A discrete-time infinite-server batch arrival queue with application to serverless computing

Veena Goswami[1,*]

[1] *School of Computer Applications, Kalinga Institute of Industrial Technology, Bhubaneswar, India*
E-mail: ⟨veena_goswami@yahoo.com⟩

Abstract. Serverless computing systems behave like an infinite server queue, dynamically expanding the number of instances to handle incoming batch tasks. In serverless computing, to leverage its scalability, efficiency, and cost-effectiveness, it is essential to apply an infinite server queue with batch task arrival. The model is used to estimate resource usage and to optimize cost-performance trade-offs. This paper analyzes a discrete-time infinite-server batch/single-arrival queue with application to a serverless computing system. We obtain the probability-generating function of the system size, which characterizes a functional equation; we then solve it recursively to find state probabilities at various epochs and some performance measures. Unlike existing discrete-time infinite-server models, this paper explicitly compares the late-arrival system with delayed access and the early-arrival system policies in the context of batch arrivals, and establishes convergence to their continuous-time counterparts. The model's numerical results have been presented in graphs and tables to illustrate the impact of batch-size distributions and system parameters.

Keywords: bulk arrival, discrete-time queueing, infinite server, scalability, serverless

Received: January 4, 2026; accepted: March 12, 2026; available online: July 8, 2026

DOI: 10.17535/crorr.2026.0025

Original scientific paper.

1. Introduction

Serverless architecture offers numerous advantages, including lower operating costs, enhanced application performance, and multi-language support for developers. Due to these benefits, serverless computing is gaining popularity among companies seeking an affordable development and implementation model. Infinite-server bulk-arrival queueing models have intriguing applications in serverless computing, particularly for handling unpredictable or high-volume workloads. Serverless computing has emerged as a notable framework in cloud-native application development, primarily due to its underlying elasticity and cost efficiency [7, 14]. This paradigm simplifies runtime resource management, leading to its widespread adoption in Function-as-a-Service (FaaS) offerings, such as AWS Lambda [28]. [26] discussed the challenge of predicting the performance of serverless functions in a dynamic environment. [27] and [25] noted that the performance variability, which demonstrates differing end-to-end response latencies across identical function invocations, is an unconsidered problem within the research community.

Serverless function performance is crucial for both practitioners seeking to optimize their applications and cloud providers aiming to improve service quality [15]. It necessitates robust methodologies for performance evaluation, often involving empirical studies to determine the number of repetitions required for a serverless function with specific inputs, accounting for observed performance fluctuations [27]. Instead of coming in one at a time, several requests

*Corresponding author.

arrive simultaneously. There is never a waiting queue because each incoming request is promptly assigned to a server. Serverless platforms and other elastic resources are used in this model system. These models are ideal for examining systems with nearly infinite scalability, such as serverless operations or cloud-based microservices.

Bulk arrival models facilitate the simulation and forecasting of serverless systems' behavior, which often encounter unexpected spikes in requests. Analysis of latency, throughput, and resource usage without bottlenecks is possible with infinite server queues. Many researchers studied the continuous-time infinite server queues with many alterations like, overlap times [18], in a semi-Markovian environment [8], system's additional tasks and impatient customers [1], phase-type arrivals [20], batch arrivals [21, 23], batch arrivals and dependent service times [19]. The transient solution of the Markovian multi-server queue with balking and catastrophes has been discussed in [16].

The arrivals and departures may co-occur at a slot's border epoch in discrete-time queueing system. Arrival-first (AF) and departure-first (DF) ruling strategies, sometimes mentioned as the early arrival system (EAS) and late arrival system with delayed access (LAS-DA), respectively, can deal with rules in the case of simultaneity, see [12, 13]. In the literature, multiserver discrete-time models with various variations have been discussed: with late and early arrival system [9], optimal control of queue [2], baling behavior [10], batch arrival [22], renewal input [3], batch arrival with renewal input [5], batch service [11].

The discrete-time batch arrival infinite-server queueing model mapping enables the performance analysis of serverless systems to predict scalability, optimize resource allocation, and model bursty, event-driven workloads. The infinite-server assumption holds well below platform concurrency limits, making it ideal for analyzing short-term auto-scaling behavior. [17] analyzed the discrete-time infinite-server queueing system with batch arrivals and bulk service, formulated a functional equation for the system occupancy, and solved it by recursion rather than by the more common transform or matrix-analytic methods. [4] examined the discrete-time infinite-server bulk-arrival renewal input queue with geometrically distributed service times. They derived both the transient and steady-state distributions of the model's system size. [6] analyzed continuous-time infinite-server batch arrival queues with Poisson arrival epochs, handling both stationary and nonstationary arrival rates and general service distributions. This study extends [4] work by switching from discrete-time geometric service models to continuous-time general service distributions.

Serverless systems, by their features, are demand-driven and event-driven. Because system actions often occur in discrete steps or in response to discrete events, continuous-time models are less appropriate. Analysis of concurrency constraints, burst handling, and scaling dynamics is made easier by representing system evolution at slot boundaries. For the study of systems with batch arrivals, synchronization effects, and time-slotted control strategies, discrete-time queueing models offer manageable frameworks. While several studies address infinite-server queues with batch arrivals, fewer works consider discrete-time formulations under different arrival–departure conventions, which are particularly relevant for slot-based serverless platforms. In this article, we consider a discrete-time batch/single arrival infinite-server queueing system to enhance the effectiveness of serverless computing. The interarrival and service times are assumed to be geometrically distributed. We find the state probabilities at arbitrary and outside observer's observation epochs for both the late arrival system with a delayed access and the early arrival system by the recursion method. Particular cases are examined by assuming various probability distributions for the arrival batch sizes. Numerical experiments of the system have been examined in the form of tables and graphs. It is established that, in the limiting case, the results of this paper tend to the continuous-time counterpart.

This paper is structured as follows. Section 2 shows a serverless computing system architecture as a $Geo^X/Geo/\infty$ queueing system. Section 3 analyzes the $Geo^X/Geo/\infty$ queue by two methods for the late arrival system with a delayed access and the early arrival system policy.

In Section 4, we discuss exceptional cases by assuming various probability distributions for the arrival batch sizes and Section 5 presents the analysis of the $Geo/Geo/\infty$ queue. Section 6 illustrates the numerical results, and Section 7 concludes the paper.

2. System model

The cloud provider can dynamically allocate machine resources through a cloud architecture called "serverless computing". Due to their elasticity, serverless architectures can effectively manage batch arrivals and are inherently scalable. With a sufficient number of servers, serverless computing can handle batch arrivals as follows.

