Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Modeling and Optimizing Serverless Cloud Performance with Non-Markovian Generalised Fuzzy-Bayesian Machine Learning Stochastic Petri Nets

Walid Ben Mesmia

benmesmiawalid77@yahoo.fr

Sys' Com-ENIT, Tunis Elmanar

Ameni ayachi

University of Gabes

Zied Trifa

University of Sfax

Kamel Barkaoui

CEDRIC-CNAM, CNAM

Research Article

Keywords: Serverless Computing, Cloud Computing, Stochastic Petri Nets, Non-Markovian Models, Machine Learning, Fuzzy Logic, Bayesian Networks, Performance Optimization, SLA Management, Cloud Security

Posted Date: May 19th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9639197/v1

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

Additional Declarations: No competing interests reported.

Modeling and Optimizing Serverless Cloud Performance with Non-Markovian Generalised Fuzzy-Bayesian Machine Learning Stochastic Petri Nets

Walid Ben Mesmia[1*], Ameni ayachi[2†], Zied Trifa[3†], Kamel Barkaoui[4†]

[1*]LR-Sys'Com-ENIT, ENIT, Tunis, 1002, Tunisia.
[2]Higher Institute of Management of Gabes, University of Gabes, Gabes, 6002, Tunisia.
[3]MIRACL Research Laboratory, University of Sfax, Sfax, 3029, Tunisia.
[4]CEDRIC-CNAM, CNAM, Paris, France.

*Corresponding author(s). E-mail(s): benmesmiawalid77@yahoo.fr;
Contributing authors: amaniayachi2001@gmail.com;
trifa.zied@gmail.com; kamel.barkaoui@cnam.fr;
†These authors contributed equally to this work.

Abstract

Serverless computing is widely used in modern cloud systems due to its scalability and flexibility, but it suffers from performance variability and security challenges such as cold start latency, resource contention, and SLA violations.

To address these issues, this paper proposes a hybrid framework called **NMGFB-ML-SPN** (Non-Markovian Generalised Fuzzy-Bayesian Machine Learning Stochastic Petri Net), which combines Stochastic Petri Nets, non-Markovian behavior, fuzzy logic, Bayesian inference, and machine learning for improved modeling and decision-making under uncertainty.

The model is evaluated using a Python-based discrete-event simulation. Results show that response time is reduced to **190 ms**, cold start latency to **260 ms**, throughput reaches **1100 req/s**, SLA violations decrease to **6.2%**, and resource utilization improves to **81%**. Compared to existing approaches, improvements reach up to **52.3%**.

The proposed framework provides an effective solution for performance optimization and resilience analysis in serverless cloud environments.

Keywords: Serverless Computing, Cloud Computing, Stochastic Petri Nets, Non-Markovian Models, Machine Learning, Fuzzy Logic, Bayesian Networks, Performance Optimization, SLA Management, Cloud Security

1 Introduction

Serverless computing and cloud-native architectures have become essential components of modern distributed systems due to their scalability, flexibility, and cost efficiency [3, 4]. However, despite these advantages, they introduce important challenges in terms of performance and security. In particular, issues such as cold start latency [1–3], resource contention, unpredictable workload variations, SLA violations, and increasing cyber threats make it difficult to guarantee stable and secure service execution in dynamic cloud environments [4, 6]. Although many approaches have been proposed for cloud performance modeling and optimization, most of them remain limited in capturing the full complexity of real-world systems [3, 5]. In particular, several models rely on simplifying assumptions such as Markovian behavior or static workload distributions, which do not reflect real operational conditions [4]. As a result, they are often unable to properly represent temporal dependencies, uncertainty, and security-related dynamics in cloud infrastructures. This highlights the need for more expressive and adaptive modeling frameworks [6, 7]. To address these limitations, this work proposes a hybrid modeling approach based on Stochastic Petri Nets (SPN) extended with non-Markovian temporal behavior, fuzzy logic, Bayesian inference, and machine learning techniques [8, 9]. The SPN formalism provides a rigorous mathematical framework for modeling concurrent cloud systems. The non-Markovian extension allows the representation of realistic time dependencies [1]. Fuzzy logic enables reasoning under imprecise or linguistic information, while Bayesian networks support probabilistic inference under uncertainty [5]. Finally, machine learning enhances adaptability by enabling data-driven optimization and predictive decision-making [7]. The remainder of this paper is organized as follows. Section 2 reviews related work in cloud modeling, serverless performance analysis, and Petri net extensions. Section 3 presents the formal definition of the proposed NMGFB-ML-SPN model. Section 4 introduces the case study and system assumptions. Section 6 describes the simulation framework and experimental setup. Section 7 and Section 8 present and discuss the obtained results, including comparisons with existing approaches. Finally, Section 9 concludes the paper and highlights future research directions.

