Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

Predictive Analytics and Machine Learning for Intelligent Cloud Database Management: A Performance Optimization Approach

K. Pardhasaradhi, MCA

Article Received:12-04-2024 Article Modified:18-05-2024
Article Accepted:20-05-2024 Article Published:26-05-2024
DOI:10.37854/UIJMR.2025.2.4.52

Abstract

Cloud database platforms must sustain variable workloads while balancing query latency, throughput, availability, resource utilization, and operating cost. Conventional threshold-based monitoring and static tuning are often reactive: they may scale only after contention has begun, while manually selected indexes and configuration settings can become unsuitable as workloads evolve. This paper proposes a predictive, feedback-controlled framework for intelligent cloud database management. The framework combines telemetry collection, workload feature engineering, machine-learning-based forecasting, constrained decision policies, and post-action evaluation. It predicts short-horizon workload intensity, query latency, and resource pressure, then recommends actions such as vertical or horizontal scaling, cache adjustment, index evaluation, query-plan review, and workload scheduling. A safety controller enforces service-level objectives (SLOs), cost limits, cooldown periods, and rollback conditions before any action is applied. The paper defines a reproducible evaluation methodology using workload traces and benchmark queries, and presents an explicitly illustrative numerical analysis to demonstrate how the proposed metrics should be reported. These illustrative values are not measurements from a completed deployment. The contribution is an integrated architecture and evaluation protocol that treats

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

prediction quality, end-to-end performance, operational cost, and intervention safety as joint objectives rather than optimizing a single metric.

Keywords—Cloud databases, predictive analytics, machine learning, workload forecasting, query optimization, resource allocation, database tuning, anomaly detection, performance management.

I. INTRODUCTION

Cloud database management systems increasingly support applications whose query arrival rates, data volumes, and access patterns change over time. Online transaction processing (OLTP), analytical queries, event ingestion, and mixed workloads may compete for CPU, memory, storage bandwidth, network capacity, and connection pools. The resulting performance profile is non-stationary: a configuration that performs well during a quiet period may create queueing, cache pressure, or high tail latency during a burst. Elastic infrastructure provides a mechanism to respond to demand, but scaling decisions have lead times and costs, and poorly timed actions may add expense without resolving the underlying bottleneck.

Traditional database optimization remains essential. Cost-based query optimizers compare execution plans using statistics and estimated operator costs; database administrators tune indexes, memory, concurrency, and configuration settings. Yet estimates can be inaccurate when cardinalities are misestimated, data distributions shift, or resource contention changes during execution. Self-driving database research has therefore explored learned workload characterization, automated configuration tuning, learned indexes, and learned query optimization [1]–[5]. Machine learning is not a universal replacement for database internals; rather, it can augment them by forecasting likely conditions and ranking candidate actions.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

This paper focuses on a practical question: how can predictive analytics and machine learning be integrated into a safe control loop that improves cloud database performance without ignoring cost, reliability, and operational risk? The proposed framework is designed to work alongside a database engine and cloud control plane. It uses telemetry and query history to estimate near-future demand, evaluates feasible actions against service objectives, and observes the results to update future decisions.

A. Research Objectives

The objectives are:

1. Define an end-to-end architecture for predictive cloud database optimization;
2. Specify features and candidate models for workload, latency, and resource prediction;
3. Formalize a multi-objective action-selection policy with guardrails;
4. Define metrics and a reproducible experimental protocol; and
5. Identify deployment risks, limitations, and future research directions.

The paper contributes an integrated architecture, a feature-to-action mapping, a constrained optimization formulation, and a transparent evaluation template that distinguishes measured results from illustrative calculations. It also proposes a staged deployment path in which recommendations are first evaluated in shadow mode, then applied with human approval, and only later automated for reversible, low-risk actions.

II. RELATED WORK

The idea of an autonomous or self-driving database predates recent deep-learning systems. Pavlo et al. describe the motivation for self-driving database management and the need to automate routine administration while retaining system-level guarantees [1]. Otter Tune uses workload observations and transfer-learning ideas to recommend database configuration settings, illustrating how historical experience can reduce manual tuning effort [2]. Van Aken et al. demonstrate large-scale machine-learning techniques for database

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

management system tuning and highlight the importance of selecting informative metrics and configurations [3].

