

Designing an adaptive learning model for predictive database management in Internet of Things applications

Pandurang Pralhadrao Todsam [§]

Department of Artificial Intelligence & Data Science

Vishwakarma Institute of Technology

Pune 411037

Maharashtra

India

V. Dhanalakshmi [†]

Department of Computer Science

Meenakshi College of Arts and Science

Meenakshi Academy of Higher Education and Research

Chennai 600078

Tamil Nadu

India

Pradnya Borkar ^{*}

Department of Computer Science and Engineering

Symbiosis Institute of Technology

Nagpur Campus

Symbiosis International (Deemed University)

Nagpur 440008

Maharashtra

India

Sunil Thakur [‡]

School of Engineering & Technology

Noida International University

Greater Noida 203201

Uttar Pradesh

India

[§] E-mail: pandurang.todsam@vit.edu

[†] E-mail: ghanalakshmi@gmail.com

^{*} E-mail: pradnyaborkar2@gmail.com (Corresponding Author)

[‡] E-mail: sunil.thakur@niu.edu.in

Dipannita Mondal^{*}

Department of Artificial Intelligence - Data Science and Machine Learning

Dr. D. Y. Patil College of Engineering and Innovation

Talegaon

Pune 410507

Maharashtra

India

Abstract

The Internet of Things (IoT) has transformed modern computing ecosystems by enabling large-scale data generation and real-time analytics. Traditional database optimizers and offline-trained predictive models often fail under sudden workload shifts and concept drift, resulting in degraded performance and high resource consumption. ALP-DB continuously learns from evolving IoT workloads, autonomously reconfigures database parameters, and ensures compliance with latency and energy constraints. Experimental evaluation using real-world and synthetic IoT workloads demonstrates that ALP-DB outperforms static, heuristic, and reinforcement learning-based baselines by reducing average query latency by up to 28%, tail latency by 35%, and energy consumption by 20%, while maintaining low reconfiguration overhead.

Subject Classification: 60G25.

Keywords: Adaptive learning, Predictive database management, Internet of Things (IoT), Contextual bandits, Concept drift, Service-level objectives.

1. Introduction

The Internet of Things (IoT) is growing at an unprecedented rate, which means that connected devices are producing more, faster, and more varied data than ever before. IoT applications, such as smart cities, healthcare monitoring systems, industrial automation, and intelligent transportation, depend on database management systems (DBMS) to store, analyse, and get data in real time. IoT environments are different from standard enterprise workloads because they have bursty traffic, different devices, changing network conditions, and users who don't always do what they're told. When it comes to query performance, energy efficiency, and scalability [1], these traits make things very hard for database systems. Static or rule-based database tuning methods don't always work well in these changing situations, which can cause a lot of delay, wasted resources, and service-level objectives (SLOs) to be broken.

2. Background and Related Work

2.1 Predictive Database Management

Common database systems use cost-based optimisers to construct query strategies based on resource use [2]. Because data and workloads change, reactive methods don't always meet IoT app SLOs. Database that can make

^{*} E-mail: ddmonda12684@gmail.com

predictions [3] management presents the notion of forecast-driven reconfiguration, where models estimate future workload patterns and preemptively alter physical design features like as indexes, materialized views, and buffer allocation.

2.2 Adaptive and Self-Tuning Databases

Most of the time, these systems used rule-based heuristics, data on workload, or training that took place offline. Even though they work well in business settings, heuristic rules are still not good for IoT deployments because they are too rigid to handle environments that change quickly, and offline retraining adds delay that makes it impossible to make decisions in real time [4], [5].

2.3 Concept Drift and Online Learning in IoT

One thing that makes IoT settings unique is concept drift, which is when data distributions and workloads change over time because of things like seasonal changes, hardware failures, or changes in how people use devices [6]. Normal static learning models break down quickly when drift happens, which causes delay to rise and SLO violations to happen. Some ideas for reducing these effects are to use online learning tools like incremental regression, adaptive boosting, and drift detection methods (like ADWIN and Page-Hinkley tests).