Automated Adjustment: Platforms without servers automatically allocate resources to meet incoming tasks. AWS Lambda system scales out in response to event volume and can manage thousands of concurrent executions. Google Cloud Functions automatically scales to handle the volume of incoming requests, ensuring each request is processed promptly. Azure Functions are suitable for batch processing; they scale out to handle multiple concurrent executions.

Event-driven invocation: Serverless functions are appropriate for batch job processing since a variety of events, such as scheduled tasks, message queues, and data processing pipelines, can trigger them.

Economy of cost: Serverless platforms bill according to the resources and real execution time, such as pay-per-use and no idle costs.

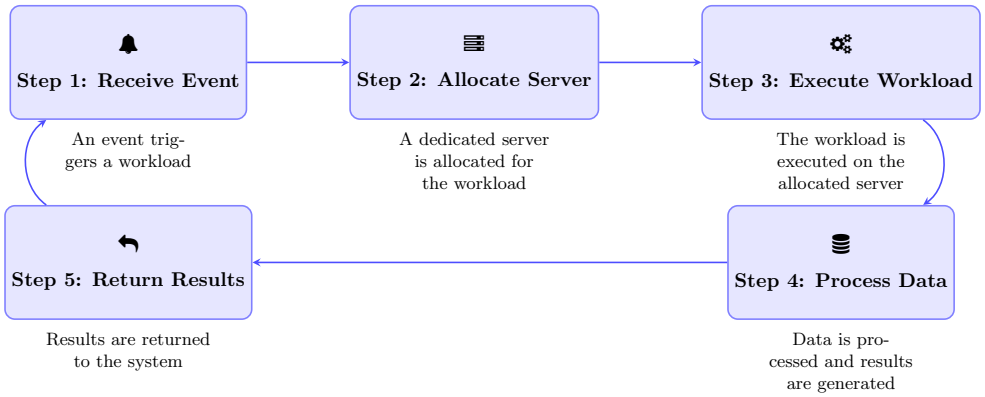

Figure 1: *Serverless architecture cycle.*

Serverless architecture processes bursty, event-driven workloads with automatic, near-infinite scale. The core principle is that every unit of work (job in a batch) gets its own dedicated execution environment (server) without waiting for previous jobs to finish. Fig. 1 depicts the serverless architecture cycle. Here is a visual representation of the core architecture.

Batch Arrival Generator (Event Sources): Here, batch arrivals originate and generate events that trigger the system, generally in bursts.

Queueing System (Trigger and Buffer Layer): After receiving the batch, this layer sends it to the compute layer. It decouples components and controls flow.

Infinite Server Farm (Compute Layer): The infinite server model consists of stateless, ephemeral compute services that scale on demand.

Shared State and Orchestration (Backing Services): The compute instances depend on managed services for durability and coordination because they are isolated and stateless.

Fig. 2 illustrates the flowchart of the serverless computing. Let us imagine a scenario where a large number of images need to be uploaded to a cloud storage bucket and then processed. Here, we consider an infinite server queueing model with packets arriving in batches. An

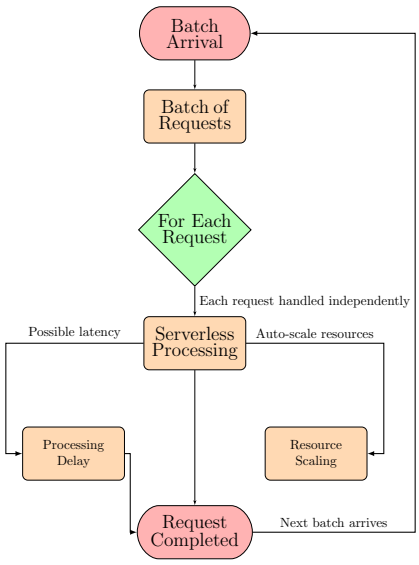

Figure 2: *Flowchart of serverless computing.*

infinite number of servers are ready to handle new assignments; a server is assigned to each task as soon as it arrives; every task is dealt with concurrently; and since there are always enough servers to manage the load, tasks never wait in a queue. This idea can be leveraged to handle such workloads in serverless computing by capitalizing on its scalability and event-driven architecture. By dynamically increasing or decreasing the number of instances to accommodate incoming batch tasks, serverless computing platforms can behave similarly to an infinite server queue. The discrete-time infinite-server batch-arrival model may not adequately explain systems that operate under capacity caps, burst-induced provisioning delays, or severe resource coupling, even while it is suitable for lightly loaded, unconstrained, and brief function executions.

3. Analytical model

We consider a discrete-time batch arrival queue wherein requests arrive in batches of random size Y with probability mass function (pmf) $g_j = P(Y = j),\ j \geq 1$, probability generating function (pgf) $G(z) = \sum_{j=1}^{\infty} g_j z^j$, $|z| \leq 1$ and mean batch size $\overline{g}$. The inter-arrival times A are independent and geometrically distributed with pmf $P(A = n) = \lambda\bar{\lambda}^{j-1}\ \ 0 < \lambda < 1,\ j \geq 1$. The batch sizes are independent of inter-arrival times. With pmf $P(S = n) = \mu\bar{\mu}^{j-1},\ 0 < \mu < 1,\ j \geq 1$ and mean service time $1/\mu$, the service time S of each of the infinite servers is independent and geometrically distributed. Additionally, given that there are k busy servers, the probability that ℓ servers finish service in the following interval is provided by

$$\Psi(\ell \mid k) = \begin{cases} \binom{k}{\ell}\mu^{\ell}\bar{\mu}^{k-\ell} & 0 \leq \ell \leq k, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Since servers are infinite, stability does not need balancing arrival and service rates, unlike finite-server queues, because an infinite number of servers prevents queue formation. Rather, stability is ensured provided that both the mean service time and the mean arrival rate are finite. As it is assumed that the arrival batches and service times are independent and identically distributed throughout slots, the system dynamics are time-homogeneous. Thus, after

the system has run long enough, the process becomes a stationary stochastic process. Performance evaluation in real-world applications, such as serverless computing platforms, usually focuses on long-term average behavior rather than short-term startup impacts. The workload produced by large user populations can often be statistically consistent over sufficiently long observation periods. The steady-state study offers useful estimates of anticipated concurrency levels, resource usage, and system capacity requirements under these circumstances.

