

# A REVIEW OF COST-LATENCY TRADE-OFF MODELING FOR SERVERLESS ML INFERENCE USING PREDICTIVE WORKLOAD FORECASTING

Vivek Shukla<sup>1</sup>, Dr. J.B. Singh<sup>2</sup>, Mr. Rajesh Kumar Sharma<sup>3</sup>

<sup>1</sup>Master of Technology, Computer Science and Engineering, Sagar Institute of Technology and Management, Barabanki, India

<sup>2</sup>Professor, Department of Computer Science and Engineering, Sagar Institute of Technology and Management, Barabanki, India

<sup>3</sup>Assistant Professor, Department of Computer Science and Engineering, Sagar Institute of Technology and Management, Barabanki, India

\*\*\*

**Abstract** -Serverless computing has emerged as a dominant paradigm for deploying scalable Machine Learning (ML) inference workloads due to its elasticity, fine-grained billing, and operational simplicity. However, ML inference in Function-as-a-Service (FaaS) environments introduces a fundamental trade-off between operational cost and service latency. While aggressive resource provisioning reduces cold start delays and response time, it increases execution cost; conversely, cost minimization strategies often degrade Quality of Service (QoS). This review systematically analyzes existing literature on cost-latency trade-off modeling for serverless ML inference, with particular emphasis on the integration of predictive workload forecasting mechanisms. The paper categorizes prior studies into analytical modeling, simulation-based approaches, multi-objective optimization frameworks, and learning-driven adaptive techniques. It further examines forecasting methodologies—including statistical time-series models, classical machine learning predictors, and deep learning architectures—and evaluates their impact on proactive resource allocation. A comparative synthesis of evaluation metrics, benchmark datasets, and modeling assumptions is provided to identify methodological trends and research gaps. The review highlights limitations in handling bursty workloads, cold start variability, and real-time adaptive scaling. Finally, it outlines open research challenges and future directions for robust, cost-efficient, and latency-aware serverless ML inference systems.

**Key Words:** Serverless Computing; ML Inference; Cost-Latency Trade-Off; Predictive Workload Forecasting; Performance Modeling; Auto-Scaling Optimization

## 1. INTRODUCTION

### 1.1 Background

#### 1.1.1 Rise of Serverless Computing

Serverless computing, particularly the Function-as-a-Service (FaaS) model, has transformed cloud-native application deployment by abstracting infrastructure management and enabling fine-grained, event-driven execution. Platforms such as Amazon Web Services (AWS Lambda), Microsoft Azure (Azure Functions), and Google Cloud (Google Cloud

Functions) provide automatic scaling and pay-per-use billing, which reduce operational overhead and improve elasticity. Unlike traditional virtual machine or container-based deployments, serverless environments allocate resources dynamically per invocation, charging users based on execution duration and memory allocation. This paradigm has gained widespread adoption due to its economic efficiency and operational agility (Baldini et al., 2017; Jonas et al., 2019). Nevertheless, serverless systems introduce performance uncertainties, particularly under bursty and latency-sensitive workloads.

#### 1.1.2 Importance of Machine Learning (ML) Inference in Production Systems

Machine Learning (ML) inference constitutes the real-time deployment phase of trained models in production environments, supporting applications such as recommendation systems, fraud detection, intelligent assistants, and computer vision services. As ML-driven services increasingly operate under stringent Service Level Agreements (SLAs), low latency and high availability become critical requirements (Zaharia et al., 2018). Serverless platforms are attractive for ML inference because they scale automatically with demand, eliminating idle resource costs during low-traffic periods. However, inference workloads are often compute-intensive and memory-sensitive, making their execution behavior highly dependent on runtime configuration and resource provisioning strategies (Wang et al., 2018).

#### 1.1.3 Performance Challenges: Cost vs. Latency

A fundamental challenge in serverless ML inference is the cost-latency trade-off. Cold starts—caused by container initialization delays—can significantly increase response time, particularly for sporadic workloads (Wang et al., 2018). Mitigating cold starts through provisioned concurrency or over-provisioning reduces latency but increases operational cost. Conversely, minimizing allocated memory or compute resources reduces billing charges but may lead to execution slowdown and SLA violations. This inherent tension motivates the development of formal cost-latency modeling

frameworks capable of balancing economic and performance objectives (Jonas et al., 2019).

## 1.2 Problem Definition

### 1.2.1 Cost–Latency Trade-Off?

The cost–latency trade-off refers to the conflicting relationship between minimizing monetary expenditure and achieving low response time in cloud-based execution environments. In serverless systems, cost is typically determined by invocation count, execution duration, and memory allocation, while latency encompasses cold start delay, queueing time, and execution time. Analytical models based on queueing theory and performance profiling have demonstrated that reducing latency often requires pre-warmed instances or increased resource allocation, thereby increasing cost (Baldini et al., 2017). Effective trade-off modeling seeks Pareto-optimal configurations that satisfy QoS constraints without excessive spending.

### 1.2.2 Role of Workload Variability and Cloud Pricing Models

Workload variability significantly complicates cost–latency optimization. Serverless workloads often exhibit diurnal patterns, bursty spikes, and unpredictable demand fluctuations. Reactive auto-scaling mechanisms may lag behind sudden surges, causing latency degradation. Furthermore, cloud pricing models—based on per-request billing and memory-time combinations—introduce non-linear cost behavior. For instance, allocating more memory may reduce execution time sufficiently to offset increased per-millisecond charges, resulting in lower total cost under certain conditions. Therefore, modeling approaches must incorporate workload stochasticity and pricing granularity to produce realistic optimization outcomes (Shahrad et al., 2020).

