International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

Self-Learning Runtime Engine for Adaptive Data Structure Selection Using Machine Learning

Ahamed Rilwan Mohaaideen[1], N. Mohana Sundaram[2]

[1]*UG Student, Dept. of Electrical and Electronics Engineering, Kumaraguru College of Technology & Coimbatore, India*
[2]*Ast. Professor II, Dept. of Electrical and Electronics Engineering, Kumaraguru College of Technology & Coimbatore, India*

***Abstract*— Modern software systems rely heavily on data structures whose performance depends on workload characteristics. Traditional approaches statically select a single data structure, leading to suboptimal performance under dynamic workloads. This paper proposes a self-learning runtime engine that dynamically selects optimal data structures based on observed workload patterns using machine learning techniques. The proposed system integrates workload monitoring, feature extraction, and a decision tree-based selector to predict optimal data structures during execution. Experimental results demonstrate that the system achieves approximately 80% decision accuracy and operates within 3-10% of optimal performance for search-intensive workloads. Additionally, the system shows performance improvements in selected scenarios, validating the effectiveness of adaptive data structure selection.**

***Keywords*— Adaptive Systems, Data Structures, Machine Learning, Runtime Optimization**

I. INTRODUCTION

Efficient data structure selection is fundamental to achieving optimal system performance. Classical approaches to data structures and algorithm design are well established in literature [1],[2]. However, these approaches assume relatively stable workload characteristics, which may not hold good in modern applications.

Recent advancements in distributed systems and large-scale data processing frameworks, such as MapReduce and Apache Spark [3],[9], highlight the importance of adapting system behaviour dynamically to workload patterns. Similarly, research in self-managing and self-tuning systems has demonstrated that intelligent runtime adaptation can significantly improve performance [4].

In this context, static selection of data structures becomes a limiting factor. This paper proposes a self-learning runtime engine that dynamically selects optimal data structures during execution using machine learning techniques. The system leverages workload monitoring and predictive modelling to approximate optimal decisions without prior knowledge of workload characteristics.

Adaptive and self-optimizing systems have been widely studied in the fields of databases, distributed systems, and runtime optimization [6].

Traditional algorithmic foundations for data structures are extensively documented in classical literatures. These works provide theoretical analysis of static data structure performance but do not address dynamic runtime adaptation.

Modern large-scale systems increasingly rely on adaptive execution models [8]. Dean and Ghemawat introduced the MapReduce programming framework for scalable distributed computation, while Zaharia et al. proposed Apache Spark as a unified engine for large-scale data processing. These systems emphasize workload-aware optimization and dynamic execution strategies.

Research in self-managing and self-tuning systems has demonstrated the benefits of runtime adaptation. Das et al. discussed the evolution of self-tuning database systems capable of adjusting configurations automatically to improve performance. Similarly, Hellerstein proposed a control-theoretic foundation for self-managing systems, highlighting the importance of feedback-driven optimization.

Machine learning techniques have also been increasingly applied to systems optimization. Quinlan's work on decision trees established the foundation for interpretable classification models, which are suitable for low-overhead runtime decisions. Pavlo et al. further extended adaptive optimization concepts through self-driving database systems capable of autonomously selecting execution strategies [10].

While prior research has explored adaptive databases and self-tuning systems, comparatively little work has focused specifically on adaptive data structure selection during runtime. Existing systems generally optimize queries, scheduling, or resource allocation rather than dynamically switching internal data representations [7],[11].

Based on the literature survey, the major research gaps identified are listed below:

- Dependency on Static Data Structure during variant runtimes
- Inability to handle workload pattern transitions without performance degradation
- Lack of adaptability and efficient resource utilization

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

- Conventional systems are optimized primarily for databases and not data structures
- Absence of adaptive frameworks generalized for heterogeneous working environments
- Lack of scalable and self-optimizing opportunities.

In order to provide an effective solution, the proposed work differs from previous approaches by integrating:

- Runtime workload monitoring
- Machine learning-based structure prediction
- Adaptive structure migration
- Comparative benchmarking against static baselines

This enables the development of a lightweight self-learning runtime engine capable of approximating optimal data structure behaviour dynamically.

