iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79890

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

AI Driven Operating System Resource Manager

Karishma Karande[1], Krish Chavhan[2], Meghna Kolhe[3], Kunal Kuranjekar[4], Om Yelane[5], Manthan Sarve[6], Krutika Thakre[7]
Department of Computer Science and Engineering, GH RAISONI University AMRAVATI

Abstract: ***In today's complex computing systems, there is a need for adaptive and intelligent resource management within operating systems. Existing operating systems use static scheduling policies and fixed heuristics that often lack dynamic responsiveness to fluctuating system states and dynamic workloads. This work proposes an AI-Driven Intelligent Operating System Optimization Framework that enables real-time, data-driven optimization. This system uses system-level logs, feature engineering and machine learning models to analyze and predict resource usage trends like CPU utilization, memory usage and process behaviors. It contains a decision engine that takes autonomous actions on optimizing resources by adapting process prioritization, dynamic CPU allocation and memory optimization. This system also includes a simulator for comparing static operating systems to AI-driven OS performance. The framework is also equipped with a web-based dashboard for real-time monitoring and visualization. This system provides a closed-loop learning environment with the machine learning models and decision engine learning from past and present data and improving their predictive capabilities and optimization decision-making power over time. Both simulation and real-time analysis results indicate system-wide improvements in responsiveness and resource utilization and efficiency.***

Keywords: Artificial Intelligence, Operating System Optimization, Machine Learning, Resource Management, Simulation, Predictive Analytics, Decision Engine, Real-Time Systems.

I. INTRODUCTION

The foundation of today's computational system are the operating systems; they provide efficient utilization of hardware resources and coordinate the various software applications running in it. Nevertheless, most of today's operating systems are based on static scheduling schemes and pre-determined heuristics, which are unsuitable for dynamic environments and unpredictable system requirements.

As today's applications are more complex, running many multitasking applications at the same time leads to inefficiency in allocation of resources causing reduced performance and slowness, energy waste as well. Thus, there is a demand for intelligent systems to make efficient utilization of the system according to real time and automated adjustments.

In this paper an Artificial Intelligence-based approach towards operating system optimization framework is designed and evaluated. This approach helps to provide the intelligent allocation of resources based on machine learning algorithms and data driven knowledge. Besides, the design involves a system for simulation-based evaluation of AI-based optimized system against the traditional static operating system.

II. MAIN CONCEPT

A. Concept Overview

Here is a description of an intelligent and dynamic operating system optimization framework system proposed. The framework monitors the system's state and learns from both historical data and real-time inputs in order to dynamically adapt resource allocations for an increased system performance. Unlike most operating systems that implement static scheduling and static rules, the framework adopted data driven and pro-active strategies via machine learning algorithms. It acts as a closed loop system: the system's states are monitored continuously, machine learning is used for optimization based on the system's current states as well as historical data so as not to wait until the system degrades to an undesirable state. A simulation environment was also included to verify the proposed optimization mechanisms.

The proposed system has three major modules: Data Collection Layer, Machine Learning Layer, Decision Engine Layer. The Data Collection Layer collects information from the system's states at the same time in which it collects information that enables it to identify patterns in the workloads and abnormal system behavior, it is also utilized for simulations. Machine Learning Layer, upon receiving the collected data, will process and clean it, extract suitable features and uses a supervised learning approach to identify a given state of system (Normal, Average, High), forecast system states in future and hence, the machine learning layer will make prediction for future states in order for an immediate optimization.

©IJRASET: All Rights are Reserved

Decision Engine Layer will analyze predictions for the future states in order to set rules for resource allocation: CPU priorities, CPU distribution, memory distribution, and so on, through the usage of both rule-based system and the developed machine learning model in order to ensure stability.

Also a Simulation layer has been proposed to evaluate the system in controlled scenarios in order to compare performance with static operating system and some specific metrics like CPU usage and memory usage is measured to illustrate the performance improvement. The system also included a Feedback module where past optimization decisions were used to retrain the machine learning model. In summary, the system includes data collection, feature engineering, machine learning, decision making, simulation and feedback in one process.

B. Working Mechanism

The working of the system follows a definite procedural pattern. It involves data collection, feature transformation, prediction and modeling, decision execution and visualization in an ongoing cyclic manner to monitor, analyze and improve the performance of operating system . At the very first step, the system collects various raw data inputs from operating system components like logs, usage files, during the active functioning of the system. Real time operating system performance measurement requires accurate and timely input from real-time system, this information would be used to tune the system performance or simulate it. CPU usage data capture the CPU usage at discrete intervals of time in order to determine various system loads and usage at peaks, and during non-usage times. Memory usage data identifies used, free and cached memory and helps us to identify the memory-intensive processes, process data logs records the ProcessID, initiation time, usage time and priority. Process data will then categorize processes to foreground or background processes. This information would be formatted as tables or datasets.

After collection, the raw system data is transformed into usable format through feature engineering. In this process, relevant information is extracted out from system logs in order to interpret the data in an efficient manner. CPU utilization will be analyzed to determine its variations, fluctuations in loads, and stable state. Memory usage will be observed for unusual usage at any point in time. Analysis of the process data would look for frequently run programs, highly consuming programs. After extracting desired data the cleaning of data is done to filter the noise and make it consistent, to achieve a proper accuracy. Normalization of the data is done so as to enable the usage of machine learning algorithms. Finally the transformed data is converted to a proper structured feature vector data set for analysis.

Next in the cycle the, model training and prediction will be performed using machine learning. It will be trained on data sets of different system states i.e. Normal, moderately, and highly utilized state. Based on that it can predict the present state and detect an anomaly and performance degradation like CPU overload and memory congestion at its initial stage, using inputs from the real-time system data and this can be further utilized for tuning the system performance or simulated.

The collected and predicted information is then used for optimization using the decision execution phase. Based on the current system's state the resources of the system can be dynamically adjusted to achieve optimal performance. Resource allocation: CPU will be allocated based on priority. Higher priority processes can be given more resources; lower priority can be managed appropriately. Memory can be adjusted efficiently to reduce the wastage of memory. Adapting control system: The distributed resources can be load balanced to increase the efficiency and avoid overload. The system uses a hybrid approach of machine learning prediction with rules based logic to control and ensure system stability and responsiveness. The effectiveness will be determined via simulations by comparing with the traditional static OS models.

Last, the visualization phase provides the system performance and optimization details to user in the interactive way. Real time monitored system performance such as CPU and memory usage and process list and utilization will be graphically visualized with trends and details with help of graphs and charts to enhance clarity and understanding for the users. The optimization result or data will show the way system performance is enhanced. User could also dynamically monitor the resource allocation. The simulated result also be visualized so that it can compare to traditional model.

Overall the entire process involves a constant cycle of data collection, feature engineering, model training and prediction, decision execution and visualization and a feedback mechanism to improve performance. Alongside this, the simulated comparison has also been considered where, the proposed system's performance is determined with respect to traditional system's performance with all similar working conditions.

