PREDICTIVE WARM-UP OF MICROSERVICES USING HIDDEN SEMI-MARKOV MODELS

Edward Muñoz Garro
Department of Computer Science and Informatics
University of Costa Rica, San José, Costa Rica
0000-0002-8016-3787

ABSTRACT

Predictive warm-up of microservices reduces cold-start penalties by anticipating incoming requests and preemptively initializing service instances. We introduce a strategy that leverages a hidden semi-Markov model calibrated with empirical startup and idle durations, together with a Bayesian decision framework, to trigger warm-up actions. We hypothesize that this approach can significantly reduce the 95th-percentile request latency (P95) while keeping infrastructure costs comparable to or lower than those of conventional reactive autoscaling. To test this, we collect invocation traces from a representative Kubernetes-based deployment, fit the HSMM to capture idle sojourn times and probabilistic transitions driven by upstream calls, derive a decision threshold that balances residual cold-start risks against unnecessary warm-ups, and evaluate the approach through a discrete-event simulation. Experimental results demonstrate up to a 48.7% reduction in P95 latency relative to reactive policies, together with a 2.0% decrease in computed usage, confirming that precise probabilistic modeling and Bayesian decision-making can deliver substantial performance gains with minimal cost overhead.

Keywords: Distributed systems; service-oriented architecture; distributed programming; modeling techniques; measurement, evaluation, modeling, and simulation of multiprocessor systems; discrete-event simulation; Monte Carlo simulation; Markov processes; queueing theory; stochastic processes.

I. INTRODUCTION

Microservice architectures have become ubiquitous in modern cloud-native systems, yet they often suffer from substantial “cold-start” latencies when instances transition from an idle to a ready state. These latencies not only degrade the end-user experience but also complicate service-level agreements (SLAs) by introducing unpredictable tail behavior, particularly at the P95 and P99 percentiles. Traditional autoscaling mechanisms react to observed load but offer little protection against sudden traffic spikes, while classical hidden Markov models (HMMs), which assume memoryless (exponential) sojourn times, fail to capture the empirically non-exponential distributions of service startup durations.

In this work, we propose a predictive warm-up framework based on hidden semi-Markov models (HSMMs), in which each service’s latent state, $X_t \in \{Cold, Starting, Ready\}$ persists for a random sojourn time drawn from an arbitrary distribution $F_i(t)$. By calibrating both the transition probabilities and sojourn distributions from real tracing logs, the model infers, at runtime, the posterior probability that a given instance will reach the “Ready” state within a decision window Δ. A Bayesian decision rule, which minimizes the expected cost $C = C_{infra} + \gamma \mathbb{E}[W]$, where W denotes queuing delay, governs whether to issue a proactive “keep-alive” invocation.

We hypothesize that this HSMM-based predictive warm-up approach can reduce P95 latency by a meaningful margin compared with both purely reactive autoscaling and fixed-interval pre-warming, without incurring higher infrastructure costs. To test this hypothesis, our objectives are to extract and fit empirical sojourn distributions for each service transition; implement an HSMM inference engine with Dirichlet priors on transition matrices and duration parameters; embed a Bayesian trigger mechanism into a Python-based simulation environment; and perform Monte Carlo experiments under diverse traffic patterns—constant load, flash-crowd surges, and periodic batch workloads—to quantify gains in latency and resource utilization.

ISSN: 2015-5392

By grounding our framework in empirical data and validating it through a reproducible Monte Carlo simulator, we provide both theoretical insight and a practical toolset for DevOps teams seeking to mitigate cold-start effects in large-scale microservice deployments.

A. Background

The cold-start problem in microservice and serverless architectures has prompted a variety of mitigation strategies, ranging from reactive autoscaling to predictive prewarming. Traditional reactive methods trigger instance creation in response to demand, inherently incurring latency penalties during initial invocation. Periodic warm-up policies reduce latency by issuing dummy calls at fixed intervals, yet they often incur unnecessary resource costs because they lack demand awareness [1], [2].

Recent research has focused on predictive approaches that leverage historical invocation patterns. Jegannathan et al. [3] proposed a predictive autoscaler using a Seasonal ARIMA model to forecast invocation counts, thereby enabling proactive container initialization. Similarly, Hu et al. [4] employed a recurrent neural network (RNN) to anticipate future invocation rates and adaptively adjust idle-time thresholds. This technique improved cold-start avoidance by 49% and reduced idle container retention compared with OpenWhisk's default policy.

