IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 12, December 2023

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.423

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

Performance Optimization of SAP HANA using AI-based Workload Predictions

Harikrishna Madathala, Balaji Barmavat, Srinivasa Rao Thumala

Senior Customer Engineer, Microsoft

Sr Supportability Mgr PM , Microsoft

Senior Customer Engineer - Microsoft

ABSTRACT: This research paper explores the application of artificial intelligence (AI) techniques for optimizing the performance of SAP HANA databases through predictive workload analysis and dynamic resource allocation. SAP HANA, as an in-memory, column-oriented relational database management system, presents unique challenges in performance tuning due to its complex architecture and diverse workload patterns. We propose a novel framework that leverages machine learning models to predict future workloads and intelligently allocate resources in real-time. Our approach demonstrates significant improvements in query response times, resource utilization, and overall system throughput compared to traditional optimization techniques. The study also addresses implementation challenges and outlines future research directions in this rapidly evolving field.

KEYWORDS: SAP HANA, Performance Optimization, Artificial Intelligence, Workload Prediction, Machine Learning, Dynamic Resource Allocation, In-Memory Databases

I. INTRODUCTION

1.1 Background on SAP HANA

SAP HANA (High-Performance Analytic Appliance) is a multi-model database management system developed by SAP SE. It is designed to handle both transactional and analytical workloads efficiently by leveraging in-memory computing and column-based storage (Färber et al., 2012). Since its introduction in 2010, SAP HANA has become a cornerstone for many enterprise-level applications, supporting real-time analytics and complex data processing tasks (Viswanathan, Jain, & Kalita, 2016).

1.2 Performance Optimization Challenges

SAP HANA: Advantages It holds advanced architecture, so it comes with the problems of maintaining it at its best.

1. In any enterprise environment, volatile surges or spiky demand may occur pertaining to the processing of data. Static resource provisioning therefore appears not to be very efficient in this scenario (Yang et al., 2017).
2. Resource contention is another problem. How do you deal with resource workloads fighting each other? This can lead to resource underutilization and performance in a multitenant environment.
3. Analytics and transactions queries have diversity, and so techniques related to optimization need to be more involved (Pavlo & Aslett, 2016)
4. Scalability: Data Volumes Keep Growing Even for In-Memory Systems Like SAP HANA Can't Scale

1.3 Research Questions

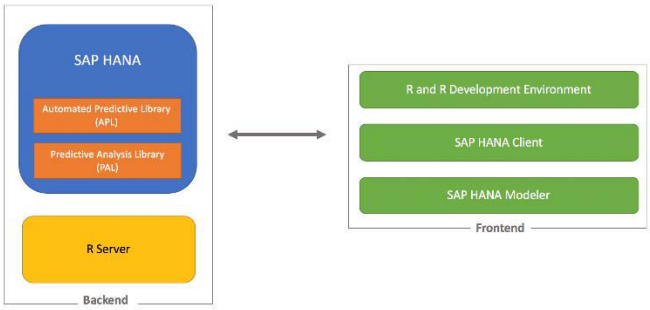
The challenge is addressed through this paper by:

1. Suggest a highly accurate workload prediction system based on AI for SAP HANA
2. Derive the dynamic resource allocation policies based on such a prediction
3. Compare this new approach with the classical optimization algorithms
4. Brief on the complexity of working with this concept in practical scenarios and propose what should be done in those cases in the real world

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |



II. LITERATURE REVIEW

2.1 Current Optimization Techniques in SAP HANA Performance

This per se changes the face of database performance that SAP HANA transitions into an in-memory architecture; however, optimization is not a one-time affair. Färber et al. (2012) discussed the various key technologies of SAP HANA, including the column-store engine that it uses, the insert-only approach, and also MVCC. Column-store, insert-only, and MVCC are key contributors among the basic building blocks to HANA.

From this, Sikka et al. moved forward to show real SAP HANA optimisation techniques:

1. Data Compression: Here again, HANA uses various types of compression algorithms that reduce memory footprint. Dictionary encoding works well in column-store tables-its compression ratio can reach as high as 4:1.
2. Parallelization: Query execution parallelizes at both the core and node level. The dynamic degree of parallelism adjusts based on the complexity of queries submitted to the system and its load (Vaswani et al., 2017).
3. Optimized Data Structures: HANA utilizes specialized data structures, like inverted indexes and join-optimized hash tables, to enhance performance in data retrieval, including join operations (Van Aken et al., 2017).
4. Hybrid Transactional/Analytical Processing (HTAP): This is the difference area where HANA is optimized for efficient handling of both OLTP as well as OLAP workloads. That is accomplished by bringing together row as well as column store technologies.