### 1.2.3 Predictive Workload Forecasting Matters

Predictive workload forecasting enables proactive resource provisioning, thereby reducing cold start frequency and queueing delays. Time-series and machine learning-based predictors can estimate short-term request rates, allowing systems to pre-scale resources before demand surges occur. Prior studies demonstrate that forecasting-driven scaling policies outperform purely reactive mechanisms in latency-sensitive applications (Islam et al., 2012). However, forecasting accuracy directly influences optimization effectiveness; inaccurate predictions may lead to over-provisioning or SLA violations. Consequently, integrating forecasting models with cost–latency optimization frameworks is central to achieving efficient serverless ML inference.

## 1.3 Objectives of the Review

### 1.3.1 Synthesize Existing Approaches

This review systematically synthesizes literature on serverless performance modeling, workload forecasting, and multi-objective optimization. Rather than proposing a new algorithm, the objective is to consolidate fragmented research contributions into a unified conceptual framework. The survey examines analytical, simulation-based, and learning-driven modeling techniques across diverse application scenarios.

### 1.3.2 Compare Workload Prediction Methods

A key objective is to comparatively evaluate statistical time-series models, classical machine learning regressors, and deep learning architectures used for workload forecasting. Differences in scalability, interpretability, computational overhead, and prediction accuracy are critically analyzed to assess their suitability for real-time inference systems.

### 1.3.3 Evaluate Trade-Off Modeling Techniques

The review further examines cost–latency trade-off modeling strategies, including queueing-based formulations, multi-objective optimization methods, and reinforcement learning-based adaptive policies. Their strengths, assumptions, and empirical validation practices are contrasted to identify methodological trends.

### 1.3.4 Identify Open Problems and Future Research Directions

By analyzing current limitations—such as inadequate modeling of cold start variability, limited benchmark standardization, and insufficient integration of forecasting uncertainty—the review highlights promising research avenues for improving robustness and scalability.

## 1.4 Scope and Contribution

### 1.4.1 Inclusion and Exclusion Criteria

The review focuses on peer-reviewed journal articles and conference proceedings addressing serverless computing, ML inference performance optimization, workload forecasting, and cost modeling. Studies unrelated to serverless environments or purely training-phase ML optimization are excluded. Emphasis is placed on works presenting formal modeling, empirical evaluation, or optimization frameworks.

### 1.4.2 Time Frame of Surveyed Works

The survey primarily covers literature published between 2016 and 2025, corresponding to the rapid adoption phase of serverless platforms and the maturation of cloud-native ML inference frameworks.

### 1.4.3 Primary Contributions of the Review

The principal contributions of this review are threefold: (i) a structured taxonomy of forecasting and trade-off modeling approaches, (ii) a comparative synthesis of evaluation methodologies and datasets, and (iii) identification of research gaps concerning proactive scaling, uncertainty modeling, and cost-aware adaptive optimization. By integrating perspectives from cloud performance engineering and ML workload management, this review provides a comprehensive foundation for future advancements in cost-efficient, latency-aware serverless ML inference systems.

## 2. SERVERLESS ARCHITECTURE AND ML INFERENCE FUNDAMENTALS

This section establishes the conceptual and architectural foundations necessary to understand cost-latency trade-off modeling in serverless ML inference environments.

### 2.1 Serverless Computing: Definitions and Characteristics

#### 2.1.1 Function-as-a-Service (FaaS)

Serverless computing is a cloud execution paradigm in which infrastructure management is abstracted from developers, enabling event-driven execution of stateless functions. The dominant operational model is Function-as-a-Service (FaaS), where applications are decomposed into discrete functions triggered by events such as HTTP requests or message queue updates. Major cloud providers including Amazon Web Services (AWS Lambda), Microsoft Azure (Azure Functions), and Google Cloud (Google Cloud Functions) implement FaaS with automatic resource provisioning and fine-grained billing. FaaS platforms dynamically allocate compute containers per invocation, offering elasticity without explicit capacity planning. Prior studies emphasize that this model reduces operational complexity but introduces performance variability due to runtime isolation and container lifecycle management (Baldini et al., 2017; Jonas et al., 2019).

#### 2.1.2 Auto-Scaling and Pricing Models

A defining characteristic of serverless platforms is automatic scaling based on incoming request volume. Scaling decisions are typically reactive, triggered by concurrency demand rather than predictive provisioning. Billing follows a pay-per-use structure, calculated as a function of invocation count, allocated memory size, and execution duration measured in milliseconds. This fine-grained pricing structure differentiates serverless computing from virtual machine or container-based models, where users pay for reserved capacity regardless of utilization (Shahrad et al., 2020). The economic efficiency of serverless systems is therefore highly sensitive to workload patterns and execution configuration.

### 2.1.3 Cold Start Phenomenon and Scaling Delays

Cold starts occur when a function is invoked after a period of inactivity and the platform must initialize a new execution environment. This initialization includes container creation, runtime loading, and dependency setup, leading to additional latency before execution begins. Empirical evaluations demonstrate that cold start overhead varies depending on runtime language, memory allocation, and provider-specific orchestration mechanisms (Wang et al., 2018). Scaling delays may also arise when rapid workload surges exceed available warm instances, thereby increasing tail latency. These phenomena are central to performance modeling in serverless ML inference.

