time to return to the normal mode. We assume that the setup times are exponentially distributed with mean $1/\alpha$. Upon service completion, if no jobs request service, the server switches to the power-saving mode and cancels its ongoing setup process if existing. Otherwise, it will immediately process the job in its current mode, and the setup process will continue if it is in the power-saving mode. During the power-saving mode, jobs are served at a rate μ_s. When the server is in the normal mode, jobs are served at a rate μ.

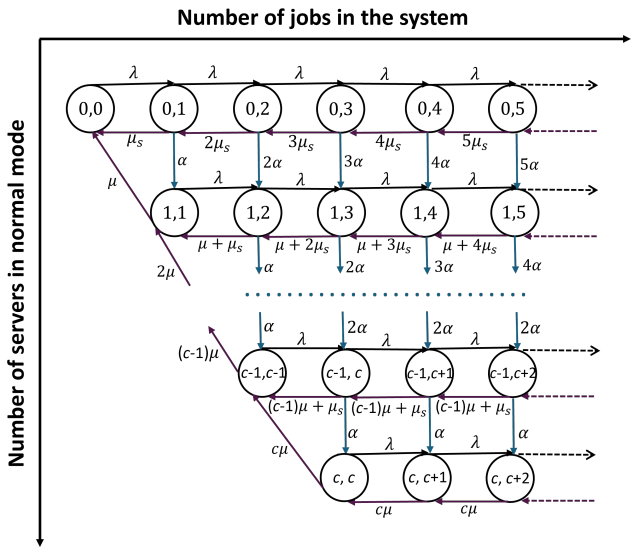

Fig. 1. State-transition diagram.

Let i denote the number of servers in the normal mode and j denote the number of jobs in the system, respectively. Then, the number of servers in the setup process is given by $\min(j-i, c-i)$. We assume that servers in this system operate under the First Come First Served (FCFS) policy. Fig. 1 shows the state transition diagram.

B. Markov chain and notations

It can be seen that the stability condition for the system is $\lambda < c\mu$ because all the servers are eventually in normal mode if the number of jobs in the system is large enough. Let $N(t)$ and $L(t)$ denote the number of servers in the normal mode and the number of jobs in the system at time t, respectively. In the current setting, the two-dimensional process $\{X(t) = (N(t), L(t)); t \geq 0\}$ forms a Markov chain in the state space

$$S = \{(i, j); i = 0, 1, \ldots, c, j = i, i+1, \ldots\}.$$

Let

$$\pi_{i,j} = \lim_{t\to\infty} \mathbb{P}(N(t) = i, L(t) = j), \quad (i, j) \in S.$$

We define generating functions $\Pi_i(z) = \sum_{j=i}^{\infty} \pi_{i,j} z^{j-i}$, for $i = 0, 1, \ldots, c$. We are interested in finding explicitly the generating functions $\Pi_i(z)$, the factorial moments defined by $\Pi_i^{(n)}(1)$, where $f^{(n)}(x)$ denotes the n-th derivative of $f(x)$, and $\pi_{i,j}$ for $(i, j) \in S$.

Definition 1. *For $x \in \mathbb{R}$, the Pochhammer symbol is defined as follows:*

$$(x)_n = \begin{cases} 1 & n = 0, \\ \prod_{k=1}^{n}(x+k-1) & n \in \mathbb{N} = \{1,2,3,\ldots\}. \end{cases}$$

III. Generating function approach

In this section, we present explicit expressions for the generating functions and the factorial moments, which yields exact solutions for the joint stationary queue length distribution of our model. These expressions do not involve *limits*, and they can be exactly calculated using a finite procedure.
The balance equations for the case $i = 0$ are given by

$$\lambda\pi_{0,0} = \mu_s\pi_{0,1} + \mu\pi_{1,1}, \quad j = 0, \tag{1}$$

$$(\lambda + j\alpha + j\mu_s)\pi_{0,j} = \lambda\pi_{0,j-1} + (j+1)\mu_s\pi_{0,j+1}, \tag{2}$$
$$j = 1, 2, \ldots, c-1,$$

$$(\lambda + c\alpha + c\mu_s)\pi_{0,j} = \lambda\pi_{0,j-1} + c\mu_s\pi_{0,j+1}, \quad j \geq c. \tag{3}$$

Letting $\widehat{\Pi}_0(z) = \sum_{j=c}^{\infty}\pi_{0,j}z^j$, we have

$$\Pi_0(z) = \sum_{j=0}^{c-1}\pi_{0,j}z^j + \widehat{\Pi}_0(z). \tag{4}$$

Multiplying (3) by z^j and summing over $j \geq c$, we have

$$(-\lambda z^2 + (\lambda + c\alpha + c\mu_s)z - c\mu_s)\widehat{\Pi}_0(z) = \lambda\pi_{0,c-1}z^{c+1} - c\mu_s\pi_{0,c}z^c. \tag{5}$$

Define $f_0(z) = -\lambda z^2 + (\lambda + c\alpha + c\mu_s)z - c\mu_s$. Because $f_0(0) = -c\mu_s < 0$, $f_0(1) = c\alpha > 0$ and $f_0(\infty) = -\infty$, $f_0(z)$ has two roots $z_0, \widehat{z}_0$ such that $0 < z_0 < 1 < \widehat{z}_0$. Then

$$z_0 = \frac{(\lambda + c\alpha + c\mu_s) - \sqrt{(\lambda + c\alpha + c\mu_s)^2 - 4\lambda c\mu_s}}{2\lambda},$$
$$\widehat{z}_0 = \frac{(\lambda + c\alpha + c\mu_s) + \sqrt{(\lambda + c\alpha + c\mu_s)^2 - 4\lambda c\mu_s}}{2\lambda}.$$

Substituting $z = z_0$ into (5), we have

$$\pi_{0,c} = \frac{\lambda}{c\mu_s}z_0\pi_{0,c-1}. \tag{6}$$

Rewriting (6), we have $\pi_{0,c} = b_c^{(0)}\pi_{0,c-1}$, where $b_c^{(0)} = \frac{\lambda}{c\mu_s}z_0$. Then, we derive a recursive scheme to determine $\pi_{0,j}$ for $j = 1, 2, \ldots, c$, by the following lemma.