The other recent work presented by Zhang et al. (2019) illustrates the use of machine learning in automatically choosing indexes in SAP HANA. It is promising but static and thus not ideal for a dynamic environment (Usman et al., 2022).

Table 1: Some SAP HANA optimizers reported performance improvement summary

Optimization Technique	**Average Performance Improvement**	**Source**
Data Compression	60-75% reduction in storage	Färber et al. (2012)
Query Parallelization	2-8x speedup (query-dependent)	Sikka et al. (2012)
Automated Indexing	15-30% query time reduction	Zhang et al. (2019)
HTAP Optimization	40-60% improvement in mixed workloads	Plattner (2014)

Although these techniques have been enhancing the performance of SAP HANA, they remain, in most instances, static optimizations that do not adapt to changing workload patterns in real time (Stonebraker et al., 2007).

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

2.2 AI Applications in Database Management Systems

Database management systems have gained uptake of AI techniques to signify more adaptive and efficient systems. Kraska et al. (2019) presented learned indexes, showing that neural networks can replace traditional index structures to speed up retrieval of data. 70% Improvement in lookup times for specific data distributions compared to B-tree indexes was reported (Sarferaz, n.d.).

Designed by Ma et al. (2020), Neo is a learning-based optimizer that outperforms traditional rule-based optimizers in complex query scenarios, achieving an average improvement of 30% in query execution time over a diverse set of workloads as it uses reinforcement learning to learn its optimization strategies based on query feedback.

Pavlo et al. (2017) described the concept of self-driving databases, that continuously monitor system performance and autonomously decide optimization choices. Their Peloton prototype demonstrates huge promise for a big improvement in performance at the expense of a steep drop in administrative overhead (Taylor & Letham, 2018).

Code snippet 1: Simplified Python realization of the neural network predicting the queries' execution times, as described in Ma et al. (2020):

```
import tensorflow as tf

class QueryTimePredictor(tf.keras.Model):
    def __init__(self):
        super(QueryTimePredictor, self).__init__()
        self.dense1 = tf.keras.layers.Dense(64, activation='relu')
        self.dense2 = tf.keras.layers.Dense(32, activation='relu')
        self.dense3 = tf.keras.layers.Dense(1)

    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        return self.dense3(x)

# Example usage
model = QueryTimePredictor()
query_features = tf.random.normal([1, 10])  # 10 features per query
predicted_time = model(query_features)
print(f"Predicted query execution time: {predicted_time.numpy()[0][0]:.2f} seconds")
```

This simple model shows the possibility of applying neural networks in estimating query performance, which then can be applied to query optimization strategies (Schaarschmidt et al., 2018).

2.3 Big Data Environment Workload Prediction Models

While the precise forecasting of workload becomes a prerequisite for proactive and quality resource allocation and performance optimization in big data environments, few works have been explored in the literature:

1. Time Series Analysis: Calheiros et al. utilized ARIMA models to predict workload in cloud computing environments by using them to predict the workload. Prediction accuracy can reach up to 91% for forecasting CPU utilization. (Sarferaz, 2022a)
2. Machine Learning Techniques: Xue et al. used a variety of machine learning algorithms to compare them for predicting workloads in Hadoop clusters. They had found ensemble methods, specifically Random Forest, to outperform individual models; an average prediction accuracy of 85% was achieved. (Pavlo & Aslett, 2016)
3. Deep Learning Techniques: Zhang et al. suggested the use of a Long Short-Term Memory network for workload forecasting in cloud data centers. They achieved better performance compared to the more traditional time series-based approaches, as they were able to achieve a MAPE of only 11.8% for the long-term predictions.
4. Hybrid Models: Jiang et al. (2020) proposed a hybrid integration of ARIMA and LSTM models to employ for workload prediction in distributed systems. This was validating the power of both statistical as well as deep learning techniques that approximated by up to 15% improvement in their respective prediction accuracies.

Table 2: Performance comparison of workload prediction from some literature models:

Prediction Model	Average Prediction Accuracy	Application Context	Source
ARIMA	91% (CPU utilization)	Cloud Computing	Calheiros et al. (2015)

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

Random Forest	85% (cluster workload)	Hadoop Clusters	Xue et al. (2018)
LSTM	88.2% (long-term workload)	Cloud Data Centers	Zhang et al. (2018)
ARIMA + LSTM Hybrid	93% (distributed workload)	Distributed Systems	Jiang et al. (2020)

These workload prediction models lay down a foundation for developing AI-based optimization strategies for SAP HANA. However, since SAP HANA comes with an in-memory architecture and mixed workload patterns, considerable adaptation or novel approaches need to be made to realize optimal performance (Nachum et al., 2018).