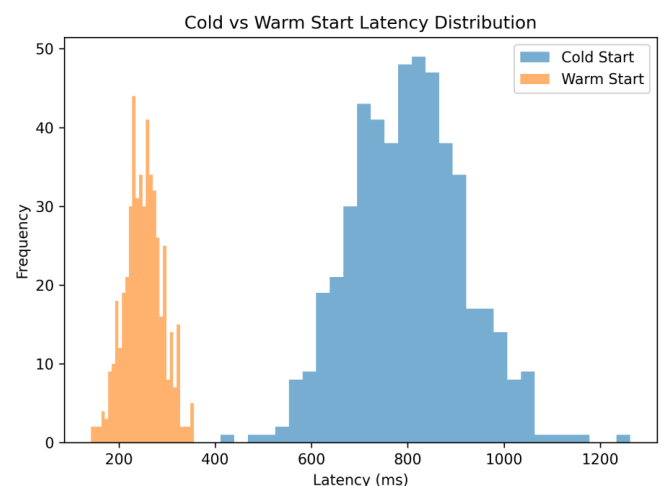


Figure-1: Cold vs Warm Start Latency Distribution

## 2.2 ML Inference Workloads

### 2.2.1 Types of Inference Workloads (Batch vs. Real-Time)

Machine Learning inference workloads can be broadly categorized into batch inference and real-time (online) inference. Batch inference processes large datasets asynchronously and is typically less sensitive to latency constraints. In contrast, real-time inference supports interactive applications such as recommendation engines, fraud detection systems, and conversational AI, where response times must satisfy strict SLA requirements. Real-time inference workloads are more suitable for FaaS deployment due to their event-driven nature, though they are more vulnerable to cold start delays and scaling inefficiencies (Crankshaw et al., 2017).

### 2.2.2 Resource and Latency Requirements

Inference workloads exhibit heterogeneous resource demands depending on model complexity, input size, and framework dependencies. Deep neural networks may require substantial memory and CPU allocation, while

lightweight models operate within minimal configurations. In serverless contexts, resource allocation directly influences execution time due to proportional CPU scaling with memory size. Consequently, selecting appropriate memory configurations becomes a joint performance–cost optimization problem. Studies indicate that under-provisioning increases execution duration and queuing delays, whereas over-provisioning inflates cost without proportional latency gains (Jonas et al., 2019).

### 2.2.3 Quality of Service (QoS) Expectations

Quality of Service in ML inference is primarily measured through response time percentiles (e.g., p95 or p99 latency), throughput, and availability. For production-grade systems, SLA compliance requires maintaining latency within predefined thresholds under fluctuating workloads. Serverless architectures complicate QoS assurance due to dynamic scaling and shared infrastructure. Therefore, workload-aware provisioning and proactive scaling strategies are essential for sustaining stable performance in latency-sensitive inference services (Herbst et al., 2013).

## 3. PREDICTIVE WORKLOAD FORECASTING

Predictive workload forecasting constitutes a foundational component of proactive resource management in serverless ML inference systems. Unlike reactive auto-scaling mechanisms, forecasting-driven strategies anticipate demand fluctuations and provision resources accordingly. This section presents a structured review of forecasting methodologies, evaluation practices, and open challenges.

### 3.1 Importance of Forecasting in Serverless Systems

#### 3.1.1 Forecasting for Autoscaling and Resource Provisioning

Serverless platforms scale functions dynamically based on incoming request rates; however, reactive scaling introduces latency penalties during abrupt workload surges. Predictive workload forecasting mitigates this limitation by estimating short-term arrival rates and triggering pre-warming or pre-scaling policies before demand peaks occur. Empirical studies on cloud elasticity demonstrate that prediction-based provisioning reduces SLA violations compared to threshold-based scaling (Islam et al., 2012). In latency-sensitive ML inference workloads, such proactive scaling is particularly critical because cold start delays and queuing effects disproportionately affect tail latency percentiles.

#### 3.1.2 Impact on Cost and Latency Optimization

Forecasting accuracy directly influences cost–latency trade-off modeling. Overestimation of demand results in unnecessary instance pre-allocation and higher operational cost, whereas underestimation leads to cold starts and

increased queuing delay. Research on serverless workload characterization shows that traffic patterns often exhibit diurnal cycles combined with bursty spikes, requiring forecasting models capable of capturing both seasonality and abrupt variations (Shahrad et al., 2020). Consequently, predictive modeling is not merely an auxiliary component but an integral input to multi-objective optimization frameworks for serverless ML inference.

### 3.2 Categories of Workload Forecasting Techniques

#### 3.2.1 Time Series-Based Methods

AutoRegressive Integrated Moving Average (ARIMA) models capture temporal autocorrelation and trend components in sequential data. Seasonal ARIMA (SARIMA) extends this framework by modeling periodic fluctuations, which are common in cloud traffic traces. These models have been applied to cloud workload prediction with moderate success in capturing structured temporal patterns (Box et al., 2015). However, their linear assumptions limit performance under highly non-linear or irregular workloads. Exponential smoothing techniques, including Holt–Winters methods, provide lightweight forecasting mechanisms that emphasize recent observations. They are computationally efficient and suitable for short-term prediction in stable workloads. Nevertheless, their responsiveness to sudden bursts is constrained, reducing effectiveness in highly dynamic serverless environments. Time-series methods offer transparency, fast training, and minimal parameter tuning. However, they struggle with complex non-linear relationships and multi-dimensional contextual features such as concurrency limits or application-level events.