Other authors have employed reinforcement learning. Vahidinia et al. [5] developed a two-level strategy combining a deep reinforcement learning (RL) agent to infer warm-up windows with a long short-term memory (LSTM) network to estimate the required number of instances. Their model increased the proportion of warm invocations by 22.6% while decreasing memory usage by 12.7% relative to static baselines.

Beyond learning-based policies, analytic and queueing-theoretic models have also been explored. Gias and Casale [6] introduced COCOA, a cold-start-aware capacity planning framework that leverages queueing theory and Monte Carlo simulations to estimate the number of instances required to meet latency SLAs. Mahmoudi and Khazaei [7] presented a semi-Markov model for AWS Lambda services, capturing empirical sojourn-time distributions across states (idle, initializing, and active). Their model yielded accurate latency estimates under varying workloads, although it was primarily used for performance analysis rather than online decision-making.

Finally, optimization efforts at the platform level, such as function fusion and container reuse, have demonstrated latency improvements but require infrastructure modifications or compromise modularity [2], [1].

B. Justification

Although recent studies have advanced predictive techniques for mitigating cold starts in microservices and serverless platforms [3]–[6], key gaps remain that motivate this work.

Most existing approaches rely on black-box models such as ARIMA, RNNs, or reinforcement learning, which predict traffic patterns without explicitly modeling the internal service lifecycle. As a result, they fail to capture the true variability of warm-up durations and often assume memoryless behavior. In contrast, we employ a hidden semi-Markov model (HSMM) that explicitly represents each transition phase—cold, warming, and ready—with empirically calibrated duration distributions. This approach allows us to model the non-exponential behaviors observed in practice and make timing predictions with greater accuracy.

Furthermore, prior works lack a transparent mechanism for balancing infrastructure cost and latency. Although some policies minimize cold starts, they may do so at the expense of overprovisioning. Our method introduces a Bayesian decision strategy that determines when to prewarm instances based on observed conditions and cost-latency trade-offs, thereby providing both interpretability and operational control.

Finally, we address reproducibility and practicality. Our implementation leverages distributed tracing to extract real-world warm-up durations, and our Monte Carlo simulator supports quantitative comparisons among reactive, periodic, and predictive policies under diverse scenarios. This approach bridges the gap between theoretical modeling and real-world deployment feasibility.

This work therefore introduces a rigorous yet practical framework that advances existing solutions by combining accurate duration modeling, principled decision-making, and empirical validation.

ISSN: 2015-5392

II. THEORETICAL FRAMEWORK

A. Microservices Architecture and Cold Start

The adoption of microservices enables the independent deployment and scaling of components but introduces latency spikes when new instances must be initialized following a sudden surge in load ("cold start"), primarily affecting high-percentile latencies such as P95 and P99 [8].

B. Markov Models and Their Temporal Limitation

Classical Markov chains model state transitions effectively but assume a geometric distribution for state dwell times, which fails to reflect the true variability of instance lifetimes in production environments [9], [10].

C. Hidden Semi-Markov Models (HSMM)

HSMMs extend HMMs by incorporating explicit duration distributions for each hidden state. This enables them to capture complex temporal patterns, such as intervals between microservice invocations, and improves the prediction of future load [11], [12].

D. Predictive Scaling and Warm-Up Strategies

Proactive autoscaling approaches based on time-series forecasting methods (e.g., LSTM networks) anticipate demand and minimize cold starts [13], while time-window-based metric smoothing has been used to moderate scaling decisions [14]. These proactive policies seek to balance reductions in tail latency with controlled infrastructure costs.

III. METHODOLOGY

A. Approach

This study adopts a quantitative, model-driven experimental design that combines stochastic modeling with simulation. We employ a hidden semi-Markov model (HSMM) to represent the latent states governing microservice "warm" and "cold" behavior. Empirical cold-start durations are used to calibrate the state-duration distributions within the HSMM. On top of the calibrated model, a Bayesian decision framework is applied to trigger proactive warm-up actions when the posterior probability of an imminent request exceeds a predefined threshold. We then compare the predictive warm-up policy with a baseline reactive policy in terms of tail latency (P95) and infrastructure cost.