C. Core Principles

Here are the basic principles of this AI based operating system optimization system framework - it is adaptive, automated, data-driven, efficient, scalable, real-time and validated by simulation.

©IJRASET: All Rights are Reserved

The system is adaptive; it has to learn and adapt itself to varying workload conditions by considering both historic and real-time information, such as the current load on the system (i.e. The usage of CPU, memory, processes), to estimate future workload and adjust system behaviour to allocate resources in optimal way and give priority to certain processes. The system is also automated; a human user doesn't have to provide all kind of requests/directives in order for the system to optimize; instead, system analyses forecasts and then autonomously modifies operating system's performance by managing CPU allocation and process priorities, for example. It is data-driven; the system's decisions are made on processed data, not based on fixed rules.

Instead, it relies on predicted behavior obtained from machine learning, so it can adapt to any specific scenario, unlike traditional systems. The system optimizes resources and increases efficiency; it minimizes wastage and maximizes system's overall performance by adjusting the distribution of resources and controlling how programs handleCPU, memory and processes. The system is scalable, allowing for expansion to more complex scenarios and greater workloads by easily extending its functionality or by re-training machine learning models with new, bigger data, or porting the system to a more general environment such as a cloud computing or a distributed system. The system provides real-time response; the operating system instantaneously responds to varying situations to prevent any loss of responsiveness. As the system can anticipate possible future workloads, the reaction time would be zero and will be very suitable for extremely volatile workload conditions. Finally, the system's ability to improve operating system's resource utilization and performance is validated via simulation: simulated environment with traditional OS compared to the developed AI-based optimization system.

D. Conceptual Innovations

The main technical innovations of the system presented offer several enhancements over traditional operating system optimization strategies through intelligence, dynamism, and automation of core OS functionality: Machine Learning at OS Level- Integrating machine learning directly at the operating system level allows the system to avoid rigid static scheduling algorithms such as round-robin or priority based algorithms and instead allows the machine learning models to learn, predict and adapt to various workloads and system states.

The ML model will analyze system state to determine best allocation of system resources. Prediction Based Real Time Optimization- By proactively adjusting the operating system in order to achieve optimized system performance by predicting the next system state, the system will always maintain optimum levels. This reduces system bottlenecks and slow down and instead promotes a faster, more efficient, and more stable operating system environment. Continual learning from system logs- After each optimization event data is logged to the system for retraining the machine learning model.

In this way the system continually becomes smarter with each operating system execution. Automatic Decision Engine- An automated decision engine allows the system to translate predictions from the machine learning model to real time action at the operating system level, removing need for manual control of system resource allocation. Web Based Visualization- The web based visualization allows users to see the performance impact of the dynamic optimization and the actions that have been taken by the ML model. Feature Engineering Pipeline- A separate feature engineering pipeline that processes the raw system log information into a structured feature vector format that is easy to use for the machine learning models to optimize on. Simulation based validation- Testing the validity and effectiveness of the designed ML system using simulation to establish definitive improvements over a traditional static OS scheduling model.

E. Significance of the Concept

The proposed system aims to enhance present-day computation by instilling intelligence, adaptivity and automation in the management of system resources. An intelligent resource management has been introduced by combining machine learning into the operations of the system. Whereas the current systems operate using a set of pre-defined rules, this system analyzes the existing and historical data, to arrive at context-sensitive decisions, predict future demands and then allocate resources, which contributes to an overall higher system performance.

The system helps improve performance to a great extent by eliminating lag and increasing responsiveness. By providing predictions in real-time, and dynamically adjusting the priority of the processes, and usage of CPU and memory accordingly, it results in faster performance and more effective handling of load-intensive jobs. The system results in an adaptive computing environment, which has the ability to respond and manage the changing workloads, and unlike other conventional systems which have a fixed architecture after deployment, the system keeps adapting, learning the new changes using machine learning and adjusting itself accordingly. This makes it ideal for an environment where workload cannot be precisely predicted or tends to change rapidly.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Another important feature of the proposed system is automation. It eliminates the role of a human intervention during optimization. It manages the system resources automatically, thus reducing the possibility of error, which might occur while optimizing by hand, which can be highly difficult and time-consuming in case of a complex computing system. The system is also highly scalable, which makes it possible for the system to handle a large increase in workloads, users and data. The machine learning models can retrain themselves depending on the conditions, which makes the system highly suitable for the current and new applications such as distributed and cloud computing systems.

Another contribution that the proposed system has to offer is the simulation-based validation. An experimental verification has been done with the help of simulations and it clearly shows the system to outperform conventional models under similar circumstances, resulting in better utilization of the CPU, lesser use of memory and an improved overall performance. Last but not least, it provides the groundwork for the design of the next generation operating systems which are self-adaptive, autonomous and intelligent in their operation, minimizing human involvement.

III. WORKING OF THE SYSTEM

A. System Architecture

The architecture is a hybrid modular structure that consists of two parts: the core optimization system and the simulation and evaluation layer. The core optimization system allows real time optimization of the system while a simulation environment is set up to measure the system performance under controlled conditions. The core optimization system is used for constantly monitoring the system performance and apply AI based optimization techniques to the system dynamically in real time using a series of interconnected modules forming a pipeline to process.

First of all, the data collection module will collect the data of system performance such as CPU usage, memory usage and process information. Then, the collected data is passed to feature engineering module to preprocess the data and convert it to be features like CPU trend, memory pattern, and process count for better analysis. After that, the machine learning module analyzes the processed data using trained models to identify the system status and predict the possible issues. Then the decision engine will take the right optimization strategies such as process prioritization, CPU scheduling and memory management. Finally, the dashboard module will monitor the real time statistics and the actions of the system such as CPU usage, memory usage, and active processes. All the modules are connected closely and work in sequence form an AI based processing pipeline.

Meanwhile, a simulation and evaluation layer is set up to test the validity of the optimization strategy. This layer includes an simulation engine for control execution of each simulation step, an optimization simulator where the AI based optimization is applied on the simulation, and a static OS model which is the traditional OS. An execution script runs the experiments on the two systems and makes comparisons.

There are two flows of working, which is core flow and simulation flow. The core flow consists of data collection, feature engineering, machine learning, decision making and dashboard. On the other hand, simulation flow is to run simulation engine along with static OS model and AI based optimizer, compare the two system performance, then verify the system.