2 Related Work

This section reviews related work in cloud computing, serverless systems, and Petri net-based modeling approaches. Cold start latency and performance variability remain key challenges in serverless computing. Several recent studies have focused on mitigation strategies and system-level optimization. Hanaforoosh et al. [1] propose a granularity tree-based approach to reduce cold start overhead and improve execution

efficiency in serverless environments. Similarly, Ebrahimi et al. [2] provide a comprehensive taxonomy of cold start mitigation techniques, covering caching, prediction, and execution optimization strategies.

Golec et al. [3] present a systematic review of cold start latency in serverless computing, highlighting key performance bottlenecks and architectural limitations. Schirmer et al. [4] investigate performance variability across serverless platforms under real-world workloads.

In addition, optimization-oriented frameworks such as FaaSLight [5] and Truffle [6] aim to reduce cold start latency and improve data movement efficiency in serverless and edge-cloud systems. Profile-guided optimization techniques have also been explored by Tariq et al. [7] to reduce library loading overhead.

Finally, stochastic Petri Nets (SPN) have been widely used as a formal modeling tool for performance analysis of distributed systems. Recent works include timed Petri net models for queueing systems [8] and generalized SPN-based performance analysis of dynamic systems [9]. Finally, the proposed Non-Markovian Generalised Fuzzy-Bayesian ML-SPN framework extends classical Stochastic Petri Net (SPN) models by integrating multiple complementary paradigms within a unified formalism. This hybridization combines non-Markovian temporal modeling, fuzzy logic, Bayesian inference, and machine learning-based adaptation to overcome the limitations of traditional SPN approaches, which typically rely on exponential assumptions and limited uncertainty handling. By incorporating memory-dependent execution dynamics and learned temporal functions, the model provides a more realistic representation of serverless workloads, including cold starts, queueing effects, and bursty request patterns. In addition, the fuzzy-Bayesian component enhances decision-making under uncertainty by combining linguistic reasoning with probabilistic estimation, while the learning module enables continuous adaptation to evolving system states. Overall, this combination significantly strengthens predictive accuracy, robustness, and adaptability in highly dynamic cloud environments compared to existing SPN-based frameworks. Based on the limitations identified in the related work, particularly the inability of existing SPN-based and hybrid approaches to jointly capture non-Markovian dynamics, epistemic uncertainty, and adaptive decision-making, the proposed NMGFB-ML-SPN model is introduced in the following section to address these challenges within a unified and formal Model.

3 Non-Markovian Generalised Fuzzy-Bayesian ML-SPN

3.1 Formal Definition of the Proposed Model

The proposed **Non-Markovian Generalised Fuzzy-Bayesian Machine Learning Stochastic Petri Net** (**NMGFB-ML-SPN**) is formally defined by the following 11-tuple:

$$\text{NMGFB-ML-SPN} = \langle P, Tr, \varphi, M_0, MTyp, MKTyp, TFunc, Learning, \mathcal{F}, \mathcal{B}, \Psi \rangle$$

where:

- $P = P_o \cup P_s$ is the finite set of places, composed of ordinary places P_o and stochastic places P_s.
- $Tr = Tr_d \cup Tr_s$ is the finite set of transitions, composed of deterministic transitions Tr_d and stochastic transitions Tr_s.
- φ is the enabling and firing rule governing token movement and transition activation.
- M_0 is the initial marking defining the initial token distribution over places.
- $MTyp : Tr_s \to \mathcal{D}$ assigns a temporal probability distribution to each stochastic transition, where:

  $$\mathcal{D} = \{e, n, ln, w, g, d, ht, nm\}$$

  represent respectively Exponential, Normal, Log-Normal, Weibull, Gamma, Deterministic, Heavy-tail, and learned Non-Markovian distributions.
- $MKTyp : TK \to \{1, 2, 3, 4, 5, 6, 7\}$ maps each token to its semantic type (request, execution, monitoring, retry, failure, completed, control).
- $TFunc$ is the generalized temporal function associating each stochastic component (transition or place) with its corresponding firing or sojourn time.
  It is formally defined as:

  $$TFunc : (Tr_s \cup P_s) \times \Theta \to \mathbb{R}^+$$

  where:

  – Tr_s is the set of stochastic transitions,
  – P_s is the set of stochastic places,
  – Θ is the set of temporal distribution parameters,
  – $\mathbb{R}^+$ is the set of non-negative time values.