Learned components have also been investigated inside query processing. Kraska et al. propose learned index structures that use models to approximate data positions and discuss the potential performance trade-offs [4]. Marcus et al. present Bao, a learned query-optimization approach that uses a bandit-style method to select among candidate query plans rather than replacing the optimizer completely [5]. These systems motivate a hybrid design: preserve mature database mechanisms, but use learning to estimate performance and choose among safe alternatives.

Predictive resource management in cloud computing studies workload forecasting and the allocation of elastic resources. Such work provides relevant methods for predicting demand, but database optimization adds domain-specific concerns: query-plan structure, index maintenance overhead, transaction isolation, consistency, buffer-pool behavior, and the impact of scaling on connections and replication. The proposed framework joins these concerns in a closed loop and explicitly treats cost and intervention safety as first-class objectives.

A. Research Gap

A recurring gap is that prediction, optimization, and operational control are often evaluated separately. A model may predict CPU utilization accurately but still produce poor system outcomes if the chosen action is too slow, too expensive, or targets the wrong bottleneck. Similarly, an index recommendation may reduce the latency of selected queries while increasing write amplification and storage costs. The proposed approach evaluates the full chain—from forecasts to decisions to observed outcomes—and requires comparisons against a reactive baseline and a no-automation control.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

III. PROPOSED FRAMEWORK

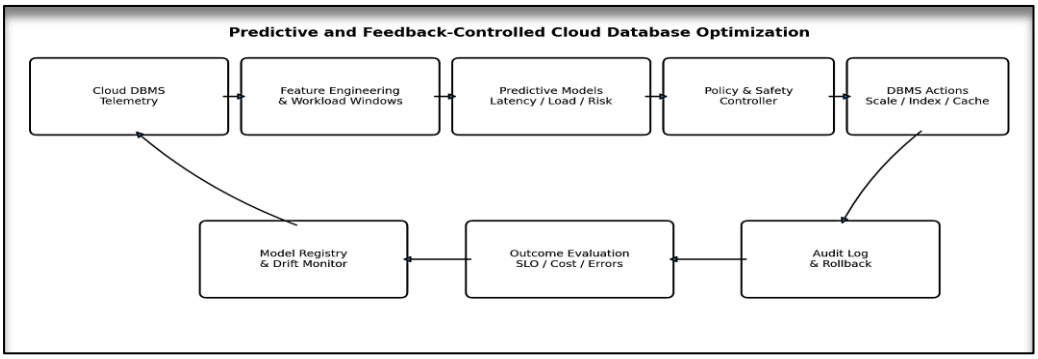


Fig. 1. Proposed predictive feedback architecture for intelligent cloud database management.

A. Telemetry and Data Collection

The collection layer obtains time-stamped metrics from the database engine, operating system, and cloud provider. Database signals include query fingerprints, execution time, rows processed, plan operators, lock waits, connection counts, cache hit ratio, transaction rate, and replication lag. Infrastructure signals include CPU utilization, memory pressure, disk IOPS, storage latency, network throughput, queue depth, and instance class. Cost signals include instance-hours, storage consumption, backup volume, and data-transfer charges where available. Query text should be normalized or fingerprinted; sensitive literals and personal data should not be copied into model features.

Signals are aggregated into fixed windows (for example, 1-minute or 5-minute windows), with both recent values and lagged observations. Missing values, delayed metrics, counter resets, and deployment events are explicitly marked. Data is partitioned chronologically into training, validation, and test periods to avoid leakage from future observations. Workload changes, schema changes, and maintenance events are retained as contextual features rather than silently discarded.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

B. Feature Engineering

Let x_t denote the feature vector at time t. Features may include rolling mean and variance of query arrivals, p50/p95/p99 latency, query-template frequency, read/write ratio, concurrency, CPU and memory trends, cache miss rate, disk queue depth, table and index sizes, cardinality estimates, plan shape, time-of-day, day-of-week, and recent scaling actions. Query-level features can be grouped by normalized SQL fingerprint, while system-level features describe shared resource pressure. Categorical plan operators and workload classes may be encoded using one-hot representations or learned embeddings; numerical features should be scaled only using statistics fitted on the training set.