We study the system under both policies, the late arrival system with delayed access (LAS-DA) and the early arrival system (EAS). Assume that the time axis is denoted as $0, 1, 2, \ldots, m$, and that it is slotted into equal-length intervals with a slot length of unity. A thorough explanation of these ideas has previously been provided in several places [12, 13]. According to the LAS-DA policy, arrivals occur at the end of the slot and departures at the start. Service cannot be provided to newly arrived clients during the same time slot. In the EAS policy, arrivals happen at the start of the slot, and departures occur at its end. If the server is idle, arrivals may receive service immediately. Systematic delay discrepancies arise from differences in event ordering between LAS-DA and EAS policies. Queueing behavior is further influenced by batch-size variability, with greater variation leading to longer lineups and delays. The service can begin immediately in EAS policy, such as digital processors or packet-switched networks. But in the case of LAS-DA policy processing, decisions occur at slot boundaries, such as synchronized or gated systems.

3.1. $Geo^X/Geo/\infty$ queue with LAS-DA policy

In LAS-DA, prospective arrivals take place in the interval $(m-, m)$, whereas prospective departures occur in the period $(m, m+)$. More precisely, different periods of time during which events take place are shown in Fig. 3.

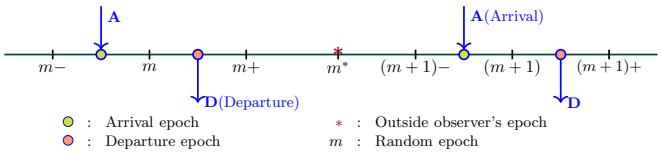

Figure 3: *Flowchart of serverless computing.*

Define the probability as $P_j(m-) = P\{j \text{ requests in the system at time } m-\}$, $j \geq 0$. Relating the system's states at two successive epochs $m-$ and $(m+1)-$, and using the notation $P_j(m)$ for $P_j(m-)$, we obtain

$$P_0(m+1) = \sum_{i=0}^{\infty} \bar{\lambda}\Psi(i|i)P_i(m), \tag{2}$$

$$P_j(m+1) = \sum_{i=j}^{\infty} \bar{\lambda}\Psi(i-j|i)P_i(m) + \sum_{k=0}^{\infty}\sum_{i=k}^{j+k-1} \lambda g_{j-i+k}\Psi(k|i)P_i(m), \quad j \geq 1. \tag{3}$$

Assuming that steady state exists, let $P_n = \lim_{m\to\infty} P_n(m)$, $n = 0, 1, \ldots$ and define pgf as $P(z) = \sum_{j=0}^{\infty} P_j z^j$. From (2) and (3), in steady-state we have

$$P_0 = \sum_{i=0}^{\infty} \bar{\lambda}\Psi(i|i)P_i, \tag{4}$$

$$P_j = \sum_{i=j}^{\infty} \bar{\lambda}\Psi(i-j|i)P_i + \sum_{k=0}^{\infty}\sum_{i=k}^{j+k-1} \lambda g_{j-i+k}\Psi(k|i)P_i, \quad j \geq 1. \tag{5}$$

Multiplying appropriate powers of z in Eqs. (4) and (5), and summing over j, after simplification, we have

$$P(z) = \left(\bar{\lambda} + \lambda G(z)\right) P\left(\mu + \bar{\mu} z\right), \tag{6}$$

which is a functional equation. From Eq. (6), using power series expansion around $z = 1$, we obtain

$$P(z) = \sum_{\ell=0}^{\infty} \frac{h_\ell}{\ell!} \, (z-1)^\ell, \tag{7}$$

where h_ℓ is given by the following recursive formula

$$h_0 = 1, \tag{8}$$

$$h_\ell = \frac{\lambda}{1-\bar{\mu}^\ell} \sum_{k=0}^{\ell-1} \binom{\ell}{k} \bar{\mu}^k \; h_k \; G^{(\ell-k)}(1), \quad \ell \geq 1, \tag{9}$$

$$= \frac{\lambda \ell!}{1-\bar{\mu}^\ell} \sum_{k=0}^{\ell-1} \bar{\mu}^k \frac{h_k}{k!} \sum_{j=\ell-k}^{\infty} \binom{j}{\ell-k} g_j, \quad \ell \geq 1, \tag{10}$$

and $G^{(n)}(1) = \frac{d^n}{dz^n} G(z)|_{z=1}$. Calculating the n-th ($n = 1, 2, \ldots$) derivatives on both sides of Eq. (6) and setting $z = 1$ yields the recursive formula mentioned above.From Eq. (7), we have

$$\sum_{\ell=0}^{\infty} P_\ell z^\ell = \sum_{\ell=0}^{\infty} z^\ell \sum_{n=\ell}^{\infty} (-1)^{n-\ell} \binom{n}{\ell} \frac{h_n}{n!}. \tag{11}$$

Thus,

$$P_\ell = \sum_{n=\ell}^{\infty} (-1)^{n-\ell} \binom{n}{\ell} \frac{h_n}{n!}, \quad \ell \geq 0. \tag{12}$$

Remark 1. *Taking inter-arrival and service times as exponentially distributed, with mean arrival and mean service rates γ and η, respectively. Divide the time axis into equal intervals $\Delta > 0$ such that $\lambda = \gamma\Delta$, $\mu = \eta\Delta$, and $\rho = \frac{\gamma\Delta}{\eta\Delta} = \rho_1$, where Δ is appropriately small. Setting $\lambda = \gamma\Delta$, $\mu = \eta\Delta$, and taking the limit as $\Delta \to 0$, we have*

$$h_\ell = \lim_{\Delta \to 0} \frac{\gamma\Delta \; \ell!}{1-(1-\eta\Delta)^\ell} \sum_{k=0}^{\ell-1} (1-\eta\Delta)^k \frac{h_k}{k!} \sum_{j=\ell-k}^{\infty} \binom{j}{\ell-k} g_j \tag{13}$$

$$= \frac{\gamma(\ell-1)!}{\eta} \sum_{k=0}^{\ell-1} \frac{h_k}{k!} \sum_{j=\ell-k}^{\infty} \binom{j}{\ell-k} g_j, \tag{14}$$

substituting h_ℓ in Eq. (12) *it tends to $M^X/M/\infty$ queue.*

Alternatively, we may solve Eq. (6) recursively, as shown below. Assume $H(z) = \left(\bar{\lambda} + \lambda G(z)\right)$. Thus,

$$P(z) = H(z) \cdot P\left(\mu + \bar{\mu} z\right), = H(z) \cdot H\left(\mu + \bar{\mu} z\right) \cdot P\left(\mu + \mu\bar{\mu} + \bar{\mu}^2 z\right) \tag{15}$$