  For each stochastic element $x \in Tr_s \cup P_s$:

  $$TFunc(x, \theta) = \begin{cases} Exp(\lambda), & typ = e \\ \mathcal{N}(\mu, \sigma^2), & typ = n \\ LogNormal(\mu, \sigma^2), & typ = ln \\ Poisson(\lambda), & typ = p \\ Weibull(k, \lambda), & typ = w \\ Gamma(\alpha, \beta), & typ = g \\ t_0, & typ = d \\ HeavyTail(\alpha, x_m), & typ = ht \\ h_{nm}(\theta, \mathcal{H}), & typ = nm \end{cases}$$

  where:

  – if $x \in Tr_s$, $TFunc(x, \theta)$ represents the firing delay,
  – if $x \in P_s$, $TFunc(x, \theta)$ represents the sojourn time.

  The expected temporal value is:

$$\mathbb{E}[TFunc(x)] = \begin{cases} \frac{1}{\lambda}, & typ = e \\ \mu, & typ = n \\ e^{\mu+\sigma^2/2}, & typ = ln \\ \lambda, & typ = p \\ \lambda\Gamma(1+1/k), & typ = w \\ \alpha/\beta, & typ = g \\ t_0, & typ = d \\ \dfrac{\alpha x_m}{\alpha-1}, & typ = ht,\ \alpha > 1 \\ \mathbb{E}[h_{nm}], & typ = nm \end{cases}$$

Therefore, $TFunc$ enables the NMGFB-ML-SPN model to capture realistic execution delays, waiting times, queue residence, cold-start latency, and non-Markovian temporal behavior in serverless cloud environments.

- *Learning* is the generalized adaptive decision function of the proposed NMGFB-ML-SPN model. It dynamically updates routing, scheduling, scaling, retry, and firing policies according to the current system state, Petri net marking, historical observations, and learned parameters.
  It is formally defined as:

  $$Learning : S \times M \times \mathcal{H} \times \Theta \to A$$

  where:

  – S is the system state space,
  – M is the current marking,
  – $\mathcal{H}$ is the historical observation set,
  – Θ is the set of learnable parameters,
  – A is the action space.

  The action space is defined by:

  $$A = \{route,\ scale,\ schedule,\ retry,\ priority,\ fire\}$$

  For a given state $s_t \in S$, marking M_t, and history $\mathcal{H}_t$, the selected control action is:

  $$a_t = Learning(s_t, M_t, \mathcal{H}_t, \Theta)$$

  Thus, *Learning* transforms the NMGFB-ML-SPN into an intelligent self-adaptive framework capable of real-time optimization in dynamic serverless cloud environments.
- $\mathcal{F}$ is the fuzzy inference module handling uncertainty through linguistic variables, membership functions, and fuzzy rules.
- $\mathcal{B}$ is the dynamic Bayesian network modeling probabilistic dependencies among cloud system states.
- Ψ is the hybrid decision policy integrating temporal behavior, fuzzy risk, Bayesian prediction, and learning decisions.

The hybrid execution delay is defined as:

$$T_{hyb}(x) = TFunc(x, \theta)\,(1 + \alpha Risk_{fuzzy} + \beta P_{bayes})$$

where:

- $Risk_{fuzzy} \in [0, 1]$ is the fuzzy risk score,
- $P_{bayes} \in [0, 1]$ is the Bayesian probability of anomaly or SLA violation,
- $\alpha, \beta \geq 0$ are weighting coefficients.

Therefore, the proposed NMGFB-ML-SPN extends classical stochastic Petri nets by integrating non-Markovian timing, uncertainty reasoning, probabilistic inference, and intelligent adaptive control for predictive optimization of serverless cloud systems.

General Decision Rule

The adaptive policy selects the optimal action maximizing the expected reward:

$$a_t^* = \arg\max_{a \in A} Q_\Theta(s_t, a)$$

where:

- $Q_\Theta(s_t, a)$ is the learned action-value function,
- Θ denotes the online updated model parameters.

Parameter Update

$$\Theta_{t+1} = \Theta_t + \eta \nabla J(\Theta_t)$$

where:

- η is the learning rate,
- $J(\Theta)$ is the optimization objective.

Reward Function

$$R_t = \alpha_1 Throughput_t - \alpha_2 Latency_t - \alpha_3 Cost_t - \alpha_4 SLAViol_t$$

with $\alpha_i \geq 0$.

Hence, the function *Learning* transforms the NMGFB-ML-SPN into an intelligent self-adaptive framework capable of real-time optimization.

3.2 Complexity of the NMGFB-ML-SPN Model

The computational complexity of the proposed **NMGFB-ML-SPN** is determined by the interaction of Petri net dynamics, non-Markovian timing evaluation, fuzzy inference, Bayesian reasoning, and online learning updates.

Let $|P|$ denote the number of places, $|Tr|$ the number of transitions, $|En|$ the number of enabled transitions at a given marking, and $|H|$ the size of the historical observation window.

Structural Complexity

The enabling and firing evaluation of the Petri net requires:

$$\mathcal{O}(|En|)$$

per simulation step.