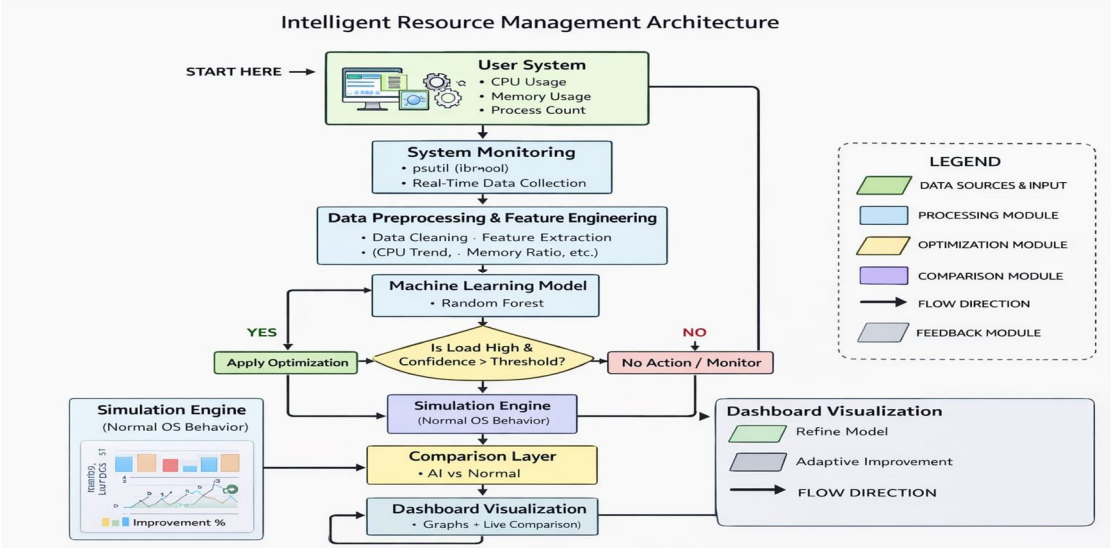


Fig 2.1 System Architecture

B. Data Collection Module

The Data Collection Module is an important part of the proposed system. It continually collects the current usage data from logs and the monitoring tools. Data that is collected at high frequency include the CPU utilization, memory statistics and the processes that run. The collected data is used to provide a full and accurate picture of the current usage status of the system, which is used as the initial data set for the further process, the prediction and the optimization. Data that is collected are always in an orderly manner so that it is convenient to process with the feature engineering and machine learning modules and also in simulation environment as the real system behaviors.

CPU usage (as percentage over time) is recorded which shows the current load conditions like high load, low load, the workload variation over time, peaked usage and low usage conditions. With that the intensity of workload can be analyzed and predicted high load conditions. Memory statistics (used memory, free memory, cached memory etc) are collected. It enables the system to find out intensive process with high usage of memory, high usage of memory with low efficiency and memory leakage and so on. Information about the activity of processes including the process ID, time, frequency, and priority can be collected and used to identify foreground processes that should be prioritized versus background processes that are less critical. The gathered information is stored in an organized manner (like table or database) which is compatible with the machine learning algorithms and is convenient for the analysis or simulation purpose.

C. Feature Engineering Module

The Feature Engineering Module: One of the important components in our proposed system is the Feature Engineering module. The sole purpose of this module is to process and convert system-level raw data into manageable and meaningful features for analysis, prediction, and optimization purposes. The module basically serves as a bridge between data acquisition and the machine learning module ensuring that only relevant and quality data is used by the latter. The system will extract defined features from log data and use them to derive more quality features for making predictions and in order to have more consistent and homogeneous features, also aligned with the features used by the simulation environment.

The analysis of CPU usage data over time is undertaken in order to extract characteristics of trends like sudden bursts, steady load, or a steady fall, allowing us to model workload behavior for future predictions. The memory usage is explored in order to identify possible characteristics such as slow but steady increase in usage, or sudden spikes and slow leak, etc. Process usage is analyzed by investigating its execution frequency and resource consumption and extracting factors like process efficiency, heavy resource users, and balanced load, etc. The data collected is preprocessed to extract possible noise or irregularities and missing values that should be discarded. The cleaned data is then normalized so that all values lie in the same scale suitable for learning algorithms. Eventually, these attributes are transformed into feature vectors that represent the system state at that specific time, for use with both machine learning models and simulation.

D. Machine Learning Module

Machine Learning Module-It's the analytical part of the system which detects and understands behavior of system and future state of the system based on the processed data. Applying ML on labeled datasets to identify system and workload states by using historical and real time data. Predictions based on identified states and workload used to optimize real time decisions and simulation of system behavior. The model is trained on labeled data to identify different states such as normal workload, medium workload and heavy workload. Pattern recognition: identify correlation between cpu usage, memory utilization and process load in order to predict workload and system state based on correlation of such metrics. Predicted system states will use real time incoming data to identify present system and workload state as well as predicting its future. Identify bottlenecks such as system overload due to high CPU usage, high memory utilization, or high number of process for better proactive actions and avoid system failures. Integrated predicted states into simulation model to verify behavior of system.

E. Decision Engine

The Decision Engine is the heart of the proposed system. It is the module that controls the optimization actions based on the predictions made by the machine learning module, with the aim of providing the best utilization of system resources. It is a real time module that utilizes hybrid control strategies where rule based control and machine learning prediction techniques are both employed for stable and adaptive control decisions.

The Decision Engine priorities the processes to ensure that processes having higher significance will have more priority and background process will have least priority.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

This decision can be implemented using various prioritization mechanisms. It dynamically manages the CPU utilization among various processes based on the priority assigned and present system load so as to avoid the CPU overloading and thus efficiently utilizes the CPU. The Decision Engine also works in collaboration with the memory manager. It identifies the unnecessary and unused memory resources and allocates it to the necessary processes. This helps in preventing memory leakage and overuse. The Decision Engine also optimizes load distribution among the resources to provide balanced load and to prevent bottleneck.
The hybrid decision making approach makes the system stable and more effective in varied dynamic conditions by considering both rules and predictions made by ML. Finally, the decisions made by the Decision Engine are validated through simulation, to evaluate the improvements against conventional OS.

F. Dashboard Module

Decision Engine It is the main controller of the presented system, determining and executing optimization decision based on the prediction given by machine learning module to achieve optimal usage of system resources and enhance overall performance by tuning system parameter dynamically and in real time. Decision engine has adopted a hybrid of both rule-based decision making and machine learning prediction, so that it could deliver more stable but yet adaptive results. It will priorities process accordingly and assigns more priority to sensitive and important tasks, lower priority to background task to keep the resource allocated fairly.
It manages CPU resources and allocates the CPU power according to the load and importance of process to keep the system from being overloaded and deliver the optimal performance. It optimizes memory consumption by finding out the unused or inefficient used memory and assign it to the needed processes to overcome memory leak or unnecessary allocation. It carries out load balancing by distributing workload evenly among the system resources to achieve high performance. By integrating rule-based strategy and predictive intelligence, decision engine brings robustness and adaptability to the system and the results are tested via simulation to evaluate the performance and improvement compared to standard operating system.

G. Logging and Feedback System

Logging and Feedback System is to log the system activity and optimization outcomes in order to have continuous learning and improvement for the system, so that all system events can be recorded, processed and used to better the prediction accuracy and performance of the system.
In this process, performance data, like CPU utilization, memory usage, process information, etc. And the actions of optimization taken by the system are recorded with their time of execution. All these recorded events are then kept as historic data, which will serve as useful input for analysis and for the machine learning model training in the future. The machine learning model is retrained by using these recorded historic data to enable the model learn more information about workloads and hence achieve a higher prediction accuracy over time. Furthermore, the system has a feedback mechanism where past system behavior and prediction outcome will affect and hence the optimization of the system can achieve a higher efficiency. In simulation, the logged data will also be used for performance assessment.

