Practical Machine Learning for Self-Tuning Buffer Pools in DBMS Mainframe Systems

Eduardo Mendizabal[1], Geraldo P. Rocha Filho[2],
Marcelo A. Marotta[1], Marcos F. Caetano[1],
João José C. Gondim[1], Lucas Bondan[1],
Aleteia Araujo[1]

[1]Dept. of Computer Science, University of Brasília (UnB), Brasília, Brazil
[2]Department of Exact and Technological Sciences,
State University of Southwest Bahia (UESB), Bahia, Brazil
{220005362, marcelo.marotta, mfcaetano, gondim, aleteia}@unb.br,
lucas.bondan@rnp.br, geraldo.rocha@uesb.edu.br

Abstract

This paper investigates the practical applicability of Machine Learning (ML) techniques for buffer pool optimization in Database Management Systems (DBMSs), a critical component traditionally tuned manually in mission-critical environments. While prior work proposes ML-based approaches, most evaluations are limited to simulated settings, leaving their feasibility in production systems—particularly on mainframes—largely unexplored. To address this gap, we present an empirical, production-level validation of an automated, data-driven optimization methodology applied to a relational DBMS running on a mainframe. Our approach integrates Bayesian Optimization based on Gaussian Process Regression (BO-GPR) with a three-phase pipeline: Exploratory Factor Analysis with clustering for metric reduction, LASSO regression for parameter selection, and BO-GPR for fine-grained configuration tuning. We validate the methodology using real workloads from a large-scale financial system, achieving substantial performance gains, including up to a 95% reduction in maximum synchronous I/O wait time. These results provide reproducible evidence that ML-based optimization is both effective and operationally viable in mainframe-based DBMSs.

Keywords: Machine Learning, Database Management Systems, Buffer Pool Optimization, Real Workloads, EFA, LASSO, BO-GPR.

1 Introduction

Database Management Systems (DBMSs) are responsible for storing, organizing, and managing data in a structured way, enabling efficient query execution and data sharing [1, 2]. As central components in corporate environments, DBMSs must ensure data integrity and performance across varied workloads [3–5]. However, DBMS tuning remains a complex and NP-hard problem due to the large number of interdependent configuration parameters, dynamic workloads, and platform heterogeneity [6, 7]. In practice, DBMS tuning requires navigating a complex search space with hundreds of knobs whose effects on performance are often non-linear and workload-dependent [8,9]. Traditional approaches rely heavily on DBA expertise, ad-hoc trial-and-error tuning, or static guidelines, which are insufficient for modern dynamic data environments [10, 11]. As corporate systems evolve to handle increasingly heterogeneous and time-sensitive workloads, there is a growing need for automated, data-driven methods that can adapt configurations in near real-time and provide consistent performance improvements across scenarios.

A growing body of research has investigated automated tuning strategies based on Machine Learning (ML), demonstrating performance gains that often surpass manual configurations performed by experienced DBAs [12–16]. Recent approaches [17–24] increasingly focus on modeling parameter interactions and adapting configurations to diverse workloads, spanning environments from cloud-native platforms to traditional on-premises databases. Within this landscape, Buffer Pool (BP) tuning stands out as a particularly critical task, as it directly influences I/O behavior by caching frequently accessed data pages in memory [17, 25–27]. Despite its substantial impact on performance, BP tuning remains predominantly manual, demanding specialized expertise and considerable effort, which ultimately constrains responsiveness and scalability in dynamic operational settings.

To mitigate these limitations, this paper introduces a fully automated buffer pool tuning pipeline for relational DBMSs on mainframes, focusing on IBM DB2 for z/OS [28]. The objective of this study is not only to apply ML techniques in a practical scenario but also to empirically evaluate their operational feasibility under real production constraints, characterizing the work as a scientific validation rather than a mere technological application. The experimental environment used is representative of corporate mission-critical scenarios, where small performance variations have direct impacts on operational costs and service levels, reinforcing the scientific relevance of the results. As a proof of concept, the solution was applied in collaboration with a major financial institution in Brazil, one of the largest in Latin America. The institution operates a mainframe infrastructure comprising over 400 distinct mainframe systems, 11 high-volume database silos, and more than 1,000 buffer pools dedicated to transaction processing. On average, the system processes over 400 billion mainframe transactions per month, supporting critical banking services at a national and international scale.

In this context, the proposed solution addresses a strategic institutional challenge: improving system responsiveness and resource efficiency without increasing infrastructure costs or operational risks. By automating buffer pool tuning (a component traditionally modified manually and infrequently), the proposed methodology enables continuous, data-driven performance optimization. This not only improves system scalability and reduces wait times but also directly contributes to service reliability, regulatory compliance, and operational cost reduction, which are critical concerns for financial institutions operating under high transactional loads.

Unlike experimental or simulated environments, the DBMS mainframe context imposes strict constraints on stability, predictability, and operational cost, making the empirical validation of ML techniques in this mission-critical scenario scientifically relevant. This study is an empirical validation conducted in a real production environment under actual loads and operational constraints, distinguishing it from approaches based on simulation or synthetic datasets commonly found in the literature.

Accordingly, the primary contribution of this work is not the introduction of novel Machine Learning techniques, but the delivery of robust, production-grade evidence that such techniques can be effectively deployed and sustained in one of the most constrained and business-critical computing environments: a relational DBMS operating on a mainframe platform. By demonstrating measurable performance gains under real corporate workloads, this study directly addresses long-standing concerns regarding the practicality, reliability, and risk of ML-driven self-tuning in industrial settings. In this regard, the present work significantly extends our earlier results presented at CLEI 2025 [29] by expanding the methodology to transaction-intensive environments, while also consolidating and advancing our prior investigation of non-transactional workloads reported in [30]. Collectively, these results establish a concrete bridge between academic self-tuning DBMS research and its adoption in mission-critical enterprise systems.

More specifically, relative to the conference version [29] and the preliminary study [30], the present paper introduces the following original contributions: (i) a substantially broader experimental scope, expanding the number of evaluated buffer pools from a limited subset to 29 distinct groups covering heterogeneous transactional workloads; (ii) a more rigorous statistical treatment, incorporating paired t-tests and effect size

analysis (Cohen's d) to validate the significance and practical magnitude of the observed improvements; (iii) stronger empirical results, achieving performance gains of up to 95% in synchronous I/O latency reduction, compared to approximately 45% in the earlier versions; (iv) a detailed case study (BP8K8) that illustrates the end-to-end application of the methodology in a single buffer pool, from diagnosis to sustained post-optimization monitoring; and (v) a comprehensive discussion section that interprets the results, addresses practical implications, and explicitly acknowledges limitations and threats to validity. It should also be noted that, although the final cleaned dataset (109,749 records) is numerically smaller than the raw collection reported in [30], this reduction results from a stricter data quality process, including consistency checks, outlier removal, and temporal alignment, which was deliberately designed to improve representativeness and statistical reliability for transaction-critical workloads rather than to maximize volume.

The remainder of this paper is organized as follows. Section 2 presents the theoretical foundations that support the proposed solution, including buffer pool fundamentals in DB2, supervised and unsupervised learning techniques, Exploratory Factor Analysis, and Bayesian Optimization based on Gaussian Process Regression. Section 3 introduces the proposed architecture, describing the end-to-end analytical workflow that integrates data processing, statistical modeling, and optimization. Section 4 reviews related work and positions this study within the current state of the art in automated DBMS tuning. Section 5 details the proposed methodology, formalizing the experimental design and the CRISP-DM-based pipeline adopted in this work. Section 6 reports the experimental results obtained in a real production environment, including quantitative performance gains and statistical validation. Section 7 discusses the implications of these results, practical considerations, and limitations. Finally, Section 8 concludes the paper and outlines directions for future research.

2 Background

This section presents the theoretical foundations of Buffer Pool and Machine Learning techniques used in this work, including supervised and unsupervised methods, exploratory factor analysis, and Bayesian Optimization.

2.1 Buffer Pool Fundamentals on DB2 z/OS

The Buffer Pool (BP) is a critical area of virtual storage in DB2 that keeps copies of data pages and index pages from disk [25, 26]. Its main role is to minimize physical I/O operations, which are significantly slower than memory access, thereby improving overall query response time. In the z/OS environment, BPs are managed by DB2's Data Manager component and are defined by several key parameters that control their size and behavior [28].

A BP's size is determined by the number of pages it can hold, and its efficiency is directly related to the hit ratio, i.e., the percentage of requests served from memory rather than disk. However, simply increasing the BP size is not always optimal, since it consumes valuable memory resources and may increase page management overhead [27]. Key parameters that govern BP behavior include:

- VPMIN (Virtual Pool Minimum): The minimum number of pages reserved for immediate use, preventing the pool from shrinking below this size;
- VPUSE (Virtual Pool Use): The current number of pages allocated to the BP;
- VPMAX (Virtual Pool Maximum): The maximum number of pages to which the BP can grow;
- DWQT (Deferred Write Threshold): The percentage of pages in the BP that, when reached, triggers asynchronous write operations to disk;
- VPSEQT (Sequential Pre-fetch Threshold): The percentage of pages in the BP that, when reached, causes sequential prefetch operations to be suspended.

The complexity arises because these parameters are interdependent and their optimal values are highly sensitive to workload (OLTP vs. Batch) and specific access patterns. For example, a high DWQT can reduce synchronous I/O waits but increases the risk of a sudden I/O surge when the threshold is eventually exceeded. Our work focuses on tuning these specific parameters to minimize Maximum Synchronous I/O Delay (ms), a critical metric for OLTP performance.

2.2 Supervised Learning

Supervised learning encompasses a class of techniques in which models are trained using labeled datasets to learn a mapping between input features and known outputs [14]. These methods are commonly categorized into regression and classification tasks [15, 16]. Regression techniques aim to estimate continuous target variables by iteratively adjusting model parameters to minimize prediction error [15]. Owing to their ability to capture quantitative relationships, regression models are widely employed in domains such as weather forecasting, financial market analysis, and time-series prediction [14]. In the context of DBMS optimization, regression-based approaches-particularly LASSO and process-based methods such as Bayesian Optimization with Gaussian Process Regression (BO-GPR)—have gained prominence due to their robustness and inherent capability to model uncertainty [8].

Classical Ordinary Least Squares (OLS) regression seeks to minimize residual error but exhibits significant limitations in high-dimensional settings. Specifically, OLS does not perform variable selection and is prone to high variance, which undermines its stability and predictive performance [31, 32]. As a result, it is ill-suited for configuration parameter selection in DBMS environments, where the number of tunable parameters can be large and highly correlated [18].

LASSO (Least Absolute Shrinkage and Selection Operator) overcomes these limitations by introducing an L1 regularization term that encourages sparsity in the model coefficients [32]. By driving irrelevant coefficients to zero, LASSO simultaneously performs regularization and feature selection, making it particularly effective for high-dimensional optimization problems. This property improves model interpretability and generalization, which are critical for reliable DBMS tuning.

Despite the emergence of more sophisticated learning techniques, linear regression remains a foundational model in machine learning and serves as a building block for many advanced approaches [16]. Bayesian linear regression extends the classical framework by incorporating prior knowledge over parameters and modeling uncertainty through posterior distributions, enabling more informed and probabilistic decision-making [33].

2.3 Unsupervised Learning

Unsupervised learning comprises techniques that analyze data without predefined labels, aiming to uncover latent structures, patterns, or relationships inherent to the dataset [15, 16]. Such methods are particularly effective for exploratory analysis tasks, including clustering and association discovery, where the objective is to identify similarities and groupings without prior assumptions.

Among clustering techniques, K-means remains one of the most widely adopted algorithms. It partitions observations into k clusters by minimizing intra-cluster variance, thereby promoting compact and well-separated groups [16]. While K-means is computationally efficient and well suited for numerical data, selecting an appropriate number of clusters poses a nontrivial challenge. In this study, K-means is employed to group workload metrics exhibiting similar behavioral patterns, enabling the identification of underlying structures in the performance data and supporting subsequent dimensionality reduction and modeling steps.