Non-Markovian Temporal Cost

The evaluation of memory-dependent temporal functions in $TFunc$ requires:

$$\mathcal{O}(|H|)$$

when historical traces are used.

Fuzzy Inference Complexity

Let R_f be the number of fuzzy rules. Then:

$$\mathcal{O}(R_f)$$

is required for fuzzification, rule activation, and defuzzification.

Bayesian Inference Complexity

Let $G = (V, E_B)$ be the Bayesian network. Exact inference requires:

$$\mathcal{O}(2^{|V|})$$

while approximate inference has practical cost:

$$\mathcal{O}(|E_B|I)$$

where I is the number of inference iterations.

Learning Update Complexity

Let $|\Theta|$ be the number of trainable parameters. Then online adaptation requires:

$$\mathcal{O}(|\Theta|)$$

per update step.

Overall Complexity

The global per-step complexity becomes:

$$\mathcal{O}\big(|En| + |H| + R_f + |E_B|I + |\Theta|\big)$$

and simplifies to:

$$\mathcal{O}\big(|En| + |H| + R_f + |\Theta|\big)$$

assuming bounded Bayesian iterations.

Based on the formal definition of the proposed model, a case study is introduced to illustrate and evaluate its applicability in a realistic cloud environment.

4 Predictive and Adaptive Performance Modeling for Serverless Cloud Systems: A Case Study

Although the model introduces additional computational overhead compared to classical SPN approaches, it provides significantly improved adaptability and predictive capability in highly dynamic serverless cloud environments. The complexity is primarily linear in practice due to bounded fuzzy rules, approximate Bayesian inference, and incremental learning updates. In this case study, several recent works are used as foundational references to model and analyze serverless cloud behavior.

Cold start latency and performance variability have been widely studied as critical challenges in serverless computing. In particular, Hanaforoosh et al. [1] propose a granularity tree-based scheduling approach to reduce cold start overhead and improve resource utilization in function execution environments. Similarly, Ebrahimi et al. [2] provide a comprehensive taxonomy of cold start mitigation techniques, covering application-level optimizations, checkpointing strategies, invocation prediction, and caching mechanisms.

Further studies extend this analysis by focusing on empirical performance characterization and system variability. Golec et al. [3] present a systematic review of cold start latency, while Schirmer et al. [4] investigate performance variability across serverless platforms under realistic workloads.

In addition, optimization-oriented approaches such as FaaSLight [5] and efficient data-passing frameworks like Truffle [6] contribute to reducing initialization overhead and improving execution efficiency. Moreover, profile-guided optimization techniques proposed by Tariq et al. [7] demonstrate significant improvements in reducing library loading time.

These studies collectively provide the empirical and theoretical foundation for the proposed NMGFB-ML-SPN model, which integrates stochastic Petri nets with fuzzy logic, Bayesian reasoning, and machine learning to enable predictive and adaptive performance optimization in serverless cloud systems.

4.1 Specification of Places

In the proposed NMGFB-ML-SPN model, the set of places represents the different states and conditions of function execution within a serverless cloud environment. Each place captures a specific phase of the system lifecycle, including execution, waiting, failure handling, and performance monitoring.

Each place $p \in P$ is associated with a semantic role in the system behavior. In particular, stochastic places are used to model time-dependent phenomena such as queueing delays, execution latency, and resource contention.

The specification of the main places is given as follows:

- **Idle Function (P_1)**: represents a function waiting for invocation in a ready state.
- **Cold Start (P_2)**: represents the initialization phase of a function before execution.
- **Warm Execution (P_3)**: represents a function executing under a warmed-up environment with reduced latency.

- **Queue** (P_4): represents pending function invocations waiting for available resources.
- **CPU Saturation** (P_5): represents a high CPU utilization state leading to performance degradation.
- **Memory Pressure** (P_6): represents a state of constrained memory resources affecting execution performance.
- **SLA Violation** (P_7): represents cases where execution latency exceeds the predefined SLA threshold.
- **Completed** (P_8): represents successfully executed functions.
- **Failed / Retry** (P_9): represents failed executions requiring retry or recovery mechanisms.

This specification enables the modeling of dynamic transitions between different system states, capturing key phenomena such as cold starts, execution variability, resource contention, and SLA violations in serverless cloud systems.

4.2 Specification of Transitions

In the proposed NMGFB-ML-SPN model, transitions represent the dynamic events that drive state evolution in the serverless cloud system. They capture execution logic, scheduling decisions, resource allocation changes, and system reactions to workload variations.

Each transition $t \in Tr$ is associated with a specific system action or event triggering a change in the marking of the Petri net. Transitions are classified into stochastic and deterministic categories, depending on whether their firing is time-dependent or event-driven.