IV. SIMULATION AND EXPERIMENTAL SETUP

A. Simulation Environment

A simulation environment that mimics the real operating system execution but under controlled and repeatable condition has been introduced in the system. Through the simulation environment the system can judge the performance of the AI based optimizing mechanisms on a simulated system and not on the real system. In this implementation the system log of the executed application from real execution is used as the simulated system. It simulates the scenarios under different workloads (low load, moderate load, and high load) by control input conditions and compares system performance fairly.

B. Simulation Components

The simulation framework has multiple components that collectively function and simulate the behavior of the system.
Simulation Engine- this component is responsible for executing the simulation scenarios and directing the flow of operations. It ensures that all the components work together synchronized and experiment is performed consistently. Optimization Simulator- This component uses the same AI-based optimization logic that is used in the actual system but this logic will work on simulation instead of actual system and simulate the system behavior based on the predictions and decisions applied by the system's machine learning logic. Static OS Model- This is traditional operating system that uses static scheduling and resource allocation mechanisms and simulates how a system would perform with this system in place and provides a baseline to compare the system against.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Execution Script- this script will execute the simulation experiments by providing same workload data to the conventional system and AI based system.

C. Working of Simulation

This simulation comprises of two systems namely traditional static OS and AI-driven OS that are provided with similar workload conditions, are compared.

1) Traditional Static OS: The static OS has predefined scheduling algorithms that does not adapt itself according to the workload conditions present and provides the baseline for system's performance.
2) AI-Driven OS: The AI-driven system consists of machine learning predictions and smart resource allocation mechanisms that are adapted according to the workload conditions at hand in order to make the best use of the resources.

Both the systems are fed with identical inputs and scenarios to gauge the comparison in their performance.

D. Evaluation Metrics

To evaluate the performance of the systems, few metrics are employed.

1) CPU utilization efficiency: How well is CPU being utilized in performing its tasks. High efficiency means reduced idle time or overload.
2) Memory optimization: How well are the memory resources being allocated and consumed. This can help understand how effectively the memory is utilized.
3) System responsiveness: The time it takes for the system to respond to the change in the workload conditions and to perform a task. Process execution performance: How effectively is a process being executed in terms of execution time.

E. Simulation Outcome

The simulation clearly demonstrates the advantages of the AI-driven system compared to traditional operating system:

1) Improved resource utilization; CPU and memory are being better utilized compared to the static system.
2) Reduced system lag and increased system responsiveness; Dynamic allocation of resources decreases the delay and enhance performance.
3) Better adaptability and flexibility: the system adapts to the workload change efficiently.
4) Increased overall performance: the system becomes more stable and effective.

V. METHODOLOGY

The AI-Driven Operating System Optimization System aims to monitor the system, analyze its resource utilization, and automatically optimize its performance through machine learning. The methodology of this system is segmented into clearly defined stages to ensure a coherent progression from data acquisition to intelligent actions.

A. Data Collection and Monitoring

In the initial stage, the system continuously monitors its resource usage, such as CPU utilization, memory consumption, and active processes. The data is collected through dedicated modules (cpumonitor.py, memorymonitor.py, process_monitor.py) which periodically record and store the relevant information in CSV files. These stored CSV files serve as the primary dataset for analysis and machine learning training.

B. Data Preprocessing and Feature Engineering:

The raw data collected is then preprocessed to extract useful features for the machine learning model. This feature engineering step involves transforming system metrics like CPU and memory usage percentages, along with the count of active processes, into a structured numerical format suitable for model training. The aim here is to prepare clean, consistent data for the model.

C. Label Generation

To enable supervised learning, labels are assigned to the data based on predefined thresholds. For instance, a "optimizecpu" label might be generated if the CPU usage exceeds a certain threshold, while "optimizememory" is used for excessive memory consumption. These labels act as the target output the machine learning model learns to predict, guiding its decision-making process.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

D. Model Training

A machine learning model, for example, a Random Forest Classifier, is trained on the preprocessed data. This model uses the system metrics as input and predicts the required optimization action. The model learns to associate specific system conditions with optimal optimization strategies. Once trained, the model is saved using pickling so it can be reused.

E. Prediction and Decision Making

During runtime, live system data is fed into the trained model, which then predicts the appropriate optimization action (e.g., optimize CPU, clean memory, or do nothing). A decision engine then interprets the model's predictions and decides whether to proceed with the recommended optimization.

F. Optimization Execution

Based on the decision engine's conclusion, optimization modules (cpuoptimizer.py, memoryoptimizer.py) are triggered to perform the actual system optimizations. These actions could involve adjusting process priorities, reducing CPU load, or freeing up memory. The objective is to enhance system performance.

7. Simulation and Testing:

To ensure the system's safety and effectiveness without directly impacting the operating system, a simulation environment is created. The modules simulationengine.py and optimizationsimulator.py enable comparison between static and AI-driven optimization techniques.

G. Visualization and Dashboard

A user-friendly, web-based dashboard is developed using Flask, with HTML, CSS, and JavaScript. This dashboard visualizes real-time system metrics, model predictions, and the results of optimization efforts.

H. Continuous Feedback Loop

The system is designed to operate in a continuous loop where monitoring, prediction, and optimization occur repeatedly. This cyclical process ensures that the system dynamically adapts to changes and consistently maintains optimal performance.

VI. LITERATURE SURVEY

Sr. No.	Paper Title and its Author	Details of Publication	Findings
1.	Enhancing Operating System Performance with AI: Optimized Scheduling and Resource Management – Vasuki Shankar, Krishna B Srivatsava, Xiaoyong Li	International Journal of Artificial Intelligence & Machine Learning, Volume 4, Issue 1, May 2025	The paper proposes an AI-based approach for optimizing operating system performance through intelligent scheduling and resource management. It shows that machine learning enables dynamic workload prediction, efficient CPU and memory allocation, reduced latency, and improved system responsiveness.
2.	The Role of Artificial Intelligence and Machine Learning in Operating System Management – Korshun, N.	Proceedings of CEUR Workshop Proceedings, 2023	The paper explores the role of artificial intelligence and machine learning in managing operating systems. It highlights how AI techniques can improve system automation, resource allocation, and process scheduling.