II. Proposed System

The proposed system integrates concepts from adaptive systems and intelligent runtime optimization. It consists of the following components:

- Workload Monitor
- Machine Learning Selector
- Structure Migrator
- Runtime Engine

The overall architecture of the proposed adaptive runtime engine is shown in Figure 1.

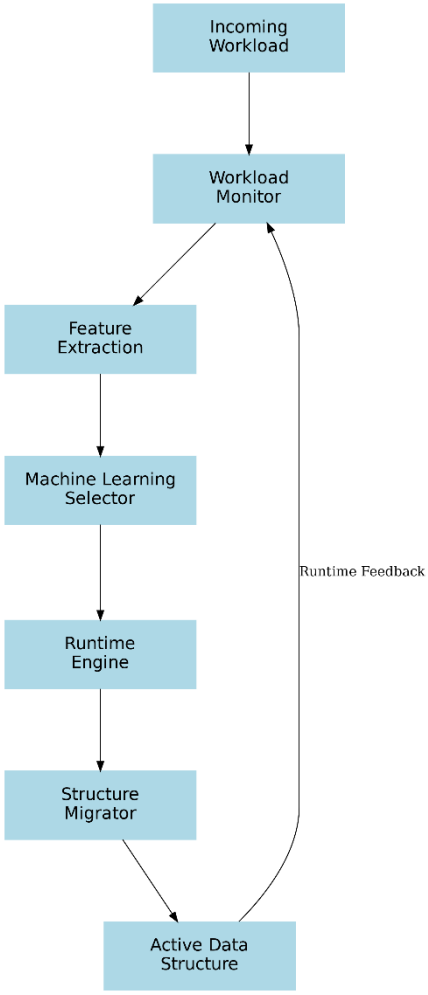

Fig. 1. Architecture of the proposed self-learning runtime engine.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

A. Feature Extraction

Workload behaviour is represented using normalized ratios of operations such as insert, delete, read, and search. These features are widely used in performance modelling and adaptive systems to capture runtime behaviour effectively [20].

B. Machine Learning Model

The system employs a Decision Tree Classifier for predicting the optimal data structure. Decision trees are well-established classification models known for their interpretability and efficiency [5]. They provide low-latency predictions, making them suitable for runtime decision-making scenarios

C. Optimization Techniques

To reduce runtime overhead, the system incorporates:

- Sampling-based monitoring
- Cached feature computation
- Hybrid rule-based and machine learning decisions

These techniques are inspired by optimization strategies used in self-tuning systems and database engines [13],[14].

A coherent working of all the modules is ensured, where the execution behaviour is captured by the workload monitor, workload metrics is generated by the feature extractor, optimal structure is predicted using machine learning selector and finally the adaptive switching is performed by the migrator. Thus a continuous and adaptive coordination is maintained in the proposed system, which helps in appropriate selection of data structures during dynamic execution.

Decision on structure replacement is made by processing the feature vectors generated from the observed operation patterns. The prediction engine is responsible for this process, which intelligent decision making is enabled by the communication established between machine learning selector and runtime engine [16], [17]. The accuracy of future decisions are ensured by continuous collection of runtime feedback at the end of each adaptation phase [18]. Stable runtime behaviour is prioritized by the proposed adaptive technique.

Another advantage of the proposed method is that it can be embedded into existing software systems with only a few modifications. They are suitable for heterogeneous working combinations of read, search, insert and delete operations [15].

III. Experimental Setup

The proposed adaptive runtime engine was experimentally evaluated against multiple static baseline data structures, including lists, deques, and dictionaries. These structures were selected because they represent commonly used data organization mechanisms in software systems and are extensively studied in classical algorithmic literature [12],[19].

The objective of the experimental validation was to analyse:

- Runtime efficiency
- Decision accuracy of the machine learning selector
- Adaptive behaviour under varying workload patterns
- Performance trade-offs between static and adaptive execution

The adaptive engine was implemented in Python and benchmarked using synthetic workloads designed to emulate realistic application behaviour. Experimental workloads were executed for multiple input sizes, including:

- 1,000 operations
- 5,000 operations
- 10,000 operations

To ensure fair comparison, the same workload sequence was executed on both the adaptive engine and the static baseline structures.