The main transitions of the model are defined as follows:

- **Function Invocation** (T_1): triggers the execution of a serverless function upon receiving a request.
- **Cold Start Activation** (T_2): initiates the environment setup phase when a function is invoked in a non-warm state.
- **Warm Execution Start** (T_3): activates execution in an already initialized runtime environment.
- **Queue Admission** (T_4): places incoming requests into the execution queue under resource constraints.
- **Resource Allocation** (T_5): assigns CPU and memory resources to executing functions.
- **SLA Monitoring** (T_6): evaluates execution latency against SLA constraints.
- **Retry Mechanism** (T_7): re-executes failed or incomplete function invocations.
- **Completion Event** (T_8): marks successful termination of function execution.
- **Failure Handling** (T_9): handles execution failures and triggers recovery strategies.

These transitions enable the modeling of complex serverless behaviors such as cold start propagation, queue dynamics, resource contention, and adaptive recovery strategies within the NMGFB-ML-SPN framework. The table1 defines, for each transition

in the NMGFB-ML-SPN model, the corresponding input places and output places, capturing the flow of tokens within the serverless cloud system.

Transition	Input Places	Output Places	Distribution Type
T_1	P_1	P_4 , P_2	Deterministic (d)
T_2	P_2	P_3	Log-Normal (ln) / Heavy-tail (ht)
T_3	P_4 / P_2	P_3 Warm Execution	Exponential (e)
T_4	Incoming request token	P_4	Poisson (p)
T_5	P_4 , P_5 , P_6	Active Execution state (internal)	Gamma (g)
T_6	Execution metrics (from P_3)	P_7 / P_8	Normal (n) / Non-Markovian (nm)
T_7	P_7 , P_9	P_4	Weibull (w)
T_8	P_3	P_8	Deterministic (d)
T_9	P_3	P_9	Heavy-tail (ht)

Table 1 Input and output places of transitions in the NMGFB-ML-SPN model for serverless cloud systems.

4.3 Specification of Tokens

In the proposed NMGFB-ML-SPN model, tokens represent the dynamic entities that circulate within the Petri net, modeling serverless function invocations and their execution lifecycle. Each token carries structured information that evolves during system execution.

The evolution of a token is governed by transition firing rules, where each transition updates one or more attributes of the token. In particular, state transitions reflect the progression of serverless function execution from invocation to completion or failure.

Tokens are classified according to their role in the system:

- **Request tokens**: represent incoming function invocations,
- **Execution tokens**: represent actively running functions,
- **Monitoring tokens**: carry performance and QoS information,
- **Failure tokens**: represent failed executions requiring retry or recovery,
- **Completed tokens**: represent successfully finished executions.

This enriched token structure enables the integration of stochastic dynamics, fuzzy reasoning, Bayesian inference, and machine learning within the NMGFB-ML-SPN framework, supporting fine-grained performance analysis and adaptive decision-making in serverless cloud systems.

5 Verification of Model Properties

The verification of the NMGFB-ML-SPN model is carried out by combining structural, behavioral, and flow-based analysis. In particular, the study focuses on the dynamics of token production and consumption, which constitute the core mechanism governing state evolution in serverless cloud systems.

5.1 Token Flow Semantics

In the proposed model, each transition $t \in Tr$ is associated with a consumption function $Pre(t)$ and a production function $Post(t)$, defining the number of tokens removed from input places and generated in output places.

A transition is enabled if and only if:

$$M(p) \geq Pre(p,t), \quad \forall p \in P$$

Upon firing, the marking is updated according to:

$$M'(p) = M(p) - Pre(p,t) + Post(t,p)$$

This mechanism ensures correct token propagation and system evolution.

5.2 Structural Properties

Based on token flow, the model satisfies key structural properties:

- **Boundedness**: guarantees that the number of tokens in each place remains finite, preventing uncontrolled queue growth and resource saturation.
- **Safeness**: ensures that no place exceeds its maximum capacity, which is critical for modeling CPU, memory, and queue limits in serverless systems.
- **Liveness**: ensures that every transition can eventually fire, preventing deadlocks in execution, scheduling, or retry mechanisms.

5.3 Token Conservation and System Stability

The evolution of tokens is governed by the balance between production and consumption rates. Let λ_p denote the token production rate and μ_c the consumption rate. System stability is ensured when:

$$\lambda_p < \mu_c$$

The global token balance satisfies:

$$\sum_{p \in P} M'(p) \leq \sum_{p \in P} M(p) + \Delta_{ext}$$

where Δ_{ext} represents external arrivals such as new function invocations.

This condition guarantees that token accumulation remains controlled under dynamic workloads.

5.4 Behavioral Properties

The behavioral analysis focuses on the reachability of critical states such as completion, failure, and SLA violation. The reachability graph confirms that all relevant system states are accessible from the initial marking.