Lemma 1.

$$\pi_{0,j} = b_j^{(0)}\pi_{0,j-1}, \quad 1 \leq j \leq c, \tag{7}$$

where

$$b_j^{(0)} = \frac{\lambda}{\lambda + j\alpha + j\mu_s - (j+1)\mu_s b_{j+1}^{(0)}},$$

for $j = c-1, c-2, \ldots, 1$.

Proof. The proof of Lemma 1 is given in Appendix A. □

The generating function $\widehat{\Pi}_0(z)$ is explicitly obtained as follows.

$$\widehat{\Pi}_0(z) = z^c\frac{A_{0,0}}{\widehat{z}_0 - z}, \tag{8}$$

where $A_{0,0} = \pi_{0,c-1}$.
Furthermore, from (8), we obtain

$$\pi_{0,j} = \frac{A_{0,0}}{\widehat{z}_0}\left(\frac{1}{\widehat{z}_0}\right)^{j-c}, \quad \text{for } j \geq c.$$

Remark 1. *At this moment, $\pi_{0,j}$ $(j \geq 1)$ are expressed in terms of $\pi_{0,0}$. In addition, $\pi_{1,1}$ is expressed in terms of $\pi_{0,j}$ $(j \geq 1)$ and then in terms of $\pi_{0,0}$ via the balance of the flows in and out the set of states $\{(0,j); j \geq 0\}$, i.e.,*

$$\mu\pi_{1,1} = \sum_{j=1}^{\infty}\min(j,c)\alpha\pi_{0,j}.$$

Differentiating (5) n times yields

$$f_0(z)\widehat{\Pi}_0^{(n)}(z) + nf_0'(z)\widehat{\Pi}_0^{(n-1)}(z) + \frac{n(n-1)}{2}f_0''(z)\widehat{\Pi}_0^{(n-2)}(z) = \lambda\pi_{0,c-1}(c-n+2)_n z^{c+1-n} - c\mu_s\pi_{0,c}(c-n+1)_n z^{c-n}.$$

Substituting $z = 1$ into this equation yields

$$\widehat{\Pi}_0^{(n)}(1) = \frac{n(\lambda - c\alpha - c\mu_s)\widehat{\Pi}_0^{(n-1)}(1) + n(n-1)\lambda\widehat{\Pi}_0^{(n-2)}(1)}{c\alpha} + \frac{\lambda}{c\alpha}\pi_{0,c-1}(c-n+2)_n - \frac{\mu_s}{\alpha}\pi_{0,c}(c-n+1)_n,$$

$$\Pi_0^{(n)}(1) = \sum_{j=0}^{c-1}\pi_{0,j}(j-n+1)_n + \widehat{\Pi}_0^{(n)}(1).$$

Now, we consider general cases where $i = 1, 2, \ldots, c-1$. The balance equations are as follows.

$$(\lambda + i\mu)\pi_{i,i} = \alpha\pi_{i-1,i} + (i\mu + \mu_s)\pi_{i,i+1} + (i+1)\mu\pi_{i+1,i+1}, \tag{9}$$

$$(\lambda + i\mu + (j-i)(\mu_s + \alpha))\pi_{i,j} = (j-i+1)\alpha\pi_{i-1,j} + \lambda\pi_{i,j-1} + (i\mu + (j-i+1)\mu_s)\pi_{i,j+1}, \quad i+1 \leq j \leq c-1, \tag{10}$$

$$(\lambda + i\mu + (c-i)(\mu_s + \alpha))\pi_{i,j} = (c-i+1)\alpha\pi_{i-1,j} + \lambda\pi_{i,j-1} + (i\mu + (c-i)\mu_s)\pi_{i,j+1}, \quad j \geq c. \tag{11}$$

Letting $\widehat{\Pi}_i(z) = \sum_{j=c}^{\infty}\pi_{i,j}z^{j-i}$, we have

$$\Pi_i(z) = \sum_{j=0}^{c-1}\pi_{i,j}z^{j-i} + \widehat{\Pi}_i(z). \tag{12}$$

Multiplying (11) by z^{j-i} and summing over $j \geq c$, we have

$$\Big(-\lambda z^2 + (\lambda + i\mu + (c-i)(\alpha + \mu_s))z - (i\mu + (c-i)\mu_s)\Big)\widehat{\Pi}_i(z) = (c-i+1)\alpha\widehat{\Pi}_{i-1}(z) + \lambda\pi_{i,c-1}z^{c-i+1} - (i\mu + (c-i)\mu_s)\pi_{i,c}z^{c-i}. \tag{13}$$