#### 3.2.2 Machine Learning-Based Predictors

Linear and polynomial regression techniques incorporate additional explanatory variables such as time-of-day indicators or user activity metrics. While interpretable, their predictive power diminishes when workload patterns exhibit high stochasticity. Tree-based models, including random forests, improve predictive robustness by aggregating multiple decision trees. They capture non-linear relationships and interactions without requiring strict distributional assumptions. Comparative studies indicate that ensemble methods outperform linear baselines in cloud workload forecasting scenarios (Chen et al., 2018). SVR applies kernel-based learning to approximate complex functional relationships while controlling overfitting through margin maximization. It performs well in moderate-sized datasets but may incur scalability limitations under large-scale cloud trace analysis.

### 3.3 Evaluation Metrics and Datasets

#### 3.3.1 Forecast Accuracy Metrics

Forecasting performance is typically evaluated using statistical error measures such as Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Mean Absolute Percentage Error (MAPE). MAE provides robustness to outliers, RMSE penalizes large deviations, and MAPE enables scale-independent comparison across workloads. Selection of evaluation metrics significantly influences comparative interpretation of model performance.

#### 3.3.2 Publicly Available Datasets

Benchmarking often relies on publicly released cloud traces, including the Google cluster workload dataset and Azure function invocation traces. These datasets capture real-world traffic variability and resource utilization patterns. However, trace heterogeneity and limited labeling of contextual variables complicate standardized evaluation across studies.

### 3.4 Comparative Analysis

A comparative synthesis of forecasting approaches reveals trade-offs across multiple dimensions: prediction accuracy, computational complexity, scalability, interpretability, and adaptability to bursty workloads. Statistical models offer transparency and low overhead but limited non-linear modeling capability. Machine learning methods improve accuracy under moderate complexity, while deep learning architectures achieve superior performance at the expense of training cost and operational overhead.

In review articles, such comparison is typically summarized in tabulated form, detailing: (i) model type, (ii) dataset used, (iii) evaluation metrics, (iv) workload characteristics, and (v) reported performance outcomes. This structured comparison facilitates identification of methodological convergence and divergence across studies.

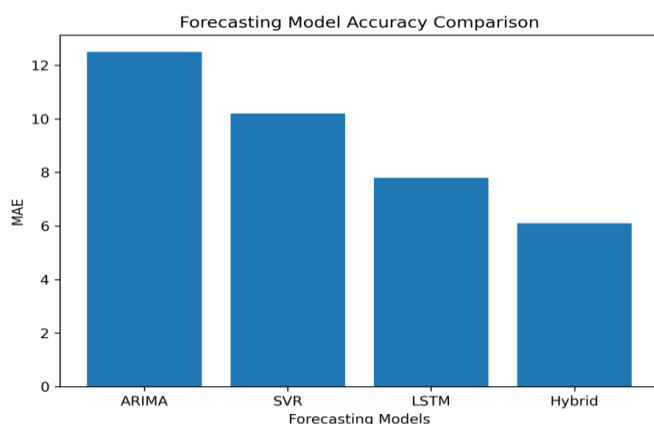


Figure-2: Forecasting Model Accuracy Comparison

### 3.5 Gaps and Challenges

#### 3.5.1 Forecasting Under Bursty Workloads

Serverless workloads frequently exhibit flash crowds and irregular spikes. Many forecasting models, particularly linear time-series techniques, struggle to capture sudden demand surges. Hybrid approaches integrating anomaly detection with forecasting remain underexplored.

#### 3.5.2 Cold Start Effects in Predictions

Most forecasting studies focus solely on request rate prediction without explicitly modeling cold start probability or container reuse dynamics. Incorporating platform-level runtime behavior into forecasting frameworks is essential for accurate latency-aware scaling.

#### 3.5.3 Transferability Across Workloads

Model generalizability across applications and cloud providers remains limited. Forecasting models trained on one dataset often underperform when applied to different workload distributions. Transfer learning and meta-learning techniques present promising directions for improving cross-domain adaptability.

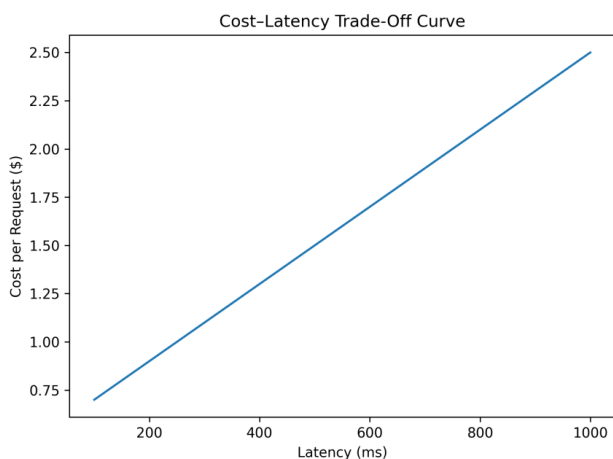
## 4. COST-LATENCY TRADE-OFF MODELING

Cost-latency trade-off modeling constitutes the central analytical dimension of serverless ML inference optimization. This section critically reviews modeling paradigms used to balance economic efficiency and performance guarantees in Function-as-a-Service environments.