In the context of DBMS performance analysis, unsupervised techniques play a critical role in handling the high dimensionality and complexity of monitoring data. Performance metrics often exhibit strong correlations and non-obvious dependencies, making manual analysis impractical. By clustering metrics with similar dynamics, unsupervised learning helps reduce redundancy, highlight dominant behavioral patterns, and improve the interpretability of subsequent modeling phases. This capability is particularly valuable in automated tuning pipelines, where uncovering latent workload characteristics enables more informed parameter selection and more robust optimization under varying operational conditions

2.4 Exploratory Factor Analysis

Exploratory Factor Analysis (EFA) is a multivariate statistical technique designed to reduce the dimensionality of complex datasets by identifying latent factors that capture the underlying structure of correlations among observed variables [34, 35]. By modeling shared variance, EFA enables the condensation of a large set of correlated metrics into a smaller number of interpretable factors, facilitating analysis and downstream modeling. The application of EFA typically begins with an assessment of data suitability, relying on established statistical tests to ensure that the correlation structure is appropriate for factor extraction.

Bartlett's Test of Sphericity evaluates whether the correlation matrix differs significantly from the identity matrix, indicating sufficient correlations among variables to justify factor analysis [34]. The test is based on a chi-square distribution and tests the null hypothesis that the variables are uncorrelated. A significant result (typically $p < 0{,}05$) rejects the null hypothesis, supporting the factorability of the data.

The Kaiser-Meyer-Olkin (KMO) measure assesses sampling adequacy by evaluating the proportion of variance among variables that may be common variance [34]. It compares observed correlations to partial correlations. Values range from 0 to 1, with values above 0.6 considered acceptable for factor analysis [35].

Factor extraction follows Kaiser's criterion [36], retaining only factors with eigenvalues greater than 1. Cattell's Scree Test [37] provides visual support for determining the number of factors to retain. After extraction, factor rotation (typically Varimax) is applied to improve interpretability by simplifying factor loadings [38].

2.5 Bayesian Optimization based on Gaussian Process Regression (BO-GPR)

Bayesian Optimization (BO) is a principled approach for optimizing black-box functions that are complex, noisy, and expensive to evaluate [39]. It relies on surrogate models to construct a probabilistic approximation of the objective function, allowing informed decision-making under uncertainty. By explicitly modeling both predicted performance and associated uncertainty, BO enables efficient exploration of the search space while minimizing the number of costly evaluations.

Gaussian Process Regression (GPR) is one of the most commonly adopted surrogate models in BO due to its non-parametric nature and strong theoretical foundations [40]. GPR defines a distribution over possible functions consistent with observed data, providing closed-form expressions for both predictive mean and variance at unseen points. Within the BO-GPR framework, these uncertainty estimates play a central role in guiding the optimization process toward regions with high potential for improvement.

The selection of the next configuration to evaluate is governed by an acquisition function, which encodes the trade-off between exploration—sampling regions of high uncertainty—and exploitation—refining regions predicted to yield high performance. Commonly used acquisition functions include:

- Expected Improvement (EI): Computes the expected improvement over the current best value, weighted by uncertainty;
- Probability of Improvement (PI): Computes the probability that a point will exceed the current best value;
- Upper Confidence Bound (UCB): Combines the mean prediction and an uncertainty term scaled by a confidence parameter.

This theoretical foundation underpins the proposed buffer pool tuning framework, providing the methodological basis for a data-driven and intelligent configuration strategy that systematically adapts to workload characteristics and performance dynamics.

3 Proposed Architecture

The proposed methodology is operationalized through a structured architecture that reflects the complete analytical workflow adopted in this work. This architecture, illustrated in Figure 1, is organized as a pipeline composed of three major stages: input data processing, analytical modeling for workload and parameter characterization, and Bayesian-based optimization. Such an organization highlights a clear separation between data preparation, statistical learning, and configuration optimization, thereby promoting clarity, modularity, and reproducibility throughout the experimental process.

The first stage, denoted as Input Data Processing, is responsible for consolidating all raw information required by the methodology. Configuration parameters are extracted directly from the DB2 system catalogs, while performance metrics are obtained from SMF records. These heterogeneous data sources are merged and submitted to a data cleaning and preprocessing step, which includes normalization, type conversion, removal of inconsistencies, and synchronization between metrics and parameters. After preprocessing, an analysis selection step defines the samples that will be used in the subsequent phases, guaranteeing that only consistent and representative observations are forwarded to the modeling components.

The second stage corresponds to the analytical core of the methodology and is divided into two complementary processes: workload characterization and parameter identification. In the workload characterization block, performance metrics are evaluated through Exploratory Factor Analysis, supported by KMO and Bartlett tests to verify statistical adequacy. The Scree Plot is used to determine the number of relevant latent factors, and K-means clustering is subsequently applied to group similar workload behaviors. This process results in a reduced and structured representation of the workload, expressed through distinct metric profiles. In parallel, the parameter identification block focuses on configuration parameters. The LASSO regression technique is applied to select the most influential parameters with respect to the observed performance behavior, producing a compact subset of configuration variables that effectively describe the system response.

The third stage implements the optimization process. The selected parameters and representative samples are used to train a Gaussian Process model, which acts as a surrogate for the real DB2 system. Based on

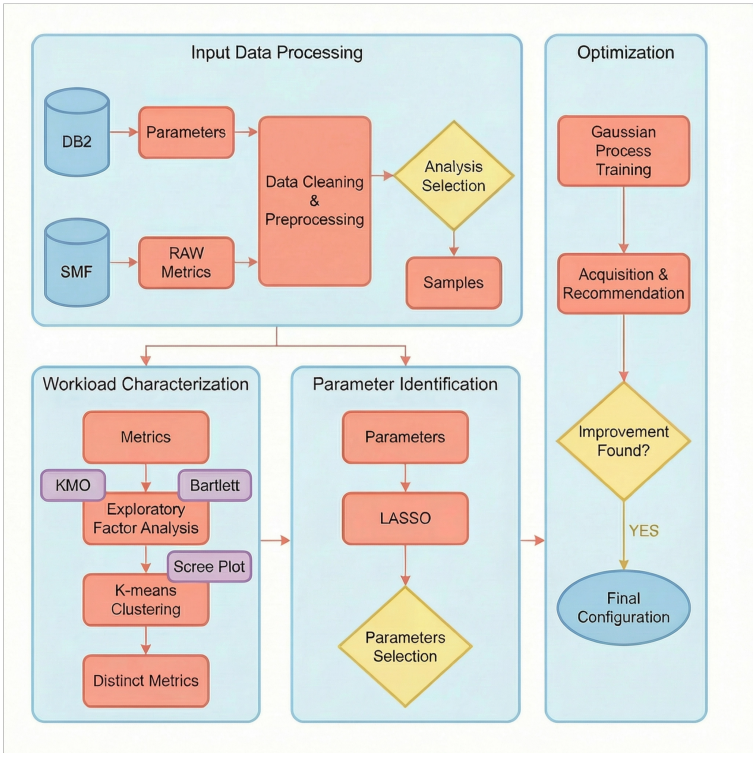

Figure 1: Proposed architecture for automated buffer pool tuning

this model, the acquisition and recommendation component applies Bayesian Optimization strategies to propose new configurations that maximize the expected performance improvement. Each recommended configuration is evaluated against the improvement criterion, and when a satisfactory gain is detected, the process converges to a final configuration. This iterative loop formalizes the architecture as a closed optimization cycle, where statistical modeling and real-system feedback are continuously combined to refine configuration recommendations.

This architecture accurately represents the end-to-end workflow of the proposed solution, from raw data acquisition to final configuration selection. It explicitly integrates statistical validation, dimensionality reduction, clustering, sparse regression, surrogate modeling, and Bayesian Optimization into a single coherent framework, ensuring that the tuning process is both data-driven and operationally feasible in production database environments.

4 Related Work

This section reviews the main advances in automated DBMS parameter tuning, with emphasis on machine learning-based approaches [8, 9, 41]. Table 1 summarizes the evolution of the state of the art, organized chronologically along three methodological dimensions: parameter selection, feature reduction, and tuning technique. Rather than suggesting a linear technological replacement, this organization highlights how different paradigms coexist, each with distinct trade-offs in computational cost, convergence speed, interpretability, and operational feasibility. In addition, the table reflects how the choice of technique is strongly influenced by the target deployment environment, particularly the contrast between traditional enterprise DBMSs and cloud-based database (CDB) platforms.

Table 1 provides a structured view of how automated DBMS tuning techniques have evolved across three orthogonal dimensions: parameter selection strategies, feature reduction mechanisms, and optimization techniques. Rather than indicating a linear technological progression, the table reveals a diversification of methodological choices driven by trade-offs among modeling complexity, computational cost, and operational feasibility. Early approaches are dominated by heuristic methods, which typically lack explicit parameter ranking and systematic metric reduction. As DBMS configurations and workloads become more complex, later works progressively incorporate statistical and learning-based mechanisms to manage the growing dimensionality of both configuratiowell-suitedd performance metrics.

An important aspect that emerges from Table 1 is that different techniques tend to dominate in different operational contexts. In classical enterprise environments, such as mainframes and on-premise relational

Table 1: Comparison of automated DBMS tuning approaches

Work	**Year**	**Selection**	**Reduction**	**Technique**
STMM [12]	2006	N/E	N/E	Heuristic
PGTune [19]	2008	N/E	N/E	Heuristic
iTuned [42]	2009	LHS	N/E	BO-GPR
OpenTuner [43]	2014	N/E	N/E	Heuristic (EA)
OtterTune [18]	2017	LASSO	EFA + K-means	BO-GPR
CDBTune [44]	2019	N/E	N/E	RL (DDPG)
iBTune [17]	2019	N/E	N/E	RL (DDPG)
Tuneful [22]	2020	N/E	N/E	BO-GPR
ReIM [45]	2020	N/E	N/E	BO-GPR
ResTune [46]	2021	N/E	N/E	BO-GPR
CGPTuner [47]	2021	N/E	N/E	BO (CGP)
UDO [48]	2021	N/E	N/E	RL (DDPG)
WATuning [23]	2021	N/E	MIM-I/O	RL (ATT-DDPG)
DB-BERT [24]	2022	NLP	N/E	RL (BERT)
LlamaTune [21]	2022	HeSBO	SHAP	BO (REMBO)
GMM/DNN [49]	2023	GMM	N/E	DL (DNN)
λ-Tune [50]	2024	LLM	ILP	LLM (GPT-4)
Centrum [51]	2025	N/E	N/E	BO (SGBE)
ADWTune [52]	2025	N/E	DBSCAN	RL (DDPG)

DBMSs, tuning is primarily concerned with reducing operational costs, improving resource efficiency, and preserving system stability. In such settings, each configuration change is expensive and risky; therefore, optimization methods must be sample-efficient, computationally lightweight, and predictable. In contrast, in cloud database (CDB) environments, tuning is often associated with elastic provisioning, rapid instantiation of new database nodes, and online adaptation to workload changes. In this latter context, higher computational overhead during training is acceptable since tuning costs are amortized over many short-lived instances and supported by abundant elastic resources.

The table also shows that Bayesian Optimization based on Gaussian Process Regression (GPR) is a recurrent and persistent choice across multiple generations of tuning systems, including iTuned, OtterTune, Tuneful, ReIM, and ResTune. This recurrence indicates that BO-GPR-based methods achieve a particularly favorable balance between sample efficiency, modeling expressiveness, and computational cost. In enterprise DBMS environments, where the main objective is cost reduction and operational efficiency, this balance is critical. BO-GPR enables effective optimization with a relatively small number of DBMS reconfigurations, making it well suited for scenarios in which each experiment has direct economic and operational implications. The stage of research of BO-GPR in Table 1 supports its role as a pragmatic and mature solution rather than merely an academic choice.

From an economic and systems perspective, adopting computationally expensive learning models to reduce CPU and memory consumption costs, limited determinism, and restricted transparency raises concerns about their impact. The aim of saving CPU and memory during the tuning process in order to optimize production undermines the very objective of optimization. For this reason, computational efficiency becomes not only a practical constraint but a first-class design requirement. BO-GPR naturally satisfies this requirement because its training and inference costs are modest, and its convergence typically requires orders of magnitude fewer interactions with the DBMS than reinforcement learning or deep neural network approaches.