After the nth recursion, we have

$$\begin{aligned} P(z) = & H(z) \cdot H\left(\mu + \bar{\mu} z\right) \ldots H\left(\mu + \mu\bar{\mu} + \mu\bar{\mu}^2 + \cdots + \mu\bar{\mu}^{n-1} + \bar{\mu}^n z\right) \\ & \cdot P\left(\mu + \mu\bar{\mu} + \mu\bar{\mu}^2 + \cdots + \mu\bar{\mu}^n + \bar{\mu}^{n+1} z\right). \end{aligned} \tag{16}$$

As $0 < \mu \leq 1$, so $0 \leq \bar{\mu} < 1$, when $n \to \infty$, in Eq. (16) the last factor is equal to one, because

$$\begin{aligned} &\lim_{n\to\infty} P\left(\mu + \mu\bar{\mu} + \mu\bar{\mu}^2 + \cdots + \mu\bar{\mu}^n + \bar{\mu}^{n+1}z\right) \\ &= \lim_{n\to\infty} P\left(\mu\left(1 + \bar{\mu} + \bar{\mu}^2 + \cdots + \bar{\mu}^n\right)\right) = P(1) = 1. \end{aligned} \tag{17}$$

Therefore, as $n \to \infty$, Eq. (16) reduces to

$$P(z) = H(z)\cdot H\left(\mu + \bar{\mu}z\right)\cdots H\left(\mu + \mu\bar{\mu} + \mu\bar{\mu}^2 + \cdots + \mu\bar{\mu}^{n-1} + \bar{\mu}^n z\right) \tag{18}$$

$$= H(z)\prod_{j=1}^{\infty} H\left(\sum_{i=1}^{j}\mu\bar{\mu}^{i-1} + \bar{\mu}^j z\right) = H(z)\prod_{j=1}^{\infty} H\left(1 - \bar{\mu}^j + \bar{\mu}^j z\right). \tag{19}$$

It is apparent from Eq. (19) that the service rate and the PGF of packet arrival batch size entirely describe the PGF of the system.

3.1.1. Outside observer's observation epoch

The outside observer's observation epoch is more likely to fall in the interval $(m+, (m+1)-)$. Since nothing happens (no arrival and no departure) in this interval, the outside observer's distribution P_n^o will be equal to the distribution of the number in the system noticed at $(m+1)-$ and hence $P_n^o = P_n$.

3.2. $Geo^X/Geo/\infty$ queue with EAS

As was previously mentioned, the order in which packets arrive and depart around a slot boundary varies between the two policies. In LAS-DA, prospective arrivals take place in the period $(m-, m)$, whereas prospective departures occur in the period $(m, m+)$. More precisely, different periods of time during which events take place are shown in Fig. 4.

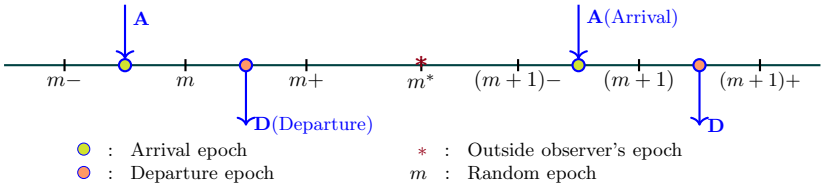

Figure 4: *Flowchart of serverless computing.*

$$Q_j = \bar{\lambda}\sum_{i=j}^{\infty} c(i-j|i)Q_i + \sum_{k=1}^{\infty}\sum_{i=j-k}^{\infty} \lambda g_k c(i-j+k|i+k)Q_i, \quad j \geq 0. \tag{20}$$

Multiplying Eq. (20) by appropriate powers of z and summing over j, after simplification, we have

$$Q(z) = \left(\bar{\lambda} + \lambda G(\mu + \bar{\mu}z)\right) Q\left(\mu + \bar{\mu}z\right), \tag{21}$$

which is a functional equation that defines the probability generating function $Q(z)$. Using power series expansion around $z = 1$ and following the same procedure as in LAS-DA policy, we obtain from Eq. (21),

$$Q(z) = \sum_{\ell=0}^{\infty} \frac{d_\ell}{\ell!}\ (z-1)^\ell, \tag{22}$$

where d_ℓ is given by the following recursive formula

$$d_0 = 1, \tag{23}$$

$$d_\ell = \frac{\lambda}{1-\bar{\mu}^\ell} \sum_{k=0}^{\ell-1} \binom{\ell}{k} \bar{\mu}^k \; d_k \; B^{(\ell-k)}(1), \quad \ell \geq 1, \tag{24}$$

where $B(z) = H(\mu + \bar{\mu}z)$. We obtain the alternative solution from Eq. (21) following the same procedure as in LAS-DA policy as

$$Q(z) = H(\mu + \bar{\mu}z)Q\left(\mu + \bar{\mu}z\right) = \prod_{j=1}^{\infty} H\left(1 - \bar{\mu}^j + \bar{\mu}^j z\right). \tag{25}$$

Taking derivative with respect to z and setting $z = 1$, we obtain $L_s = Q^{(1)}(z)|_{z=1} = \frac{\lambda\bar{\mu}\bar{g}}{\mu}$.

3.2.1. Outside observer's observation epoch

At the outside observer's observation epoch in EAS, let Q_n^o be the outside observer's distribution, that is, there are n requests in the system. In this instance, the observation epoch of the external observer is likely to occur between a possible arrival and a possible departure. Since we already know the system's number distribution at m, and since there is only one event—an arrival or no arrival—after m, the Q_n^o is determined by

$$Q_0^o = \bar{\lambda} Q_0, \tag{26}$$

$$Q_k^o = \bar{\lambda} Q_k + \lambda \sum_{j=1}^{k} g_j Q_{k-j}, \; k \geq 1. \tag{27}$$

The PGF at outside observer's observation epoch in EAS using Eq. (25) simplifies as

$$Q^o(z) = \left(\bar{\lambda} + \lambda G(z)\right) Q(z) = H(z) \prod_{j=1}^{\infty} H\left(1 - \bar{\mu}^j + \bar{\mu}^j z\right). \tag{28}$$