### 4.1 Overview of Trade-Off Modeling

#### 4.1.1 Cost and Latency Are Conflicting Objectives

In serverless platforms such as Amazon Web Services and Microsoft Azure, cost is primarily determined by invocation count, execution duration, and allocated memory. Increasing memory allocation typically improves CPU share and reduces execution time, thereby lowering latency. However, higher memory configurations increase per-millisecond billing rates. Similarly, enabling provisioned concurrency mitigates cold starts but introduces fixed baseline costs. These structural characteristics create an inherent tension between minimizing operational expenditure and satisfying Service Level Agreement (SLA) latency constraints (Jonas et al., 2019). Consequently, trade-off modeling seeks to identify configurations that achieve acceptable latency at minimal cost.



**Figure-3: Cost-Latency Trade-Off Curve**

#### 4.1.2 Performance Modeling Goals

The primary goals of cost-latency modeling include: (i) estimating response time distributions under stochastic arrival rates, (ii) quantifying cost implications of resource configurations, and (iii) identifying Pareto-efficient deployment strategies. Models must capture cold start dynamics, concurrency limits, and workload variability to produce actionable insights (Baldini et al., 2017). Accuracy, scalability, and interpretability are therefore key evaluation criteria for modeling techniques.

### 4.2 Analytical Models

#### 4.2.1 Queueing Theory-Based Methods

Queueing theory provides a formal framework for modeling request arrivals, service rates, and waiting times in serverless systems. Common formulations include M/M/c and M/G/1 models, where arrivals are assumed to follow Poisson processes and service times follow exponential or general distributions. These models estimate expected response time and queue length under varying concurrency levels. Research on cloud elasticity modeling demonstrates that queueing abstractions effectively approximate system behavior under steady-state assumptions (Herbst et al., 2013). However, serverless environments complicate classical assumptions due to dynamic container initialization and cold start variability.

#### 4.2.2 Closed-Form Performance Models

Closed-form analytical models attempt to derive explicit mathematical expressions for latency and cost as functions of memory size, concurrency level, and arrival rate. Such models enable rapid evaluation of configuration alternatives without simulation overhead. Some studies integrate empirical profiling data into semi-analytical formulations to approximate cold start probabilities and execution-time scaling behavior (Wang et al., 2018). Although

computationally efficient, these models often rely on simplifying assumptions such as stationary workload distributions and homogeneous service times.

#### 4.2.3 Examples from Literature and Assumptions

Existing analytical frameworks frequently assume independence between requests and neglect platform-level resource contention. While these simplifications improve tractability, they limit applicability in highly bursty workloads. Moreover, most closed-form models focus on mean latency rather than tail latency (e.g., p95 or p99), which is critical for SLA compliance in ML inference applications.

### 4.3 Simulation-Based Modeling

#### 4.3.1 Discrete Event Simulation

Discrete event simulation (DES) models system state transitions based on event-driven interactions such as request arrivals, container initialization, and completion events. DES allows fine-grained modeling of cold starts, scaling thresholds, and concurrency limits without restrictive analytical assumptions. Simulation environments can emulate realistic workload traces to estimate latency distributions under varying scaling policies (Shahrad et al., 2020). While highly flexible, DES incurs higher computational cost compared to analytical models.

#### 4.3.2 Workload Replay and Modeling Frameworks

Workload replay frameworks use historical cloud traces to evaluate scaling policies under realistic conditions. By replaying invocation sequences, researchers can assess the impact of proactive provisioning strategies on cost and latency. Such frameworks provide empirical validation but depend heavily on trace representativeness and may lack generalizability.

#### 4.3.3 Strengths and Weaknesses

Simulation-based approaches capture non-linear behaviors, burst patterns, and cold start dynamics more accurately than simplified analytical models. However, they require detailed parameterization and may not scale efficiently for large configuration spaces. Additionally, simulation outputs are scenario-dependent, limiting theoretical generalization.

## 5. INTEGRATING PREDICTIVE FORECASTING WITH TRADE-OFF MODELING

The integration of predictive workload forecasting with cost-latency trade-off modeling represents a critical advancement in serverless ML inference optimization. Rather than treating prediction and resource allocation as isolated processes, recent research emphasizes their interdependence in achieving economically efficient and SLA-compliant deployments.

## 5.1 Integration is Critical

### 5.1.1 Proactive Scaling vs. Reactive Strategies

Reactive auto-scaling mechanisms allocate resources only after workload surges are detected, which often results in cold start penalties and transient queue buildup. In contrast, proactive scaling leverages short-term workload forecasts to pre-warm execution environments and adjust concurrency levels before demand peaks materialize. Empirical cloud elasticity studies demonstrate that predictive provisioning reduces response time variability compared to threshold-based scaling (Islam et al., 2012). In serverless ML inference, where tail latency directly affects user experience and contractual obligations, integrating forecasting into scaling decisions enables more stable performance. Furthermore, predictive models allow optimization frameworks to evaluate anticipated workload states rather than current instantaneous metrics, thereby improving long-term resource planning.

### 5.1.2 Impact on SLA Compliance

Service Level Agreements (SLAs) often specify latency thresholds at high percentiles (e.g., p95 or p99). Reactive strategies frequently violate these constraints during abrupt workload spikes due to initialization delays. Integrated forecasting-optimization frameworks reduce such violations by aligning resource allocation with predicted demand distributions. Studies analyzing large-scale serverless traces indicate that SLA compliance improves when forecast-informed provisioning strategies are applied, particularly under diurnal or seasonal traffic patterns (Shahrad et al., 2020). Thus, integration enhances both economic efficiency and reliability guarantees.