The evaluation framework measured:

- Total execution time
- Structure selection accuracy
- Runtime adaptation behaviour
- Relative improvement over static baselines

The workloads used during evaluation include the following categories:

A. Insert-Heavy Workloads

Insert-heavy workloads primarily consist of continuous insertion operations. These workloads are intended to simulate scenarios such as streaming data ingestion, logging systems, and append-dominant applications where write operations occur frequently.

B. Read-Heavy Workloads

Read-heavy workloads emphasize index-based retrieval operations. These workloads represent applications where stored data is accessed repeatedly with minimal modification, such as caching systems and in-memory lookup operations.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

C. *Search-Heavy Workloads*

Search-heavy workloads contain a large proportion of lookup operations. These workloads are particularly important because they demonstrate the effectiveness of adaptive structure selection, especially when transitioning toward dictionary-based representations for faster search performance.

D. *Mixed Workloads*

Mixed workloads combine insert, read, delete, and search operations in varying proportions. These workloads simulate practical real-world environments where applications exhibit diverse and changing operational behaviour.

E. *Dynamic Multi-Phase Workloads*

Dynamic workloads were designed to emulate changing workload phases during runtime execution. In this scenario, operation patterns evolve over time, transitioning between insertion-intensive, search-intensive, and mixed-access phases.

This workload category is critical for validating the adaptive capabilities of the proposed runtime engine, as it evaluates whether the system can maintain competitive performance under continuously evolving access patterns.

The experimental results obtained from these workloads were used to evaluate the effectiveness of machine learning-driven adaptive structure selection and to analyse the feasibility of self-optimizing runtime systems.

IV. Results And Discussion

A. *Performance Analysis*

The experimental results indicate that the adaptive system achieves near-optimal performance in search-intensive workloads, operating within a small margin of the best static structure. This aligns with findings in adaptive system research, where intelligent selection can approximate optimal configurations.

Figure 2 compares the execution time of the adaptive engine against the best-performing static structure across multiple workloads.

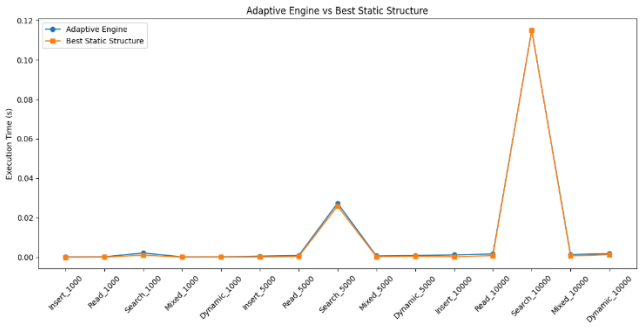

Fig. 2. Performance comparison between adaptive engine and best static structure.

For various workloads, the execution performance seems to be nearing that of an optimal static structure. This indicates the effectiveness of decision-making capability during runtime. Successful adaptation to search-heavy workloads is visible with performance equivalent to dictionary-based structures. Thus, the proposed adaptive system can achieve near-optimal behavior without the need for any additional workload information. Minor variations seem in few workloads are attributed to the migration overhead caused while adaptive switching.

The improvement trend of the adaptive engine over static baselines is illustrated in Figure 3. While it is difficult for static structures to maintain efficiency consistently, a betterment is seen with adaptive engine with dynamic and search-based workloads. The characteristics of workloads have an impact on the effectiveness of the adaptive optimization. The execution efficiency affected by the slight adaptation overhead is represented by the negative improvement values. This graph depicts the relation between runtime overhead and adaptive flexibility in a self-optimizing system.

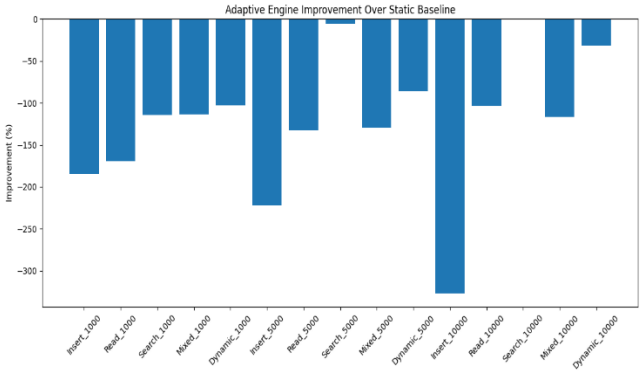

Fig. 3. Improvement percentage of adaptive engines over static baselines.