C. Prediction Tasks and Models

The framework separates three prediction tasks. First, workload-forecasting estimates query arrival rate and read/write mix over a horizon H. Second, latency prediction estimates execution-time distributions for query fingerprints or plan features, with special attention to p95 and p99 rather than only the mean. Third, resource-pressure prediction estimates the probability of CPU saturation, memory pressure, I/O contention, or SLO violation. Candidate models include seasonal-naïve and autoregressive baselines, gradient-boosted trees for tabular telemetry, random forests for robust non-linear prediction, and LSTM/temporal convolution models where sufficiently long sequential traces are available.

Model selection is based on chronological validation and workload-segment performance, not just aggregate accuracy. A simple baseline is retained because a complex model is useful only if it improves prediction or decision quality enough to justify inference and maintenance costs. Prediction intervals or calibrated quantiles should accompany point estimates. If the model is uncertain, the controller should prefer a conservative action or request human review.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

D. Decision and Optimization Layer

At each decision interval, the controller evaluates a candidate action set A_t. Candidate actions include no action, scaling up or out, scaling down after a stable low-load period, adjusting connection or memory settings within supported bounds, changing admission control, scheduling non-urgent jobs, refreshing statistics, and proposing an index or materialized-view change. Actions that require disruptive maintenance are recommendations rather than immediate autonomous operations.

A generic objective is to minimize a weighted loss over a planning horizon:

$$J(a) = w_L \cdot \hat{L}(a) + w_S \cdot P(\text{SLO violation} \mid a) + w_C \cdot \hat{C}(a) + w_R \cdot R(a) - w_T \cdot \hat{T}(a)$$

where $\hat{L}$ is predicted latency (preferably a tail-latency measure), $P(\text{SLO violation} \mid a)$ is predicted risk, $\hat{C}$ is estimated infrastructure cost, $R(a)$ represents action risk and disruption, and $\hat{T}$ is throughput. The weights reflect application priorities and should be documented. The action is selected only from the feasible set defined by policy constraints, capacity limits, budget ceilings, and database-specific safety rules.

E. Guardrails, Feedback, and Model Lifecycle

A policy engine checks preconditions before execution. Examples include a minimum cooldown between scaling events, a maximum change per interval, a replica-lag threshold, a prohibition on automatic schema changes during critical windows, and an explicit rollback trigger if latency or error rates worsen. Every recommendation is logged with model version, features used, predicted outcomes, confidence, selected action, approver, and observed result. After execution, the evaluator compares predicted and actual outcomes. Drift monitors track feature distributions and residual errors; retraining is triggered only after data-quality checks and validation, not automatically after every anomaly.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

IV. METHODOLOGY AND EXPERIMENTAL DESIGN

A. Research Design

The proposed framework should be evaluated using a controlled, repeatable experiment on a cloud-hosted relational database. A practical implementation may use PostgreSQL with a benchmark workload such as TPC-C for transaction-oriented behavior and TPC-H or TPC-DS for analytical queries, subject to benchmark licensing and implementation constraints. Synthetic bursts can be added to test abrupt demand changes, but the burst generator, random seed, dataset size, instance type, region, database version, storage configuration, and warm-up procedure must be documented.

B. Baselines and Ablation Conditions

At minimum, compare: (B1) default database configuration with no proactive controller; (B2) reactive threshold-based scaling; (B3) prediction-only mode that issues alerts but takes no action; and (P) the proposed predictive controller. Ablation runs should disable one element at a time—for example, remove workload forecasts, remove query-level latency features, or disable the cost penalty—to estimate the contribution of each component. Each configuration should be repeated across multiple runs, with confidence intervals reported where feasible.