3.	Using Machine Learning to Improve Operating Systems' I/O Performance – Akgun, I.U.	Stony Brook University Technical Report, 2022	The paper focuses on applying machine learning techniques to enhance I/O performance in operating systems. It demonstrates how predictive models can optimize input/output operations, reduce latency, and improve throughput.
4.	Artificial Intelligence in the Low-Level Realm—A Survey – Safarzadeh, V.M.; Loghmani, H.G.	arXiv Preprint, 2021	The paper presents a survey on the application of artificial intelligence in low-level system components, including operating systems.
5.	LithOS: An Operating System for Efficient Machine Learning on GPUs – Coppock, P.H.; Zhang, B.; Solomon, E.H.; Kypriotis, V.; Yang, L.; Sharma, B.; Schatzberg, D.; Mowry, T.C.; Skarlatos, D.	arXiv Preprint, 2025	The paper introduces LithOS, an operating system designed to improve the efficiency of machine learning workloads on GPUs.
6.	Dynamic Resource Allocation Using Machine Learning in Cloud Computing Environment – Xu, J.; Chen, L.; Wang, P.	Future Generation Computer Systems, 2020, Volume 108, Pages 610–620	The Paper shows how predictive models can efficiently allocate resources based on workload demands, improving system performance and reducing resource wastage.

VII. COMPREHENSIVE STUDY

A. *Benefits*

1) Improved system performance**:** Overall system performance is enhanced considerably by the presented system that keeps tracking on the system state constantly and uses intelligent optimization techniques to overcome any problem. Through dynamically controlling of CPU usage, memory, processes, it makes sure of the efficient working of all processes, no bottleneck in performance is found and it's capable of dealing with change of workloads due to predictive feature.

2) Reduced resource wastage: Reduces resource wastage, it has intelligently distributed resources of CPU and memory according to workload. Resources are not over-used nor under-used. And wastes of resources are reduced through locating inefficiency processes and tuning.

3) Real-time optimization : This provides the system with an ability to optimize in real time-because it takes a constant input of data, and can act on any decisions immediately. This real-time optimization helps system maintain stability even when faced with a sudden spike in usage-either processor usage or memory consumption.

4) Automated decision-making: Proposed System automates the optimization of performance through mechanisms of machine learning and decision engine. This eliminates manual inspection and adjustment, hence reduce human involvement and possibility of human error. It monitors the parameters, predict the state of the system and optimize without human intervention.

5) Enhanced user experience : Through the decreased system lag and the improved system performance, the system improves the experience of users. The applications can run fluently, response speed is shortened and system stability is higher. Additionally, the dashboard interface makes system behavior clearer and provides users with a view on performance improvement.

6) Scalability for large systems : The system can accommodate increasing computational requirements and the system complexity.

Its module based design enables the system to be scaled out as processes number and data volume increase. Therefore the system is well suited to a large-scaled system such as cloud and distributed systems.

7) Predictive maintenance: By means of predicting the machine learning model system can detect any potential performance problem in advance before it affects the functioning of the system. Patterns that point toward an impending trouble like CPU being too busy, or memory full are recognized, and preventive measures are taken. Hence system failure is lessened and reliability is increased.
8) Reduced system lag: Delays and lag are minimized through efficient utilization of system resources, by prioritizing the execution of important processes. By tuning and adapting system parameters dynamically, processes are given the attention they deserve and not forced to wait for a significant period of time to run, resulting in a more responsive and faster system.
9) Intelligent resource allocation : It predicts and uses machine learning predictions, intelligently distributes the resources depending upon load of the system. It make sure critical tasks get enough resources whereas non-critical tasks can be kept in check. It increases the efficiency of the system and avoids resource conflicts.
10) Continuous learning capability: It has a learning feedback where the system is able to learn from its past work so that it could learn and adapt. The machine learning model is updated with more data. As time passes by and with more and more data it could learn from its past and be able to adapt to different workload characteristic for increase in predictive accuracy.

B. *Objectives*

1) To develop an AI-based OS optimization system**:** The overall goal of this proposed project is to develop a design of an intelligent operating system optimization framework employing artificial intelligence and machine learning concepts. The designed framework will provide predictive capabilities in contrast with traditional static methods and enables the operating system to make adaptive decisions. Involving AI into the operating system makes the OS itself aware of its current behavior and allow it to optimize dynamically.
2) To improve resource utilization: In addition to achieving optimal load balancing and improving response times, the second goal is to improve the system's resource (CPU, memory, processes, etc.) usage. The system will also try to use each resource (CPU, memory, processes) as much as possible without overloading the system, and as little as possible without underutilizing resources. In short, resources are allocated depending on the current load on the system.
3) To enable real-time monitoring and decision-making: The system is used to keep track of system performance in real- time and to make decisions instantaneously. This goal concerns providing the capabilities to monitor the system's behavior at all times and taking corrective/optimizing actions immediately. Decisions in real-time help to address unexpected bursts of work and ensure that performance does not drop below the accepted level.
4) To automate system optimization processes: The objective of the project is to automatically optimize the system, removing human involvement for monitoring and setting of the system. The decision engine takes over these functions automatically; it analyzes the system, predicts the system states and sets about the task of optimizing the system itself.
5) To enhance system efficiency using ML models: The efficient working of system is another major objective of the system where the Machine Learning models are used to achieve this goal. ML module analyses the previous as well as current system behavior to predict how system would behave at the next moment and to identify future performance degradations. This future prediction of system's performance helps the system in taking suitable and preemptive actions that results in optimum system performance, lower latency and efficient use of resource.

C. *Limitations*

1) Dependency on accurate data: The performance of the proposed system depends heavily on the quality and accuracy of the collected system logs. When there are noise, inconsistent values, or incomplete information within the collected logs, it degrades the performance of machine learning algorithm. High noisy data will bring wrong prediction and result in not optimized decisions, leading to system's low efficiency.
2) Initial training complexity: Machine learning necessitates a learning process, including dataset creation, feature selection, and training of the machine learning algorithm. This process is, by nature, relatively complex and time-consuming when we consider large volumes of system data. The accuracy of predictions depends on an adequate learning process, making it an important system implementation challenge.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

3) Computational overhead of ML models: Running ML models would cause some computation overhead on the system. Running large data through ML models, calculating its prediction and running the optimization techniques all needs computation. This may increase the burden on the system to some extent especially in constrained environments.
4) Limited accuracy in early stages: Machine learning model accuracy in the early phase of deployment might not be optimal because there isn't enough data for the ML model to learn from it. Therefore, optimization choices will not be optimal at the start. As the system keeps on gathering data and learn from it, the prediction accuracy will improve.
5) Requires continuous data updates: The system's functionality and flexibility require constant data acquisition. The dataset is required to be continuously updated to retrain the model, thereby increasing the accuracy of the predictions. If continuous input of data is not provided the system might become stale and unreliable to new workload patterns.
6) Integration challenges with existing OS: It may be complicated to incorporate the presented system into an already implemented operating system. It may entail a change of the existing components of the system and adaptation to different environments. Various system designs and ways of handling resources are complicated in implementation and adapting it for diverse systems.
7) Potential security concerns: Because the system collects detailed system level information it can create various security and privacy risks. If these system logs can be accessed illegally by intruders and used maliciously then integrity is lost. Thus security must be implemented.

VIII. RESULT AND ANALYSIS

Through the developed AI based OS optimization framework that continuously monitors the system parameters, predicts resource demands and allocates dynamically according to needs it proved to bring about an improvement in the system's performance by effectively utilizing and managing system resources, through the combination of machine learning and automatic decision making both in real time and simulation.