Note that $Q^o(z) = P^o(z) = P(z)$, which implies $Q_n^o = P_n$, meaning that the system size distribution at the observation epoch of the outside observer is equivalent in both LAS-DA and EAS. Taking derivative with respect to z and putting $z = 1$, we have

$$L_s^o = Q^{o(1)}(z)|_{z=1} = \frac{\lambda\bar{g}}{\mu}. \tag{29}$$

Remark 2. *Similar findings can be obtained from the observation epoch probabilities of outside observers in the EAS policy, even in the limiting case. This leads to the conclusion that both LAS-DA and EAS queue results tend to be the same in continuous time.*

4. Special cases

In this section, we assume various probability distributions for the arrival batch sizes and derive the system size distribution. The distribution of batch sizes significantly impacts queue variability and latency when arrivals occur in batches. Geometric, Shifted Poisson, Shifted Bernoulli, and Shifted Binomial probability distributions are discussed for arrival batch sizes.
Geometric batches. Let us assume the batch size is Geometrically distributed with pmf

$P(X=\ell)=\varphi(1-\varphi)^{\ell-1}$, where $0<\varphi<1$. Here, pgf of geometric distribution is $G(z)=\frac{\varphi z}{1-\bar{\varphi}z}$, and $H(z)=\bar{\lambda}+\frac{\lambda\varphi z}{1-\bar{\varphi}z}$. Applying this in Eq. (19), we have

$$P(z)=\left(\bar{\lambda}+\frac{\lambda\varphi z}{1-\bar{\varphi}z}\right)\prod_{j=1}^{\infty}\left(\bar{\lambda}+\frac{\lambda\varphi\left(1-\bar{\mu}^j+\bar{\mu}^j z\right)}{1-\bar{\varphi}\left(1-\bar{\mu}^j+\bar{\mu}^j z\right)}\right). \tag{30}$$

Shifted Poisson batches. We consider that the batch arrival is shifted Poisson distributed with pmf $P(X=i)=\frac{\varphi^{i-1}e^{-\varphi}}{(i-1)!}, i\geq 1$, where $\varphi>0$. Here, $G(z)=ze^{-\varphi(1-z)}$, and $H(z)=\bar{\lambda}+\lambda ze^{-\varphi(1-z)}$. Applying this in (19), we have

$$P(z)=\left(\bar{\lambda}+\lambda ze^{-\varphi(1-z)}\right)\prod_{j=1}^{\infty}\left[\bar{\lambda}+\lambda\left(1-\bar{\mu}^j+\bar{\mu}^j z\right)e^{-\varphi\bar{\mu}^j(1-z)}\right]. \tag{31}$$

Shifted Bernoulli batches. Let us assume the batch size is a shifted Bernoulli random variable that takes only two values: $P(X=1)=\bar{\varphi}$ and $P(X=2)=\varphi$. So, $G(z)=z(\bar{\varphi}+\varphi z)$, which gives $H(z)=\bar{\lambda}+\lambda z(\bar{\varphi}+\varphi z)$. In this case, the PGF is

$$P(z)=\left(\bar{\lambda}+\lambda z(\bar{\varphi}+\varphi z)\right)\prod_{j=1}^{\infty}\left[1-\lambda\bar{\mu}^j(1-z)\left(1+\varphi-\varphi\bar{\mu}^j(1-z)\right)\right]. \tag{32}$$

Shifted Binomial batches. Let us assume the batch size is shifted binomial random variable and its pmf is $P(X=i)=\binom{n}{i-1}\varphi^{i-1}\bar{\varphi}^{n-(i-1)}, i=1,2,\ldots,n+1$ and $G(z)=z(\bar{\varphi}+\varphi z)^n$, which gives $H(z)=\bar{\lambda}+\lambda z(\bar{\varphi}+\varphi z)^n$. In this case, the pgf is

$$P(z)=\left(\bar{\lambda}+\lambda z(\bar{\varphi}+\varphi z)^n\right)\prod_{j=1}^{\infty}\left[\bar{\lambda}+\lambda\left(1-\bar{\mu}^j(1-z)\right)\left(1-\varphi\bar{\mu}^j(1-z)\right)^n\right]. \tag{33}$$

Table 1 shows the impact of several probability distributions for the arrival batch sizes on queueing performance.

Distribution	Queueing performance
Geometric	High variability; bursty traffic patterns
Shifted Poisson	Analytical tractability; moderate randomness;
Shifted Bernoulli	Sparse arrivals; low variability
Shifted Binomial	Smoother traffic; moderate variability

Table 1: *Impact of batch-size distribution on the queueing performance.*

The geometric distribution typically exhibits the highest effective burstiness among these distributions, because it allows unbounded batch sizes, which can increase the expected number of jobs in the system and lead to larger fluctuations in instantaneous workload. In contrast, shifted Bernoulli and shifted binomial arrivals produce more regulated traffic patterns. They are frequently employed as baseline models for systems that are not heavily loaded or subject to regulations. The shifted Poisson distribution is used due to its memorylessness and mathematical tractability. As a result, the batch-size distribution selection should account for anticipated fluctuations in actual workloads, especially in burst-sensitive settings such as serverless computing systems.

5. *Geo/Geo/*∞ queue with LAS-DA policy

Taking $g_1=1$ and $g_i=0, \forall i\geq 2$, the above model reduce to $Geo/Geo/\infty$ queue with LAS-DA policy. Here, the functional equation is

$$P(z)=\left(\bar{\lambda}+\lambda z\right)P\left(\mu+\bar{\mu}z\right). \tag{34}$$

Taking derivative of Eq. (34) with respect to z and and putting $z = 1$, we obtain

$$P(z)^{(1)}|_{z=1} = \sum_{n=0}^{\infty} nP_n = \frac{\lambda}{\mu} = L_s. \tag{35}$$

Since we have as many servers as requests in the system, the average number of requests in the queue and the average waiting time in the queue is zero. The sojourn time is the mean service time, that is, $W_s = \frac{1}{\mu}$. Applying power series expansion about $z = 1$, from Eq. (34), we get Eq. (7), where

$$d_k = \frac{\ell\lambda\bar{\mu}^{k-1}}{1-\bar{\mu}^k} d_{k-1}, \quad d_0 = 1. \tag{36}$$

Using recursively the above, we get

$$d_k = k!\lambda^k \prod_{j=0}^{k-1} \frac{\bar{\mu}^j}{1-\bar{\mu}^{j+1}}, \quad k \geq 1, \quad \text{thus,} \quad \sum_{\ell=0}^{\infty} P_\ell z^\ell = \sum_{\ell=0}^{\infty} \frac{z^\ell}{\ell!} \sum_{k=\ell}^{\infty} \frac{(-1)^{k-\ell} d_k}{(k-\ell)!}. \tag{37}$$