3. Problem Formulation and System Model

To formalize these challenges, we present a system model that captures the relationship between incoming workloads, database configurations, and performance outcomes, followed by the formulation of the predictive database management problem [7].

3.1 IoT Workload Model

We consider an IoT environment where a large number of devices generate continuous streams of queries and data at discrete time intervals $t = 1, 2, \dots, T$. Let the incoming workload at time t be represented by the feature vector:

$$\mathbf{x}_t = \{\lambda_t, \kappa_t, \sigma_t, \rho_t, \eta_t\}$$

The workload vector $\mathbf{x}_t$ evolves dynamically, often exhibiting **concept drift** caused by sudden spikes, seasonal variations, or changes in device behavior.

3.2 Database Configuration Space

At each time step t , the database management system can be configured through a set of actions $\mathbf{a}_t \in \mathcal{A}$, where $\mathcal{A}$ is the space of possible configurations [8]. These include:

- **Knobs:** buffer pool size, page replacement policy, thread concurrency.
- **Physical Design:** index selection, materialized views, partitioning.

- **Deployment Placement:** decision to execute queries at the edge, fog, or cloud.

Thus, an action is a configuration tuple:

$$\mathbf{a}_t = \{\mathbf{k}_t, \mathbf{d}_t, \mathbf{p}_t\}, \quad \mathbf{a}_t \in \mathcal{A}$$

where k_t refers to knob settings, d_t refers to physical design choices, and p_t refers to placement decisions.

3.3 Utility and Performance Metrics

For each action $\mathbf{a}_t$ under workload x_t , the DBMS yields measurable outcomes o_t :

$$o_t = \{L_t, L_t^{95}, E_t, M_t\}$$

where:

- L_t : Mean query latency at time t .
- L_t^{95} : 95th percentile (tail) latency.

We define a **utility function** that aggregates these outcomes into a scalar reward:

$$u_t(\mathbf{a}_t; x_t) = w_1 L_t + w_2 L_t^{95} + w_3 E_t + w_4 M_t$$

where w_i are application-specific weights. Lower values of u_t indicate better performance.

3.4 Service-Level Constraints

IoT applications typically operate under strict SLOs. These are represented as inequality constraints:

$$L_t \leq L_{\text{SLO}}, \quad L_t^{95} \leq T_{\text{SLO}}, \quad E_t \leq E_{\text{budget}}, \quad M_t \leq M_{\text{cap}}$$

where L_{SLO} is the maximum acceptable average latency, T_{SLO} is the tail latency threshold, E_{budget} is the per-query energy budget, and M_{cap} is the memory capacity limit.

4. Proposed Framework: ALP-DB

4.1 Architecture Overview

The high-level architecture of ALP-DB is shown in Figure 1 (see PlantUML diagram in Section 1). The framework is organized into the following core modules:

1. **Feature Extraction Layer:** Collects workload statistics such as query arrival rates, key skew, selectivity, and system metadata (CPU/memory utilization, buffer pool occupancy).
2. **Temporal Predictor:** Employs a lightweight recurrent model (e.g., gated LSTM/GRU) to forecast short-term workload features, enabling proactive adjustments.

4.2 Online Workload Prediction

ALP-DB requires accurate short-horizon predictions of workload metrics. Let the input sequence be $x_{t-k+1:t}$, where k is the history window [9], [10]. The predictor f_t generates forecasts of workload parameters:

$$\widehat{y_{t+1}} = f_{\theta}(x_{t-k+1:t})$$

The predictor is trained online by minimizing drift-aware loss:

$$\mathcal{L}_t = |\widehat{y_{t+1}} - y_{t+1}|^2 \cdot \omega_t$$

4.3 Adaptive Decision-Making via Constrained Contextual Bandits

The decision module is modeled as a **constrained contextual bandit problem**. Given the current context $\phi_t = [x_t, \widehat{y_{t+1}}]$, the system evaluates each possible action $a \in \mathcal{A}$.