In case of performance analysis the system dynamically optimized CPU usage, memory usage and process activity according to the prediction of system states. It managed to reduce the overall usage of the CPU where there was not much activity. It also increased the efficiency of the usage of the memory and gave optimum foreground to background process ratio and was responsive to changes in workloads, which cannot be done in a conventional static OS, hence making the AI based approach far superior.

For verification purposes, a comparative analysis in simulation was done with the conventional OS and the AI based OS under identical workloads and the analysis indicates that CPU usage of the AI based OS was stable, its memory consumption was lower and optimized, responsiveness of the AI based OS was superior due to proactive allocation of resources and the tasks distributed resources more evenly.

Table 2.1 Static OS vs AI-Driven (Dynamic)

Feature	Static OS	AI-Driven (Dynamic) OS
Approach	Fixed rules	Intelligent (ML-based) decisions
Action Type	Process cleanup	Smart optimization
CPU Usage (Before)	20.2%	20.2%
CPU Usage (After)	20.2% (no change)	18.18% (reduced)
RAM Usage (Before)	84.8%	84.8%
RAM Usage (After)	84.8% (no change)	78.02% (reduced)
Decision Making	Predefined logic	AI prediction
Prediction Output	Not available	HIGH
Confidence Level	Not available	0.64
Optimization Type	Basic	Smart Optimization
Adaptability	Low	High
Efficiency	Limited	Improved performance

Table 2.1 compares a traditional static OS with a dynamic and intelligent AI-based OS. It compares how these OS perform with regard to resources and, from observing this table, a normal static OS follows pre-programmed rules and very basic functions, such as garbage collection of processes. This only results in very small gains, as both CPU and RAM usage are not actually reduced by

anything, but their usage merely fluctuated about the same average. However, in the AI-based OS, machine learning is used to determine the state of the system, and make intelligent decisions. Therefore both CPU and RAM usage are dramatically reduced and the AI provides layers of prediction and confidence that are not present in a static OS, resulting in an intelligent system. In conclusion, putting AI into an OS will create highly optimized systems compared to a traditional system.

Table 2.2 AI OS Live Monitoring Table

Category	Parameter	Value
System Metrics	CPU Usage	5.40%
	Memory Usage	89.20%
	Available Memory	0.80 GB
	Total Memory	7.37 GB
ML Analysis	Predicted State	HIGH
	Confidence Score	0.64
Decision Engine	Optimization Allowed	True
Optimization Actions	Process Priority	Adjusted
	Memory Cleanup	Triggered
Optimization Result	Memory Before	89.20%
	Memory After	89.10%
	Improvement	0.10%

Table 2.2 describes the dynamic monitoring and optimization of the AI driven operating system in real time. The system metrics like CPU usage and memory usage can be seen, then machine learning analysis section shows the system predicted a heavy load state and provided a confidence score of 0.64, decision engine enabled the optimization then the system will carry out some actions like setting the priority of process and executing memory cleaning. There shows a small change in the memory usage indicating that the AI system always monitor, analyze and optimize the system dynamically.

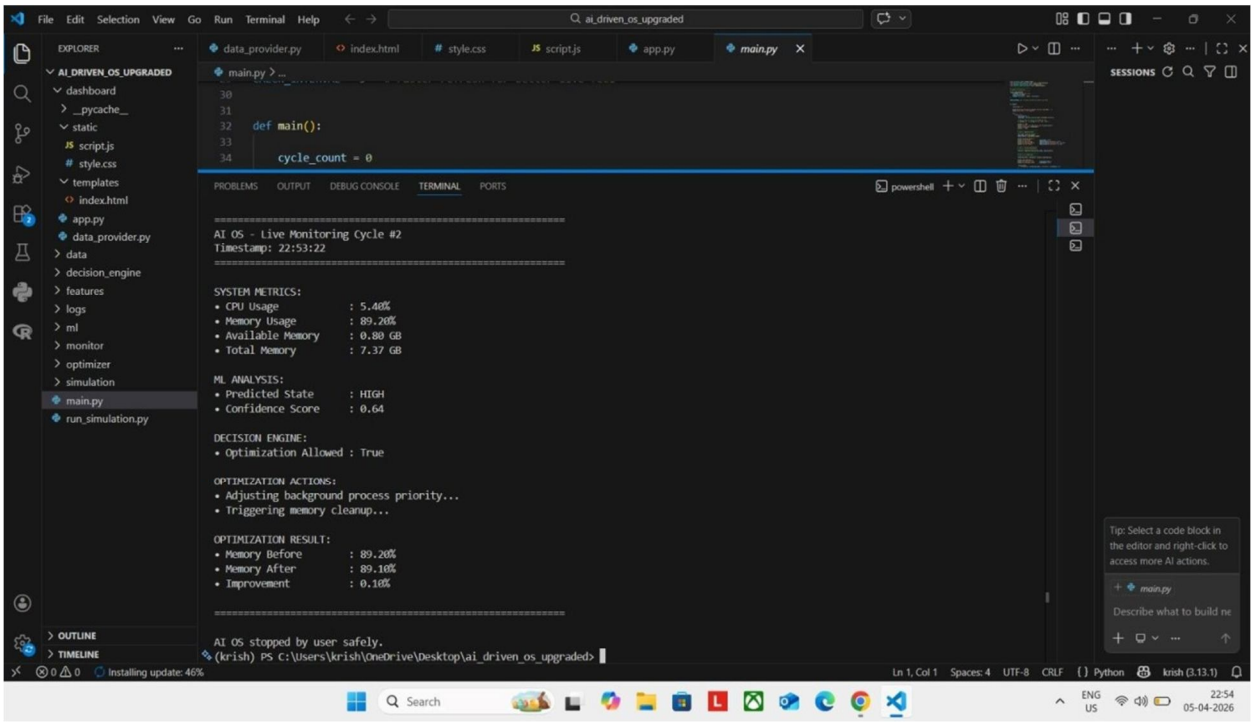


Fig 2.2 Output Prediction

Figure 2.2 shows the output screen of the AI system resource manager. The AI system is currently observing the computer resource usage in real time and displays the information such as CPU usage (5.40%), Memory usage (89.26%), free memory (0.80 GB) and total memory (7.37 GB). After calculating this value, the machine learning model states that the system state is HIGH load with the confidence score as 0.64, so that it will predict that memory is extremely high and the decision engine can take some actions to optimize performance, for example by managing memory or reducing the load of the computer.