Remark 3. *Taking inter-arrival and service times as exponentially distributed, with mean arrival and mean service rates γ and η, respectively. Divide the time axis into equal intervals $\Delta > 0$ such that $\lambda = \gamma\Delta$, $\mu = \eta\Delta$, and $\rho = \frac{\gamma\Delta}{\eta\Delta} = \rho_1$, where Δ is appropriately small. Setting $\lambda = \gamma\Delta$, $\mu = \eta\Delta$, and taking the limit as $\Delta \to 0$, we have*

$$\lim_{\Delta\to 0} d_k = k! \lim_{\Delta\to 0} (\gamma\Delta)^k \prod_{j=0}^{k-1} \frac{(1-\eta\Delta)^j}{1-(1-\eta\Delta)^{j+1}} = \left(\frac{\gamma}{\eta}\right)^k = \rho_1^k. \tag{38}$$

So,

$$\sum_{\ell=0}^{\infty} P_\ell z^\ell = \sum_{\ell=0}^{\infty} \frac{\rho_1^\ell z^\ell}{\ell!} \sum_{k=\ell}^{\infty} \frac{(-1)^{k-\ell}\rho_1^{k-\ell}}{(k-\ell)!} = e^{-\rho_1} \sum_{\ell=0}^{\infty} \frac{(\rho_1 z)^\ell}{\ell!}, \tag{39}$$

which matches with the results of Shortle et al. [24].

5.1. *Geo/Geo/∞* queue with EAS policy

Using the same procedure as discussed in previous subsection, we get the following expressions

$$Q(z) = \left(\bar{\lambda} + \lambda(\mu + \bar{\mu}z)\right) Q\left(\mu + \bar{\mu}z\right), \tag{40}$$

which is a functional equation. We obtain the probabilities after simplification

$$Q_j = \sum_{k=j}^{\infty} \binom{k}{j} (-1)^{k-j} \lambda^k \prod_{i=1}^{k} \frac{\bar{\mu}^i}{1-\bar{\mu}^i}, \quad j \geq 0. \tag{41}$$

The PGF of outside observer's observation epoch is $Q^o(z) = \left(\bar{\lambda} + \lambda z\right) Q(z)$, and

$$Q_j^o = \bar{\lambda} \sum_{k=j}^{\infty} \binom{k}{j} (-1)^{k-j} \lambda^k \prod_{i=1}^{k} \frac{\bar{\mu}^i}{1-\bar{\mu}^i} + \sum_{k=j-1}^{\infty} \binom{k}{j-1} (-1)^{k-j+1} \lambda^{k+1} \prod_{i=1}^{k} \frac{\bar{\mu}^i}{1-\bar{\mu}^i}. \tag{42}$$

The mean number of tasks in the system at outside observer's epoch is $L_s^o = \frac{\lambda}{\mu}$, which is same as of LAS-DA policy.

6. Numerical illustrations

In this section, we examine how model parameters affect several probability distributions and various performance metrics. The numerical experiments demonstrate some of the qualitative aspects of the system under study and validate the analytical results obtained in the sections above. The numerical calculations were performed on a PC with an Intel(R) Core(TM) i7-1165G7 CPU @ 2.80 GHz and 16GB RAM, using Maple 22 software.

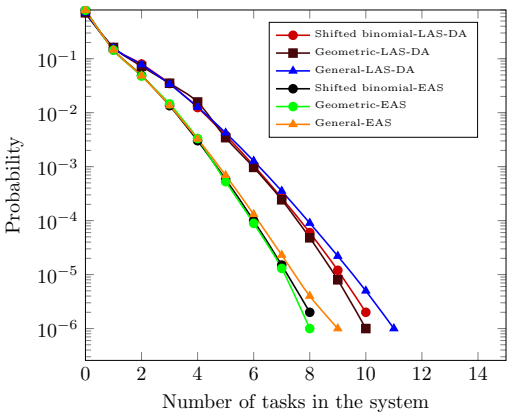

Figure 5: *Number of tasks vs probability.*

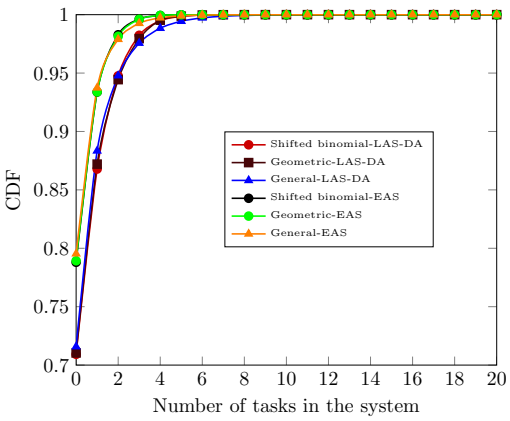

Figure 6: *Number of tasks vs CDF.*

Fig. 5 illustrates the probabilities of several requests for various probability distributions of arrival batch sizes for EAS and LAS-DA policies. As the number of tasks increases, the probability decreases. For a shifted binomial batch, the probability is lower than for geometric- and general-distributed batch sizes, as the batch size is small or bounded, resulting in more compactly focused arrival forms. In contrast, distributions with higher variability, such as the arbitrary and geometric batch size distributions, exhibit noticeably larger fluctuations. Namely, large batches can push many jobs into the system simultaneously, increasing the likelihood of transient workload spikes. Fig. 6 depicts the cumulative distribution function (CDF) of the number of tasks for various distributions of fixed mean batch size. The CDF increases with the number of tasks in the system, and eventually attains its maximum value. As expected, the number of tasks is maximum when the probability distributions of arrival batch sizes are arbitrary and minimum when the distribution is shifted binomial. Fig. 7 shows the probability distribution of the number of tasks in the system for both EAS and LAS-DA policies. When the system is empty, the probability in EAS is greater than in LAS-DA, because arrivals are admitted immediately before departures. When the system is not idle, EAS provides immediate service and reduces delays, but it can create sharper peaks in resource demand. Contrary, LAS-DA imposes a one-slot delay, which can be advantageous for systems that require batching or synchronization, as it reduces the risk of overload at slot boundaries.