In contrast, Reinforcement Learning approaches, especially those based on DDPG, appear mainly in works that aim for fully autonomous, end-to-end tuning in cloud-oriented scenarios, such as CDBTune, iBTune, and UDO. Fill methods offer high expressive power and strong online adaptation capabilities. Table 1 shows that their adoption is comparatively limited and concentrated in recent years. This reflects their substantial computational overhead, long exploration phases, and a large number of interactions required for convergence. Such properties are difficult to reconcile with enterprise production environments, but they are compatible with CDB settings, where elasticity, rapid scaling, and online learning are more important than minimizing tuning costs.

Similarly, Deep Learning and LLM-based approaches occupy a smaller portion of the landscape summarized in Table 1. Works such as GMM/DNN and λ-Tune illustrate an exploratory trend toward leveraging complex neural architectures and unstructured knowledge sources. However, their limited presence in the

table, combined with their typically experimental evaluation settings, indicates that these techniques are still at an early stage of maturity. Their results are often difficult to reproduce, their operational costs are high, and consistent performance gains over established BO or RL baselines have not yet been systematically demonstrated. Consequently, they should be interpreted as exploratory research directions rather than as consolidated alternatives for production-grade DBMS tuning.

The automated tuning workflow is usually decomposed into four stages [8], namely parameter configuration, feature selection, tuning method, and transfer techniques, which are described as follows:

- **(i) Parameter Configuration:** define the knobs to be optimized. Each parameter (κ) is associated with a configuration value (ν_κ), which can be tuned within continuous intervals (ν^{min}, ν^{max}) or discrete sets ($\nu^1, \ldots, \nu^k$). This step may rely on expert knowledge or data-driven ranking techniques such as LASSO [18] and CART [21];
- **(ii) Feature Selection:** characterize and reduce the dimensionality instead of using cs. Metrics typically include workload indicators and DBMS internal counters. Dimensionality reduction is commonly performed using EFA [18], Principal Component Analysis (PCA) [53], or clustering methods such as K-means [18];
- **(iii) Tuning Method:** solve the optimization problem using the selected parameters and features. Heuristic approaches rely on static rules or DBA expertise. Bayesian Optimization (BO) employs surrogate models such as Gaussian Processes [18], SMAC [21], and TPE [54]. Deep Learning (DL) approaches use models such as DNNs [13] and CNNs [17], while Reinforcement Learning (RL) approaches rely on agents such as DDPG [44] and DDQN [48];
- **(iv) Transfer Techniques:** reuse historical tuning knowledge across workloads or DBMS instances, ranging from simple configuration mappings [18] to full model reuse [23].

Historically, the first generation of automated tuning systems was dominated by heuristic approaches such as STMM [12] and PGTune [19], which encode DBA knowledge in fixed rules. These methods are simple and robust, but lack adaptability to workload changes and offer limited support for systematic exploration of the configuration space.

The introduction of Bayesian Optimization marked a fundamental shift. iTuned [42] formalized tuning as a black-box optimization problem using Gaussian Process Regression (GPR) as surrogate model and Latin Hypercube Sampling (LHS) for initial exploration. OtterTune [18] consolidated this paradigm by integrating LASSO for parameter ranking and EFA for metric reduction, showing that statistically grounded BO pipelines can achieve strong performance with relatively small numbers of experiments. A key advantage of BO-GPR is its favorable computational profile: training and inference costs are significantly lower than those required by deep neural networks or reinforcement learning agents, and convergence is typically achieved with orders of magnitude fewer DBMS interactions. This property is critical in production environments, where each configuration change is expensive, risky, and subject to strict operational constraints.

In contrast, Reinforcement Learning approaches such as CDBTune [44], iBTune [17], and UDO [48], largely based on DDPG, require long exploration phases and a large number of environment interactions to converge. While RL offers expressive modeling capacity and strong online adaptation, its computational cost, instability during training, and limited interpretability pose substantial barriers to adoption in mission-critical enterprise DBMS environments. Similarly, Deep Learning approaches based on DNNs [49] introduce high training overhead and require large labeled datasets, which are rarely available in real corporate deployments.

Despite recent interest in Large Language Models (LLMs) for DBMS tuning, such as λ-Tune [50], these approaches remain at an early research stage. Their evaluations are typically conducted in controlled or synthetic environments, their experimental setups are difficult to reproduce, and their performance improvements over established BO-GPR or RL baselines are not yet consistently demonstrated. Moreover, their high operational cost, limited determinism, and restricted transparency raise concerns regarding direct deployment in production DBMS infrastructures. They should therefore be viewed as exploratory research directions rather than as mature optimization solutions.

Despite the diversity of techniques, a persistent limitation in the literature is the heavy reliance on simulated environments, experimental clusters, or benchmark workloads. Very few studies demonstrate end-to-end viability under real production constraints, such as restricted experimentation budgets, stability requirements, and compliance with operational change management processes. This gap is particularly pronounced for RL, DNN, and LLM-based approaches, whose experimental demands are difficult to reconcile with corporate mainframe environments.

Our work positions itself within this gap. By adopting a BO-GPR strategy, we prioritize computational efficiency, low experimentation cost, and reproducibility, while maintaining strong optimization performance.

This choice reflects a pragmatic trade-off between modeling expressiveness and operational feasibility, aiming to bridge academic advances in DBMS tuning with the concrete requirements of industrial, mission-critical production systems.

5 Proposed Methodology

The central hypothesis of this work is that Machine Learning techniques are not only predictive-effective but also operationally viable for optimizing buffer pool parameters in a DBMS running on a mainframe, even under real performance, availability, and computational cost constraints. The proposed approach does not rely on a specific environment configuration but on the systematic use of observable DBMS metrics, reducing the risk of overfitting to a particular scenario and favoring reproducibility. In the conducted experiment, the independent variable corresponds to buffer pool parameter configurations associated with ML strategies, while the dependent variables are DBMS performance metrics such as latency, I/O efficiency, and effective memory utilization. The success criterion considers obtaining measurable improvements in DBMS performance metrics, particularly significant latency reductions, while preserving operational stability and complying with the production constraints of the mainframe environment.

For this purpose, the experiment follows the CRISP-DM framework (Cross-Industry Standard Process for Data Mining) [55, 56] and is structured in three main stages: (1) Workload characterization and metric reduction; (2) Parameter identification and ranking; and (3) Bayesian Optimization for parameter tuning. Observed variables include DBMS performance metrics (e.g., cache hit rates, I/O times) and buffer pool configuration parameters. The operational success criterion is the ability to identify configurations that significantly reduce synchronous I/O wait time without degrading other performance metrics and with an optimization computational cost justifiable in production. Although confidentiality constraints prevent full disclosure of data and infrastructure, the proposed methodology is implementation-agnostic and was designed to be replicable in other environments that provide equivalent operational metrics, in line with reproducible research principles in computational science [57].

To facilitate reproducibility and provide a concise overview of the end-to-end optimization process, Algorithm 2 summarizes the complete pipeline, from data preparation through final configuration recommendation. Each stage is detailed in the subsequent subsections. It should be noted that, for computational tractability, the GPR surrogate model is trained on a stratified subsample comprising 25% of the cleaned dataset (preserving the proportional representation of each buffer pool), with a training cap of 4,000 observations selected at random from the training partition. Model selection employs early stopping with a patience of 25 epochs, monitored on the validation-set Negative Log Predictive Density (NLPD). The acceptance policy hyperparameters (*MIN_IMPROV_PCT* and $N_{\text{CANDIDATES\_BP}}$) are jointly calibrated through a grid search that maximizes a composite score balancing adoption rate and predicted latency reduction.

5.1 Workload Characterization and Metric Reduction

This stage establishes the analytical foundation for subsequent phases. Its goal is to understand the origin, structure, and quality of available information, removing inconsistencies, standardizing formats, and assessing the dataset's statistical adequacy for dimensionality reduction and predictive modeling. The process integrates BP configuration parameters extracted from the DB2 catalog and operational time series metrics collected at 15-minute granularity. The target variable, Maximum Synchronous I/O Delay (ms), was standardized using *StandardScaler*, ensuring comparability and stabilizing distribution behavior.

After cleaning, the dataset was reduced from 45 to 35 metrics and from 7 to 6 parameters, removing about 22% of monitored variables. The determinant of the correlation matrix (1.132118×10^{-20}) and a rank equal to 32 confirmed that removing redundant columns preserved the linear structure necessary for factor analysis. Robustness was ensured by excluding records with null fields, resulting in an intermediate sample of 122,618 complete observations prior to outlier removal. Table 2 lists the removed metrics and their selection criteria.

The metric selection stage aimed to identify among the collected indicators those capable of representing BP behavior in a stable and non-redundant manner. The overall process, combining EFA and K-means clustering, is summarized in Figure 3. The process began with EFA for dimensionality reduction and latent factor discovery [34, 35]. The global KMO index reached 0.97, classifying the sample as "excellent", and Bartlett's test of sphericity was significant ($p < 0.001$), confirming sufficient correlation for factor extraction [38]. Initial factor extraction is visually supported by the Scree Plot in Figure 4, based on Cattell's test [37], which clearly indicates the number of factors to retain.

Extraction followed Kaiser's criterion [36] (eigenvalues > 1), yielding ten latent factors. Prior to factor extraction, an iterative KMO-based selection procedure was applied: at each iteration, the metric with the lowest individual KMO value was removed until the global KMO index exceeded the adequacy threshold of

Input: Raw performance metrics $\mathbf{M}$ (15-min granularity), BP configuration parameters $\mathbf{P}$, target variable y (Max Synchronous I/O Delay)
Output: Optimized configuration $\mathbf{p}^*$ for each buffer pool, or baseline preservation

Stage 1: Workload Characterization and Metric Reduction
Clean $\mathbf{M}$: remove zero-variance and highly correlated (≥ 0.995) features
Iteratively remove metrics with individual KMO < 0.6 until global KMO ≥ 0.6
Apply EFA with Varimax rotation; extract n factors via Kaiser's criterion (eigenvalues > 1)
Apply K-means on factor loadings with optimal k (Silhouette + Pham's criterion)
For each cluster, select representative metric closest to centroid $\rightarrow$ reduced set $\mathbf{M}_r$

Stage 2: Parameter Identification and Ranking
Normalize $\mathbf{P}$ using StandardScaler
Compute LASSO regularization path: vary λ from $\lambda_{\max}$ to $\lambda_{\min}$
Rank parameters by activation step (first λ where $|\hat{\beta}_j| > 0$)
Retain parameters with early or moderate activation $\rightarrow$ reduced set $\mathbf{P}_r$

Stage 3: Bayesian Optimization (BO-GPR)
Subsample dataset via stratified sampling by BPOOL (25%); split 80/20 train/validation
Train GPR on $(\mathbf{P}_r, y)$ with composite kernel (RBF + Matérn + Linear + ARD)
  Optimizer: Adam; early stopping on NLPD (patience = 25); training cap = 4,000 samples
Calibrate (MIN_IMPROV_PCT, N_{CAND}) via grid search on composite score
for each buffer pool $b \in \mathcal{B}$ **do**
  Define baseline $y_0 \leftarrow$ last observed latency for b
  Sample N_{CAND} candidates from feasible parameter space (uniform + local perturbation)
  Predict $\mu(\mathbf{x}), \sigma(\mathbf{x})$ for all candidates via GPR posterior
  Compute $\gamma = (y_{0,z} - \mu_z)/\sigma_z$
  Evaluate EI, PI, and UCB acquisition functions for each candidate
  Select best candidate per strategy; choose overall winner by lowest predicted latency
  $\Delta \leftarrow (y_0 - \hat{y}^*)/y_0$
  if $\Delta \geq MIN\_IMPROV\_PCT$ **then**
    Accept recommended configuration for b
  else
    Preserve baseline configuration for b
  end if
end for
return Set of accepted configurations $\{\mathbf{p}_b^*\}$