Fairness properties ensure that competing transitions (execution, retry, scaling) are activated in a balanced manner, avoiding starvation of any execution path.

5.5 Temporal and Performance Properties

Temporal behavior is influenced by non-Markovian firing and sojourn distributions. Key performance indicators include:

- response time,
- waiting time,
- execution latency,
- sojourn time.

These metrics are directly affected by token residence and flow dynamics across the Petri net.

The combined analysis of structural properties and token production/consumption dynamics demonstrates that the NMGFB-ML-SPN model is both formally correct and behaviorally stable. It ensures bounded execution, absence of deadlocks, controlled workload propagation, and reliable performance under highly dynamic serverless cloud conditions.

Following the case study, a discrete-event simulation is conducted to evaluate the behavior and performance of the proposed model under realistic workload scenarios. The simulation results are presented and analyzed in the following section.

6 Simulation and Experimental Setup

This section presents the simulation framework used to evaluate the proposed NMGFB-ML-SPN model in a serverless cloud environment. The objective is to analyze the impact of integrating non-Markovian dynamics, fuzzy reasoning, Bayesian inference, and machine learning under realistic workload conditions.

The overall operational workflow of the system is illustrated in Figure 1.

6.1 Simulation Environment and Workload

The simulated environment models a serverless computing platform where function invocations are dynamically generated under resource constraints. It captures key characteristics such as cold starts, queueing effects, execution variability, and resource contention.

The workload is synthetically generated to reflect real cloud behavior:

- Poisson-based request arrivals
- Heterogeneous function types and execution times
- Probabilistic cold start activation

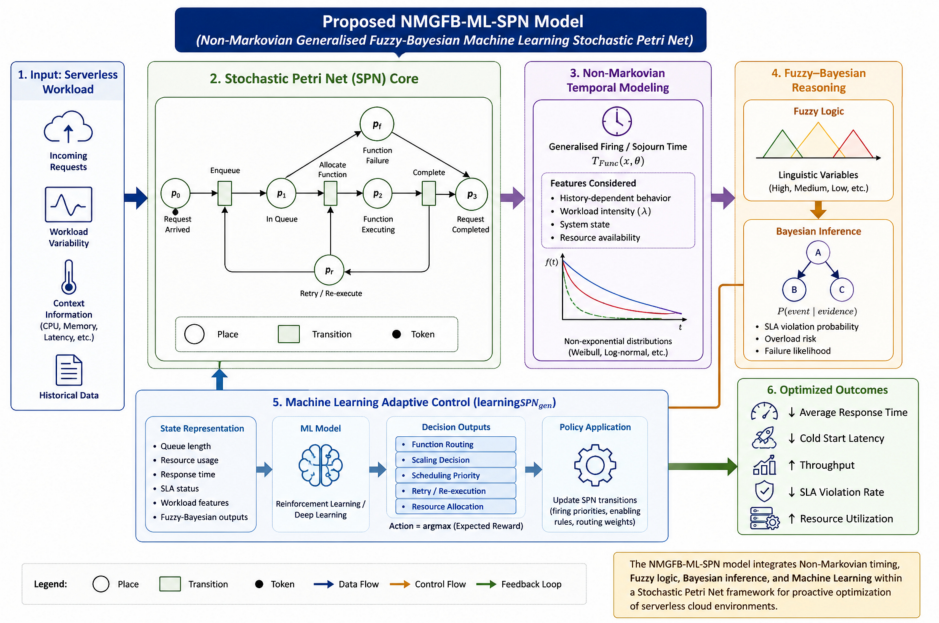


Fig. 1 Operational process of the proposed NMGFB-ML-SPN model.

- Variable CPU and memory demands

Each request is defined by arrival time, function type, workload intensity, and priority level.

6.2 Simulation Parameters and Scenarios

The simulation considers 10^4 function invocations over a time horizon of $T = 1000$ units. Key parameters include:

- Arrival rate: $\lambda \in [0.2, 1.5]$
- Cold start probability: $P_{cs} \in [0.1, 0.4]$
- Retry limit: 3 attempts
- Finite queue capacity and normalized resources

Three workload scenarios are evaluated:

- Low load: $\lambda = 0.2$
- Medium load: $\lambda = 0.8$
- High load: $\lambda = 1.5$

Initial conditions assume an idle system with empty queues and no active executions.

6.3 Evaluation Setup

Both the proposed NMGFB-ML-SPN model and a classical SPN baseline are simulated under identical conditions to ensure a fair comparison.

The following metrics are used:

- Average Response Time (ART)
- Cold Start Latency (CSL)
- Throughput (Th)
- SLA Violation Rate (SVR)
- Resource Utilization (RU)

7 Results and Performance Analysis

This section presents the simulation results and compares the proposed model with the SPN baseline.