For serverless systems, this difference translates into cost and performance trade-offs. Therefore, whether the objective is to minimize delay or to stabilize resource consumption under bursty arrival patterns, managers must align the system selection with operational requirements. Fig. 8 presents the number of tasks in the system when the arrival of tasks is single for various ρ. Here, we consider LAS-DA policy. For fixed ρ, as the number of tasks in the system increases, the probability decreases. For a fixed number of tasks in the system, the probability decreases as ρ increases. In addition to the qualitative trends observed in the figures, several quantitative insights emerge regarding the impact of the batch-size distribution on system performance. Specifically, the findings emphasize the importance of arrival variability. For distributions with the same mean batch size, the variance significantly affects the expected

number of jobs in the system.

Table 2 depicts steady-state probabilities of the $Geo^X/Geo/\infty$ queue for various probability distributions for the arrival batch sizes. The parameters are $\lambda = 0.1, \mu = 0.4$ and mean batch size $\bar{g} = 2$. In all cases, the probability decreases as the number of tasks increases. Probabilities in the case of an idle server are higher when the arrival batch size follows a Poisson distribution. The structure of arrival bursts influences system performance in addition to the mean arrival rate. Large numbers of function instances may be present in serverless computing systems, even when the average workload remains constant; distributions with greater variance can significantly raise the peak number of concurrent executions when launched. This observation underscores the importance of accounting for arrival variability when designing resource provisioning techniques, scaling policies, and concurrency constraints.

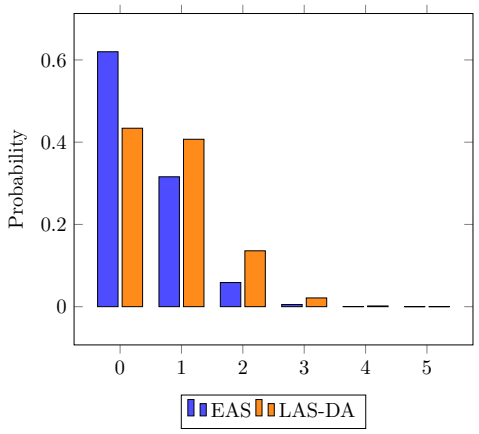

Figure 7: *Distribution of the number of tasks in the system.*

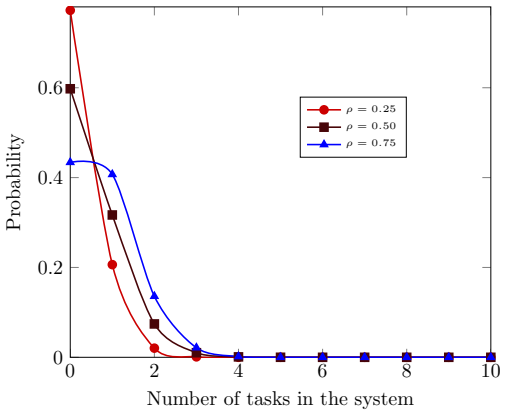

Figure 8: *Distribution of the number of tasks in the system with single server.*

	$\lambda = 0.1, \mu = 0.4, \bar{g} = 2$					
	Geometric		Shifted Poisson		Shifted Binomial	
n	P_n	Q_n	P_n	Q_n	P_n	Q_n
0	0.715786	0.795318	0.709933	0.788815	0.709210	0.788012
1	0.167613	0.142052	0.159749	0.145255	0.158627	0.145723
2	0.064377	0.041546	0.077744	0.048202	0.079778	0.049075
3	0.027881	0.013679	0.033875	0.013610	0.034745	0.013471
4	0.012738	0.004743	0.012725	0.003269	0.012435	0.003020
5	0.005994	0.001691	0.004229	0.000691	0.003817	0.000582
10	0.000166	0.000011	0.000005	0.000000	0.000002	0.000000
15	0.000005	0.000000	0.000000	0.000000	0.000000	0.000000
20	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
Sum	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Table 2: *Steady-state probabilities of the $Geo^X/Geo/\infty$ queue.*

An effective model for comprehending serverless dynamics is the discrete-time batch-arrival infinite-server queue. Managers should use it as a forecasting tool to ensure governance, budget for volatility, and anticipate bursts. This model yields several practical qualitative insights for managers operating serverless policies. In serverless computing, the infinite server queue enables instant allocation of compute resources without waiting. Managers must anticipate cost surges when large batches trigger simultaneous function executions. Due to discrete time modelling, time-slotted arrivals mirror real-world scheduling and help managers forecast peak-demand windows and optimize resource allocation. Pay-per-use billing can cause budget shocks

when large batches arrive in short succession, leading to volatile costs. Infinite servers eliminate queue delays but complicate system observability, and managers must track thousands of concurrent executions. Large batch arrivals may involve sensitive data; managers must ensure regulatory compliance during high-volume processing. The infinite-server batch-arrival queue model captures serverless elasticity; managers should exploit it for bursty workloads. Use predictive analytics to forecast batch arrival patterns and set budget alerts. Establish monitoring dashboards for batch arrivals and execution concurrency. Diversify across providers to reduce lock-in and compliance exposure. Infinite servers can scale beyond budget; managers must set usage thresholds. Batch arrivals may mask inefficiencies; invest in real-time analytics. Engage compliance teams early when handling sensitive batch workloads.

7. Conclusions

This paper examines the effectiveness of serverless computing in a discrete-time infinite-server batch/single-arrival queue, where customers arrive in batches rather than individually. Each customer is immediately served by one of an infinite number of servers, thereby eliminating waiting times due to unlimited processing capacity. We obtain state probabilities at various epochs, along with their corresponding performance measures. We illustrate numerical illustrations in the form of graphs and tables. The batch-arrival infinite-server queue paradigm in a serverless architecture is ideal for managing concurrent workloads on cloud platform. The discrete-time model offers valuable insights for serverless computing architectures, as it can capture key aspects of event-driven, auto-scaling systems. Thus, using discrete-time queueing models to analyze serverless system performance helps predict scalability, optimize resource allocation, and forecast scalability. In serverless computing platforms, where each incoming request is executed in a separate execution environment and, in theory, may be served instantly without waiting in line, this paradigm is particularly pertinent. Naturally, traffic spikes caused by external factors, such as IoT signals, user surges, and planned jobs, are modelled as batch arrivals. However, the infinite-server assumption imposes significant restrictions. Infrastructure limitations may result in implicit resource coupling, and providers typically set concurrency limits at the account or region level. It can be further extended for greater realism and practical relevance, such as assuming an infinite number of servers, a massive but finite number, leading to discrete-time $GI^X/G/c$ models, or state-dependent service distributions to model initialisation delays. Generalising discrete-time batch-arrival models along these lines would serve as a bridge between analytically tractable infinite-server queues and the operating conditions of contemporary cloud-native and serverless systems.