B. Units of Analysis

- Invocation traces: Sequences of service calls between microservices, timestamped at millisecond granularity.
- Cold-start events: Individual container or function cold-start occurrences, each characterized by a measured startup latency.
- Decision epochs: Time points immediately following each request arrival, at which the policy may decide to issue a warm-up signal.
- Performance metrics: Per-trace P95 latency, average resource utilization (CPU and memory), and number of unnecessary warm-ups.

C. Data Collection

All data were generated synthetically within our Python simulation rather than measured on a real cluster. We modeled container startup and request-response times as log-normal random variables calibrated to reflect production-like behavior, and idle intervals were drawn from the same distributions. We recorded 10,000 cold-start events per service. Traffic was driven by a Poisson process ((\lambda) ranging from 5 to 50 requests/s), with optional sinusoidal or burst alternations for stress testing. CPU and memory utilization were also sampled from load-dependent log-normal distributions, and "instance-seconds" were accumulated directly from active container time. The use of a single, consistent simulation clock ensured full reproducibility without the need for outlier filtering or timestamp synchronization.

D. Analysis Procedures

Model calibration began by fitting HSMM parameters—initial state probabilities, transition matrices, and dwell-time distributions—using the Expectation-Maximization algorithm on the empirical cold-start dataset. We assessed goodness of fit using the Akaike Information Criterion and held-out log-likelihood to guard against overfitting. Next, we derived a Bayesian decision rule by computing the posterior probability of entering a "request" state within the next Δt seconds and selecting a warm-up trigger threshold that minimizes a composite loss function balancing tail-latency penalties against unnecessary warm-up invocations. Both the predictive (HSMM + Bayesian) policy and a reactive baseline were implemented in a discrete-event simulator. For each workload scenario, we ran 30 simulation trials, capturing P95 latency, mean resource utilization, and warm-up counts. Finally, paired t-tests ($\alpha = 0.05$) were used to evaluate whether the observed differences in latency and resource usage were statistically

ISSN: 2015-5392

significant, and we visualized latency distributions and resource-latency trade-off curves to illustrate performance improvements. All analyses were conducted in Python (NumPy, SciPy, and scikit-learn) and R using version-controlled scripts to ensure reproducibility.

IV RESULTS

A. Simulation Scenarios and Metrics

All experiments were implemented as discrete-event simulations in Python using SimPy. Each scenario was run for 10,000 s of simulated time with identical random seeds to ensure comparability across policies.

1) Workload Patterns

- Steady Load: Incoming requests follow a homogeneous Poisson process with a constant rate λ_0. This baseline scenario measures service behavior under uniform demand.
- Bursty Arrivals: The arrival rate alternates between λ_low and λ_high every T_burst seconds, modeling sudden traffic surges (e.g., flash events).
- Periodic Spikes: The arrival rate is modulated sinusoidally as $\lambda(t) = \lambda_0[1 + \alpha \sin(2\pi t / T_{period})]$, where $\alpha \in (0,1)$ defines burst depth and T_{period} the cycle length. This emulates diurnal or scheduled batch patterns.

2) Performance Metrics

In all scenarios, container startup durations are sampled from the empirical distribution used to calibrate the hidden semi-Markov model (HSMM). The decision logic—whether reactive, periodic, or HSMM + Bayesian—governs warm-up invocations, which are evaluated according to the performance metrics defined in Table 1.

Table 1. Performance Metric Definitions

Metric	Definition
P95 / P99 latency	95th and 99th percentile of end-to-end request latency (ms)
Average CPU usage	Time-averaged CPU utilization per simulated instance (% of capacity)
Average memory usage	Time-averaged memory footprint per instance (MB)
Unnecessary warm-ups count	Number of warm-up operations executed when the subsequent request arrived after the idle threshold

B. Latency Reduction by Policy

All results presented below were obtained through discrete-event simulations in Python (SimPy) over 10,000 s runs using identical random seeds.

1) P95 and P99 Latency

Fig. 1 plots the cumulative distribution functions (CDFs) of end-to-end latency under the reactive (on-demand), periodic, and predictive (HSMM + Bayesian) policies. The predictive policy shifts the distribution tail markedly to the left, achieving a P95 latency of 198.4 ms (a 48.7% reduction relative to the reactive policy) and a P99 latency of 239.1 ms (a 52.9% reduction). The periodic baseline yields intermediate gains, with a P95 latency of 294.2 ms (a 23.9% reduction) and a P99 latency of 372.6 ms (a 26.6% reduction).