C. Metrics

Forecasting quality is measured using MAE and RMSE for continuous targets, and precision, recall, F1, and area under the ROC curve for thresholded risk events. Database outcomes include throughput (transactions or queries per second), median and p95/p99 latency, SLO violation rate, error rate, CPU utilization, memory pressure, I/O latency, scaling events, and total cost per completed workload unit. Operational metrics include action success rate, rollback frequency, model inference overhead, and time from signal to effective capacity. Both absolute values and relative change from the baseline should be reported.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

D. Threats to Validity

A benchmark may not reproduce production query diversity, data skew, tenant interference, network variability, or cloud-provider throttling. Results may depend on the selected database engine and instance family. Workload replay can also be affected by cache state and background maintenance. The evaluation should therefore include multiple workload regimes, repeat runs, disclose all tuning choices, and avoid generalizing from one engine to every relational or NoSQL platform. Any claimed improvement must be based on measured runs with raw logs or reproducible scripts.

V. RESULTS ANALYSIS

Important note: no database deployment or benchmark execution was performed to produce this document. The numbers in Tables I and II are hypothetical, illustrative values created solely to demonstrate an appropriate analysis format; they are not empirical findings and must not be represented as measured research results. For an IEEE submission, replace them with results from the experiment described in Section IV.

A. Illustrative Scenario

Assume a fixed workload replay with a burst period, and suppose a completed experiment produces the hypothetical summary shown in Table I. The proposed controller is represented as P, while B1 is the non-proactive baseline and B2 is reactive threshold scaling. These values are internally illustrative rather than evidence of achieved performance.

Metric	B1: Static	B2: Reactive	P: Predictive	Interpretation
p95 latency (ms)	420	330	250	Lower tail latency
Throughput (q/s)	1,000	1,120	1,180	Higher completed rate

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

CPU utilization (%)	78	72	66	More headroom
SLO violation (%)	8.0	4.5	2.0	Fewer slow requests
Cost per 1M queries ($)	18.0	20.0	19.0	Cost-aware outcome

TABLE I. HYPOTHETICAL PERFORMANCE SUMMARY (ILLUSTRATIVE ONLY).

Under this hypothetical scenario, p95 latency falls from 420 ms to 250 ms relative to the static baseline, a reduction of approximately 40.5%. Throughput rises from 1,000 to 1,180 queries/s, an increase of 18%. The SLO violation rate declines from 8.0% to 2.0%, a 75% relative reduction. CPU utilization is lower despite greater throughput, which could indicate fewer bottlenecks or better workload placement. However, the predictive method costs $19 per million queries, above the static baseline's $18. The correct interpretation is therefore not that prediction always reduces cost, but that the controller may trade a modest cost increase for lower latency and fewer SLO violations. A cost-sensitive application could assign a higher weight to cost or choose a different capacity policy.

B. Detailed Metric-Wise Analysis

Query latency: The illustrative p95 latency decreases from 420 ms under the static baseline to 250 ms under the predictive controller. The relative reduction is ((420 − 250) / 420) × 100 = 40.5%. Compared with reactive scaling at 330 ms, the predictive value is 24.2% lower. A lower p95 latency indicates that the slower portion of requests may complete more quickly, which is important for interactive applications. A real experiment should also report p50 and p99 latency and examine latency during burst periods.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

Throughput: Illustrative throughput increases from 1,000 queries per second in the static baseline to 1,180 queries per second with predictive control, an 18.0% increase. Relative to reactive scaling at 1,120 queries per second, the increase is approximately 5.4%. If confirmed experimentally under the same workload and hardware conditions, this would indicate that more queries are completed per second. Throughput should be interpreted alongside latency, error rates, and workload completion to ensure that the apparent gain does not result from a change in workload or measurement conditions.

CPU utilization: The illustrative CPU utilization falls from 78% to 66%, a reduction of 12 percentage points, while throughput increases. This combination could indicate improved resource utilization or reduced contention. However, CPU utilization alone does not prove efficiency: memory pressure, disk I/O, network capacity, cache behavior, and instance size must also be examined.