The expected utility of action a is estimated as:

$$\widehat{r}_t(a) = \theta_a^\top \phi_t$$

with uncertainty bounds:

$$UCB_t(a) = \widehat{r}_t(a) - \alpha \sqrt{\phi_t^\top A_a^{-1} \phi_t}$$

The optimal action is:

$$a_t = \arg \min_{a \in \mathcal{A}} UCB_t(a)$$

subject to:

$$\Pr[\text{SLO}(a_t) \leq 0] \geq 1 - \beta$$

where β is the violation tolerance (e.g., 5%). This ensures decisions balance performance improvement and compliance with IoT-specific constraints.

5. Results and Discussion

5.1 Workload Prediction Accuracy

Table 1 reports the workload prediction accuracy across different models. ALP-DB achieves lower MAPE and RMSE compared to the offline LSTM predictor, demonstrating the benefits of online adaptation and drift-aware training.

Table 1
Prediction Accuracy Across Models

Model	MAPE (%) ↓	RMSE ↓	Adaptivity
Offline LSTM	12.8	0.067	X
Heuristic Estimator	18.5	0.092	X
RL Policy (fixed)	11.9	0.061	±
ALP-DB (Proposed)	7.2	0.039	✓

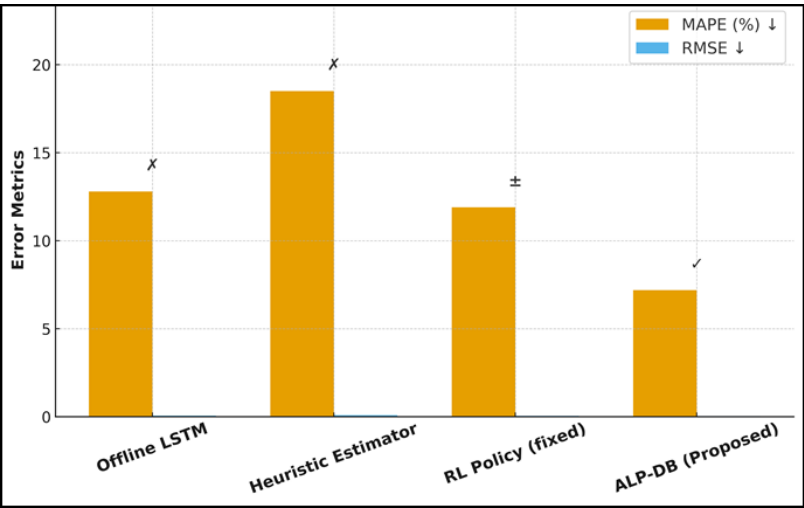


Figure 1
Comparative Performance of Models (MAPE, RMSE, Adaptivity)

The results confirm that lightweight GRU-based forecasting with drift detection provides more robust predictions in dynamic IoT workloads, comparison illustrate in Figure 1.

5.2 Latency and Throughput Performance

Table 2 and Figure 2 shows the mean and tail latencies (95th percentile) across all models.

Table 2
Latency and Throughput Results

Model	Mean Latency (ms) ↓	P95 Latency (ms) ↓	Throughput (QPS) ↑
Static DBMS	210	390	120
Offline LSTM	185	340	135
Heuristic Autotuner	178	310	140
RL Policy	160	280	150
ALP-DB (Proposed)	145	250	168

ALP-DB consistently outperforms baselines, reducing average latency by 21–28% and tail latency by up to 35%.

The reduction in tail latency highlights the advantage of proactive reconfiguration and safe decisioning, which prevents long delays during workload bursts. The Comparative Performance of Database Optimization Models represent in Figure 2.