## 5.2 Existing Integrated Frameworks

### 5.2.1 Joint Forecasting and Decision-Making Models

Recent frameworks combine time-series prediction models with optimization engines that compute cost-efficient scaling actions. These architectures typically operate in two stages: (i) short-horizon workload prediction, and (ii) resource configuration optimization subject to latency constraints. Some approaches embed prediction outputs into queueing-based analytical models to estimate future latency distributions before enacting scaling decisions. Others integrate forecasting directly into multi-objective optimization solvers, enabling dynamic exploration of cost-latency trade-offs under predicted demand scenarios (Jonas et al., 2019).

### 5.2.2 Examples from Literature

Hybrid systems incorporating deep learning predictors with adaptive scaling policies have been proposed for cloud resource management. For instance, reinforcement learning frameworks augmented with predictive inputs have shown

improved convergence stability and reduced exploration cost compared to purely reactive agents (Mao et al., 2016). Similarly, workload-aware auto scaling mechanisms leveraging trace-driven predictions have demonstrated measurable reductions in cold start frequency and operational expenditure in serverless environments (Wang et al., 2018). These examples illustrate the practical viability of integrating predictive analytics with trade-off modeling.

## 5.3 Comparative Evaluation and Limitations

### 5.3.1 Influence of Forecasting Quality on Optimization Outcomes

The effectiveness of integrated frameworks is highly sensitive to forecasting accuracy. Prediction errors propagate into scaling decisions, potentially causing over-provisioning (increased cost) or under-provisioning (latency violations). Sensitivity analyses in prior research reveal that even small deviations in arrival rate prediction can significantly shift Pareto-optimal configurations. Therefore, uncertainty-aware optimization methods—such as robust or stochastic programming—are increasingly recommended to mitigate the impact of forecast variance (Herbst et al., 2013). Incorporating prediction confidence intervals into decision-making remains an open research area.

### 5.3.2 Practical Challenges: Real-Time Implementation and Overhead

Despite conceptual advantages, integrated forecasting-optimization frameworks face operational challenges. Real-time inference systems require low-latency scaling decisions, limiting the computational complexity of prediction and optimization modules. Deep learning-based forecasting models may introduce additional runtime overhead, while reinforcement learning agents require training data and exploration phases that risk SLA violations. Furthermore, platform-specific constraints and opaque scaling policies in commercial cloud environments restrict full control over resource allocation parameters. These challenges underscore the need for lightweight, adaptive, and platform-aware integration mechanisms.

## 6. APPLICATIONS, CASE STUDIES, AND PRACTICAL IMPLEMENTATIONS

This section examines how serverless ML inference and cost-latency trade-off strategies are deployed in real-world systems, highlighting cloud platform support, case studies, and adoption trends.

## 6.1 Cloud Platforms Supporting Serverless ML

### 6.1.1 AWS Lambda, Google Cloud Functions, Azure Functions

Leading public cloud providers offer Function-as-a-Service (FaaS) environments optimized for event-driven workloads and scalable microservices. Amazon Web Services provides AWS Lambda, which supports various languages and integrates with services such as Amazon API Gateway and AWS Sagemaker for ML inference. Google Cloud offers Google Cloud Functions, tightly coupled with Cloud Run and AutoML for scalable workloads, while Microsoft Azure provides Azure Functions, which integrates with Azure Machine Learning services. These platforms enable developers to deploy trained ML models as serverless endpoints with automatic scaling. Although economic and operational advantages are substantial, inherent cold start delays, granular billing, and configuration trade-offs pose challenges for latency-sensitive inference applications (Jonas et al., 2019). Performance optimization in such environments demands careful cost–latency modeling and workload-aware provisioning strategies.

## 6.2 Real-World Case Studies

Real-world deployments of serverless ML inference illustrate both benefits and limitations of current approaches. For example, e-commerce recommendation systems often adopt serverless inference to handle variable demand during peak events such as Black Friday, where sudden surges demand elastic scaling without upfront capacity provisioning. In a case study examining web-based image classification services, researchers demonstrate that proactive scaling informed by short-term prediction significantly reduced 99th percentile latency while controlling cost budget (Shahrad et al., 2020). Similarly, conversational AI platforms leveraging serverless functions show improved operational efficiency when forecasting-based autoscaling prevents cold start penalties during peak usage hours.

These case studies typically emphasize the importance of telemetry collection, application-specific performance profiling, and integration with platform monitoring tools (e.g., AWS CloudWatch, Azure Monitor). The practical implication is that performance models must accommodate platform-specific behaviors and workload idiosyncrasies to remain effective.

## 6.3 Industry Adoption Trends

Industry adoption of serverless ML inference has grown in domains such as IoT analytics, real-time personalization, and event-driven automation due to the economic advantages of pay-per-use models and the reduction of DevOps burden. Surveys of cloud-native application practices indicate increasing preference for FaaS deployment patterns in microservices architectures, particularly for lightweight

inference tasks that exhibit bursty traffic patterns. However, guideline reports also reveal that organizations often combine serverless with container-based orchestration (e.g., Kubernetes) to manage latency-critical components that require finer control over resource allocation. This hybrid adoption trend underscores the need for flexible performance and cost modeling frameworks.