Define $f_i(z) = -\lambda z^2 + (\lambda + i\mu + (c-i)(\alpha + \mu_s))z - (i\mu + (c-i)\mu_s)$. Because $f_i(0) = -(i\mu + (c-i+1)\mu_s) < 0$, $f_i(1) = (c-i)\alpha > 0$ and $f_i(\infty) = -\infty$, $f_i(z)$ has two roots $z_i, \widehat{z}_i$ such that $0 < z_i < 1 < \widehat{z}_i$. Then

$$z_i = \frac{(\lambda + i\mu + (c-i)(\mu_s+\alpha)}{2\lambda} - \frac{\sqrt{(\lambda + i\mu + (c-i)(\mu_s+\alpha))^2 - 4\lambda(i\mu + (c-i)\mu_s)}}{2\lambda},$$

$$\widehat{z}_i = \frac{(\lambda + i\mu + (c-i)(\mu_s+\alpha)}{2\lambda} + \frac{\sqrt{(\lambda + i\mu + (c-i)(\mu_s+\alpha))^2 - 4\lambda(i\mu + (c-i)\mu_s)}}{2\lambda}.$$

Substituting $z = z_i$ into (13), we have

$$\pi_{i,c} = \frac{(c-i+1)\alpha\widehat{\Pi}_{i-1}(z_i) + \lambda\pi_{i,c-1}z_i^{c-i+1}}{(i\mu + (c-i)\mu_s)z_i^{c-i}}. \tag{14}$$

Rewriting (14), we have

$$\pi_{i,c} = a_c^{(i)} + b_c^{(i)}\pi_{i,c-1},$$

where

$$a_c^{(i)} = \frac{(c-i+1)\alpha\widehat{\Pi}_{i-1}(z_i)}{(i\mu + (c-i)\mu_s)z_i^{c-i}}, \quad b_c^{(i)} = \frac{\lambda z_i}{i\mu + (c-i)\mu_s}.$$

Then, we derive a recursive scheme to determine $\pi_{i,j}$ for $i+1 \le j \le c$ by the following lemma

Lemma 2.

$$\pi_{i,j} = a_j^{(i)} + b_j^{(i)}\pi_{i,j-1}, \quad j = i+1, i+2, \ldots, c, \tag{15}$$

where

$$a_j^{(i)} = \frac{(j-i+1)\alpha\pi_{i-1,j} + (i\mu + (j-i+1)\mu_s)a_{j+1}^{(i)}}{\lambda + i\mu + (j-i)(\mu_s+\alpha) - (i\mu + (j-i+1)\mu_s)b_{j+1}^{(i)}}, \tag{16}$$

$$b_j^{(i)} = \frac{\lambda}{\lambda + i\mu + (j-i)(\mu_s+\alpha) - (i\mu + (j-i+1)\mu_s)b_{j+1}^{(i)}}, \tag{17}$$

for $i = c-1, c-2, \ldots, i+1$. Moreover, we have

$$0 < a_j^{(i)}, \quad 0 < b_j^{(i)} < \frac{\lambda}{i\mu + (j-i)\mu_s}.$$

The generating function $\widehat{\Pi}_i(z)$ $(i = 1, 2, \ldots, c-1)$ is explicitly obtained by

$$\widehat{\Pi}_i(z) = z^{c-i}\sum_{j=0}^{i}\frac{A_{i,j}}{\widehat{z}_j - z}, \tag{18}$$

where

$$A_{i,j} = (c-i+1)\alpha\frac{A_{i-1,j}\widehat{z}_j}{f_i(\widehat{z}_j)},$$

$$A_{i,i} = \pi_{i,c-1} - (c-i+1)\alpha\sum_{j=0}^{i-1}\frac{A_{i-1,j}\widehat{z}_j}{f_i(\widehat{z}_j)}.$$

Proof. The proof of Lemma 2 is given in Appendix B. □

Remark 2. *At this moment, $\pi_{i,j}(j \ge i)$ is expressed in terms of $\pi_{0,0}$. In addition, $\pi_{i+1,i+1}$ is expressed in terms of $\pi_{i,j}$ $(j = i+1, i+2, \ldots)$ and then in terms of $\pi_{0,0}$ via the balance of the flows in and out the set of states $\{(m,j); 0 \le m \le i, j \ge m\}$, i.e.,*

$$(i+1)\mu\pi_{i+1,i+1} = \sum_{j=i+1}^{\infty}\min(j-i, c-i)\alpha\pi_{i,j}.$$

Differentiating (13) n times yields

$$f_i(z)\widehat{\Pi}_i^{(n)}(z) + nf_i'(z)\widehat{\Pi}_i^{(n-1)}(z) + \frac{n(n-1)}{2}f_i''(z)\widehat{\Pi}_i^{(n-2)}(z)$$
$$= (c-i+1)\alpha\widehat{\Pi}_{i-1}^{(n)}(z) + \lambda\pi_{i,c-1}(c-i-n+2)_n z^{c-i-n+1}$$
$$- (i\mu + (c-i)\mu_s)\pi_{i,c}(c-i-n+1)_n z^{c-i-n}.$$

Substituting $z = 1$ into this equation yields

$$\widehat{\Pi}_i^{(n)}(1) = \frac{c-i+1}{c-i}\widehat{\Pi}_{i-1}^{(n)}(1) + \frac{\lambda\pi_{i,c-1}(c-i-n+2)_n}{(c-i)\alpha} - \frac{\pi_{i,c}(i\mu + (c-i)\mu_s)(c-i-n+1)_n}{(c-i)\alpha} + \frac{n(\lambda - i\mu - (c-i)(\mu_s+\alpha))\widehat{\Pi}_i^{(n-1)}(1)}{(c-i)\alpha} + \frac{n(n-1)\lambda\widehat{\Pi}_i^{(n-2)}(1)}{(c-i)\alpha},$$

which is a recursive formula to compute all the factorial moments $\widehat{\Pi}_i^{(n)}(1)$ $(n \in \mathbb{N})$. It can be noted that $\widehat{\Pi}_i^{(0)}(1) = \widehat{\Pi}_i(1)$ and $\widehat{\Pi}_{i-1}^{(n)}(1)$ $(n \in \mathbb{N})$ are already obtained.