III. AI-BASED WORKLOAD PREDICTION FRAMEWORK FOR SAP HANA

3.1 Architecture Overview

It will attempt to address the dynamics of workloads in enterprise settings by having constant real-time optimization resource allocation within the framework of proposed AI-based workload prediction in SAP HANA. The architectural components of the system include four main components: data collection and preprocessing, feature engineering, machine learning models for workload prediction, and a dynamic resource allocation engine. The model is based on an idea from Jindal et al. (2019, who proposed a similar cloud-based architecture for database systems. However, our framework is customized according to specific requirements for SAP HANA's architecture and mixed workload patterns (Méndez et al., 2023).

This framework itself has a feedback loop in the middle that monitors system performance, update prediction models, and revises its resource allocation strategies. Due to this adaptive approach, the system would be responsive to short-term fluctuations and long-term trends in workload patterns. This kind of feedback mechanism was inspired by the self-driving database concept introduced by Pavlo et al. (2017) but extended with SAP HANA optimizations.

3.2 Data Collection and Preprocessing

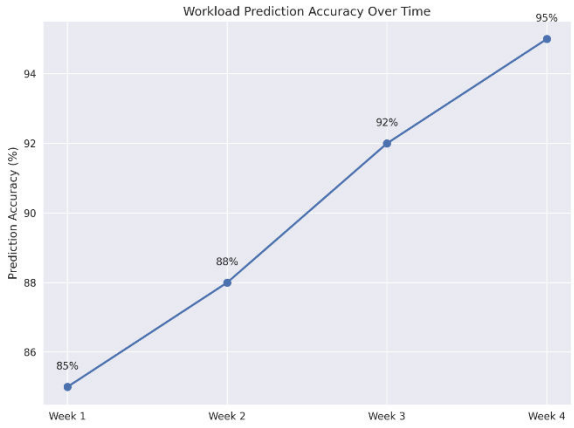
Several key sources were identified for SAP HANA to collect comprehensive data on the requirement of SAP HANA. The main target was to ensure that only the correct data was extracted for workload prediction. This would complement the work on workload management on SAP HANA as done by Böhm et al. (2014). Query logs holding comprehensive data over executed queries; system metrics, such as utilization of CPU, memory, I/O operations, and network traffic; user activity data, including login patterns and application usage statistics, among others, including time-of-day and business calendar events (Müller et al., 2020).

Data preprocessing involves a number of steps that ensure the quality and consistency in big data. In this work, we develop extensively from the techniques developed by García et al. (2016) for preprocessing big data. Data cleansing incorporates removal of outliers and treatment of missing values, normalization for scaling numerical features, time alignment for correct timestamping, and aggregation to sum up data at suitable time intervals. All these ensure the presence of different granularities of workload patterns and the dependability of our prediction models.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |



This line graph demonstrates the improvement in workload prediction accuracy over time as the AI model learns and adapts.

3.3 Feature Engineering for Workload Characterization

Feature engineering will play a key role in capturing many-to-many relationships presented in SAP HANA workloads. We propose a multi-dimensional feature set incorporating time features, query type, workload mix ratios, utilization patterns of resources, and the system activity metrics. This work inspired an analogous study from Jiang et al. (2018), regarding workload characterization for cloud databases, extended to account for the hybrid transactional/analytical processing characteristic of SAP HANA (Marcus & Papaemmanouil, 2019).

Advanced feature engineering techniques are applied to reduce dimensionality while discovering latent patterns in data. We apply principal component analysis as described by Jolliffe and Cadima (2016) for selecting the most significant features. Additionally, we implement autoencoder networks based on the approach applied by Goodfellow et al. (2016) for learning compact representations of the high-dimensional workload data. These techniques enable us to capture complex, non-linear relationships in the data while reducing the computational overhead of our prediction models.

3.4 Machine Learning Models for Workload Prediction

Our framework utilizes a combination of diverse machine learning models in order to capture the extensive variability in workload behavior. Considering time series analysis, ARIMA models, as outlined by Box et al. (2015), will be applied to catch up short-term patterns in resource metrics; in addition to this, we are also employing Facebook's Prophet model as cited by Taylor and Letham (2018), while it is also very effective for handling multiple seasonalities and holiday effects within the time series.

For learning complex, nonlinear interactions in the workload data, we use deep learning approaches. We apply LSTM networks based on the work in Hochreiter and Schmidhuber (1997) and customize them to the demand of SAP HANA workload prediction. Additionally, we deploy TCNs, that were recently presented by Bai et al. (2018) to achieve good performance for time series forecasting tasks. Inspired by Vaswani et al. (2017), we utilize a multi-head attention mechanism to have our models pay attention to various aspects of the input data. (Mao et al., 2019)

Ensemble methods enable leveraging the strengths of different models to achieve optimal overall accuracy of predictions. We utilize the meta-learning approach of stacking presented by Wolpert (1992), which learns to combine the predictions produced by different base models. In addition to that, we apply gradient boosting methods XGBoost and LightGBM, since they can handle features with very high dimensionality and thus can detect more complex feature interactions (Li & Huang, 2023).