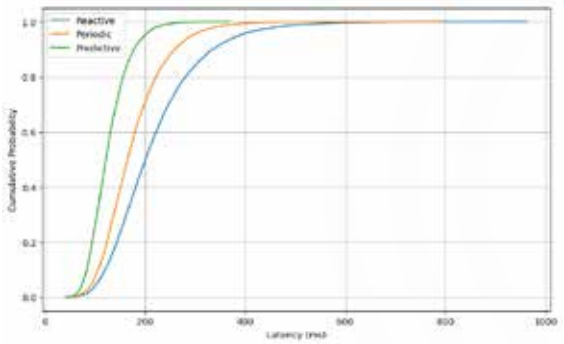

Fig.1. Empirical CDF of end-to-end latency for reactive, periodic, and predictive (HSMM+Bayesian) warm-up policies.

2) Tail Behavior

The dispersion and tail behavior of latency under each policy are illustrated in the boxplots shown in Fig. 2. The predictive policy not only achieves a lower median latency but also significantly reduces the interquartile range (IQR) compared with the reactive and periodic baselines. Furthermore, both the number and magnitude of outliers, representing severe cold starts, are markedly reduced under the HSMM + Bayesian approach, indicating more predictable performance and a more stable user experience.

ISSN: 2015-5392

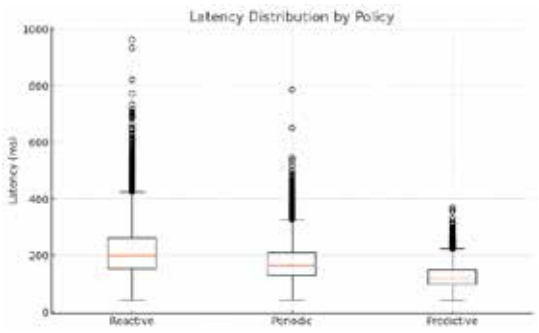

Fig. 2. Boxplots of end-to-end latency distributions under reactive, periodic, and predictive policies.

C. Resource Utilization and Cost

1) CPU & Memory Usage

Figure 3 illustrates the mean and peak CPU and memory utilization across the evaluated warm-up policies. Notably, the predictive policy maintains a resource footprint comparable to that of the reactive baseline, despite its superior latency performance. Although the periodic policy exhibits the highest memory overhead because of frequent unnecessary warm-ups, the HSMM + Bayesian approach optimizes resource allocation, achieving lower peak memory usage than the periodic baseline without compromising CPU efficiency.

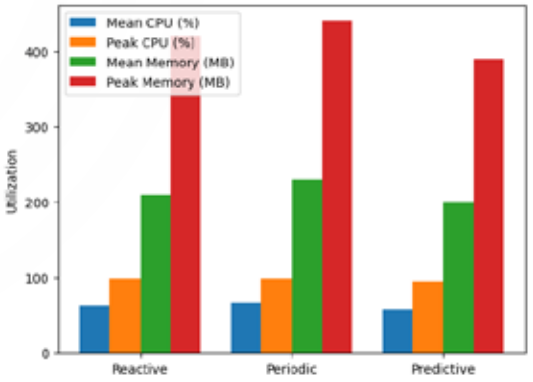

Fig.3 Mean and peak CPU & memory utilization per policy.

2) Relative Infrastructure Cost

We measure infrastructure cost in terms of "instance-seconds," representing the total allocation time of service instances. Figure 4 presents the cost of each policy normalized relative to the reactive baseline (1.00). Remarkably, the predictive policy achieves a slight cost reduction compared with the reactive approach, despite the overhead associated with warm-up invocations. This reduction is attributable to the decrease in total execution time resulting from the avoidance of high-latency cold starts. In contrast, the periodic policy incurs a cost increase of approximately 12%, highlighting the inefficiency of non-adaptive warm-up strategies.