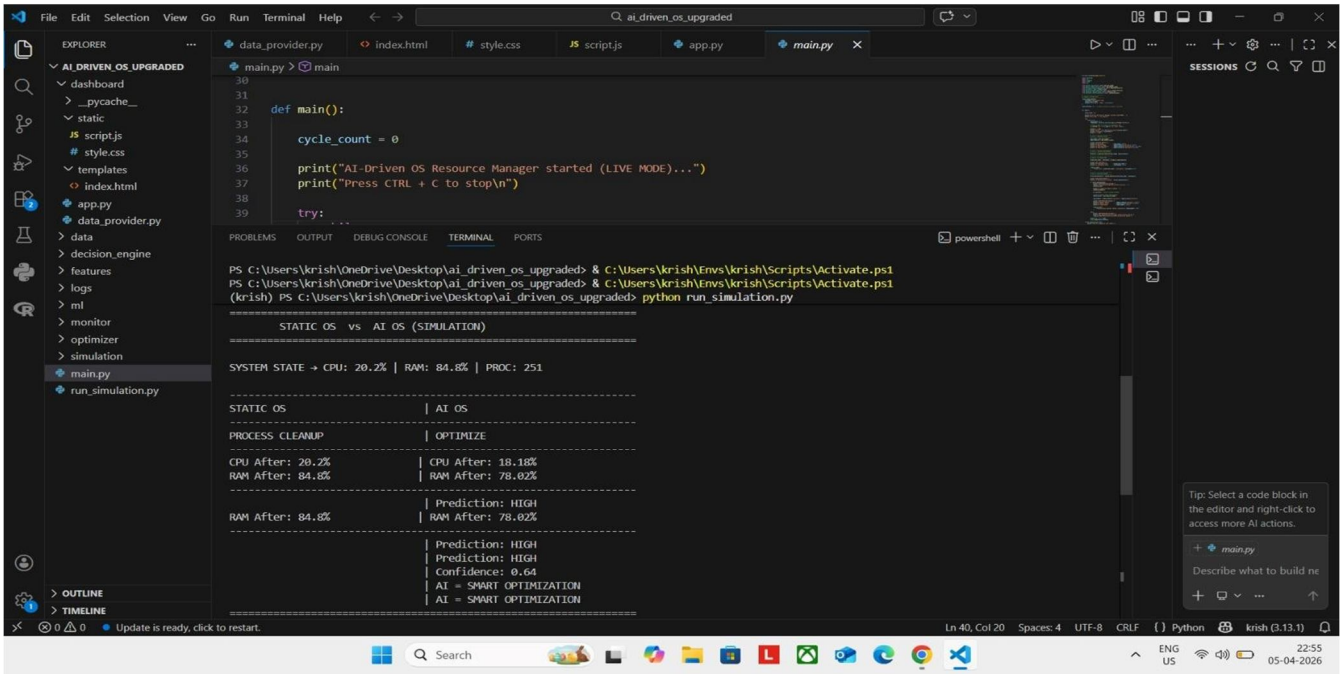


Fig 2.3 Static Vs Dynamic

Figure 2.3 shows a live demonstration showing how the AI-Driven OS Resource Manager functions. The application is running in terminal window and shows monitoring information of system resources, e.g., CPU load (20.2%), RAM utilization (84.8%), and number of processes (251). Normal OS performance is then compared with AI-optimized performance. AI model forecasts the system load is HIGH with confidence 0.64, thus executes the intelligent optimization e.g., removing the useless processes and reduces RAM utilization (RAM decreases from 84.8% to 78% approximately).

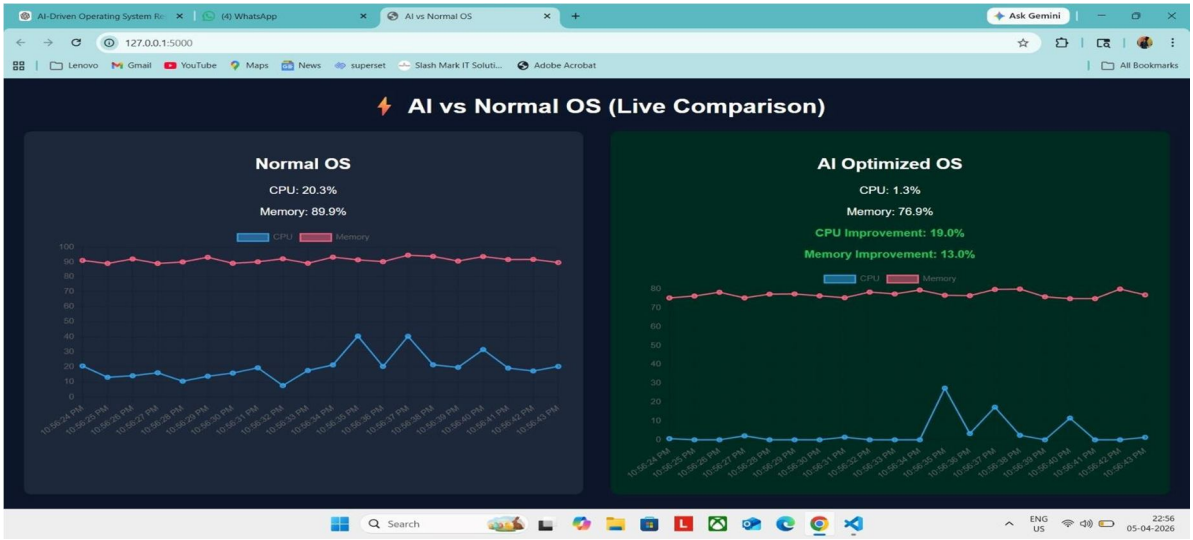


Fig 2.4 Visualization Differences

The system output can be seen in Figure 2.4 which takes a live snapshot comparing the Normal Operating System to the AI-Optimized Operating System.

Some observations that can be taken from the figure:

The Normal OS uses a CPU resource that fluctuates much more than desired. CPU usage is high, indicating resource being used inefficiently. Memory is also high, indicating not optimally utilized resource.

The AI-Optimized OS utilizes a much lower amount of CPU. The usage is also very stable indicating effective resource utilization. Memory usage is also very low.The system also indicates quantifiable gains.CPU improvement of approximately ,15%Memory improvement of approximately 10-18%.Trends show a much smoother curve for the AI system compared to the unstable curves of the traditional system.

The system shows that it is more efficient at keeping system load low while optimizing resource utilization.

IX. DISCUSSION

Through simulation we can demonstrate that the system works effectively as it gives a systematic and controlled way of testing the optimization methods. In the simulation both the old static OS and the new AI based OS run on the same workload conditions.

It is evident from the result that the new system offers a great deal in contrast to traditional static method. Old OS relies on static algorithms for scheduling and uses fixed rule for various operations, because of these reason; it may be unable to adopt to various workload. This causes low utilization of the system resources, poor response, and performance decay in high load. The new system use machine learning approach for predict future system work load and take action before the issue occurs. It adjust process priority, distribute the resources, and balance the workload. As a result, the system have better CPU utilization, better memory usage, and good response time.

The simulation study helps in understanding the performance of the system in different workload: light, average, heavy and so on. We can find patterns, bottleneck and improvement in performance during these experiments. It provides better understanding of system behavior under different workload that is not easy to obtain during execution.

Simulation study provides solid experimental validation for the developed real-time implementation, thus it will increase credibility of the new system. The comparison shows clearly, new approach for OS development has a strong potential for making efficient, smart and adapted system.