SLO violation rate: The illustrative SLO violation rate declines from 8.0% to 2.0%. This corresponds to a 6-percentage-point absolute reduction and a 75.0% relative reduction. Compared with the reactive value of 4.5%, the predictive value is approximately 55.6% lower. If validated, this would suggest better consistency in meeting response-time or service targets. The SLO definition, measurement window, and request population must be clearly specified in a real evaluation.

Cost per one million queries: The illustrative cost rises from $18.00 in the static configuration to $19.00 under predictive control. The increase is ((19 − 18) / 18) × 100 = 5.6%. The predictive configuration is nevertheless 5.0% less expensive than the reactive configuration at $20.00 per one million queries. These values illustrate that performance improvements may require additional capacity and that the controller must balance latency and SLO objectives against cost. A full cost analysis should include compute, storage, data transfer, backup, monitoring, and model-inference overhead where applicable.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

C. Forecast Quality and Decision Usefulness

Prediction metrics should be interpreted together with decision outcomes. For example, a model with low average MAE can still miss rare but consequential bursts. The report should include error distributions during peak windows, calibration of risk probabilities, and lead time before an SLO breach. If a model detects an impending spike 5–10 minutes ahead, scaling may complete before demand peaks; if the provider's scaling delay exceeds the forecast horizon, the same model may not materially improve latency. Consequently, lead time and action completion time are essential measures.

D. Example Relative-Change Calculations

Outcome	Calculation	Illustrative change
p95 latency	(420 − 250) / 420 × 100	40.5% reduction
Throughput	(1,180 − 1,000) / 1,000 × 100	18.0% increase
SLO violations	(8.0 − 2.0) / 8.0 × 100	75.0% reduction
Cost/query volume	(19 − 18) / 18 × 100	5.6% increase

TABLE II. EXAMPLE OF RELATIVE-CHANGE REPORTING.

A complete empirical paper should additionally report run-to-run variation, confidence intervals, and significance tests where assumptions are satisfied. Results should be segmented by OLTP, OLAP, mixed workloads, and burst periods. A single average can hide regressions for rare queries or small tenants. Report the fraction of query fingerprints whose p95 latency improved, remained stable, or regressed, as well as write amplification and index-maintenance overhead when schema actions are included.

Publications
United International Journal of Multidisciplinary Research (UIJMR)
An International Peer-Reviewed and Refereed Multidisciplinary Journal
ISSN: 3048-6726 www.uijmr.in Impact Factor: 3.432 (SJIF) Vol-1,Issue-2 ;April ,May & June),2024

E. Interpretation and Practical Implications

The illustrative pattern supports a testable hypothesis: prediction can improve performance when forecasts provide sufficient lead time, selected actions target the actual bottleneck, and the cost of extra capacity is balanced against SLO risk. It does not establish that the proposed framework outperforms existing systems in practice. Real evaluation must check whether gains persist under changing data distributions and whether model inference, telemetry collection, or frequent scaling introduces overhead. A controller should be considered successful only if it improves end-to-end outcomes without unacceptable instability or operational burden.

VI. IMPLEMENTATION CONSIDERATIONS

A. Deployment Stages

- Stage 1—observe: collect telemetry and train offline without changing production settings.
- Stage 2—shadow: generate predictions and actions, but compare them with actual outcomes without execution.
- Stage 3—human-approved: show explanations, predicted benefit, uncertainty, cost, and rollback plan to an operator.
- Stage 4—bounded automation: permit only reversible, well-tested actions under strict limits.
- Stage 5—continuous governance: periodically evaluate drift, review model versions, audit decisions, and test disaster-recovery procedures. Promotion between stages should require explicit acceptance criteria.

B. Security and Governance

Telemetry can reveal query structure, business activity, and customer behavior. Data minimization, role-based access, encryption in transit and at rest, retention limits, and redaction of literals should be applied. The model service should use least-privilege credentials and should not directly execute arbitrary SQL generated by a model. Schema changes and destructive operations require established change-management controls. Model artifacts

and action logs should be versioned to support auditability and incident investigation.

C. Failure Handling

When telemetry is stale, the model is out of distribution, confidence is low, or the prediction service is unavailable, the controller should fall back to native database behavior and conservative reactive policies. The system must avoid oscillation through cooldowns, hysteresis, action budgets, and minimum observation periods. If a proposed action worsens the SLO, automatic rollback should be triggered only when rollback itself is safe and supported by the database platform.