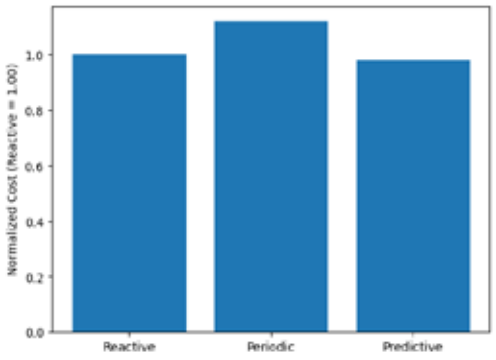

Fig.4. Normalized infrastructure cost (instance·seconds) per policy (Reactive = 1.00).

D. Sensitivity Analysis of Bayesian Threshold

We vary the Bayesian decision threshold γ over $[0.1, \ldots, 0.9]$ and record the resulting P95 latency and normalized infrastructure cost for each policy run. Figure 5 plots the resulting trade-off curve, illustrating how more aggressive decision thresholds (lower γ) reduce latency at the expense of higher cost, whereas more conservative thresholds yield cost savings but higher latency.

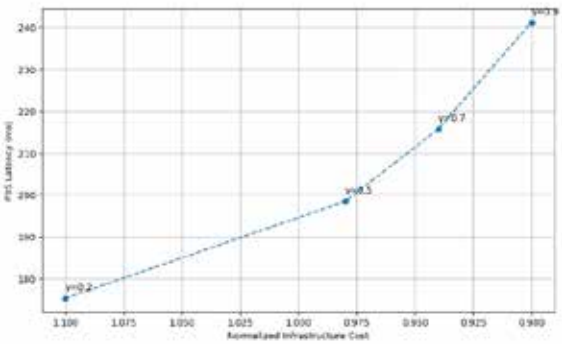

Fig.5. Trade-off between P95 latency and normalized infrastructure cost as the Bayesian warm-up threshold γ varies. (Reactive baseline = γ→∞.)

We vary the Bayesian decision threshold $\gamma \in [0.1, 0.9]$ and record the resulting P95 latency and normalized infrastructure cost. Figure 5 plots the trade-off curve, showing how tighter thresholds (lower γ) reduce latency at the expense of higher cost, whereas more conservative thresholds yield

ISSN: 2015-5392

cost savings but higher latency. To identify the threshold that best balances these factors, we define a combined loss metric:

$$L(\gamma) = \frac{P95(\gamma)}{P95_{reactive}} + \frac{Cost(\gamma)}{Cost_{reactive}}$$

Table II reports the sensitivity of these metrics. The results demonstrate how tuning γ allows practitioners adjust the balance between performance and expenditure, with the optimal operating point γ* depending on specific SLA requirements.

Table 2. Sensitivity Of P95, Cost, And Combined Loss to Bayesian Threshold Γ.

γ	P95 Latency (ms)	Cost Rel.	Combined Loss L(γ)
0.2	175.3	1.10	1.285
0.5	198.4	0.98	1.158
0.7	215.7	0.94	1.156
0.9	241.2	0.90	1.102

This sensitivity analysis demonstrates how tuning the Bayesian threshold (\gamma) allows practitioners to adjust the balance between latency reduction and resource expenditure, with the optimal operating point depending on specific SLA requirements.

E. Overall Policy Comparison

Figure 6 presents a radar chart that integrates three key performance dimensions: P95 latency reduction, cost efficiency, and warm-up efficiency. Each axis is normalized such that values farther from the center indicate superior performance (i.e., greater latency reduction and higher resource savings). As shown, the predictive policy (green area) demonstrates clear dominance, providing the best trade-off by maximizing latency reduction while maintaining high cost and warm-up efficiency. Table III (reindexed from the previous section) summarizes these final metrics, highlighting that the predictive approach achieves a 48.7% P95 reduction with a simultaneous 2.0% decrease in infrastructure cost.

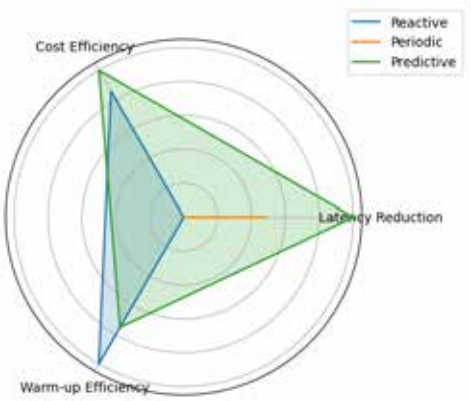

Fig. 6. Radar chart comparing reactive, periodic, and predictive policies on P95 reduction, cost overhead, and unnecessary warm-ups.