Finally, we consider the case $i = c$ which needs some special treatment. Balance equations are given by

$$(\lambda + c\mu)\pi_{c,c} = \alpha\pi_{c-1,c} + c\mu\pi_{c,c+1}, \quad j = c, \tag{19}$$

$$(\lambda + c\mu)\pi_{c,j} = \lambda\pi_{c,j-1} + \alpha\pi_{c-1,j} + c\mu\pi_{c,j+1} \quad j \ge c+1. \tag{20}$$

Letting $\widehat{\Pi}_c(z) = \sum_{j=c}^{\infty}\pi_{c,j}z^{j-c}$, we have $\Pi_c(z) = \widehat{\Pi}_c(z)$. Multiplying (20) by z^{j-c} and summing over $j \ge c$ yields

$$(-\lambda z^2 + (\lambda + c\mu)z - c\mu)\widehat{\Pi}_c(z) = \alpha\widehat{\Pi}_{c-1}(z) - c\mu\pi_{c,c}. \tag{21}$$

Define $f_c(z) = -\lambda z^2 + (\lambda + c\mu)z - c\mu = (z-1)(c\mu - \lambda z)$. Then

$$f_c(z)\widehat{\Pi}_c(z) = \alpha\widehat{\Pi}_{c-1}(z) - c\mu\pi_{c,c},$$

or,

$$\widehat{\Pi}_c(z) = \frac{\alpha\widehat{\Pi}_{c-1}(z) - c\mu\pi_{c,c}}{z-1}\frac{1}{c\mu - \lambda z} = \frac{\alpha(\widehat{\Pi}_{c-1}(z) - \widehat{\Pi}_{c-1}(1))}{z-1}\frac{1}{c\mu - \lambda z}. \tag{22}$$

Applying l'Hopital's rule, we obtain

$$\widehat{\Pi}_c(1) = \frac{\alpha\widehat{\Pi}'_{c-1}(1)}{c\mu - \lambda}.$$

Substituting $\widehat{\Pi}_{c-1}(z)$ in the form of (18) with $i = c-1$ into (22), we obtain

$$\widehat{\Pi}_c(z) = \sum_{j=0}^{c} \frac{A_{c,j}}{\widehat{z}_j - z}, \tag{23}$$

where

$$\widehat{z}_c = \frac{c\mu}{\lambda}, \quad A_{c,j} = \alpha \frac{A_{c-1,j}\widehat{z}_j}{f_c(\widehat{z}_j)}, \quad j = 0, 1, \ldots, c-1,$$
$$A_{c,c} = -\alpha \sum_{j=0}^{c-1} \frac{A_{c-1,j}\widehat{z}_j}{f_c(\widehat{z}_j)}.$$

Differentiating (21) n times and rearranging the results and then applying l'Hopital's rule yields

$$\Pi_c^{(n)}(1) = \frac{\alpha \widehat{\Pi}_{c-1}^{(n+1)}(1) + n(n+1)\lambda \Pi_c^{(n-1)}(1)}{(n+1)(c\mu - \lambda)}.$$

It can be noted that $\Pi_{c-1}^{(n+1)}(1)$ and $\Pi_c^{(0)}(1) = \Pi_c(1)$ are already known.

At this moment, $\pi_{i,j} (j \leq c)$ and the generating functions $\widehat{\Pi}_i(z)$ $(i = 0, 1, \ldots, c)$ are expressed in terms of $\pi_{0,0}$ which is uniquely determined using the following normalization condition.

$$\Pi_0(1) + \Pi_1(1) + \ldots + \Pi_c(1) = 1.$$

Remark 3. *We can obtain explicit results for the factorial moments and the joint stationary distribution using $A_{i,j}$ $(c \leq i \leq j \leq c)$ and $\widehat{z}_i$ $(i = 0, 1, \ldots, c)$ since expressions for the generating functions are given. Furthermore, $\pi_{i,j}$ $(j \geq c)$ is explicitly extracted from (8), (18) and (23).*

Remark 4. *If we use a general method to solve boundary equations as the matrix analytic methods in [9], [14], the computational complexity is $O(c^6)$. In contrast, the complexity of the generating function approach is only of order $\sum_{i=0}^{c} i = c(c+1)/2 = O(c^2)$. Indeed, we calculate $A_{i,j}$ and $\pi_{i,j}$ $(0 \leq i \leq j \leq c)$ in the following order.*

$$\begin{aligned}(0,0) \to (0,1) \to (0,2) \to \cdots \to (0,c)\\ \to (1,1) \to (1,2) \to \cdots \to (1,c)\\ \vdots\\ \to (c-1,c-1) \to (c-1,c)\\ \to (c,c).\end{aligned}$$

It should be noted that the recursive procedure for calculating $\pi_{i,j}$ $(0 \leq i \leq j \leq c)$ involves only positive numbers, i.e., $a_j^{(i)}$ and $b_j^{(i)}$.