Figure 2: Complete optimization pipeline for automated buffer pool tuning

Table 2: Removed metrics and selection criteria

Removed Metric	**Reason**
Pending Time: Device Busy Delay (ms)	Zero variance
Decrypted Read Rate (kB/s)	Zero variance
Encrypted Write Rate (kB/s)	Zero variance
Read/Write Access Method (kB/s)	Correlation ≥ 0.995
Unchanged Pages in Buffer Pool (kB)	Correlation ≥ 0.995
Random Read (I/Os/s)	Correlation ≥ 0.995
Getpages: Buffer Pool Hits (Getpages/s)	Correlation ≥ 0.995
Read Operations (LR/s)	Correlation ≥ 0.995
VPPSEQT	Zero variance (refinement)

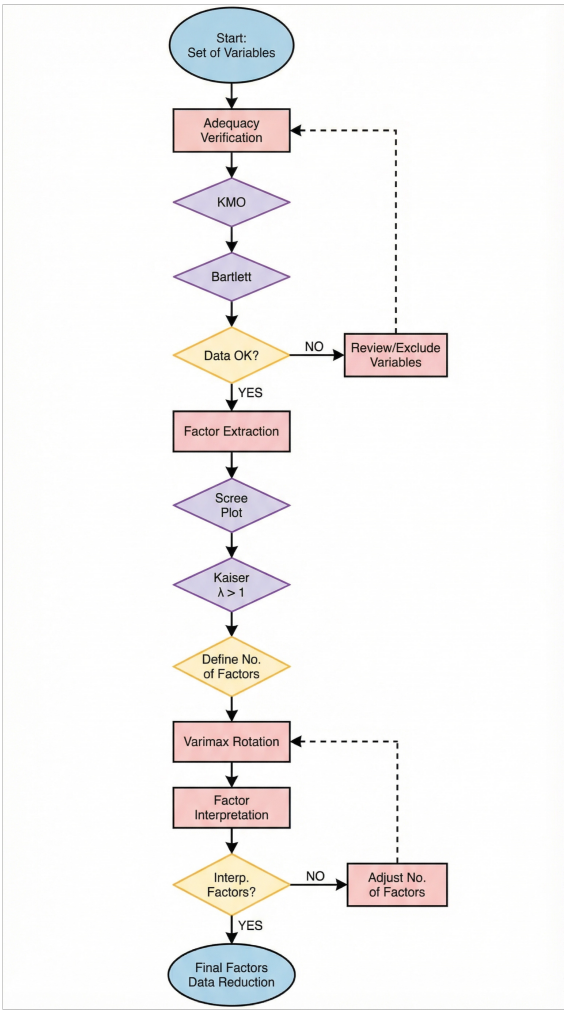

Figure 3: Workload characterization and metric reduction process.

0.6, ensuring that only metrics with sufficient sampling adequacy were retained for factor analysis. Subsequently, K-means clustering was applied to the resulting factor loadings to identify groups of metrics with statistically similar behavior. The optimal number of clusters was determined by combining elbow analysis, the Silhouette coefficient [58], and Pham's criterion, with both Silhouette and Pham indicating a common optimum. Within each cluster, the representative metric was selected as the one closest to the cluster centroid in the factor loading space (minimum Euclidean distance), ensuring that each selected metric best summarizes the latent behavior of its group. Resulting clusters are analyzed in Figure 5, showing metric distribution in factor space.

Table 3 summarizes numerical results, indicating a clear optimal solution at $k = 11$, simultaneously supported by the Silhouette coefficient and Pham's criterion, both reaching their best values at this configuration. The final model established eleven coherent clusters, each representing a distinct behavioral pattern among the monitored metrics. Table 4 lists representative metrics for each cluster, chosen based on factorial dominance and internal proximity.

The resulting set, composed of eleven representative metrics distributed across ten latent factors and eleven clusters, forms the empirical basis for subsequent parameter selection and modeling phases. This structure reduces the original dimensionality and enhances statistical interpretability, allowing the optimization steps to rely on more consistent metrics correlated with overall BP performance.

Moreover, the clustering results revealed a strong structural consistency between the EFA and the K-means stages. Each cluster is dominated by a single latent factor, indicating that the clustering process preserves the statistical structure uncovered by factor analysis instead of introducing artificial groupings. In this sense, EFA provides the latent representation of the metric space, while K-means reorganizes this space into operationally meaningful regimes. This near one-to-one correspondence between factors and clusters reinforces the methodological coherence of the dimensionality reduction process and ensures that each selected representative metric is associated with a well-defined operational behavior of the DBMS, such

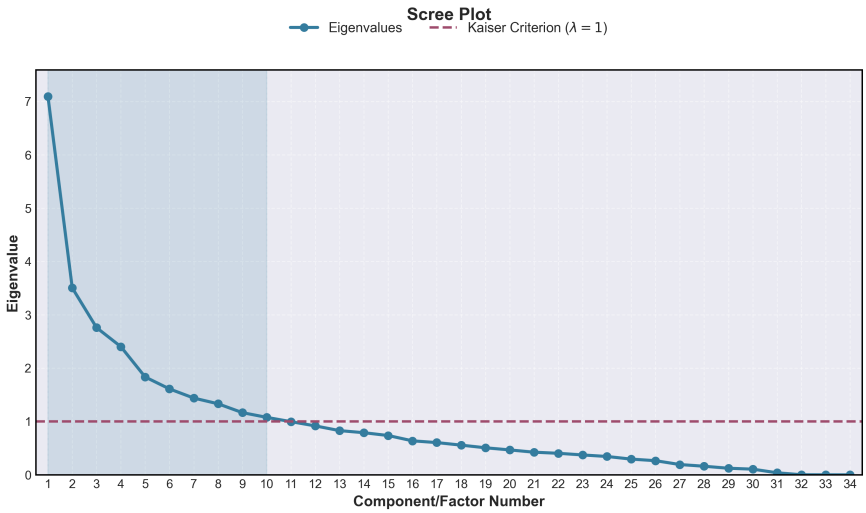

Figure 4: Scree plot for factor extraction

Table 3: K-means clustering results for different values of k

k	**SSE**	**PS**	**Silhouette**
2	12.8522	—	0.2206
3	10.7914	0.9106	0.2519
4	8.8387	0.8676	0.2591
5	7.5557	0.8939	0.3036
6	6.2711	0.8608	0.3290
7	5.2389	0.8616	0.3960
8	4.3639	0.8555	0.3730
9	3.6335	0.8525	0.4145
10	3.0531	0.8581	0.4386
11	2.3301	0.7779	0.4550
12	1.8768	0.8196	0.4498
13	1.5679	0.8489	0.4455
14	1.2636	0.8180	0.4165
15	1.1336	0.9096	0.3934

as synchronous I/O latency, asynchronous I/O efficiency, cache effectiveness, buffer pool utilization, and throughput patterns.

5.2 Parameter Identification

The Ordinary Least Squares (OLS) regression method, although commonly used, is not ideal for parameter selection in DBMSs due to two main shortcomings: (i) estimates have low bias but high variance, which increases as more features are included, compromising prediction accuracy and variable selection; and (ii) estimates become harder to interpret as the feature set grows because irrelevant features are never removed [18].

As an alternative, we chose the LASSO (Least Absolute Shrinkage and Selection Operator) regression technique [31, 32], which adds a penalty proportional to the sum of absolute coefficients to the model's cost function. This induces sparsity and forces some coefficients to be exactly zero. The penalty enables LASSO to perform variable selection and coefficient estimation simultaneously, yielding more stable and interpretable models.

The technique uses regularization to select variables that most influence system performance, removing small weights and discarding less impactful features. Metrics were normalized using Scikit-Learn's StandardScaler [59] to bring all variables to the same scale. Based on the LASSO path analysis, parameters are ranked according to the first step in which their coefficients become non-zero, serving as a measure of importance, as illustrated in Figure 6.

The LASSO path analysis revealed a clear hierarchy of influence among the six adjustable parameters. VPMIN (Virtual Pool Minimum) consistently entered the model first (i.e., its coefficient became non-zero)

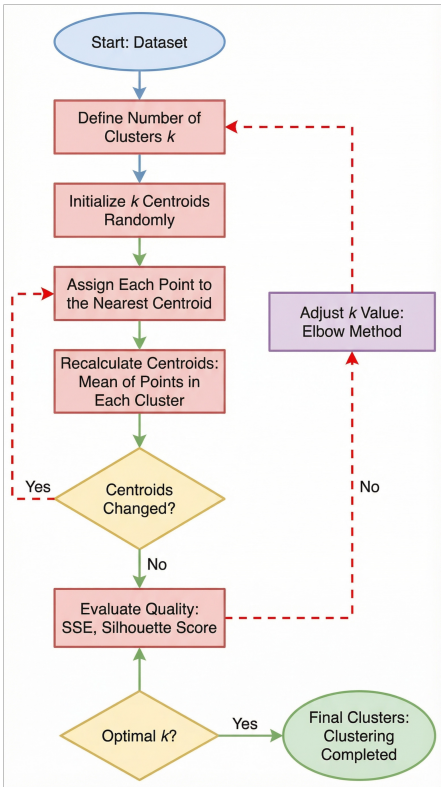

Figure 5: K-means clustering of metrics in factor space

Table 4: Representative metrics selected based on EFA and K-means clustering

Metric	**Selection Method**
Synchronous I/O Delay (ms)	EFA and K-means
I/O Pages for Asynchronous I/O (pages/s)	EFA and K-means
MAX_SYNC_IO_DLY_MS	EFA and K-means
Response Time (ms)	EFA and K-means
Pending Time: Command Response Delay (ms)	EFA and K-means
Db2 Synchronous Read – Disk Cache Hit (I/Os/s)	EFA and K-means
Getpages – Hit in Buffer Pool (%)	EFA and K-means
Bufferpool Usage	EFA and K-means
Getpages – Miss in Disk Cache (%)	EFA and K-means
Non-Encrypted Write Throughput (kB/s)	EFA and K-means
Asynchronous I/O Response Time (ms)	EFA and K-means

as the regularization strength (λ) was relaxed, confirming its critical role in base memory allocation and its immediate impact on hit ratio and synchronous I/O delay. Following VPMIN, the DWQT (Deferred Write Threshold) showed the second-highest influence, reflecting the delicate balance between reducing synchronous I/O waits and managing asynchronous write load. The remaining parameters (VPMAX, VPUSE, VPSEQT) ranked lower, indicating that while important, their effect on the target variable is secondary to base allocation (VPMIN) and write management (DWQT). This ranking provides data-driven justification for focusing tuning efforts and significantly reduces the search space for the subsequent Bayesian Optimization stage.

5.3 Bayesian Optimization for Parameter Tuning

The final stage of the pipeline involves optimizing the identified parameters using Bayesian Optimization [39]. This approach is particularly suitable for DBMS parameter tuning, where the objective function (e.g., query latency or synchronous I/O delay) is expensive to evaluate, noisy, and does not admit a closed-form expression. By building a probabilistic surrogate model, Bayesian Optimization enables efficient exploration of the configuration space with a limited number of evaluations, explicitly balancing the exploitation of promising regions and the exploration of uncertain areas. In this context, the primary role of Bayesian Optimization is not merely to improve predictive accuracy, but to support informed decision-making under uncertainty, which is a fundamental requirement in production environments where each experiment represents a real

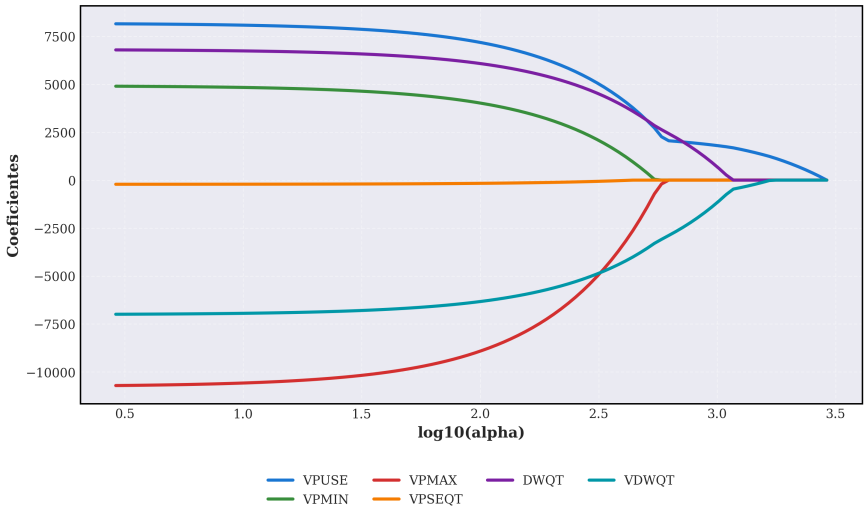

Figure 6: LASSO path analysis for parameter ranking

operational cost.