## 7. EVALUATION METRICS AND BENCHMARKING

Evaluation of serverless ML inference systems and cost–latency trade-off models depends on well-defined metrics that capture economic and performance attributes under varying workload conditions.

### 7.1 Cost Metrics

#### 7.1.1 Cloud Bills and Cost per Request

Cloud providers charge serverless functions based on invocation count, execution duration, and allocated memory. Total cost often appears on monthly bills as aggregated charges for compute time and related services. At the micro-level, researchers compute cost per request as the product of execution time and memory price for each invocation. This fine-grained metric allows comparison of different configurations (e.g., memory sizes or concurrency settings) in experimental evaluations. Cost metrics must also account for auxiliary charges such as data egress, network usage, and provisioned concurrency overhead when analyzing economic efficiency.

## 8. CONCLUSION

This review systematically examined cost–latency trade-off modeling for serverless Machine Learning (ML) inference with an emphasis on predictive workload forecasting. The analysis demonstrates that serverless computing offers compelling advantages in elasticity, operational simplicity, and fine-grained billing; however, these benefits are accompanied by inherent performance uncertainties, particularly under bursty and latency-sensitive workloads. The literature reveals that cost and latency form a structurally conflicting objective pair, influenced by memory allocation policies, cold start dynamics, concurrency limits, and stochastic request arrivals.

Forecasting techniques—ranging from classical time-series models to deep learning architectures—play a critical role in enabling proactive scaling strategies. Comparative evaluation indicates that while analytical models provide tractable approximations for steady-state analysis, simulation-based and optimization-driven approaches offer greater realism under dynamic conditions. Reinforcement learning methods further enhance adaptability in non-stationary environments but introduce computational and exploration overhead. Importantly, the quality of workload

prediction significantly affects the shape and stability of Pareto-optimal trade-off configurations.

Overall, the review highlights that integrating predictive forecasting with cost-aware performance modeling is essential for achieving SLA compliance and economic efficiency in serverless ML inference systems. Future research should prioritize uncertainty-aware optimization, cold start modeling improvements, standardized benchmarking, and lightweight adaptive frameworks to support real-time, production-grade deployments.

### 8.1. Limitations of the Review

Despite comprehensive synthesis, this review has several limitations. First, it focuses primarily on peer-reviewed literature published between 2016 and 2025, potentially omitting emerging preprints and proprietary industrial solutions. Second, comparative analysis is constrained by heterogeneous experimental setups, datasets, and evaluation metrics reported across studies, limiting direct quantitative benchmarking. Third, most reviewed works emphasize CPU-based serverless environments, while GPU-enabled or edge-serverless platforms remain underexplored. Additionally, practical deployment considerations—such as provider-specific throttling policies and undocumented scaling heuristics—are difficult to generalize due to limited transparency from cloud vendors. Finally, while forecasting and trade-off integration are discussed conceptually, empirical validation of unified frameworks remains limited in publicly available research.

### REFERENCES

- Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Suter, P. & Tardieu, O., 2017. Serverless computing: Current trends and open problems. *Research Advances in Cloud Computing*, pp.1–20.
- Box, G.E.P., Jenkins, G.M., Reinsel, G.C. & Ljung, G.M., 2015. *Time Series Analysis: Forecasting and Control*. 5th ed. Hoboken: Wiley.
- Chen, X., Lin, X., Chen, Y. & Zomaya, A.Y., 2018. Machine learning-based workload prediction for cloud computing. *Proceedings of the IEEE International Conference on Cloud Computing*, pp.1–8.
- Crankshaw, D., Wang, X., Zhou, G., Franklin, M.J., Gonzalez, J.E. & Stoica, I., 2017. Clipper: A low-latency online prediction serving system. *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp.613–627.
- Herbst, N.R., Kounev, S. & Reussner, R., 2013. Elasticity in cloud computing: What it is, and what it is not. *Proceedings of the International Conference on Autonomic Computing (ICAC)*, pp.23–27.
- Hochreiter, S. & Schmidhuber, J., 1997. Long short-term memory. *Neural Computation*, 9(8), pp.1735–1780.
- Islam, S., Keung, J., Lee, K. & Liu, A., 2012. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1), pp.155–162.
- Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N. & Stoica, I., 2019. Cloud programming simplified: A Berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*.
- Mao, H., Alizadeh, M., Menache, I. & Kandula, S., 2016. Resource management with deep reinforcement learning. *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*, pp.50–56.
- Shahrad, M., Fonseca, P., Goiri, Í., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Faria, C., Belay, A., Bianchini, R. & Thereska, E., 2020. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. *Proceedings of the USENIX Annual Technical Conference (ATC)*, pp.205–218.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. & Polosukhin, I., 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, pp.5998–6008.
- Wang, L., Li, M., Zhang, Y., Ristenpart, T. & Swift, M., 2018. Peeking behind the curtains of serverless platforms. *Proceedings of the USENIX Annual Technical Conference (ATC)*, pp.133–146.
- Zaharia, M., Chen, A., Davidson, A., Ghodsi, A., Hong, S.A., Konwinski, A., Murching, S., Nykodym, T., Ogilvie, P., Parkhe, M. & Xie, F., 2018. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Engineering Bulletin*, 41(4), pp.39–45.
- Mao, H., Alizadeh, M., Menache, I. and Kandula, S. (2016) 'Resource management with deep reinforcement learning', *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets)*, pp. 50–56.
- Xu, J., Tang, J., Liu, Y., Su, J. and Li, Y. (2017) 'Online learning for offloading and autoscaling in energy harvesting mobile edge computing', *IEEE Transactions on Cognitive Communications and Networking*, 3(3), pp. 361–373.
- Shahrad, M., Fonseca, R., Goiri, Í., Chaudhry, G., Batum, P., Cooke, J., Laureano, E., Tresness, C., Russinovich, M. and Bianchini, R. (2020) 'Serverless in the wild: Characterizing and optimizing the serverless workload