IV. Performance measures

In this section, we utilize the analytical results to numerically evaluate the trade-off between performance and power consumption, employing the existing cost function used in [9], [17]–[19]. Accordingly, the power used when running at speed s, $P(s)$, is a function of the processor speed s, i.e., $P(s) = \frac{s^\sigma}{\beta}$, where $\sigma > 1$ is a constant scaling factor and β controls the relative cost of delay, called delay aversion in [9], [17]–[19]. In our analysis, we consider the two common performance metrics in the literature, that are sum of energy and response time ERWS [9], [17]–[19] and their product ERP [5], [7], to analyze the performance-energy trade-off. Then, ERWS and ERP are given by

$$\text{ERWS} = \text{E}[R] + \text{E}[P],$$
$$\text{ERP} = \text{E}[R] * \text{E}[P],$$

where $\text{E}[R]$ denotes the average response time, and $\text{E}[P]$ denotes the average power consumption per time unit.

The average number of jobs in the system $\text{E}[L]$ is given by

$$\text{E}[L] = \sum_{(i,j)\in S} j\pi_{i,j} = \sum_{i=0}^{c} (\Pi_i'(1) + i\Pi_i(1)),$$

where $\sum_{i=0}^{c} \Pi_i'(1)$ represents the average number of jobs in the system minus the average number of jobs served in normal mode, and $\sum_{i=0}^{c} i\Pi_i(1)$ is the average number of servers in normal mode.

From Little's law, the average response time $\text{E}[R]$ is given by

$$\text{E}[R] = \frac{1}{\lambda}\text{E}[L].$$

The average power consumption per time unit $\text{E}[P]$ is given by

$$\begin{aligned}\text{E}[P] =& \frac{1}{\beta} \sum_{(i,j)\in S} \Big((\mu_s^\sigma + \phi\mu^\sigma)\min(j-i, c-i)\\ &+ \omega\mu_s^\sigma \max(c-j, 0) + \mu^\sigma i \Big)\pi_{i,j}\\ =& \frac{1}{\beta} \left(\sum_{i=0}^{c} \sum_{j=i}^{c-1} \Big((\mu_s^\sigma + \phi\mu^\sigma(j-i) + \omega\mu_s^\sigma(c-j) \Big)\pi_{i,j} \right.\\ &\left. + (\mu_s^\sigma + \phi\mu^\sigma) \sum_{i=0}^{c-1} (c-i)\widehat{\Pi}_i(1) + \mu^\sigma \sum_{i=0}^{c} i\Pi_i(1) \right),\end{aligned}$$

where ϕ and ω represent the power consumption during setup and idle periods, respectively. It should be noted that $\min(j-i, c-i)$ is the number of busy servers in power-saving mode or the number of setup servers, $\max(c-j, 0)$ is the number of idle servers in power-saving mode, and i is the number of servers in normal mode.

For comparison, we consider the corresponding ON-OFF model, that is, M/M/c/Setup model without the power-saving policy. Then, the average response time and average power consumption per time unit in this model are given by

$$\mathrm{E}[R]_{\text{ON-OFF}} = \frac{1}{\lambda}\sum_{i=0}^{c}(\Pi_i'(1) + i\Pi_i(1)),$$

$$\mathrm{E}[P]_{\text{ON-OFF}} = \frac{1}{\beta}\left(\sum_{i=0}^{c}\sum_{j=i}^{c-1}\phi\mu^{\sigma}(j-i)\pi_{i,j} + \phi\mu^{\sigma}(c-i)\widehat{\Pi}_i(1) + \mu^{\sigma}\sum_{i=0}^{c} i\Pi_i(1)\right).$$

We assume that the power consumption required to set up a server is equivalent to the power used for processing a job at the server's peak rate, i.e., $\phi = 1$. As in [9], [17]–[19], we consider the case $\sigma = 2$. Furthermore, we note that an idle server consumes around 60% of the power consumption for processing a job in the power-saving mode [2], allowing us to establish $\omega = 0.6$.

V. Numerical Examples

In this section, we first compare the computational complexity of obtaining performance measures by the generating function method with that by the matrix geometric method in [9]. We use the CPU: Intel(R) Core(TM) i5-8400 @ 2.81 GHz, RAM: 8 GB, Windows 10 Pro system, and Python 3.10.9 to test the running time. We then illustrate some numerical examples to compare our model with the M/M/c/Setup model to assess the effect of the power-saving policy.

TABLE I
Running time of algorithms using the generating function approach and matrix geometric method (s).

Number of servers	Generating function method (s)	Matrix geometric method (s) [9]
5	0.0004	0.0025
10	0.0012	0.0042
50	0.0145	0.1953
100	0.0348	10.9378
200	0.1157	402.9612
300	0.2392	memory limit exceeded error
500	0.6862	memory limit exceeded error
1000	2.7661	memory limit exceeded error
1500	6.1295	memory limit exceeded error

Table I shows the running times needed to calculate the system performance based on the generating function and matrix geometric methods. We observe that the generating function approach has a much shorter running time than the matrix geometric method, especially when the number of servers is large. The matrix geometric method takes approximately 1600 times longer than our method for the case of $c = 200$. Specifically, given our computer configuration, the matrix geometric method is not feasible for cases with many servers, such as cases with more than 200 servers. However, the generating function approach remains efficient even for systems with many servers, requiring about 6 seconds to process a case with 1500 servers. Therefore, the larger the number of servers, the more efficient our method is.