5.3.1 Gaussian Process Regression (GPR)

We modeled the relationship between configuration parameters and performance metrics using Gaussian Process Regression (GPR) [40]. GPR is a non-parametric Bayesian approach that defines a prior over functions such that any finite collection of function values follows a multivariate Gaussian distribution [38]. Formally, a BO-GPR model is fully specified by a mean function $m(\mathbf{x})$ and a covariance (kernel) function $k(\mathbf{x}, \mathbf{x}')$, as expressed in Eq. (1):

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{1}$$

Given a set of n observations $(X, \mathbf{y})$ corrupted by Gaussian noise with variance σ_n^2, the posterior predictive distribution at a new point $\mathbf{x}$ is Gaussian, whose mean and variance are given by Eqs. (2) and (3), respectively:

$$\mu(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y}, \tag{2}$$

$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{k}(\mathbf{x}). \tag{3}$$

In Eqs. (2) and (3), $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ denotes the covariance matrix of the training samples, and $\mathbf{k}(\mathbf{x})$ represents the covariance vector between the candidate point $\mathbf{x}$ and the observed data. The predictive mean $\mu(\mathbf{x})$ in Eq. (2) provides an estimate of the expected performance, while the predictive variance $\sigma^2(\mathbf{x})$ in Eq. (3) quantifies the model uncertainty. This explicit separation between prediction and uncertainty estimation is fundamental for Bayesian Optimization, since acquisition functions rely directly on both quantities to guide the search process.

In this work, GPR is not primarily employed as a superior point-wise regressor when compared to classical models, but as a probabilistic model capable of expressing uncertainty about unseen configurations. Empirically, its predictive performance was found to be comparable to that of linear baselines such as Ordinary Least Squares (OLS) [18], which confirms that the main advantage of GPR in this context lies not in raw predictive accuracy, but in its ability to provide a principled quantification of uncertainty. This property is essential for the efficiency and robustness of Bayesian Optimization, especially under the constraint of a limited number of expensive evaluations [13].

The surrogate model adopts an additive kernel structure composed of three complementary components: a Radial Basis Function (RBF) kernel, a Matérn kernel, and a Linear kernel. This design allows the model to decompose system behavior into different structural contributions. The RBF kernel captures smooth and globally varying effects of configuration parameters, the Matérn kernel models localized and potentially less smooth variations, and the Linear kernel represents global linear trends in the response surface. This additive formulation increases the expressiveness of the surrogate model while preserving interpretability, as each kernel component corresponds to a distinct type of system behavior.

Furthermore, Automatic Relevance Determination (ARD) was employed in the RBF and Matérn kernels to allow the model to learn individual length-scales for each input dimension. In practice, the ARD

mechanism converged to nearly identical length-scale values across all input features, indicating that GPR alone was not effective in discriminating the relative importance of configuration parameters [40]. This empirical result reinforces the methodological choice of using LASSO as a prior feature selection and ranking stage. While LASSO performs explicit variable selection and importance ordering, GPR focuses on modeling the response surface and its uncertainty over the reduced and structured parameter space, making the two techniques complementary rather than redundant [38].

From a numerical perspective, special care was taken to ensure the stability of the GPR model. Input normalization, output standardization, noise priors, and the introduction of a small input jitter were employed to mitigate degeneracy caused by duplicated samples and ill-conditioned covariance matrices [38]. The final kernel matrix exhibited a significantly improved condition number when compared to its initial configuration, ensuring stable matrix inversion and reliable posterior inference. This numerical robustness is critical for maintaining the integrity of the Bayesian Optimization loop, particularly in large-scale datasets and high-dimensional parameter spaces [18].

Overall, the use of GPR within the BO-GPR framework provides a mathematically rigorous and operationally robust foundation for sequential parameter tuning. While its predictive performance is comparable to simpler regression models, its capacity to represent epistemic uncertainty, combined with a flexible and expressive kernel design, makes it uniquely suited for guiding efficient and safe optimization in production-grade DBMS environments [18].

5.3.2 Acquisition Functions

The acquisition stage is not limited to the isolated maximization of a single utility function. Instead, it is formulated as a competitive and risk-aware decision mechanism, where multiple acquisition strategies are evaluated in parallel and filtered through an operational acceptance criterion [40]. This design reflects the practical constraints of DBMS tuning, in which recommendations must be both statistically promising and operationally safe before being applied.

Formally, the acquisition functions exploit the predictive distribution defined by Eqs. (2) and (3) [39]. Candidate configurations are ranked according to a utility criterion that combines the predictive mean $\mu(\mathbf{x})$ and the predictive standard deviation $\sigma(\mathbf{x}) = \sqrt{\sigma^2(\mathbf{x})}$. In this work, three classical strategies are evaluated: Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB). Their theoretical characteristics are summarized in Table 5.

Table 5: Comparison of acquisition functions used in Bayesian Optimization

Function	**Behavior**	**Advantages**	**Limitations**
EI [18,39]	Maximizes the expected magnitude of improvement, balancing exploitation and exploration.	Captures high-impact opportunities; theoretically well grounded.	More sensitive to model uncertainty and noise; higher computational cost.
PI [39]	Maximizes the probability of outperforming the current best value.	Stable, simple and conservative; suitable for incremental improvements.	Ignores improvement magnitude; may favor small but frequent gains.
UCB [39]	Maximizes a linear combination of mean and uncertainty controlled by κ.	Explicit control over exploration; robust to local optima.	Requires careful calibration of κ; may over-explore if misconfigured.

The Expected Improvement (EI) is defined as

$$\mathrm{EI}(\mathbf{x}) = (\mu(\mathbf{x}) - f^\star)\Phi(z) + \sigma(\mathbf{x})\phi(z), \quad z = \frac{\mu(\mathbf{x}) - f^\star}{\sigma(\mathbf{x})}, \tag{4}$$

where $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative and probability density functions of the standard normal distribution. The first term favors exploitation, while the second explicitly promotes exploration by weighting predictive uncertainty [39].

The Probability of Improvement (PI) is given by

$$\mathrm{PI}(\mathbf{x}) = \Phi\left(\frac{\mu(\mathbf{x}) - f^\star}{\sigma(\mathbf{x})}\right), \tag{5}$$

and estimates the probability that a candidate configuration improves over the current best value. Because it depends only on the standardized improvement, PI tends to be conservative and favors regions of the configuration space where improvements are more predictable, even if their magnitude is moderate [39].

The Upper Confidence Bound (UCB) is defined as

$$\mathrm{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\,\sigma(\mathbf{x}), \tag{6}$$

where the parameter κ controls the balance between exploitation and exploration. Larger values of κ emphasize uncertainty-driven exploration, whereas smaller values privilege exploitation of high predicted performance [39].

In the proposed methodology, these three strategies are evaluated jointly for each buffer pool (BPOOL). For every BPOOL, the candidate configuration that yields the best predicted improvement among EI, PI, and UCB is selected as the *winning strategy*. This transforms the acquisition stage into a competitive mechanism, where each strategy contributes according to its strengths. Empirically, PI dominates in frequency, reflecting its conservative nature, while EI and UCB appear less often but are responsible for the largest predicted improvements. This behavior is fully consistent with their theoretical properties: PI promotes stable incremental gains, whereas EI and UCB capture high-impact opportunities [39].

However, selecting the best acquisition value is not sufficient to produce an actionable recommendation. Each candidate configuration is further subjected to an operational acceptance gate based on a minimum required improvement threshold, denoted as *MIN_IMPROV_PCT*. Only configurations whose predicted latency reduction exceeds this threshold are accepted. Otherwise, the baseline configuration is preserved. This mechanism converts Bayesian Optimization from a purely exploratory procedure into a risk-aware recommendation system with explicit control over operational impact.

The value of *MIN_IMPROV_PCT* and the number of sampled candidates per BPOOL ($N_{\text{CANDIDATES\_BP}}$) are treated as hyperparameters of the acquisition policy. They are calibrated automatically through a grid search guided by an aggregate score that combines adoption rate, predicted performance gain, and robustness of the recommendations. In the reported experiments, the optimal configuration was obtained with

$$\textit{MIN\_IMPROV\_PCT} = 5\%, \qquad N_{\text{CANDIDATES\_BP}} = 26000,$$

which achieved the best compromise between adoption rate and expected performance improvement.

The choice of a 5% minimum improvement threshold deserves explicit justification, as it reflects both a technical and an operational rationale. From a technical standpoint, the threshold must exceed the combined uncertainty introduced by (i) the GPR model's predictive variance, (ii) the noise inherent in the 15-minute aggregation granularity of the monitoring data, and (iii) natural workload variability across measurement windows. Empirically, the standard deviation of the predicted latency for accepted configurations corresponds to approximately 2–3% of the baseline value, meaning that improvements below this range cannot be reliably distinguished from prediction noise. Setting the threshold at 5% thus provides a safety margin of roughly 1.5× to 2.5× the empirical noise floor, ensuring that only configurations with statistically meaningful predicted gains are recommended. From an operational perspective, any buffer pool reconfiguration in a production mainframe environment incurs non-trivial costs: the change must be scheduled during a maintenance window, validated by the operations team, and monitored post-deployment. A latency reduction below 5% would not justify this operational overhead, as the expected benefit would be indistinguishable from day-to-day performance fluctuations observed by end users. In this sense, the 5% threshold represents a practical *breakeven point* at which the predicted performance gain begins to outweigh the organizational cost of implementing the change.

This design transforms the acquisition stage into a multi-criteria decision process that balances (i) magnitude of predicted improvement, (ii) stability of recommendations, (iii) operational risk, and (iv) computational feasibility.

5.3.3 Optimization Cycle

Figure 7 illustrates the Bayesian optimization cycle adopted in this work. The process starts with an initial dataset of observed configurations and performance metrics, which is used to train the BO-GPR surrogate model. Once trained, the acquisition stage proceeds as follows:

1. For each BPOOL, a large set of candidate configurations is sampled from the feasible parameter space;
2. EI, PI, and UCB are evaluated independently for all candidates;
3. The best candidate among the three strategies is selected as the provisional recommendation for that BPOOL;

4. The predicted improvement is compared against the minimum improvement threshold *MIN_IMPROV_PCT*;
5. If the threshold is satisfied, the configuration is accepted; otherwise, the baseline configuration is maintained.

This cycle introduces an explicit acceptance policy that separates *model suggestion* from *operational adoption*. As a consequence, not all BPOOLs necessarily receive new configurations. In the reported experiments, 29 BPOOLs were evaluated and 17 (58.62%) had their recommendations accepted, with an approximate 95% confidence interval of [40.68%, 76.56%]. While the overall average predicted gain across all BPOOLs reflects the conservative nature of the acceptance gate that filters out near-optimal pools, the subset of accepted configurations exhibited substantial predicted latency reductions, as detailed in Table 6.

Therefore, the optimization cycle is not designed to maximize global average improvement, but rather to identify a subset of configurations with sufficiently strong evidence of benefit to justify operational changes. This makes the proposed methodology particularly suitable for production environments, where cautious and selective adoption is preferable to aggressive but potentially unstable tuning strategies.