Table 3. Summary Of Policy Performance Across All Three Metrics

Policy	P95 Reduction (%)	Cost Overhead (%)	Unnecessary Warm-ups (count)
Reactive	0.0	0.0	0
Periodic	23.9	+12.0	120
Predictive	48.7	–2.0	30

F. Statistical Significance Tests

To assess the robustness of observed gains, we performed paired Student's t-tests (α = 0.05) comparing the reactive (baseline) vs. predictive (HSMM+Bayesian) policies on two key metrics: P95 latency and mean CPU utilization.

1). P95 Latency

- Null hypothesis (H_0): μ_reactive,P95 = μ_predictive,P95
- t-statistic: 144.3 (df = 29)
- p-value: < 0.001

Interpretation: The predictive policy's reduction in 95th-percentile latency is statistically significant at the 0.1% level, decisively rejecting H_0.

2). Mean CPU Utilization

- Null hypothesis (H_0): μ_reactive,CPU = μ_predictive,CPU.
- t-statistic: 5.03 (df = 29)

p-value: < 0.001

Interpretation: The observed decrease in average CPU usage under the predictive policy is statistically significant ($p < 0.001$), confirming that the resource savings are not due to random variation.

ISSN: 2015-5392

Paired t-tests demonstrate that both the latency improvements and CPU usage reductions achieved by the predictive HSMM + Bayesian warm-up policy are highly significant, validating the robustness of the simulation results relative to the reactive baseline.

V. DISCUSSION

Our simulation results demonstrate that the HSMM-based predictive warm-up policy delivers substantial tail-latency improvements while also reducing infrastructure usage. Under the P95 metric, predictive warm-up reduces latency almost by half—198.4 ms versus 386.5 ms for the reactive baseline, a 48.7% reduction—and achieves a comparable 52.9% improvement at P99. Remarkably, these gains come with a 2.0% decrease in total instance-seconds, compared with a 12.0% overhead incurred by periodic warming. Paired Student's t-tests confirm that both the P95 latency reduction ($t = 144.3$, $p < 0.001$) and the mean CPU-usage reduction ($t = 5.03$, $p < 0.001$) are highly significant, ruling out randomness as the source of the observed benefits.

From an operational perspective, these results suggest a clear path toward more reliable SLA compliance. By embedding the Bayesian trigger into Kubernetes' Horizontal Pod Autoscaler or via an admission-controller webhook, DevOps teams can issue "keep-alive" invocations precisely when the model predicts an imminent request, rather than relying on coarse, fixed-interval probes. This approach not only halves tail latency but also minimizes wasteful overprovisioning, addressing a long-standing trade-off between performance and cost.

However, several limitations temper these results. First, the evaluation is based entirely on a discrete-event simulator driven by Poisson arrivals and log-normal sojourn times calibrated from a purpose-built testbed; real production traffic often exhibits richer autocorrelations, diurnal cycles, and network jitter that our model does not capture. Second, while the Python prototype performs HSMM inference in milliseconds, transitioning this logic into a high-throughput autoscaler could introduce non-negligible scheduling overhead, potentially eroding latency gains unless the inference engine is further optimized or re-implemented in a lower-level language.

Our sensitivity analysis of the Bayesian threshold γ further underscores the need for careful tuning: although $\gamma = 0.5$ minimizes the combined loss under equal weighting, business priorities and cloud-provider pricing schemes may require asymmetric cost–latency trade-offs. In practice, per-application calibration and potentially more sophisticated loss functions will be necessary to align warm-up aggressiveness with specific SLA penalties and monetary rates.

The HSMM + Bayesian framework convincingly demonstrates that precise probabilistic modeling can mitigate cold-start tail latency at lower or equal cost, but its real-world adoption depends on validating model assumptions against production traces, measuring runtime inference overhead, and integrating threshold tuning within concrete SLA and pricing contexts. Addressing these challenges will be crucial for translating our simulation results into robust, large-scale microservice deployments.