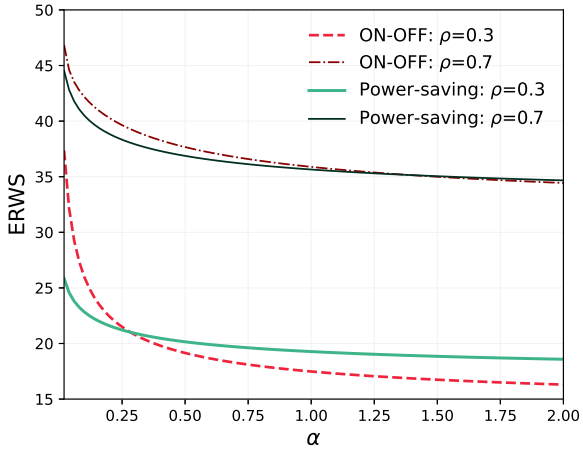

Fig. 2. ERWS versus the setup rate α.

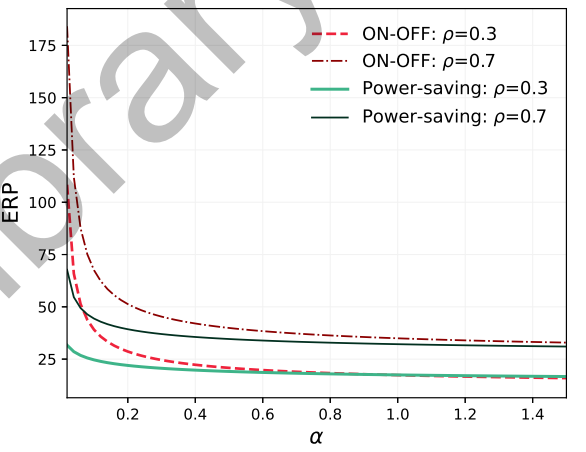

Fig. 3. ERP versus the setup rate α.

To analyze the efficiency of the power-saving policy, we compare it with the M/M/c/Setup model (ON-OFF model), which operates without this policy. We define the traffic intensity $\rho = \lambda/(c\mu)$. The ERWS and ERP metrics for our model and the ON-OFF model are plotted as a function of the setup rate α in Fig. 2 and Fig. 3, respectively. We set the number of servers $c = 30$, the service rate $\mu = 1.2, \mu_s = 0.5$, and the delay aversion value $\beta = 1$. We observe from Fig. 2 and Fig. 3 that the ERWS and ERP decrease with the setup rate as expected. The power-saving policy leads to lower ERWS and ERP levels when the setup time is large, while it may cause higher ERWS and ERP levels with a short setup time. Furthermore, the intercepts of the curves corresponding to

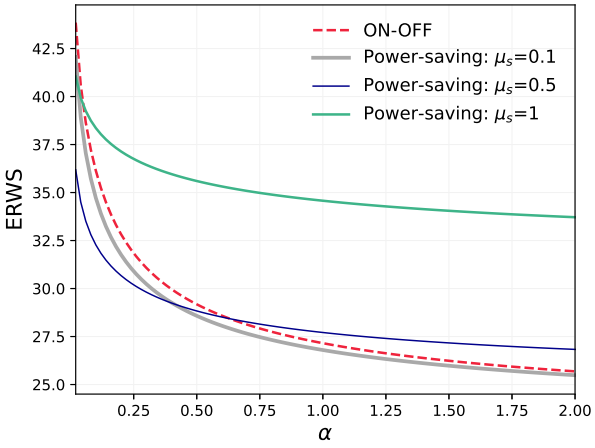

Fig. 4. ERWS versus the setup rate $\alpha(\rho = 0.5)$.

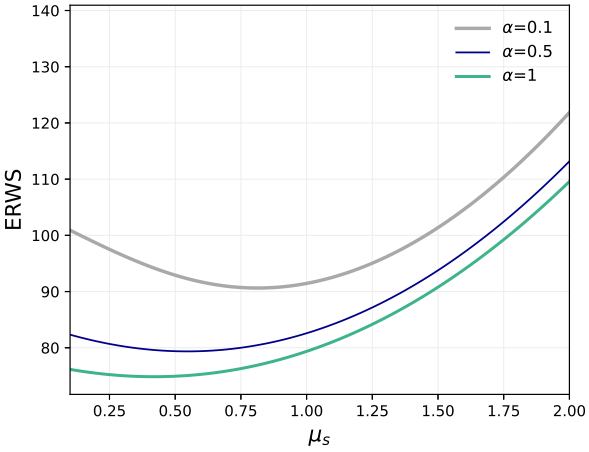

Fig. 6. ERWS versus the service rate μ_s.

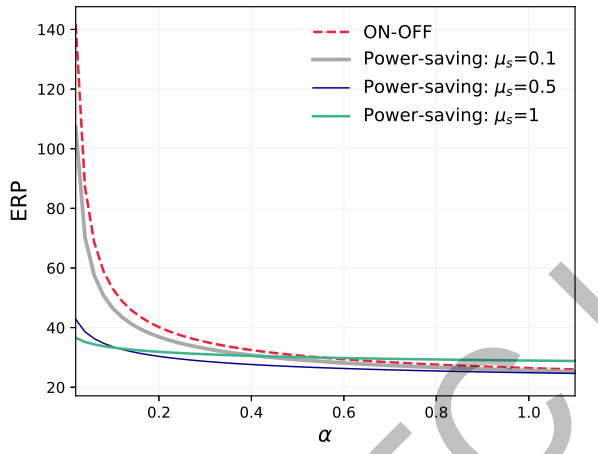

Fig. 5. ERP versus the setup rate $\alpha(\rho = 0.5)$.