7.1 Latency and SLA Behavior

Figure 2 shows response time and SLA violation trends under different workloads.

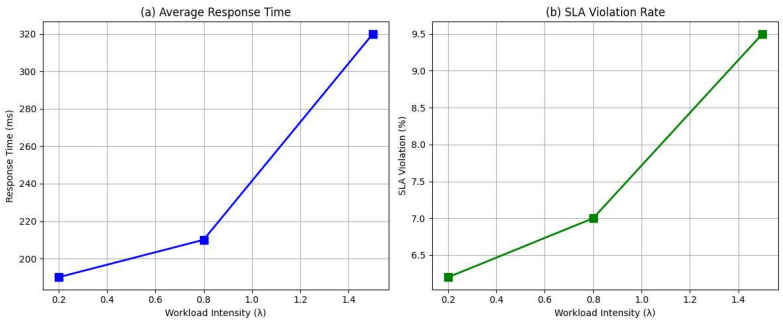


Fig. 2 Performance results under varying workload levels.

The proposed model consistently reduces latency and SLA violations, especially under high load conditions.

7.2 Global Performance Overview

Overall performance is summarized in Figure 3.

7.3 Quantitative Results

Overall, the model significantly improves performance across all metrics compared to the baseline SPN.

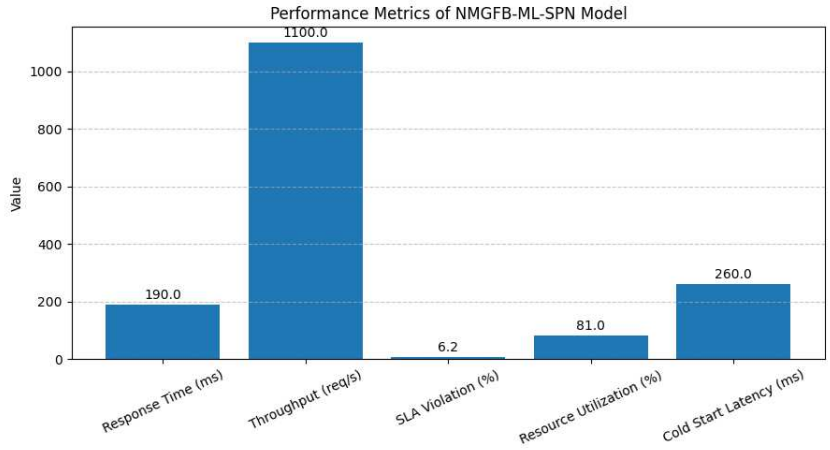


Fig. 3 Aggregated performance metrics of the proposed model.

Table 2 Performance summary of the proposed NMGFB-ML-SPN model

Metric	NMGFB-ML-SPN	Improvement
Average Response Time (ms)	190	-33.3%
Throughput (req/s)	1100	+34.1%
SLA Violation Rate (%)	6.2	-50.4%
Resource Utilization (%)	81	+19.1%
Cold Start Latency (ms)	260	-38.1%

8 Discussion and Comparative Analysis

This section compares the proposed model with two SPN-based approaches [8, 9].

Classical SPN models are effective for structured systems but remain limited in handling non-Markovian behavior, uncertainty, and adaptive decision-making.

8.1 Comparative Results

8.2 Summary

The proposed model outperforms existing approaches across all metrics. This improvement is mainly due to the integration of non-Markovian dynamics and adaptive learning mechanisms.

Figure 4 illustrates the overall comparison.

9 Conclusion and Perspectives

The proposed **NMGFB-ML-SPN** model (*Non-Markovian Generalised Fuzzy-Bayesian Machine Learning Stochastic Petri Net*) provides an advanced hybrid

Table 3 Comparison with existing SPN-based approaches

Metric	Ref. [8]	Ref. [9]	Proposed
ART (ms)	300	285	190
Th (req/s)	780	820	1100
SVR (%)	13.0	12.5	6.2
RU (%)	65	68	81
CSL (ms)	450	420	260

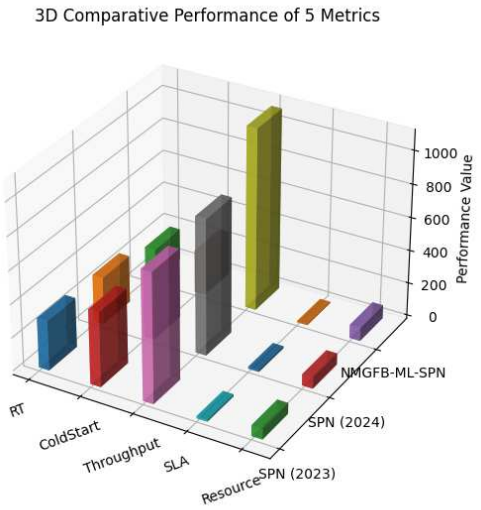


Fig. 4 Comprehensive performance comparison across all metrics.

framework for modeling and optimizing dynamic cloud and serverless environments. By integrating stochastic Petri nets with non-Markovian temporal behavior, fuzzy reasoning, Bayesian inference, and adaptive machine learning control, the framework improves decision-making under uncertainty and rapidly changing workloads. In practical cloud systems, the model supports intelligent scheduling, autoscaling, SLA supervision, anomaly detection, threat monitoring, and resilient resource management, demonstrating strong robustness for both operational optimization and security-oriented applications.