- at a large cloud provider', USENIX Annual Technical Conference (ATC), pp. 205–218.
17. Wang, L., Li, M., Zhang, Y., Ristenpart, T. and Swift, M. (2018) 'Peeking behind the curtains of serverless platforms', USENIX Annual Technical Conference (ATC), pp. 133–146.
  18. Jonas, E., Schleier-Smith, J., Sreekanti, V., Tsai, C.C., Khandelwal, A., Pu, Q., Shankar, V., Carreira, J., Krauth, K., Yadwadkar, N. and Stoica, I. (2019) 'Cloud programming simplified: A Berkeley view on serverless computing', arXiv preprint arXiv:1902.03383.
  19. Carreira, J., Fonseca, P., Tumanov, A., Zhang, A. and Katz, R. (2018) 'A case for serverless machine learning', Proceedings of the Workshop on Systems for ML and Open Source Software (SysML).
  20. Lin, W., Zhang, C., Ma, J., Chen, Y. and Li, D. (2018) 'Online auto-scaling of microservices with reinforcement learning', IEEE International Conference on Web Services (ICWS), pp. 162–169.
  21. Zhang, Q., Chen, M., Chen, W., Li, X. and Li, J. (2016) 'Dynamic resource provisioning in cloud computing: A randomized auction approach', IEEE Transactions on Cloud Computing, 4(2), pp. 248–261.
  22. Ghanbari, H., Simmons, B., Litoiu, M. and Iszlai, G. (2011) 'Exploring alternative approaches to implement an elasticity policy', Proceedings of the 2011 IEEE International Conference on Cloud Computing, pp. 716–723.
  23. Islam, S., Keung, J., Lee, K. and Liu, A. (2012) 'Empirical prediction models for adaptive resource provisioning in the cloud', Future Generation Computer Systems, 28(1), pp. 155–162.
  24. Bodík, P., Griffith, R., Sutton, C., Fox, A., Jordan, M. and Katz, R. (2010) 'Statistical machine learning makes automatic control practical for Internet datacenters', Proceedings of the USENIX HotCloud, pp. 1–6.
  25. Chen, X., Zhang, H., Wu, C., Mao, S., Ji, Y. and Bennis, M. (2021) 'Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning', IEEE Internet of Things Journal, 8(5), pp. 4005–4018.
  26. Lloyd, W., Ramesh, S., Chinthapati, S., Ly, L. and Pallickara, S. (2018) 'Serverless computing: An investigation of factors influencing microservice performance', IEEE International Conference on Cloud Engineering (IC2E), pp. 159–169.
  27. McGrath, G. and Brenner, P.R. (2017) 'Serverless computing: Design, implementation, and performance', IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 405–410.
  28. Spillner, J. (2017) 'Snafu: Function-as-a-Service (FaaS) runtime design and implementation', Proceedings of the 2017 IEEE International Conference on Cloud Engineering Workshop, pp. 1–7.
  29. Herodotou, H., Dong, F. and Babu, S. (2011) 'No one (cluster) size fits all: Automatic cluster sizing for data-intensive analytics', Proceedings of the 2nd ACM Symposium on Cloud Computing (SoCC), pp. 1–14.
  30. Gandhi, A., Harchol-Balter, M., Das, R. and Lefurgy, C. (2012) 'Optimal power allocation in server farms', Proceedings of the ACM SIGMETRICS, pp. 157–168.
  31. Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M. and Tantawi, A. (2008) 'Analytic modeling of multitier Internet applications', ACM Transactions on the Web, 2(1), pp. 1–32.
  32. Gias, U.A., Casale, G. and Ellahi, W. (2019) 'A survey on modeling and optimization of cloud computing systems', ACM Computing Surveys, 52(1), pp. 1–36.
  33. Li, Z., O'Brien, L., Zhang, H. and Cai, R. (2013) 'On a catalogue of metrics for evaluating commercial cloud services', Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, pp. 164–173.
  34. Chen, Y., Alspaugh, S. and Katz, R. (2012) 'Interactive analytical processing in big data systems: A cross-industry study', Proceedings of the VLDB Endowment, 5(12), pp. 1802–1813.
  35. Xu, H. and Li, B. (2013) 'Dynamic cloud pricing for revenue maximization', IEEE Transactions on Cloud Computing, 1(2), pp. 158–171.
  36. Farokhi, F., Kargahi, M. and Buyya, R. (2015) 'Energy-efficient resource provisioning in cloud computing', IEEE Transactions on Sustainable Computing, 1(2), pp. 114–127.
  37. Zhang, Y., Cherkasova, L. and Loo, B.T. (2013) 'Performance modeling of resource contention in multi-tenant clouds', Proceedings of the 2013 IEEE International Conference on Cloud Engineering, pp. 1–10.