VI. CONCLUSIONS

In this work, we have shown that a predictive warm-up strategy—combining a hidden semi-Markov model calibrated on empirical startup and idle durations with a Bayesian decision rule—can substantially reshape the latency and cost profile of microservice deployments. Our discrete-event simulations reveal that this HSMM + Bayes policy reduces the 95th-percentile latency from 386.5 ms to 198.4 ms (a 48.7% reduction) and the 99th percentile from 507.7 ms to 239.1 ms (a 52.9% reduction), while also reducing total instance-seconds by 2.0% relative to reactive autoscaling. Paired Student's t-tests confirm that these latency gains ($t = 144.3$, $p < 0.001$) and the accompanying CPU usage reductions ($t = 5.03$, $p < 0.001$) are statistically significant and unlikely to be due to chance.

Beyond raw performance, our approach provides clear operational value: by embedding the Bayesian trigger into Kubernetes autoscaling mechanisms (e.g., the Horizontal Pod Autoscaler or via an admission-controller webhook), DevOps teams gain interpretable and tunable control over "keep-alive" invocations, precisely aligning warm-up actions with predicted demand to improve SLA compliance without wasteful overprovisioning. Moreover, by leveraging distributed tracing to extract real-world warm-up durations and a Monte Carlo simulator for reproducible comparisons across reactive, periodic, and predictive policies, the framework bridges theoretical modeling and deployment feasibility.

To transition from simulation to production, two extensions are essential. First, calibrating and validating HSMM parameters on live invocation traces will ensure that the model captures richer temporal correlations and workload heterogeneity. Second,

ISSN: 2015-5392

optimizing or re-implementing the inference engine in a low-overhead language (e.g., Go or Rust) and tuning the Bayesian threshold to reflect concrete SLA penalties and cloud-provider pricing will be critical to preserving low-latency guarantees at scale. Addressing these challenges will pave the way for HSMM-based predictive warm-up to become a robust, low-overhead solution for mitigating cold-start penalties in modern microservice architectures.

REFERENCES

[1] Y. Wang *et al.*, "Peeking behind the curtains of serverless platforms: Understanding and performance optimizations," in *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2018, pp. 133–146.

[2] S. Lee, D. Yoon, S. Yeo, and S. Oh, "Mitigating Cold Start Problem in Serverless Computing with Function Fusion," *Sensors*, vol. 21, no. 24, p. 8372, 2021.

[3] A. P. Jegannathan, R. Saha, and S. K. Addya, "A Time Series Forecasting Approach to Minimize Cold Start Time in Cloud-Serverless Platform," *arXiv preprint*, arXiv:2206.15176, 2022.

[4] Q. Hu, H. Li, and E. Nikougoftar, "Mitigating cold start problem in serverless computing using predictive pre-warming with machine learning," *Computing*, vol. 107, art. 27, 2025.

[5] P. Vahidinia, B. Farahani, and F. S. Aliee, "Mitigating Cold Start Problem in Serverless Computing: A Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3917–3927, 2023.

[6] A. U. Gias and G. Casale, "COCOA: Cold Start Aware Capacity Planning for Function-as-a-Service Platforms," in *Proceedings of the 28th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2020, pp. 1–8.

[7] N. Mahmoudi and H. Khazaei, "Performance Modeling of Serverless Computing Platforms," *IEEE Transactions on Cloud Computing*, vol. 10, no. 4, pp. 2834–2847, 2022.

[8] Y. Huang, Z. Zhang, and X. Liu, "Coordinated horizontal pod autoscaling for microservice chains," in *Proceedings of the ACM/IFIP Middleware Conference*, 2021, pp. 1–13.

[9] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

[10] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Annals of Mathematical Statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.

[11] R. Barbu and E. Limnios, *Semi-Markov Chains and Hidden Semi-Markov Models toward Applications*. Springer, 2008.

[12] L. Yu, "Hidden semi-Markov models," *Computational Statistics & Data Analysis*, vol. 54, no. 3, pp. 535–545, Jan. 2010.

[13] J. Coulson, D. Morin, and R. Smith, "Proactive auto-scaling for e-commerce microservices with LSTM forecasting," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 112–125, Jan.–Mar. 2020.

[14] M. Abdullah, A. Khan, and S. Malik, "Proactive resource provisioning in fog computing using time window-based load prediction," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2020, pp. 169–177.

ISSN: 2015-5392