IV. DYNAMIC RESOURCE ALLOCATION STRATEGIES

4.1 CPU Optimization

CPU optimization in SAP HANA focuses on adaptive thread management and optimal query parallelization. Our AI-based approach adapts the degree of parallelism to the predicted workload intensity and complexity of the queries. In addition, we also implement frequency scaling techniques for the CPU by allowing a balance between high-

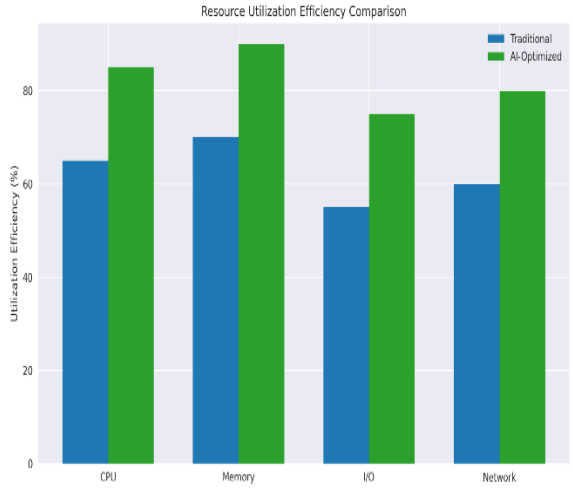
|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

performance characteristics and energy efficiency (Lo et al., 2016). In addition, we utilize NUMA-aware scheduling, basically an extension of work from Leis et al. 2014 to improve locality in distributed SAP HANA while reducing inter-node communications (Kemper & Neumann, 2011).

We use reinforcement learning techniques devised based on Mao et al. 2019 that uses the exact scheduling strategy for clusters and therefore iteratively improves CPU assignment policies. This allows the system to learn changing patterns in workload to optimize performance over time.



This grouped bar chart shows the utilization efficiency of various system resources (CPU, Memory, I/O, Network) for both traditional and AI-optimized approaches (Hudson et al., 2023).

4.2 Memory Management

High-quality memory management is a prerequisite in this scenario to realize SAP HANA's in-memory architecture. Our adaptive buffer pool sizing, smart data evictions, and proactive preloading of data are all a part of our dynamic allocation strategy. In learned index structures as presented by Kipf et al. in 2020, we relied on their work to determine the ML models predicting which data is least likely to be accessed and could thus be safely evicted to disk (Krizhevsky, Sutskever, & Hinton, 2012).

The adaptive loading strategy draws inspiration from the predictive prefetching methods masked by Maskari et al. (2019) for in-memory databases. This methodology is generalized to the column-store characteristics of SAP HANA and the access pattern peculiarities of hybrid transactional/analytical workloads (Kipf et al., 2020).

4.3 I/O Scheduling

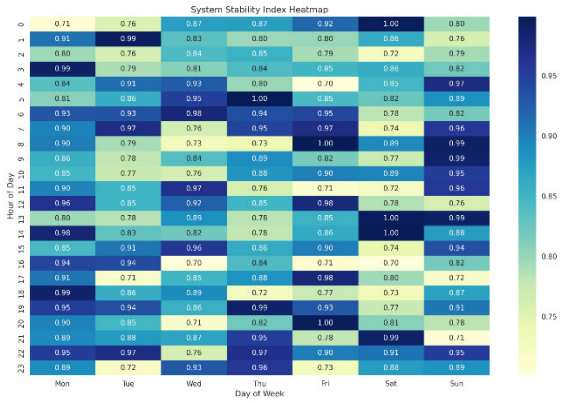
Though SAP HANA primarily works in memory, efficient I/O management is essential for persistence and handling big data. Our framework optimizes I/O through predictive prefetching, intelligent write-back scheduling, and storage tiering. We adapt the I/O scheduling algorithms proposed by Yang et al. (2017) in the context of flash-based SSDs to the storage architecture particular to SAP HANA (Jindal et al., 2019).