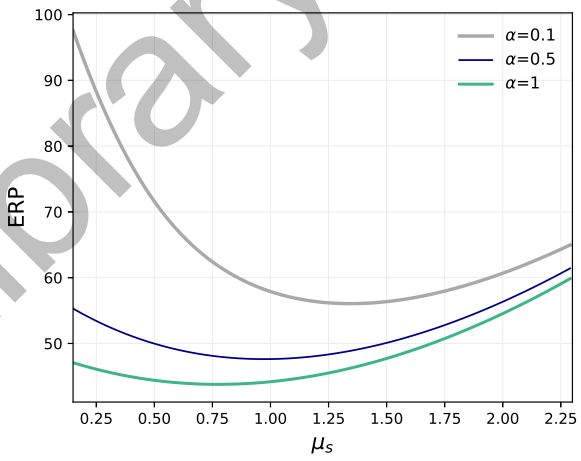

Fig. 7. ERP versus the service rate μ_s.

$\rho = 0.7$ are shifted to the right relative to those associated with $\rho = 0.3$. This observation suggests that power-saving policy exhibits greater effectiveness under high traffic intensity and vice versa. Indeed, under a low traffic regime, remaining servers in the power-saving mode results in higher power consumption than immediately turning them off and restarting them when needed, especially when the setup time is short.

In Fig. 4 and Fig. 5, we plot the ERWS and ERP for the same setting as Fig. 2 and Fig. 3 but under load $\rho = 0.5$ and $\mu_s = 0.1, 0.5, 1$. In the limit $\mu_s \to 0$ and $\alpha \to \infty$, our model resembles the M/M/c/Setup model. These figures illustrate that an appropriate value of the service rate μ_s can effectively minimize the ERWS and ERP, while excessively high service rates in the power-saving mode may be inefficient, especially when the setup time is negligible. Our model exhibits greater efficiency when trade-offs are evaluated using the ERP rather than ERWS. Nevertheless, both the metrics indicate that higher service rates μ_s effectively minimize the ERP and ERWS, particularly in cases with prolonged setup times. In practice, setup times in data centers often approximate 200 seconds, while typical service times are less than 1 second [13]. This substantial difference highlights the feasibility of adopting the power-saving policy.

Fig. 6 and Fig. 7 illustrate ERWS and ERP as a function of the service rate in power-saving mode μ_s, respectively. These figures show that the ERWS and ERP decrease with the service rate μ_s when μ_s is small and vice versa. Hence, the trade-off between power consumption and delay is apparent, with lower power consumption on the left-hand side, lower delay on the right-hand side, and an optimum in between. We observe from Fig. 6 and Fig. 7 that the optimal points of the ERP and ERWS curves shift to the right as the setup rate α increases. This observation indicates that a shorter setup time requires a higher service rate in the power-saving mode to enhance the

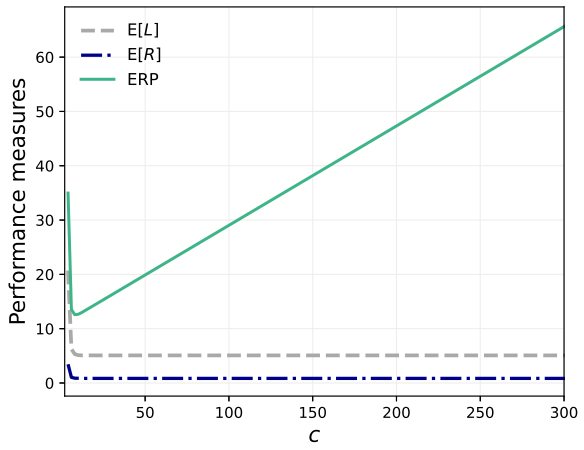

Fig. 8. Performance measures versus the number of servers c ($\lambda = 6$).

probability of completing tasks before the setup process ends and transitions to the normal mode, which consumes more energy.

Finally, performance measures E[L], E[R], and ERP for our model are plotted as a function of the number of servers c in Fig. 8. The generating function approach makes it possible to conduct extensive numerical experiments for large-scale systems with numerous servers, which are beyond the capabilities of existing methods. We fix $\mu = 1.6, \mu_s = 0.6, \alpha = 0.5$ and $\lambda = 6$. We observe in Fig. 8 that the mean number of jobs in the system E[L] and the mean response time E[R] decreases with c as expected. Meanwhile, ERP decreases with c as c is small and vice versa. This indicates the existence of an optimal c that minimizes ERP, reflecting the trade-off between delay and power consumption as c changes.

VI. Conclusions

This paper analyzed the M/M/c/Setup model incorporating the power-saving policy for data centers. Using the generating function approach, we derived exact expressions for the joint stationary queue length distribution, generating functions, and factorial moments of any order. This approach significantly reduced computational complexity compared to existing methods, making it more practical for large-scale systems. Based on the analytical results, we assessed the trade-off between system performance and power consumption using an existing speed-based cost function. We also conducted numerical experiments to compare our model with the M/M/c/Setup model, which allows us to evaluate the effectiveness of the power-saving policy. The results showed that the power-saving policy leads to a better balance between energy consumption and system performance.

Appendix

A. Proof of Lemma 1

Proof. We use induction for the proof of this lemma. It is easy to see that (7) is true for $i = c$. Assuming that (7) holds in the case $j + 1$, i.e., $\pi_{0,j+1} = b_{j+1}^{(0)}\pi_{0,j}$, for some $j < c - 1$. Substituting this expression into (2) and rearranging the result we derive (7). □

B. Proof of Lemma 2

Proof. We use induction for the proof of (15) similar as in Lemma 1. Next, we prove the inequalities. Lemma 2 holds in the case $j = c$ since

$$0 < a_c^{(i)}, \quad 0 < b_c^{(i)} < \frac{\lambda}{i\mu + (c-i)\mu_s},$$

because $0 < z_i < 1$.