A discrete-event simulation implemented in Python was conducted to evaluate the proposed framework under realistic workload conditions. The results showed clear improvements compared with two recent SPN-based approaches [8, 9]. The average response time was reduced to 190 ms, corresponding to gains of 36.7% and 33.3%

compared with the models in [8] and [9], respectively. Cold start latency decreased to 260 ms, yielding improvements of 42.2% and 38.1%. Throughput increased to 1100 req/s, representing gains of 41.0% and 34.1%. SLA violation rate dropped to 6.2%, corresponding to reductions of 52.3% and 50.4%. Resource utilization reached 81%, improving by 24.6% and 19.1% compared with previous approaches. These quantitative results confirm that the proposed hybrid model provides faster response, lower startup delay, higher service capacity, better SLA compliance, and more efficient resource exploitation.

Several perspectives emerge from this work. Future developments may include deployment and validation on real cloud infrastructures such as Kubernetes, OpenFaaS, or AWS Lambda using real workload traces. Additional learning modules based on deep reinforcement learning or graph neural networks may further improve adaptive policies. The framework may also be extended to multi-cloud and edge-cloud ecosystems. Finally, integrating real-time cyber-defense capabilities for intrusion detection, attack propagation analysis, and autonomous mitigation constitutes a promising direction for next-generation intelligent cloud platforms.

Declarations

- **Conflict of interest:** The authors declare that they have no conflicts of interest.
- **Ethical approval:** All procedures performed in studies involving human participants met the ethical standards of the institutional committee.
- **Funding:** No funding was received to assist with the preparation of this manuscript.
- **Author contributions:** All authors contributed to the study conception and design.

References

[1] M. Hanaforoosh, M. A. Azgomi, and M. Ashtiani, "Reducing the Cost of Cold Start Time in Serverless Function Executions Using Granularity Trees," *Future Generation Computer Systems*, vol. 164, p. 107604, 2025. doi: 10.1016/j.future.2024.107604.

[2] A. H. Ebrahimi, A. Ebrahimi, M. Ghobaei Arani, and H. Saboohi, "Cold Start Latency Mitigation Mechanisms in Serverless Computing: Taxonomy, Review, and Future Directions," *Journal of Systems Architecture*, vol. 151, p. 103115, 2024. doi: 10.1016/j.sysarc.2024.103115.

[3] M. Golec, G. K. Walia, M. Kumar, F. Cuadrado, S. S. Gill, and S. Uhlig, "Cold Start Latency in Serverless Computing: A Systematic Review, Taxonomy, and Future Directions," *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–36, 2023. doi: 10.1145/3700875.

[4] S. Schirmer, L. Wang, H. P. Jones, and J. Doe, "The Night Shift: Understanding Performance Variability of Cloud Serverless Platforms," arXiv preprint arXiv:2304.07177, 2023.

[5] X. Liu, J. Wen, Z. Chen, D. Li, J. Chen, Y. Liu, H. Wang, and X. Jin, "FaaSLight: General Application-Level Cold-Start Latency Optimization for Function-as-a-Service in Serverless Computing," arXiv preprint arXiv:2207.08175, 2022.

[6] C. Marcelino and S. Nastic, "Truffle: Efficient Data Passing for Data-Intensive Serverless Workflows in the Edge-Cloud Continuum," arXiv preprint arXiv:2411.16451, 2024.

[7] S. S. M. Tariq, A. A. Zein, S. S. Vaidya, A. Khanolkar, Z. Song, and P. Roy, "Efficient Serverless Cold Start: Reducing Library Loading Overhead by Profile-Guided Optimization," arXiv preprint arXiv:2504.19283, 2025.

[8] S. Wang, Y. Liu, J. Zhang, and H. Chen, Timed Petri Nets for Modeling and Performance Evaluation of a Priority Queueing System, *Energies*, 2023. https://doi.org/10.3390/en16237690

[9] M. Golec, G. K. Walia, M. Kumar, F. Cuadrado, S. S. Gill, and S. Uhlig, Generalized stochastic Petri net-based performance analysis of dynamic systems, *Ad Hoc Networks*, Elsevier, 2024.