References

[1] Altman, E. and Yechiali, U. (2008). Infinite-server queues with system's additional tasks and impatient customers. Probability in the Engineering and Informational Sciences, 22(4), 477–493. doi: 10.1017/S0269964808000296

[2] Artalejo, J. R. and Hernández-Lerma, O. (2003). Performance analysis and optimal control of the Geo/Geo/c queue. Performance Evaluation, 52(1), 15–39. doi: 10.1016/S0166-5316(02)00161-X

[3] Chan, W. C. and Maa, D. Y. (1978). The GI/Geom/N queue in discrete time. INFOR: Information Systems and Operational Research, 16(3), 232–252. doi: 10.1080/03155986.1978.11731705

[4] Chaudhry, M. L. and Kim, J. D. (2004). Complete analytic and computational analyses of the discrete-time bulk-arrival infinite-server system: $GI^X/Geom/\infty$. Computers & Operations Research, 31(13), 2119–2135. doi: 10.1016/S0305-0548(03)00167-9

[5] Chaudhry, M. L., Gupta, U. C. and Goswami, V. (2001). Modeling and analysis of discrete-time multiserver queues with batch arrivals: $GI^X/Geom/m$. INFORMS Journal on Computing, 13(3), 172–180. doi: 10.1287/ijoc.13.3.172.12627

[6] Daw, A. and JPender, J. (2019). On the distributions of infinite server queues with batch arrivals. Queueing Systems, 91(3), 367–401. doi: 10.1007/s11134-019-09603-4

[7] Finol, G., París, G., García-López, P. and Sánchez-Artigas, M. (2024). Exploiting inherent elasticity of serverless in algorithms with unbalanced and irregular workloads. Journal of Parallel and Distributed Computing, 190, 104891. doi: 10.1016/j.jpdc.2024.104891

[8] Brian H Fralix, B. H. and Adan, I. J. (2009). An infinite-server queue influenced by a semi-markovian environment. Queueing Systems, 61(1), 65–84. doi: 10.1007/s11134-008-9100-y

[9] Goswami, V. and Gupta, U. C. (1998). Analyzing the discrete-time multiserver queue Geom/Geom/m queue with late and early arrivals. Information and Management Sciences, 9(2), 55–66. url: eudml.org/doc/249750 [Accessed 4/1/2025]

[10] Goswami, V. and Mund, G. B. (2017). Computational analysis of multi-server discrete-time queueing system with balking, reneging and synchronous vacations. RAIRO-Operations Research, 51(2), 343–358. doi: 10.1051/ro/2016025

[11] Goswami, V., Gupta, U. C. and Samanta, S. K. (2006). Analyzing discrete-time bulk-service $Geo/Geo^b/m$ queue. RAIRO-Operations Research, 40(3), 267–284. doi: 10.1051/ro:2006021

[12] Gravey, A. and Hébuterne, G. (1992). Simultaneity in discrete-time single server queues with Bernoulli inputs. Performance Evaluation, 14(2), 123–131. doi: 10.1016/0166-5316(92)90014-8

[13] Hunter, J. J. (1983). *Mathematical Techniques of Applied Probability, Volume II, Discrete Time Models: Techniques and Applications*. New York: Academic Press.

[14] Kaffes, K., Yadwadkar, N. J. and Kozyrakis, C. (2021). Practical scheduling for real-world serverless computing. arXiv preprint arXiv:2111.07226. doi: 10.48550/arXiv.2111.07226

[15] Kelly, D., Glavin, F. and Barrett, E. (2020). Serverless computing: Behind the scenes of major platforms. In: *2020 IEEE 13th International Conference on Cloud Computing(CLOUD)*, 304–312. doi: 10.1109/CLOUD49709.2020.00050

[16] Kumar, R. (2017). A transient solution to the M/M/c queuing model equation with balking and catastrophes. Croatian Operational Research Review, 8(2), 577–591. doi: 10.17535/crorr.2017.0037

[17] Nassar, H. (2003). A novel approach to analyzing the occupancy of the $Geo^X/Geo^Y/\infty$ queueing system. AEU-International Journal of Electronics and Communications, 57(6), 391–394. doi: 10.1078/1434-8411-54100190

[18] Palomo, S. and Pender, J. (2024). Overlap times in the infinite server queue. Probability in the Engineering and Informational Sciences, 38(1), 21–27. doi: 10.1017/S0269964822000456

[19] Pang, G. and Whitt, W. (2012). Infinite-server queues with batch arrivals and dependent service times. Probability in the Engineering and Informational Sciences, 26(2), 197–220. doi: 10.1017/S0269964811000337

[20] Ramaswami, V. and Neuts, M. F. (1980). Some explicit formulas and computational methods for infinite-server queues with phase-type arrivals. Journal of Applied Probability, 17(2), 498–514. doi: 10.2307/3213039

[21] Reynolds, J. F. (1968). Some results for the bulk-arrival infinite-server Poisson queue. Operations Research, 16(1), 186–189. doi: 10.1287/opre.16.1.186

[22] Rubin, I. and Zhang, Z. (2002). Message delay and queue-size analysis for circuit-switched TDMA systems. IEEE Transactions on Communications, 39(6), 905–914. doi: 10.1109/26.87180

[23] Shanbhag, D. N. (1966). On infinite server queues with batch arrivals. Journal of Applied Probability, 3(1), 274–279. doi: 10.2307/3212053

[24] Shortle, J. F., Thompson, J. M., Gross, D. and Harris, C. M. (2018). Fundamentals of queueing theory. New York: John Wiley & Sons, Inc. doi: 10.1002/9781119453765

[25] Sinha, P., Kaffes, K. and Yadwadkar, N. J. (2024). Shabari: Delayed decision-making for faster and efficient serverless functions. arXiv preprint arXiv:2401.08859. doi: 10.48550/arXiv.2401.08859

[26] Wen, J., Chen, Z., Sarro, F. and Liu, X. (2023). Revisiting the performance of serverless computing: An analysis of variance. arXiv preprint arXiv:2305.04309. doi: 10.48550/arXiv.2305.04309

[27] Wen, J., Chen, Z., Sarro, F. and Wang, S. (2025). Unveiling overlooked performance variance in serverless computing. Empirical Software Engineering, 30(2), 30–59. doi: 10.1007/s10664-025-10615-3

[28] Zhu, L., Giotis, G., Tountopoulos, V. and Casale, G. (2021). RDOF: deployment optimization for function as a service. In: *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, 508–514. doi: 10.1109/CLOUD53861.2021.00066