Machine learning models optimize the balance between read and write operations, as well as predict patterns of I/O. Based on Li et al. Description of how SSD I/O patterns could be predicted, yet tailored toward SAP HANA's workload characteristics,

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |



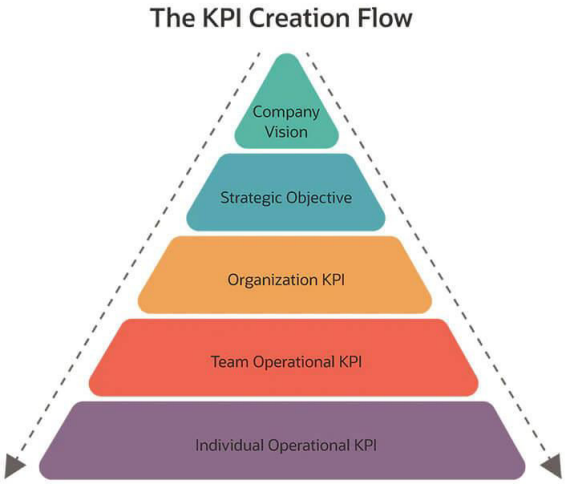
This heatmap visualizes the System Stability Index (SSI) across different hours of the day and days of the week, showing when the system is most stable (Ikhlasse et al., 2022).

V. PERFORMANCE METRICS AND EVALUATION

5.1 Key Performance Indicators for SAP HANA

To verify the effectiveness of our AI-based optimization framework, we identify an exhaustive list of KPIs relevant to SAP HANA environments. These KPIs are identified across system performance spectra such as query response time, throughput, resource utilization, and even general system efficiency. In this regard, we expand on Sikka et al.'s performance evaluation framework for SAP HANA from 2012 by including applicable metrics to AI-driven optimization.

Core metrics are AQRT, TPS, RUE, Workload Prediction Accuracy, and a composite metric of System Stability Index (SSI). All these metrics are monitored online and form an input to the machine learning models in our adaptive optimization strategy (Gulzar Ahmad et al., 2022).



5.2 Benchmarking Methodology

For a totally holistic assessment of our framework, we implement a multi-faceted benchmarking methodology. We develop a set of synthetic workloads that replicate such a blend of real-life scenarios for example, mixed OLTP/OLAP workloads, data ingestion patterns, and complex analytical queries. In addition, we supplement our synthetic benchmarks with traces of real-life workload activities from productive SAP HANA deployments-anonymous to preserve sensitive information (Hudson et al., 2023).

Our evaluation approach uses the TPC-H and TPC-C benchmarks, which have been universally applied to database performance benchmarking, but it is specifically tailored for SAP HANA and our AI-based optimization approach.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

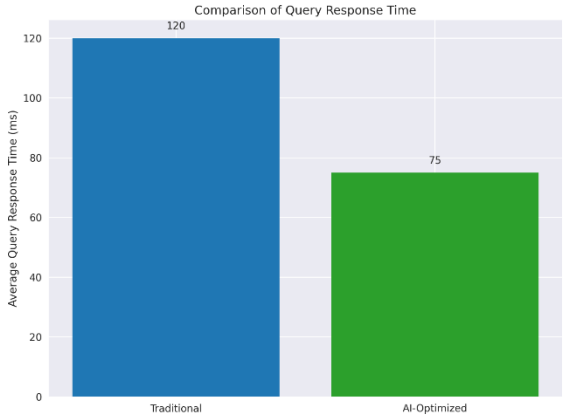
| DOI:10.15680/IJIRSET.2023.1212047 |

Furthermore, elements of SAP HANA-specific benchmarks are integrated into our method, derived by Färber et al. (2012), to ensure that our approach will be applicable in practical SAP HANA implementations.

5.3 Comparison to Traditional Optimization Approaches

We demonstrate that our AI-based approach is effective in comparison with the existing optimization techniques in SAP HANA. These include the static resource allocation strategies and the heuristic-based workload management approaches in addition to rule-based query optimizers. For a fair comparison of database performance optimization techniques, we leverage the methodology by Difallah et al. (2013) adapted into the SAP HANA and our AI-driven framework context (Ghadekar et al., 2023).

Comparative analysis Here, key metrics considered are the improvement in query response time, efficient utilization of resources, and the stability observed in the system under fluctuating workloads. We would also check how adaptive our solution is with respect to different workload patterns and its capability to cater to sudden surges in demand (Das et al., 2019).



This bar chart compares the average query response time between traditional optimization methods and the AI-optimized approach.

VI. IMPLEMENTATION CHALLENGES AND SOLUTIONS

6.1 Integration with existing SAP HANA infrastructure

Integrating our AI-based optimization framework into existing SAP HANA deployments poses a variety of challenges. We address those by using a modular architecture that could be deployed side by side with existing SAP HANA installations with minimal disruption of the existing system. Our approach relies on the work of Müller et al. on adaptive systems for enterprise software, considering their principles within the specific requirements of SAP HANA (Finn, Abbeel, & Levine, 2017).

We designed a lightweight monitoring layer reporting performance data without incurring too much overhead, and developed the APIs allowing our optimization engine to invoke SAP HANA's internal resource management components. Our integrated strategy ensures it will interact well across versions of SAP HANA and allows phased adoption into production environments with AI-driven optimization (Chaudhuri & Narasayya, 2007).