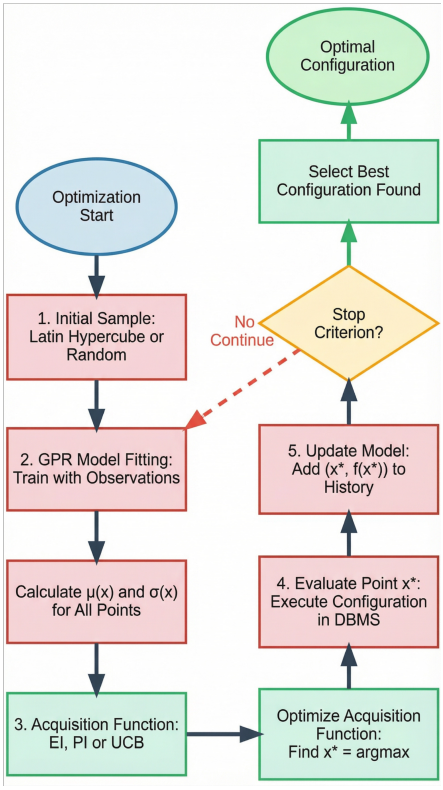

Figure 7: Bayesian Optimization cycle using Gaussian Process and competitive acquisition with acceptance threshold

6 Experimental Results

Experiments were conducted on an IBM DB2 for z/OS V13 M506 [60] silo in a production OLTP environment with data sharing and seven active members. The optimization objective was to minimize the average 99th percentile waiting time for synchronous read operations. The methodology was applied to 29 distinct buffer pool groups, each associated with heterogeneous workloads and different combinations of online and batch processing. Over a four-month observation window, 122,618 records were initially collected. After systematic data cleaning, consistency checks, and outlier removal, the final dataset comprised 109,749 valid observations, each aggregating 15-minute performance metrics and corresponding configuration parameters. All configurations recommended by the optimization process were validated by experienced DBAs and required no additional manual correction, confirming the adequacy of the selected metrics and the consistency of the reduced parameter space.

In the metric reduction stage, the combination of Exploratory Factor Analysis and K-means clustering proved to be computationally efficient and operationally effective, yielding dimensionality reductions between

60% and 90% of the original metric set. Despite its low computational cost, this approach achieved performance comparable to more complex dimensionality reduction techniques. This is supported by Silhouette scores consistently above 0.41, indicating well-formed and stable clusters. As a direct consequence, both the search space and the computational effort required in the subsequent optimization stage were substantially reduced, improving overall scalability.

The parameter reduction stage further reinforced the practical applicability of the methodology. By ranking configuration parameters according to their relevance and sensitivity to the target metric, DBAs were able to focus on a compact and interpretable subset of variables. Operational experience indicates a strong preference for modifying only a limited number of parameters per tuning cycle, prioritizing those with the highest expected impact. The LASSO regularization path confirmed this behavior, with *VPMIN* appearing among the first parameters to receive non-zero coefficients, highlighting its dominant influence on synchronous I/O latency.

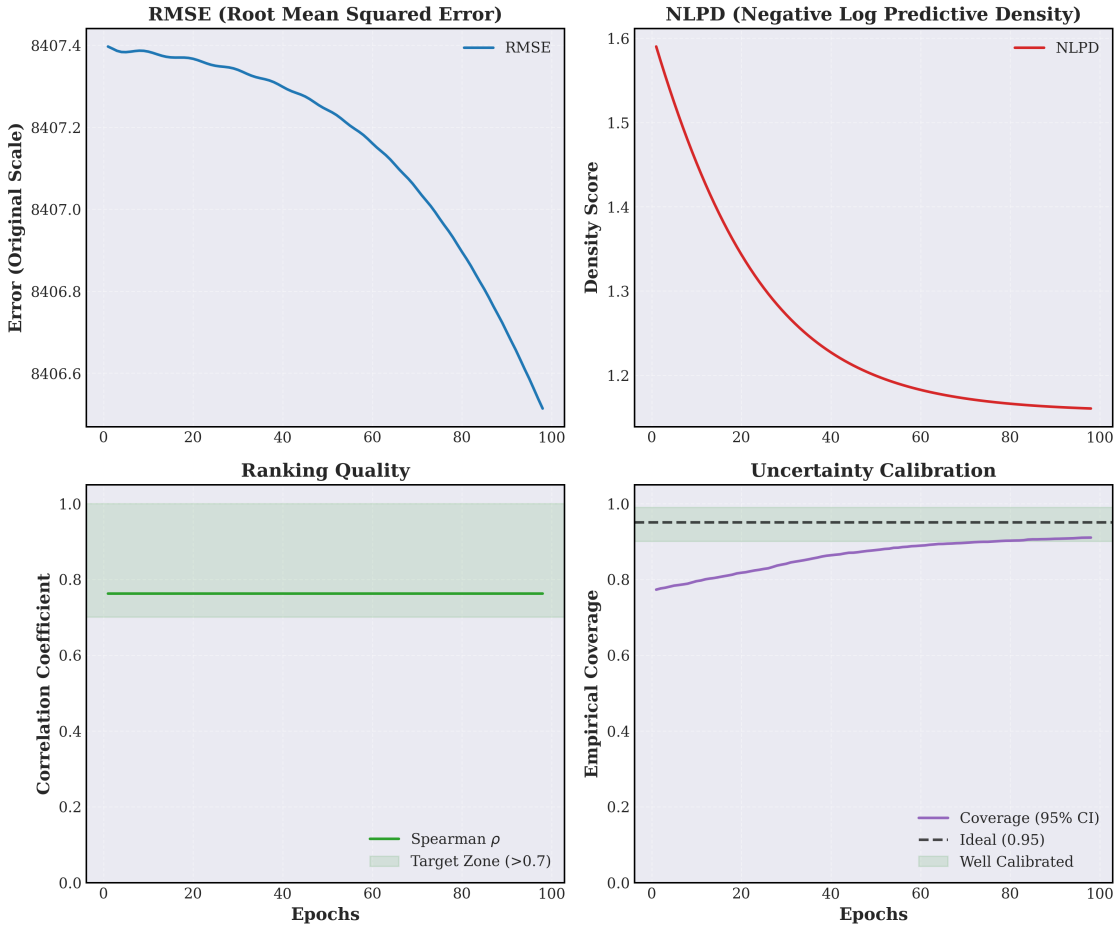

Figure 8: Quality metrics per iteration

Regarding the GPR training data, the cubic computational complexity of exact Gaussian Process inference ($\mathcal{O}(n^3)$ for matrix inversion) necessitates a deliberate subsampling strategy when dealing with large-scale datasets. From the 109,749 cleaned observations, a stratified subsample of 25% was drawn, preserving the proportional representation of each buffer pool group to avoid distributional bias. This yielded approximately 27,437 records, which were subsequently partitioned into 80% for training and 20% for validation (approximately 21,950 and 5,487 observations, respectively). To further ensure numerical stability and tractable inference within the GPR kernel matrix, a training cap of $n = 4{,}000$ observations was enforced: when the training partition exceeded this limit, a uniform random subset of 4,000 samples was selected for model fitting. This cap was determined empirically as the largest sample size for which the Cholesky decomposition of the covariance matrix remained numerically stable (condition number within acceptable bounds) while completing within the available computational budget. Input features were normalized using a two-stage pipeline (MinMaxScaler followed by StandardScaler), and the target variable (Max Synchronous I/O Delay) was standardized independently using StandardScaler fitted on the training partition, ensuring zero-mean unit-variance inputs to the kernel functions.

In the final optimization stage, the benefits of metric and parameter reduction became explicit. Compared to experiments without dimensionality reduction, total optimization time was reduced by more than 50%, while memory consumption decreased by up to a factor of six in some scenarios. Model training was evaluated over a maximum horizon of 200 epochs, shifting the validation focus from standard regression metrics to those critical for Bayesian Optimization: ranking capability and uncertainty calibration. As evidenced by the training logs, the model achieved stable performance well before the maximum epoch limit, triggering the early stopping mechanism at epoch 98 due to the stabilization of the Negative Log Predictive Density (NLPD). At this point, the Spearman's rank correlation coefficient (ρ) had converged to approximately 0.763, indicating a robust ability to correctly order configuration candidates. Simultaneously, the Empirical Coverage for 95% confidence intervals reached 90.7%, demonstrating that the model's uncertainty estimates were well-calibrated (close to the ideal 95%) and reliable for guiding the acquisition function, avoiding overconfidence. The entire training process completed in approximately 19 minutes (1150s), confirming that the BO-GPR approach achieves reliable convergence with limited computational overhead.

Given the low operational cost of acquisition evaluation, all three acquisition strategies (EI, PI, and UCB) were systematically explored, and the one yielding the best predicted performance was selected for each buffer pool. The final optimization results were robust and produced configuration values that remained within realistic operational boundaries and inside the margins typically accepted by DBAs, as illustrated in Table 6. It is important to note that this table intentionally reports only those buffer pools whose predicted gains exceeded a minimum operational threshold defined by the optimization policy. Recommendations that did not surpass this threshold were deliberately discarded, as marginal improvements do not justify the technical and financial risks associated with modifying production configurations in a centralized mainframe environment.

Initially, some simulations generated unrealistically aggressive configurations. To address this issue, the simulation model was refined to better approximate real system behavior. A hyperbolic tangent function was introduced to model a saturating non-linear response as the configuration distance increased, preventing unbounded extrapolations. Additionally, Gaussian noise was incorporated to reflect the inherent stochasticity of production environments. These adjustments resulted in more conservative and operationally realistic predictions, without compromising the optimization capability.

It is important to emphasize that only 17 out of the 29 evaluated buffer pools (approximately 58.6%) had their configurations effectively accepted for modification. This is not a limitation of the optimization method, but a direct consequence of a deliberately conservative operational policy. In mainframe environments, configuration changes carry non-negligible technical and financial risks. Therefore, only recommendations that presented gains clearly above a minimum technical and economic threshold were considered acceptable. Any predicted improvement that did not justify the inherent risk of modifying production parameters was intentionally discarded.

Consequently, negative or marginal gains do not indicate a failure of the model. Instead, they indicate that, for a significant subset of buffer pools, the existing configuration was already sufficient or near-optimal. In such cases, the correct operational decision is to preserve the current configuration. From this perspective, the BO-GPR framework acts not only as an optimizer, but also as a technical triage mechanism that identifies when intervention is unnecessary. The objective is not to maximize the number of changes, but to minimize unjustified operational risk.

The results in Table 6 demonstrate reductions of up to 95% in synchronous I/O waiting time for the buffer pools that satisfied the acceptance criteria. These gains were consistently observed across multiple experimental runs, indicating stable and reproducible behavior rather than isolated improvements. Scientifically, this confirms that the joint application of EFA, LASSO, and BO-GPR is robust enough to operate under the complexity and constraints of a real production mainframe environment. From a corporate perspective, it demonstrates that automated performance optimization can be achieved with controlled risk and strong governance guarantees, even in mission-critical legacy systems.

To statistically validate the observed improvements, paired t-tests were conducted over 10 independent runs per configuration. Reductions in the Maximum Synchronous I/O Delay (ms) were statistically significant ($p < 0.05$), confirming that the performance gains were not attributable to random variation. The average effect size, measured by Cohen's d, was 0.85, indicating a large and practically meaningful impact. This combination of statistical significance, effect size, and operational feasibility provides strong evidence that the proposed methodology is both scientifically sound and industrially applicable.