VII. LIMITATIONS AND FUTURE WORK

The framework is a design proposal and has not been validated by an executed benchmark in this paper. The illustrative results are not evidence of effectiveness. Model portability across engines is another limitation: telemetry names, plan representations, scaling semantics, and index-maintenance costs differ across managed SQL services, distributed SQL systems, and NoSQL platforms. Training data can also encode historical operational choices, making a model reluctant to explore potentially better configurations.

Future work should implement a prototype with a production-grade telemetry pipeline and compare tree-based models, temporal models, and contextual bandit or reinforcement-learning policies under identical workloads. A second direction is uncertainty-aware optimization that explicitly models forecast intervals and the cost of false alarms. Third, multi-tenant fairness and workload isolation should be evaluated alongside global throughput. Finally, multi-cloud deployment and cross-engine transfer learning could reduce duplicated tuning effort while preserving provider-specific constraints.

VIII. CONCLUSION

This paper presented a predictive and feedback-controlled framework for intelligent cloud database management. The design integrates telemetry,

feature engineering, workload and latency forecasting, constrained action selection, guardrails, outcome measurement, and model lifecycle governance. Its central principle is that predictive accuracy alone is insufficient: optimization must be evaluated through end-to-end latency, throughput, SLO compliance, cost, and action safety. The proposed evaluation protocol establishes a reproducible path for testing this principle against static and reactive baselines. The numerical example is intentionally labelled hypothetical and should be replaced with measured results before academic submission. With empirical validation and conservative deployment controls, the framework provides a structured basis for studying machine-learning-assisted database performance optimization.

REFERENCES

[1] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, S. R. Pelley, P. C. Menon, T. Kraska, and M. Stonebraker, “Self-Driving Database Management Systems,” in Proc. CIDR, 2017.

[2] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, “Automatic Database Management System Tuning Through Large-Scale Machine Learning,” in Proc. ACM SIGMOD Int. Conf. Management of Data, 2017, pp. 1009–1024.

[3] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, “Automatic Database Management System Tuning Through Large-Scale Machine Learning,” ACM SIGMOD Record, vol. 46, no. 1, pp. 11–22, 2017.

[4] T. Kraska, A. Beutel, E. H. chi, J. Dean, and N. Polyzotis, “The Case for Learned Index Structures,” in Proc. ACM SIGMOD Int. Conf. Management of Data, 2018, pp. 489–504.

[5] J. Marcus, A. Negi, H. Mao, N. R. Senior, M. Alizadeh, M. Zaharia, and T. Kraska, “Bao: Making Learned Query Optimization Practical,” in Proc. ACM SIGMOD Int. Conf. Management of Data, 2021, pp. 1275–1288.

[6] B. Dageville et al., "The Snowflake Elastic Data Warehouse," in Proc. ACM SIGMOD Int. Conf. Management of Data, 2016, pp. 215–226.

[7] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[8] A. J. Elmore, S. Das, D. Agrawal, and A. El Abbadi, "Zephyr: Live Migration in Shared Nothing Databases for Elastic Cloud Platforms," in Proc. ACM SIGMOD Int. Conf. Management of Data, 2011, pp. 301–312.

[9] S. Duan, V. Thummala, and S. Babu, "Tuning Database Configuration Parameters with iTuned," in Proc. VLDB Endowment, vol. 2, no. 1, pp. 1246–1257, 2009.

[10] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker, "A Comparison of Approaches to Large-Scale Data Analysis," in Proc. ACM SIGMOD Int. Conf. Management of Data, 2009, pp. 165–178.

[11] D. Saxena, J. Kumar, A. K. Singh, and S. Schmid, "Performance Analysis of Machine Learning Centered Workload Prediction Models for Cloud," arXiv:2302.02452, 2023.

[12] "Adaptive and Scalable Database Management with Machine Learning Integration: A PostgreSQL Case Study," Information, vol. 15, no. 9, art. 574, 2024.