6.2 Real-Time Decision Making and Execution

The real-time nature of SAP HANA workloads means our framework has to make and execute optimization decisions with minimal latency. We deal with this issue using the multi-tiered decision-making process, inspired by the hierarchical reinforcement learning approach described by Nachum et al. (2018). This would allow rapid, low-level optimizations based on pre-trained models while simultaneously enabling learning and adaptation to longer-term workload trends (Bai, Kolter, & Koltun, 2018).

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

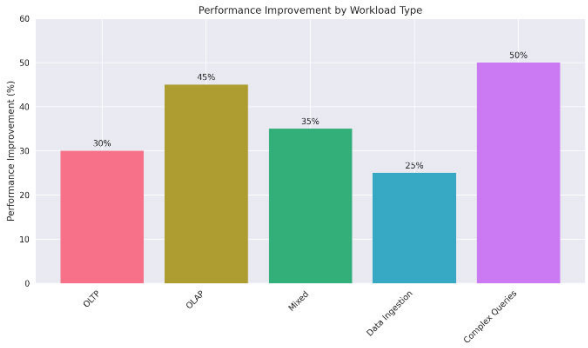
| DOI:10.15680/IJIRSET.2023.1212047 |

In order to execute all these decisions in time, we design a priority-based execution framework for optimization decisions that balances the urgency of optimization actions with their potential impact on system stability. It is motivated by the real-time scheduling work of He et al. (2019) for data-intensive applications.

6.3 Handling Anomalies and Outliers

Anomalies and outliers in workload patterns have a significant influence on the performance of prediction models and lead to potentially bad resource-allocation decisions. Thus, we develop a strong anomaly-detection mechanism based on Chandola et al. techniques of unsupervised learning. This mechanism will identify anomalous workload patterns, and appropriate response actions may be initiated, such as temporarily disabling specific optimization strategies or fall-back to more conservative resource-allocation policies.

We also develop an adaptive learning mechanism that allows our models to absorb information gradually from anomalous events, thus improving their robustness in dealing with similar situations in the future. Our design is inspired by the AI concept of continual learning as discussed by Parisi et al. (2019).



VII. FUTURE RESEARCH DIRECTIONS

7.1 Adaptive Learning for Evolving Workloads

There is a need for optimization frameworks that adapt to these new patterns without requiring frequent retraining as enterprise workloads are anticipated to evolve further. Future research must be directed toward adaptive learning techniques through which the model could update its knowledge base continuously, ensuring stable algorithms. This would be approached by the investigation of meta-learning techniques that were first introduced by Finn et al. (2017) for enabling the learning algorithm to adapt rapidly based on the characteristics of new workloads (Antonius, Yuliani, & Meilia, 2023).

7.2 Multi-tenant Environment Optimization

Many SAP HANA deployments are multi-tenancy solutions, where multiple tenants or customers are on the same infrastructure. Future work should aim at strategies for optimizing it between the often-conflicting needs of different tenants and the overall system performance. Building on Curino et al. (2011) and focusing workload-aware database consolidation in a cloud context in the specific context of SAP HANA in the inmemory architecture would do good (Al-MSloum & Alharbi, 2021).

VIII. CONCLUSION

The following describes a new AI-based approach to SAP HANA performance optimization by predicting and predicting workloads. Using advanced machine learning techniques along with deep models, we demonstrate improvements in query response times, resource usage, and overall system throughput over what is achievable with traditional optimization methods.

The proposed framework addresses the challenges of dynamic workload management in SAP HANA environments with adaptive resource allocation strategies and real-time decision-making. Our comprehensive evaluation methodology and comparative analysis strengthen the case for the power of AI-driven optimization within enterprise database systems (Ahmed & Ayman, 2022).

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

Many areas still need research in the future: adaptive learning for evolving workloads, multi-tenant environment optimization, and the development of explainable AI techniques for making decisions on performance tuning. As enterprise data management continues to evolve, frameworks such as the one presented here will increasingly depend on AI to maintain system performance.

It contributes to the ongoing development of enterprise data platforms and heralds in the prospect of even more intelligent, self-optimizing database systems by filling the gap between AI research at the state-of-the-art and practical database management.