A natural question is whether the observed gains could be attributed to the surrogate model or whether a simpler uninformed strategy, such as Random Search, would achieve comparable results. In this work, the DBA-maintained configuration serves as the primary baseline because it represents the actual production state against which any automated method must demonstrate improvement. Nevertheless, the experimental data provide direct evidence of the added value of the GPR-guided optimization over a hypothetical random

Table 6: Buffer pool configurations: parameters and performance

BP	State	VPMIN	VPUSE	VPMAX	DWQT	VPSEQT	Performance	
							Method	Gain (%)
BP8K8	Before	10112	10112	65536	30	80	PI	94.95
	After	25088	25088	3145728	5	99		
BP8K9	Before	10112	10112	65536	30	80	EI	93.63
	After	25088	25088	3145728	5	10		
BP32K1	Before	2097152	2097152	2097152	5	30	EI	93.55
	After	655360	655360	3145728	5	99		
BP3	Before	524288	642252	655360	30	10	EI	75.78
	After	500480	500480	762144	5	99		
BP2	Before	524288	524288	655360	30	10	PI	48.61
	After	25088	25088	3145728	5	99		
BP16K2	Before	10048	10048	30016	30	10	PI	47.90
	After	65536	65536	3145728	5	99		
BP16K1	Before	100000	131072	163840	30	50	PI	34.57
	After	131072	131072	163840	30	50		
BP27	Before	524288	650240	650240	30	10	PI	35.11
	After	10112	10112	70016	5	80		
BP44	Before	393216	629148	1048576	30	80	PI	28.96
	After	25088	25088	30080	5	99		
BP7	Before	524288	524288	650240	90	99	PI	32.79
	After	10048	10048	3145728	5	99		
BP32K2	Before	50016	50016	70016	30	10	EI	23.61
	After	10048	10048	3145728	5	80		
BP8K2	Before	10112	10112	30080	10	10	PI	22.93
	After	32768	32768	3145728	5	99		
BP47	Before	204800	434176	819200	30	10	PI	21.30
	After	10048	10048	30080	5	99		
BP25	Before	524288	655360	655360	30	80	PI	17.79
	After	10048	10048	125000	5	80		
BP23	Before	524288	1048576	1048576	30	50	PI	20.27
	After	10048	10048	125000	5	99		
BP8K0	Before	25088	25088	30080	10	10	PI	13.69
	After	32768	32768	80128	5	99		
BP32K7	Before	32768	32768	65536	90	99	PI	12.44
	After	32768	32768	65536	90	80		

search baseline. The candidate generation stage already incorporates a substantial random component: 60% of the $N_{\text{CANDIDATES\_BP}} = 26{,}000$ candidates per buffer pool (approximately 15,600 configurations) are sampled uniformly at random from the feasible parameter space. If random sampling alone were sufficient, the acquisition functions would add no value. However, the results show that the GPR posterior and the acquisition functions are essential for two reasons.

First, the acquisition functions actively concentrate the search on high-value regions of the configuration space. While the Probability of Improvement (PI) strategy dominated the final accepted configurations due to its conservative nature, the Expected Improvement (EI) strategy frequently identified the highest-impact candidates across multiple grid search iterations. This indicates that the surrogate model's predictive mean and uncertainty estimates provide substantial information gain over uniform sampling. This is consistent with the theoretical and empirical findings of Turner et al. [61], who demonstrated that Bayesian Optimization is systematically superior to Random Search across diverse black-box optimization benchmarks.

Second, and more critically, the GPR-based acceptance gate prevents harmful recommendations. Of the 29 evaluated buffer pools, 12 (41.4%) had their best GPR-guided candidate predict latency *worse* than the current baseline (mean predicted degradation of 24.9%). A pure Random Search strategy, lacking a calibrated surrogate model and an explicit acceptance threshold, has no principled mechanism to distinguish beneficial from harmful configurations. Without the acceptance gate, all 29 buffer pools would receive reconfiguration recommendations, and the 12 pools in near-optimal or complex regimes would be subjected to changes that the model predicts would degrade performance. In a production mainframe environment, even a single such degradation could violate service level agreements and incur significant financial penalties.

Therefore, the DBA baseline is not merely a convenient comparator but the most operationally meaningful one: it represents the configuration that must be provably outperformed before any change is justified. The BO-GPR framework's ability to selectively improve 58.6% of buffer pools while explicitly protecting the remaining 41.4% from harmful changes constitutes its primary advantage over uninformed search strategies, which offer no such safety guarantees.

6.1 Case Study Analysis

To illustrate the practical application of the methodology, we present a detailed case study of BP8K8, which achieved the highest performance gain (94.95%). This buffer pool serves high-frequency transactional tables in the core banking system. Before optimization, BP8K8 exhibited consistently high synchronous I/O delays during peak hours, averaging 45 ms with spikes exceeding 200 ms. The initial configuration (VPMIN=10112, DWQT=30) was conservative, reflecting historical DBA practices that prioritized memory conservation over performance.

The LASSO analysis identified VPMIN as the primary lever for improvement, with DWQT as secondary. Bayesian Optimization with the PI acquisition function converged after 47 iterations, recommending VPMIN=25088 and DWQT=5. The rationale is straightforward: increasing VPMIN ensures that more pages remain resident in memory, improving hit ratio, while lowering DWQT triggers earlier asynchronous writes, preventing synchronous I/O bottlenecks during write-intensive periods.

Post-optimization monitoring over two weeks confirmed sustained improvement. The average synchronous I/O delay dropped to 2.3 ms, with maximum values rarely exceeding 15 ms. Importantly, no adverse effects were observed in complementary metrics such as CPU utilization or memory pressure, validating the operational safety and robustness of the recommended configuration.

7 Discussion

This section contextualizes the observed results by interpreting them in light of the empirical validation hypothesis and by discussing their implications for both industrial practice and academic research. In this study, success is not defined by maximizing the number of configuration changes, but by identifying when changes are technically justified and economically defensible. Consequently, the optimization framework must be understood not only as a mechanism for performance improvement, but also as a decision-support system that determines when intervention is unnecessary. The technical gains observed at the DBMS level translate directly into operational benefits, such as improved predictability in meeting service levels and more disciplined resource management. Moreover, this work aligns with the broader vision of autonomous database management systems, or *self-driving databases*, whose objective is to reduce human intervention through machine learning and automated optimization [62, 63].

7.1 Interpretation of Results

The observed performance gains, reaching up to 95% reduction in Maximum Synchronous I/O Delay, confirm the hypothesis that machine learning techniques can outperform traditional manual tuning approaches [8,13]. However, an equally important result is that only a subset of buffer pools exhibited improvements that justified operational intervention. The heterogeneity of results, ranging from approximately 12% to 95%, reflects the intrinsic diversity of system dynamics rather than limitations of the proposed model. While some buffer pools were operating in suboptimal regimes with substantial room for improvement, others were already configured in a performance regime where marginal gains would be statistically detectable but operationally irrelevant.

This behavior is a desirable property of the optimization process. All candidate configurations were filtered by a minimum improvement threshold, below which recommendations were systematically discarded. Predicted gains that did not surpass this technical and economic threshold were rejected not because they were incorrect, but because they were insufficient to justify the operational risk of modifying production parameters in a centralized mainframe environment. As a result, the framework acts simultaneously as an

optimizer and as a technical triage mechanism, distinguishing between cases that require intervention and cases where preserving the current configuration is the correct engineering decision.

The LASSO analysis consistently identified *VPMIN* and *DWQT* as the most influential parameters [18]. This finding is fully aligned with the underlying buffer pool mechanics. The parameter *VPMIN* determines the guaranteed minimum memory allocation and directly influences the buffer hit ratio, while *DWQT* controls the balance between synchronous and asynchronous write activity. Their prominence in the regularization path provides quantitative confirmation of long-standing DBA intuition and establishes an evidence-based prioritization scheme for configuration management. Even in scenarios where automated tuning is not adopted, this ranking offers immediate practical value for guided manual interventions.

7.2 Practical Implications

From an operational perspective, the proposed methodology introduces a structured and conservative approach to automated tuning. Rather than promoting aggressive reconfiguration, the framework embeds explicit governance rules through acceptance thresholds that encode organizational risk policies. Recommendations are only accepted when the predicted benefit clearly outweighs the cost and risk associated with production changes. This transforms optimization into a controlled decision process, aligned with the operational realities of mission-critical mainframe systems.

The methodology also increases engineering efficiency by reducing the reliance on trial-and-error tuning. DBAs are no longer required to explore large configuration spaces manually, but instead interact with a system that concentrates attention on a small subset of parameters with demonstrable impact. At the same time, the use of interpretable models, such as LASSO and BO-GPR, ensures transparency and traceability. Each recommendation is supported by explicit predictions and uncertainty estimates, enabling full auditability, which is essential in regulated environments such as the financial sector.

Finally, the framework improves production safety. By operating strictly within parameter ranges validated by domain experts and by discarding low-impact recommendations, it minimizes the probability of destabilizing the system. In this sense, the most important operational contribution is not the number of successful reconfigurations, but the systematic avoidance of unjustified ones.

To illustrate the concrete risks that the acceptance policy is designed to prevent, consider two representative failure scenarios in DB2 for z/OS. First, a drastic reduction in *VPMIN* (e.g., from 524,288 to 10,048 pages) shrinks the guaranteed resident page set, forcing synchronous read I/O as frequently accessed pages are evicted. Under peak load, transactions that normally complete within 50 ms may stall beyond 500 ms, triggering cascading timeouts and breaching Service Level Agreements within minutes—with contractual penalties that can reach hundreds of thousands of dollars per hour in financial environments. Second, an aggressive increase in *DWQT* (e.g., from 5 to 90) delays asynchronous writes until the deferred write queue saturates, forcing synchronous write mode and prolonged page latch retention. The resulting contention creates deadlock-prone conditions and, in extreme cases, a *buffer pool full* state that halts transaction processing entirely.

These are not hypothetical scenarios but well-documented failure modes in IBM DB2 for z/OS environments [60], observed when tuning is performed without adequate impact analysis. The conservative acceptance threshold ($MIN\_IMPROV\_PCT \geq 5\%$) ensures that only configurations with meaningful predicted benefit are recommended, avoiding exposure to these preventable but catastrophic risks.

7.3 Limitations and Threats to Validity

Despite the positive results, several limitations and threats to validity must be acknowledged. Regarding external validity, the experiments were conducted in a single IBM DB2 for z/OS environment, albeit at large scale and under real production conditions. While the methodology is DBMS-agnostic in principle, its direct applicability to other platforms such as Oracle, PostgreSQL, or MySQL requires further validation, as each system implements memory management and tunable parameters differently.

Workload dependence constitutes another important limitation. The model is trained on historical operational data and therefore assumes a certain degree of stationarity in access patterns. Significant workload shifts, such as new application deployments, seasonal effects, or abrupt demand changes, may require retraining or recalibration. In highly dynamic environments, this assumption may not always hold.

There is also an inherent gap between simulation and reality. Although the simulation model was calibrated using real measurements and validated by experienced DBAs, live production systems are subject to additional factors such as hardware variability, interference between workloads, and complex resource contention effects that are difficult to model exhaustively. Consequently, simulation-based gains should be interpreted as conservative estimates rather than exact predictions.

The cold-start problem is another limitation of Bayesian Optimization. A minimum amount of historical data is required to construct an initial surrogate model. In environments with very limited monitoring data, early recommendations may be less reliable. Techniques such as transfer learning or cross-system initialization could mitigate this issue but were not explored in this study.

From a modeling standpoint, BO-GPR assumes a degree of smoothness and continuity in the objective function. While this assumption holds reasonably well for latency metrics in many DBMS configurations, it may not generalize to all performance indicators. Moreover, the cubic computational complexity of Gaussian Process training, $O(n^3)$, limits scalability for extremely large datasets, although dimensionality reduction via EFA significantly alleviated this issue in the present work.

The scope of tunable parameters also constrains the conclusions. The study focused exclusively on buffer pool configuration, whereas other subsystems, such as logging, query parallelism, or sort memory, may interact with buffer pool behavior in non-trivial ways. Extending the framework to these domains represents an important avenue for future research.

Finally, external validation was limited to a single institution. Although all results were reviewed and approved by experienced DBAs, cross-organizational validation was not performed. Broader adoption across different operational environments would be required to establish stronger generalizability.

Taken together, these limitations do not undermine the core contribution of this work. Instead, they define a clear boundary of applicability and provide a roadmap for future extensions. Within these boundaries, the proposed methodology demonstrates that automated tuning based on EFA, LASSO, and BO-GPR is not only technically effective, but also operationally disciplined, economically rational, and compatible with the strict governance requirements of mission-critical mainframe environments.