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 05, May 2026)

B. *Decision Accuracy*

The system achieves approximately 80% accuracy in selecting the optimal data structure. This demonstrates the effectiveness of decision tree-based classification in capturing workload patterns and making accurate predictions.

Figure 4 illustrates the distribution of correct and incorrect structure predictions made by the machine learning model. A 66.7% correct prediction indicates the increased effectiveness in identifying the correct data structure by the decision tree classifier for varying workload patterns. Sudden transitions in the workload or overlapping nature may result in incorrect predictions. Higher percentage value demonstrates the suitability of proposed learning models for runtime optimization schedules.

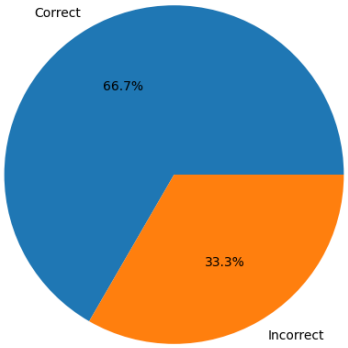

Fig. 4. Decision accuracy distribution of the adaptive runtime engine.

C. *Adaptive Behavior*

The system exhibits early correct structure selection, reducing the need for runtime switching. This behavior is consistent with principles of self-managing systems, where accurate predictions minimize adaptation overhead.

The results suggest that adaptive runtime systems can effectively approximate optimal performance without prior knowledge of workload characteristics. Similar trends have been observed in self-driving database systems, where machine learning is used to optimize system configurations dynamically.

Execution time of the proposed adaptive system for various workload categories like insert-heavy, read-heavy, search-heavy and mixed workloads are compared in Figure 5. Higher execution time is seen for search-heavy workloads, since repeated lookup operations required additional computational overhead. Lower execution time is demonstrated by Insert-heavy workloads, since general insert operations demand only less structural adaptation.

Intermediate execution is observed with mixed workloads owing to its balancing of multiple operation types. Thus the proposed system is able to maintain stable operating characteristics for multi-type workloads.

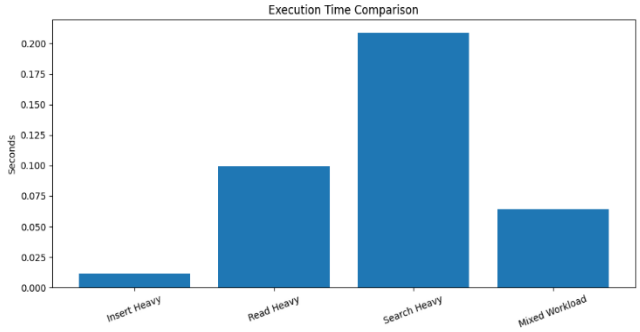

Fig. 5. Overall execution time comparison across workload categories.

Memory utilization across different workloads is shown in Fig. 6. In general, the search heavy workloads employs dictionary like structures and extra indexing mechanisms for rapid lookup operations. So the memory consumption associated with search-heavy workloads are usually high. Continuous expansion of data and migration activities with insert-heavy and mixed workloads demands moderate memory. Lower memory requirements are demonstrated by the read-heavy workloads, since the structural modification operations are limited. The trade-off between memory utilization and execution speed in the proposed adaptive runtime system is indicated by the memory variation observed in the Figure 6. This helps in planning for memory management strategies for future research works in this area.

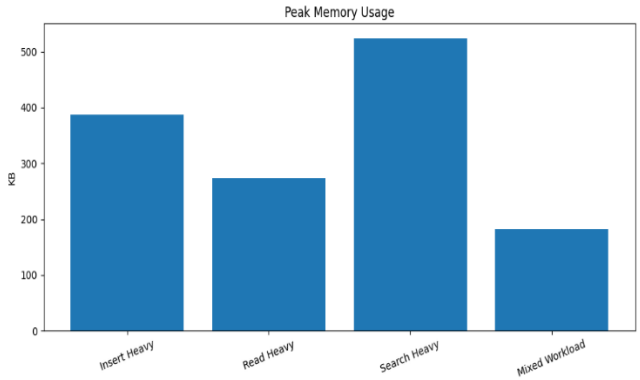

Fig. 6. Peak memory usage under different workload conditions.