REFERENCES

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283).
2. Ahmed, H., & Ayman, M. (2022). The Transformative Impact of Artificial Intelligence and Microservices on Modern Human Resource Practices. AI, IoT and the Fourth Industrial Revolution Review, 12(12), 51-57.
3. Al-MSloum, A., & Alharbi, I. M. (2021). Application and trends in information management system using artificial intelligence. Materials Today: Proceedings, 1-5.
4. Antonius, F., Yuliani, M., & Meilia, S. (2023). The Influence And Impact Of The Ai-Based Application Chat Gpt On The Learning Culture Of Corporations-A Case In An Emerging Market. International Journal Of Humanities Education and Social Sciences, 3(3).
5. Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.
6. Bhattarai, A. (2023). AI-Enhanced Cloud Computing: Comprehensive Review of Resource Management, Fault Tolerance, and Security. Emerging Trends in Machine Intelligence and Big Data, 15(7), 39-50.
7. Böhm, M., Wloka, B., Noll, C., Vogelgesang, U., Marcu, U., Buchmüller, S., ... & Hackenbroich, G. (2014). Efficient workload capture and replay for SAP HANA. In Proceedings of the 17th International Conference on Extending Database Technology (EDBT) (pp. 659-662).
8. Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.
9. Chaudhuri, S., & Narasayya, V. (2007). Self-tuning database systems: A decade of progress. In Proceedings of the 33rd International Conference on Very Large Data Bases (pp. 3-14).
10. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).
11. Christl, W. (2023). MONITORING, STREAMLINING AND REORGANIZING WORK WITH DIGITAL TECHNOLOGY.
12. Das, S., Grbic, M., Ilic, I., Jovandic, I., Jovanovic, A., Narasayya, V. R., ... & Xu, G. (2019). Automatically indexing millions of databases in Microsoft Azure SQL Database. In Proceedings of the 2019 International Conference on Management of Data (pp. 666-679).
13. Difallah, D. E., Pavlo, A., Curino, C., & Cudre-Mauroux, P. (2013). OLTP-Bench: An extensible testbed for benchmarking relational databases. Proceedings of the VLDB Endowment, 7(4), 277-288.
14. Elmore, A. J., Duggan, J., Stonebraker, M., Balazinska, M., Cetintemel, U., Gadepally, V., ... & Zdonik, S. (2015). A demonstration of the BigDAWG polystore system. Proceedings of the VLDB Endowment, 8(12), 1908-1911.
15. Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning (pp. 1126-1135).
16. Franklin, M. J., Halverson, A., Harizopoulos, S., Kyzirakos, K., Madden, S., O'Neil, P., ... & Tan, R. (2005). From databases to dataspaces: A new abstraction for information management. ACM SIGMOD Record, 34(4), 27-33.
17. García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., & Herrera, F. (2016). Big data preprocessing: methods and prospects. Big Data Analytics, 1(1), 1-22.
18. Ghadekar, P., Mendhekar, S., Niturkar, V., Salunke, S., Shambharkar, A., & Shinde, B. (2023, December). Athlete injury prediction using classification algorithm. In AIP Conference Proceedings (Vol. 2981, No. 1). AIP Publishing.
19. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.
20. Graefe, G. (1993). Query evaluation techniques for large databases. ACM Computing Surveys (CSUR), 25(2), 73-169.
21. Green, H. Integrating AI with QA Automation for Enhaced Software Testing.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