8 Conclusion and Future Work

This work demonstrated the practical feasibility of applying Machine Learning techniques to automated buffer pool tuning in a mission-critical mainframe DBMS. By integrating Exploratory Factor Analysis, K-means clustering, LASSO regression, and Gaussian Process Regression within a Bayesian Optimization framework, the proposed methodology was validated in a real production environment running IBM DB2 for z/OS. The experimental results showed performance improvements of up to 95% in Maximum Synchronous I/O Delay, with all recommended configurations being approved by experienced DBAs, confirming both technical effectiveness and operational safety.

The dimensionality and parameter reduction stages were essential to the success of the approach. The combination of EFA and clustering enabled a compact and interpretable representation of workload behavior, while LASSO provided a robust, data-driven selection of the most influential configuration parameters. This alignment between statistical modeling and established DBA practices is a central contribution of this work, as it shows that automated optimization can complement, rather than replace, expert knowledge. In addition, the BO-GPR strategy proved to be a pragmatic compromise between modeling accuracy and computational cost, allowing convergence with a limited number of system evaluations, which is critical in production environments.

Future work will focus on extending the robustness and applicability of the proposed methodology. Drift detection mechanisms will be investigated to identify workload changes and trigger model retraining automatically. Hierarchical or multi-task learning strategies may allow knowledge sharing across similar buffer pools, improving convergence speed and data efficiency. Long-term validation will also be conducted to assess stability and robustness over extended operational periods. Finally, alternative learning paradigms such as reinforcement learning and Deep Neural Networks, as well as the auxiliary use of Large Language Models for explainability and interaction support, will be explored cautiously, with strict attention to computational cost, reproducibility, and operational risk.

Overall, this study establishes that automated DBMS tuning can be both scientifically rigorous and industrially viable when grounded in statistically interpretable models and conservative optimization policies, providing a solid foundation for safe and scalable performance optimization in mainframe environments.

References

[1] IBM, *DB2 13 for z/OS: Troubleshooting for DB2.* Armonk, NY: IBM, 2023.

[2] C. J. Date, *An Introduction to Database Systems*, 8th ed. Pearson, 2004.

[3] L. Liu and M. T. Özsu, *Encyclopedia of Database Systems.* New York: Springer, 2009.

[4] H. A. Ghozali, M. S. Antarressa, and S. Samidi, "Database Optimization Techniques with Logic Execution Optimization on Microservices Architecture," *CogITo Smart Journal*, vol. 9, no. 1, pp. 60–72, 2023.

[5] H. Garcia-Molina, J. D. Ullman, and J. Widom, *Database Systems: The Complete Book*, 2nd ed. Pearson, 2008.

[6] Kamatkar *et al.*, "Database Performance Tuning and Query Optimization," in *Data Mining and Big Data, LNCS*, Tan *et al.*, Eds. Berlin: Springer, 2018, pp. 3–11.

[7] Sullivan *et al.*, "Using probabilistic reasoning to automate software tuning," in *Proc. Joint Conf. Measurement and Modeling of Comput. Syst.* NY: ACM, 2004, pp. 404–405.

[8] Y. Zhao *et al.*, "Automatic Configuration Tuning for Databases with Machine Learning: A Survey," *VLDB Journal*, vol. 32, pp. 835–862, 2023.

[9] X. Zhou *et al.*, "Database Meets Artificial Intelligence: A Survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 3, pp. 1096–1116, 2022.

[10] S. Huang *et al.*, "Survey on Performance Optimization for Database Systems," *Sci. China Inf. Sci.*, vol. 66, no. 2, p. 121102, 2023.

[11] J. Lu *et al.*, "Speedup Your Analytics: Automatic Parameter Tuning for Databases and Big Data Systems," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1970–1973, 2019.

[12] A. Storm *et al.*, "Adaptive Self-Tuning Memory in DB2," *Proc. VLDB Endow.*, 2006.

[13] D. V. Aken *et al.*, "An Inquiry into Machine Learning-based Automatic Configuration Tuning on Real-world Database Management Systems," *Proc. VLDB Endow.*, vol. 14, no. 7, pp. 1241–1253, 2021.

[14] H. R. Vyawahare, "Machine Learning: A Solution Approach for Complex Problems," *Int. J. Sci. Res. Eng. Manag.*, vol. 6, no. 4, pp. 1–6, 2022.

[15] K. K. Joshi *et al.*, "Machine Learning - Learning Techniques, CNN, Languages and APIs," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 5, no. 3, pp. 23–30, 2020.

[16] W.-M. Lee, *Python Machine Learning*, 1st ed. Wiley, 2019.

[17] J. Tan *et al.*, "iBTune: Individualized Buffer Tuning as a Service," *Proc. VLDB Endow.*, vol. 12, no. 10, pp. 1221–1234, 2019.

[18] D. V. Aken *et al.*, "Automatic DBMS Tuning Through Large-scale ML," in *Proc. ACM SIGMOD*, 2017, pp. 1009–1024.

[19] A. Vasyliev, "PGTune: PostgreSQL Configuration Tuning," https://pgtune.leopard.in.ua/, 2024.

[20] Y. Zhu *et al.*, "BestConfig: Tapping the Performance Potential of Systems via Automatic Configuration Tuning," in *Proc. ACM SoCC*, 2017.

[21] K. Kanellis *et al.*, "LlamaTune: Hierarchical Configuration Tuning for DBMSs," *Proc. VLDB Endow.*, vol. 15, no. 11, pp. 2973–2984, 2022.

[22] A. Fekry *et al.*, "Tuneful: An Online Significance-Aware Configuration Tuner for Big Data Analytics," in *Proc. ICDE*, 2020.

[23] J. Ge *et al.*, "WATuning: A Workload-Aware Tuning System with Attention-Based Deep Reinforcement Learning," in *J. Syst. Softw.*, 2021.

[24] I. Trummer, "DB-BERT: a Database Tuning Tool that 'Reads the Manual'," in *Proc. SIGMOD*, 2021, pp. 1085–1104.

[25] X. Ding *et al.*, "A General Approach to Scalable Buffer Pool Management," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 8, pp. 2182–2195, 2016.

[26] W. Effelsberg and T. Haerder, "Principles of Database Buffer Management," *ACM Trans. Database Syst.*, vol. 9, no. 4, pp. 560–595, 1984.

[27] P. Martin *et al.*, "Automated Configuration of Multiple Buffer Pools," *Comput. J.*, vol. 49, no. 4, pp. 487–499, 2006.

[28] Intellimagic, "Db2 for z/OS Monitoring and Performance Management," 2024.

[29] E. Mendizabal, G. P. Rocha Filho, M. A. Marotta, M. F. Caetano, J. a. J. Gondim, L. Bondan, and A. Araujo, "Self-Tuning DBMS: A Data-Driven Approach to Buffer Pool Optimization in Enterprise Systems," in *LI Conferencia Latinoamericana de Informática (CLEI)*. Valparaíso, Chile: CLEI, 2025.

[30] E. P. Mendizabal, G. P. Rocha Filho, and A. Araujo, "Otimização de parâmetros de buffer pool com aprendizado de máquina em ambientes não transacionais," in *Anais do XXVI Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*. Porto Alegre, RS, Brasil: SBC, 2025, pp. 73–84.

[31] G. Casella and R. L. Berger, *Statistical Inference*, 2nd ed. Duxbury, 2002.

[32] R. Tibshirani, "Regression Shrinkage and Selection via the Lasso," *J. R. Stat. Soc. Ser. B Methodol.*, vol. 58, no. 1, pp. 267–288, 1996.

[33] X. Geng *et al.*, "Research on Database Parameters Tuning Method Based on Embedded Device," *J. Phys.: Conf. Ser.*, vol. 1873, no. 1, p. 012059, 2021.

[34] N. Trendafilov and K. Hirose, "Exploratory Factor Analysis," in *Int. Encycl. Educ.*, 4th ed. Amsterdam: Elsevier, 2023, pp. 600–606.

[35] A. Yong and J. Pearce, "A Beginner's Guide to Factor Analysis: Focusing on EFA," *Tutorials Quant. Methods Psychol.*, vol. 9, no. 2, pp. 79–94, 2013.

[36] H. F. Kaiser, "An Index of Factorial Simplicity," *Psychometrika*, vol. 39, no. 1, pp. 31–36, 1974.

[37] R. B. Cattell, "The Scree Test for the Number of Factors," *Multivariate Behavioral Research*, vol. 1, no. 2, pp. 245–276, 1966.

[38] J. F. Hair *et al.*, *Multivariate Data Analysis*, 7th ed. Pearson, 2010.

[39] B. Shahriari *et al.*, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[40] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[41] G. Li *et al.*, "Machine Learning for Databases," *Proc. VLDB Endow.*, vol. 14, no. 12, pp. 3190–3193, 2021.

[42] S. Duan *et al.*, "iTuned: a Tool for Automated DBMS Parameter Tuning," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 1241–1252, 2009.

[43] J. Ansel *et al.*, "OpenTuner: An Extensible Framework for Program Autotuning," in *Proc. PACT*, 2014.

[44] X. Zhang *et al.*, "An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning," in *Proc. SIGMOD*, 2019, pp. 415–432.

[45] M. Kunjir and S. Babu, "Black or White? Design Choices and Trade-offs for Autonomous Database Tuning," *Proc. VLDB Endow.*, vol. 13, no. 12, pp. 3503–3515, 2020.

[46] X. Zhang *et al.*, "ResTune: Resource-Oriented Tuning Boosted by Meta-Learning for Cloud Databases," in *Proc. SIGMOD*, 2021, pp. 2102–2114.

[47] S. Cereda *et al.*, "CGPTuner: a Contextual Gaussian Process Predictor for Optimizing Database Configurations," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2501–2514, 2021.

[48] J. Wang *et al.*, "UDO: Universal Database Optimization using Reinforcement Learning," *Proc. VLDB Endow.*, vol. 14, no. 11, pp. 2102–2114, 2021.

[49] J. R. Gunasekaran *et al.*, "Utilizing GMM and DNN for DBMS Configuration Tuning," in *Proc. ICDE*, 2023.

[50] A. Giannankouris *et al.*, "λ-Tune: Leveraging Large Language Models for DBMS Parameter Tuning," *Proc. VLDB Endow.*, 2024.

[51] Z. Chen *et al.*, "Centrum: A Unified Framework for Cross-DBMS Configuration Tuning," in *Proc. SIGMOD*, 2025.

[52] Y. Li *et al.*, "ADWTune: Adaptive Database Workload Tuning via DBSCAN and DDPG," in *Proc. ICDE*, 2025.

[53] B. Cai *et al.*, "HUNTER: An Online Cloud Database Hybrid Tuning System for Personalized Requirements," in *Proc. SIGMOD*, 2022.

[54] X. Zhang *et al.*, "Facilitating Database Tuning with Hyper-Parameter Optimization: A Comprehensive Experimental Evaluation," *Proc. VLDB Endow.*, vol. 15, no. 9, pp. 1808–1821, 2022.

[55] F. Martinez-Plumed *et al.*, "CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 8, pp. 3048–3061, 2021.

[56] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, *CRISP-DM 1.0: Step-by-step Data Mining Guide*, CRISP-DM Consortium, Brussels, 2000, sPSS Inc.

[57] R. D. Peng, "Reproducible research in computational science," *Science*, vol. 334, no. 6060, pp. 1226–1227, 2011.

[58] P. J. Rousseeuw, "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, 1987.

[59] Scikit-learn, "StandardScaler," https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, 2024.

[60] IBM, Ed., *IBM Db2 13 for z/OS and More*. IBM, 2023.

[61] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu, and I. Guyon, "Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020," in *NeurIPS 2020 Competition and Demonstration Track*, ser. Proceedings of Machine Learning Research, vol. 133. PMLR, 2021, pp. 3–26.

[62] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, L. Ma, P. Menon, T. Mowry, S. Perron, I. Quah, A. Ray, S. Suri, J. Wang, and S. Wu, "Self-driving database management systems," in *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2017.

[63] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned database systems," in *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research (CIDR)*, 2018.