Assuming that $0 < a_{j+1}^{(i)}$, $0 < b_{j+1}^{(i)} < \frac{\lambda}{i\mu+(j-i+1)\mu_s}$, we have

$$\lambda + i\mu + (j-i)(\mu_s + \alpha) > \lambda + i\mu + (j-i)(\mu_s + \alpha) - (i\mu + (j-i+1)\mu_s)b_{j+1}^{(i)} > i\mu + (j-i)(\mu_s + \alpha),$$

which together with (16) and (17) yield

$$0 < \frac{\lambda}{\lambda + i\mu + (j-i)(\mu_s + \alpha)} < b_j^{(i)} < \frac{\lambda}{i\mu + (j-i)(\mu_s + \alpha)} < \frac{\lambda}{i\mu + (j-i)\mu_s},$$
$$0 < \frac{(j-i+1)\alpha\pi_{i-1,j} + (i\mu + (j-i+1)\mu_s)a_{j+1}^{(i)}}{\lambda + i\mu + (j-i)\mu_s + (j-i)\alpha} < a_j^{(i)}.$$

Next, we prove (18) using mathematical induction. Substituting

$$\widehat{\Pi}_{i-1}(z) = z^{c-i+1}\sum_{j=0}^{i-1}\frac{A_{i-1,j}}{\widehat{z}_j - z}$$

into (13) and rearranging the result, we obtain (18). It should be noted that (24) is used to decompose $\widehat{\Pi}_i(z)$ into simple form.

$$\frac{1}{(a-z)(b-z)} = \frac{1}{b-a}\left(\frac{1}{a-z} - \frac{1}{b-z}\right), \quad \forall a \neq b. \quad (24)$$

□

References

[1] Electricity, I.E.A. Analysis and Forecast to 2026. *IEA Report (URL: https://www.iea.org/reports/electricity-2024)* (2024)

[2] Barroso, L. & Hölzle, U. The case for energy-proportional computing. *Computer*. **40**, 33-37 (2007)

[3] Artalejo, J., Economou, A. & Lopez-Herrero, M. Analysis of a multi-server queue with setup times. *Queueing Systems*. **51**, 53-76 (2005)

[4] Phung-Duc, T. & Kawanishi, K. Delay performance of data-center queue with setup policy and abandonment. *Annals of Operations Research*. **293**, 269-293 (2020)

[5] Le-Anh, T. & Phung-Duc, T. Energy-performance tradeoffs in server farms with batch services and setup times. *Performance Evaluation*. **168**, 102468 (2025)

[6] Pender, J. & Phung-Duc, T. A law of large numbers for M/M/c/delayoff-setup queues with nonstationary arrivals. *Proceeding of 23rd International Conference on Analytical and Stochastic Modelling Techniques and Applications, ASMTA 2016, Cardiff, UK, August 24-26, 2016*. pp. 253-268 (2016)

[7] Gandhi, A., Harchol-Balter, M. & Kozuch, M. Are sleep states effective in data centers?. *2012 International Green Computing Conference (IGCC)*. pp. 1-10 (2012)

[8] Ma, Z., Guo, S. & Wang, R. The virtual machines scheduling strategy based on M/M/c queueing model with vacation. *Future Generation Computer Systems*. **138**, 43-51 (2023)

[9] Le-Anh, T. & Phung-Duc, T. Analysis of multi-server queueing systems with setup times and power-saving modes. *Proceedings of 17th EAI International Conference on Performance Evaluation Methodologies and Tools, Milan, Italy, December 12-13 2024*. (2024)

[10] Wu, D. & Takagi, H. M/G/1 queue with multiple working vacations. *Performance Evaluation*. **63**, 654-681 (2006)

[11] Tao, L., Liu, Z. & Wang, Z. The GI/M/1 queue with start-up period and single working vacation and Bernoulli vacation interruption. *Applied Mathematics And Computation*. **218**, 4401-4413 (2011)

[12] Servi, L. & Finn, S. M/M/1 queues with working vacations (M/M/1/WV). *Performance Evaluation*. **50**, 41-52 (2002)

[13] Gandhi, A., Doroudi, S., Harchol-Balter, M. & Scheller-Wolf, A. Exact analysis of the M/M/k/setup class of Markov chains via recursive renewal reward. *Proceedings of the ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*. pp. 153-166 (2013)

[14] Van Houdt, B. & Leeuwaarden, J. Triangular M/G/1-type and tree-like quasi-birth-death Markov chains. *INFORMS Journal On Computing*. **23**, 165-171 (2011)

[15] Phung-Duc, T. Exact solutions for M/M/c/setup queues. *Telecommunication Systems*. **64**, 309-324 (2017)

[16] Doroudi, S., Fralix, B. & Harchol-Balter, M. Clearing analysis on phases: Exact limiting probabilities for skip-free, unidirectional, quasi-birth-death processes. *Stochastic Systems*. **6**, 420-458 (2017)

[17] Lu, X., Aalto, S. & Lassila, P. Performance-energy trade-off in data centers: Impact of switching delay. *2013 22nd ITC Specialist Seminar on Energy Efficient and Green Networking (SSEEGN)*. pp. 50-55 (2013)

[18] Phung-Duc, T., Rogiest, W. & Wittevrongel, S. Single server retrial queues with speed scaling: Analysis and performance evaluation. *Journal of Industrial and Management Optimization*. **13**, 1927-1943 (2017)

[19] Wierman, A., Andrew, L. & Tang, A. Power-aware speed scaling in processor sharing systems. *IEEE INFOCOM 2009*. pp. 2007-2015 (2009)