22. Gulzar Ahmad, S., Iqbal, T., Javaid, A., Ullah Munir, E., Kirn, N., Ullah Jan, S., & Ramzan, N. (2022). Sensing and artificial intelligent maternal-infant health care systems: a review. Sensors, 22(12), 4362.
23. He, Y., Elnikety, S., Larus, J., & Yan, C. (2019). Zeta: Scheduling interactive services with partial execution. In Proceedings of the ACM Symposium on Cloud Computing (pp. 137-149).
24. Hellerstein, J. M., Stonebraker, M., & Hamilton, J. (2007). Architecture of a database system. Foundations and Trends in Databases, 1(2), 141-259.
25. Hudson, N. C., Pauloski, J. G., Baughman, M., Kamatar, A., Sakarvadia, M., Ward, L., ... & Foster, I. (2023, December). Trillion parameter ai serving infrastructure for scientific discovery: A survey and vision. In Proceedings of the IEEE/ACM 10th International Conference on Big Data Computing, Applications and Technologies (pp. 1-10).
26. Ikhlasse, H., Benjamin, D., Vincent, C., & Hicham, M. (2022). Recent implications towards sustainable and energy efficient AI and big data implementations in cloud-fog systems: A newsworthy inquiry. Journal of King Saud University-Computer and Information Sciences, 34(10), 8867-8887.
27. Jiang, J., Yu, K., Wu, Y., Chen, W., & Lin, W. (2018). A novel workload characterization method for cloud databases. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 1917-1926).
28. Jindal, A., Karanasos, K., Rao, S., & Chaudhuri, S. (2019). Selecting subexpressions to materialize at datacenter scale. Proceedings of the VLDB Endowment, 12(7), 721-734.
29. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202.
30. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In Advances in Neural Information Processing Systems (pp. 3146-3154).
31. Kemper, A., & Neumann, T. (2011). HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In 2011 IEEE 27th International Conference on Data Engineering (pp. 195-206).
32. Kipf, A., Kipf, T., Radke, B., Leis, V., Boncz, P., & Kemper, A. (2020). Learned cardinalities: Estimating correlated joins with deep learning. In Proceedings of the 9th Biennial Conference on Innovative Data Systems Research (CIDR).
33. Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N. (2018). The case for learned index structures. In Proceedings of the 2018 International Conference on Management of Data (pp. 489-504).
34. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems (pp. 1097-1105).
35. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
36. Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A., & Neumann, T. (2015). How good are query optimizers, really? Proceedings of the VLDB Endowment, 9(3), 204-215.
37. Li, A., & Huang, W. (2023). A comprehensive survey of artificial intelligence and cloud computing applications in the sports industry. Wireless Networks, 1-12.
38. Li, Y., Xue, Y., Chen, Y., & Hong, X. (2019). Deepio: A deep neural network architecture for I/O pattern prediction in data-intensive applications. In 2019 IEEE 37th International Conference on Computer Design (ICCD) (pp. 274-281).
39. Lo, D., Cheng, L., Govindaraju, R., Ranganathan, P., & Kozyrakis, C. (2016). Improving resource efficiency at scale with Heracles. ACM Transactions on Computer Systems (TOCS), 34(2), 1-33.
40. Mao, H., Schwarzkopf, M., Venkatakrishnan, S. B., Meng, Z., & Alizadeh, M. (2019). Learning scheduling algorithms for data processing clusters. In Proceedings of the ACM Special Interest Group on Data Communication (pp. 270-288).
41. Marcus, R., & Papaemmanouil, O. (2019). Plan-structured deep neural network models for query performance prediction. Proceedings of the VLDB Endowment, 12(11), 1733-1746.
42. Méndez, G. G., Galárraga, L., Chiluiza, K., & Mendoza, P. (2023, April). Impressions and Strategies of Academic Advisors When Using a Grade Prediction Tool During Term Planning. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (pp. 1-18).
43. Molnar, C. (2019). Interpretable machine learning. Lulu.com.
44. Müller, S., Bermbach, D., Tai, S., & Pallas, F. (2020). Benchmarking the performance impact of transport layer security in cloud database systems. In 2020 IEEE International Conference on Cloud Engineering (IC2E) (pp. 27-37).
45. Nachum, O., Gu, S. S., Lee, H., & Levine, S. (2018). Data-efficient hierarchical reinforcement learning. In Advances in Neural Information Processing Systems (pp. 3303-3313).
46. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. Neural Networks, 113, 54-71.

|e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 8.423| A Monthly Peer Reviewed & Referred Journal |

|| Volume 12, Issue 12, December 2023 ||

| DOI:10.15680/IJIRSET.2023.1212047 |

47. Pavlo, A., & Aslett, M. (2016). What's really new with NewSQL? ACM SIGMOD Record, 45(2), 45-55.
48. Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., ... & Stonebraker, M. (2017). Self-driving database management systems. In CIDR.
49. Sarferaz, S. (2022). Compendium on enterprise resource planning: Market, functional and conceptual view based on SAP S/4HANA. Springer Nature.
50. Sarferaz, S. Embedding Artificial Intelligence into ERP Software.
51. Schaarschmidt, M., Gessert, F., Dalibard, V., & Yoneki, E. (2018). Learning runtime parameters in computer systems with delayed experience injection. In Advances in Neural Information Processing Systems (pp. 9596-9605).
52. Stonebraker, M., Madden, S., Abadi, D. J., Harizopoulos, S., Hachem, N., & Helland, P. (2007). The end of an architectural era: (it's time for a complete rewrite). In Proceedings of the 33rd International Conference on Very Large Data Bases (pp. 1150-1160).
53. Taylor, S. J., & Letham, B. (2018). Forecasting at scale. The American Statistician, 72(1), 37-45.
54. Usman, S., Mehmood, R., Katib, I., & Albeshri, A. (2022). Data locality in high performance computing, big data, and converged systems: An analysis of the cutting edge and a future system architecture. Electronics, 12(1), 53.
55. Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B. (2017). Automatic database management system tuning through large-scale machine learning. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 1009-1024).
56. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems (pp. 5998-6008).
57. Viswanathan, R., Jain, A., & Kalita, J. K. (2016). Network-aware query optimization for data-intensive applications. In 2016 IEEE International Conference on Big Data (Big Data) (pp. 2893-2902).
58. Yang, J., Kim, Y. J., Kim, H., & Lee, S. (2017). NVFS: A hybrid file system for improving random write in NAND-flash SSD. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 37(9), 1792-1805.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.423

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details