V. Conclusion And Future Scope

This paper presents a machine learning-based adaptive runtime system capable of selecting optimal data structures dynamically. The proposed ML based prediction feature helps in achieving an efficient runtime decision making with reduced computational costs. Under both the categories of mixed and multi-phase operational patterns, a stable execution behaviour is maintained by the adaptive runtime engine. The system demonstrates high accuracy and competitive performance, validating the feasibility of self-optimizing software systems. Further, the experimental analysis suggests the presence of an intelligent runtime adaptation. This modular framework can be extended on a large scale suitable for both distributed and real time applications. This proposed work forms the basis for future development of an autonomous self-optimization software systems. Other possible research considerations are Reinforcement learning-based adaptation, Multi-threaded implementation, Memory-aware optimization and Integration with compiled languages.

REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge, MA: MIT Press, 2009.

[2] D. E. Knuth, The Art of Computer Programming, 3rd ed. Reading, MA: Addison-Wesley, 1997.

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in Proc. 6th Symp. Operating System Design and Implementation (OSDI), San Francisco, CA, USA, 2004, pp. 137–150.

[4] J. M. Hellerstein, "Self-managing systems: A control theory foundation," Computer, vol. 40, no. 2, pp. 36–42, Feb. 2007.

[5] J. R. Quinlan, "Induction of decision trees," Mach. Learn., vol. 1, no. 1, pp. 81–106, 1986.

[6] J. Dean and L. A. Barroso, "The tail at scale," Commun. ACM, vol. 56, no. 2, pp. 74–80, Feb. 2013.

[7] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in Proc. 5th European Conf. Computer Systems (EuroSys), Paris, France, 2010, pp. 265–278.

[8] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia, "Spark SQL: Relational data processing in Spark," in Proc. ACM SIGMOD Int. Conf. Management Data, Melbourne, VIC, Australia, 2015, pp. 1383–1394.

[9] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: A unified engine for big data processing," Commun. ACM, vol. 59, no. 11, pp. 56–65, Nov. 2016.

[10] A. Pavlo, M. Butrovich, A. Joshi, and C. Curino, "Self-driving database management systems," in Proc. 8th Biennial Conf. Innovative Data Systems Research (CIDR), Chaminade, CA, USA, 2017.

[11] T. Kraska, A. Beutel, E. H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in Proc. ACM SIGMOD Int. Conf. Management Data, Houston, TX, USA, 2018, pp. 489–504.

[12] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2nd ed. Cambridge, MA: MIT Press, 2018.

[13] S. Kulkarni, A. Marda, and K. Vaidhyanathan, "Towards self-adaptive machine learning-enabled systems through QoS-aware model switching," in Proc. IEEE/ACM Int. Conf. Automated Software Engineering (ASE), Luxembourg, 2023.

[14] F. Quin, D. Weyns, and O. Gheibi, "Reducing large adaptation spaces in self-adaptive systems using machine learning," 2023, arXiv:2306.01404.

[15] G. Georgakoudis, K. Parasyris, C. Liao, D. Beckingsale, T. Gamblin, and B. de Supinski, "Machine learning-driven adaptive OpenMP for portable performance on heterogeneous systems," 2023, arXiv:2303.08873.

[16] X. Chen, R. Zhu, B. Ding, S. Wang, and J. Zhou, "Lero: Applying learning-to-rank in query optimizer," VLDB J., vol. 33, no. 4, pp. 1307–1331, 2024.

[17] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," ACM Trans. Auton. Adapt. Syst., vol. 19, no. 1, pp. 1–38, 2024.

[18] M. Alidoost Nia, R. Calinescu, M. Kargahi, and A. Abate, "Efficient model verification at runtime through adaptive dynamic approximation," ACM Trans. Auton. Adapt. Syst., vol. 20, no. 1, pp. 1–45, 2024.

[19] C. Li, J. Wang, J. Shi, L. Liu, and S. Zhang, "ADWTune: An adaptive dynamic workload tuning system with deep reinforcement learning," Complex Intell. Syst., vol. 11, no. 1, pp. 1–18, 2025.

[20] M. Jam, E. Petit, P. de Oliveira Castro, D. Defour, G. Henry, and W. Jalby, "MLKAPS: Machine learning and adaptive sampling for HPC kernel auto-tuning," 2025, arXiv:2501.05811.