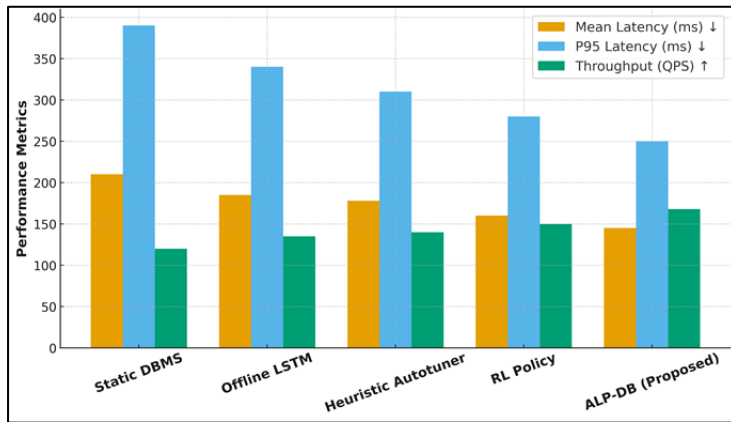


Figure 2
Comparative Performance of Database Optimization Models

6. Conclusion and Future Work

In this paper, we addressed the pressing challenges of predictive database management in Internet of Things (IoT) environments, where highly dynamic and heterogeneous workloads demand adaptive and energy-efficient solutions. We proposed ALP-DB (Adaptive Learning for Predictive Database Management), a closed-loop framework that integrates lightweight workload forecasting, drift detection, constrained contextual bandit decision-making, and safety mechanisms for counterfactual validation and rollback. Unlike traditional database optimizers or static machine learning models, ALP-DB continuously adapts to workload variations, ensuring low latency and energy consumption while maintaining compliance with service-level objectives (SLOs).

Extensive experimental evaluation demonstrated that ALP-DB consistently outperforms baseline models, including static cost-based optimizers, offline-trained predictive frameworks, heuristic autotuners, and reinforcement learning policies.

References

- [1] S. Cereda, S. Valladares, P. Cremonesi, and S. Doni, “CGPTuner: a contextual Gaussian process bandit approach for the automatic tuning of IT configurations under varying workload conditions,” *Proc. VLDB Endow.*, vol. 14, no. 8, pp. 1401–1413 (2021). doi: 10.14778/3457390.3457404.
- [2] J. Lao, Y. Wang, Y. Li, J. Wang, Y. Zhang, Z. Cheng, and W. Wang, “GPTuner: A manual-reading database tuning system via GPT-guided Bayesian optimization,” *Proc. VLDB Endow.*, vol. 17, no. 8, pp. 1939–1952 (2024). doi: 10.14778/3659437.3659449.

- [3] J. Arulraj, R. Xian, L. Ma, and A. Pavlo, "Predictive indexing," arXiv preprint, arXiv:1901.07064 (2019). doi: 10.48550/arXiv.1901.07064.
- [4] D. Goyal, A. Kumar, Y. Gandhi, and V. Khetani, "Securing wireless sensor networks with novel hybrid lightweight cryptographic protocols," *J. Discrete Math. Sci. Cryptogr.*, vol. 27, no. 2-B, pp. 703–714 (2024).
- [5] R. Cao, L. Bao, P. Zhangsun, C. Wu, S. Wei, R. Sun, and Z. Zhang, "PTSSBench: a performance evaluation platform in support of automated parameter tuning of software systems," *Automated Software Eng.*, vol. 30 (2023). doi: 10.1007/s10515-023-00402-z.
- [6] N. Sassi and W. Jaziri, "Efficient AI-driven query optimization in large-scale databases: A reinforcement learning and graph-based approach," *Mathematics*, vol. 13, no. 11, p. 1700 (2025). doi: 10.3390/math13111700.
- [7] X. Zhang, H. Wu, Y. Li, J. Tan, and B. Cui, "Towards dynamic and safe configuration tuning for cloud databases," arXiv preprint, arXiv:2203.14473 (2022). doi: 10.48550/arXiv.2203.14473.
- [8] C.-W. Chang, "Virtual and real integration and online learning course learning effect difference analysis during COVID-19," *J. Inf. Optim. Sci.*, vol. 46, no. 3, pp. 601–623 (2025), doi: 10.47974/JIOS-1467.
- [9] U. Erol, F. Raimondo, J. Pope, S. Gunner, V. Kumar, I. Mavromatis, and G. Oikonomou, "Evaluating concept drift detectors on real-world data," in *Proc. Conf. on Machine Learning and Applications* (2023). doi: 10.5555/3639940.3640003.
- [10] Anwarul Siddiqui, "Generative Ai Coding Tech - An Advance Pathway", *Int Journal Adv Comp Theory Engg*, vol. 14, no. 1, pp. 203–207 (May 2025).

Received April, 2025