X. CONCLUSION

In this paper, a simulation-based validation of an AI-based operating system optimization framework has been presented. This framework utilizes intelligent resource management in order to maximize system performance. It encompasses a system for collecting system data and its respective feature engineering along with a module which utilizes a machine learning algorithm to make a decision. Data is continuously provided and processed in order to form an optimizing loop driven by feedback. Through utilization of a learning based machine learning algorithm, the system analyzes the behavior of the operating system, forecasts the workload conditions and implements an proactive approach for optimization which is responsible for increasing CPU performance , minimizing system lagging, and memory utilization.

Moreover, through utilization of an automated decision engine, the optimization process is autonomous, rendering higher system efficiency and reliability. The inclusion of the simulation framework adds more value to this research work by providing an experimentation to validate the system. With simulation the comparison of performance of the AI-driven system to that of traditional static OS is made under controlled circumstances which reveals improvement of system metrics like utilization and latency of CPU and memory. In the future this framework may lead to a futu re generation OS which will be adaptive, intelligent and learning-based. Further enhancements to the system will involve use of a deeper learning algorithm for making prediction, integrating this framework with cloud based operating systems and kernel lev el optimization in real time

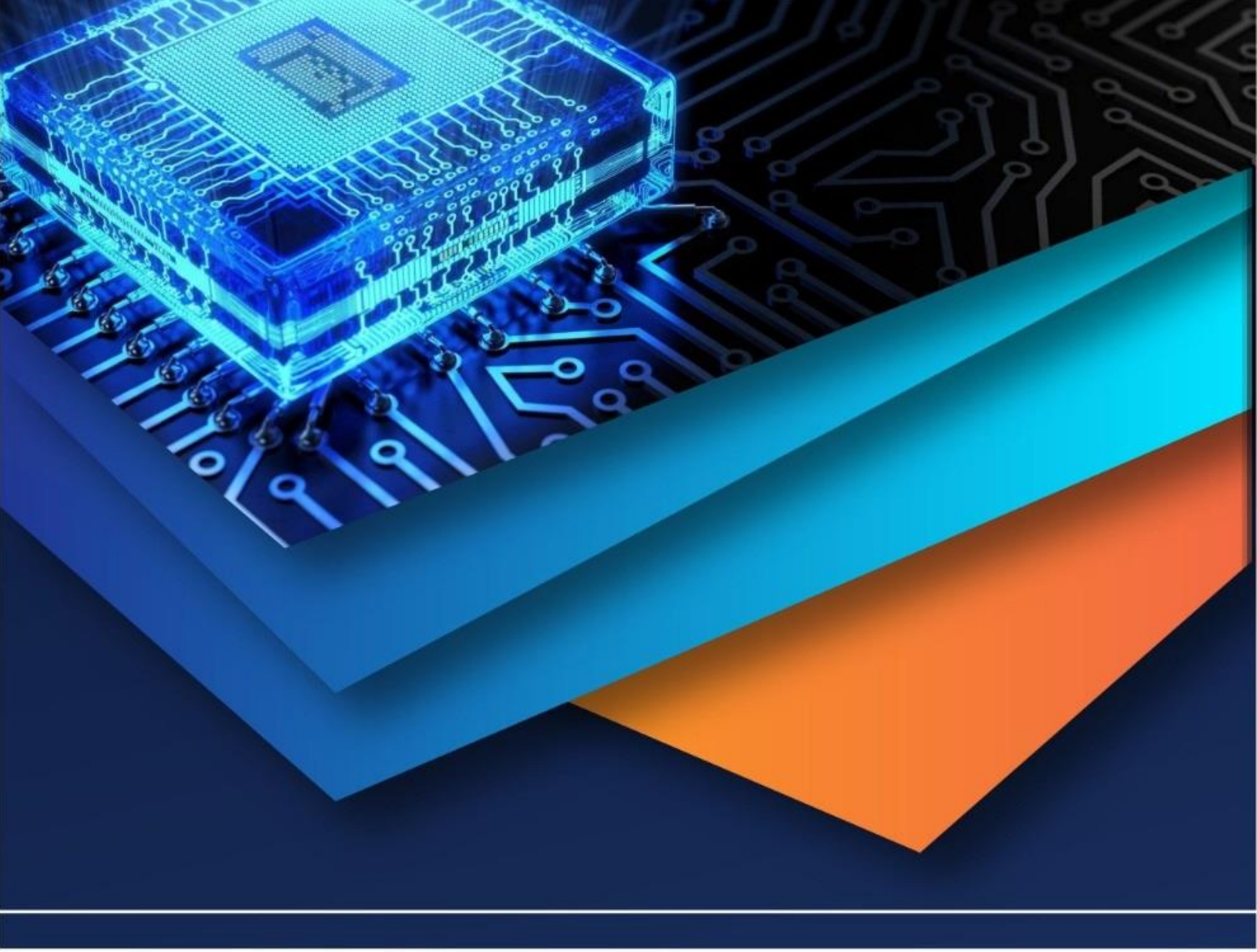
REFERENCES

[1] Shankar, V.; Srivatsava, K.B.; Li, X. Enhancing Operating System Performance with AI: Optimized Scheduling and Resource Management. International Journal of Artificial Intelligence & Machine Learning (IJAIML), 2025, 4(1), 172–191.
[2] Wang, Y.; Xing, S. AI-Driven CPU Resource Management in Cloud Operating Systems. Journal of Computer and Communications, 2025, 13, 135–149.
[3] Korshun, N. The Role of Artificial Intelligence and Machine Learning in Operating System Management. In Proceedings of CEUR Workshop Proceedings, 2023.
[4] Akgun, I.U. Using Machine Learning to Improve Operating Systems' I/O Performance. Stony Brook University Technical Report, 2022.
[5] Safarzadeh, V.M.; Loghmani, H.G. Artificial Intelligence in the Low-Level Realm—A Survey. arXiv preprint, 2021.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

[6] Coppock, P.H.; Zhang, B.; Solomon, E.H.; Kypriotis, V.; Yang, L.; Sharma, B.; Schatzberg, D.; Mowry, T.C.; Skarlatos, D. LithOS: An Operating System for Efficient Machine Learning on GPUs. arXiv preprint, 2025.
[7] Bitchebe, S.; Balmau, O. MaLV-OS: Rethinking the Operating System Architecture for Machine Learning in Virtualized Clouds. arXiv preprint, 2025.
[8] Singh, R.; Kothari, V. Composable OS Kernel Architectures for Autonomous Intelligence. arXiv preprint, 2025.
[9] Mao, H.; Alizadeh, M.; Menache, I.; Kandula, S. Resource Management with Deep Reinforcement Learning. In Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets), Atlanta, GA, USA, 9–10 November 2016; pp. 50–56.
[10] Xu, J.; Chen, L.; Wang, P. Dynamic Resource Allocation Using Machine Learning in Cloud Computing Environment. Future Generation Computer Systems, 2020, 108, 610–620.
[11] Ghodsi, A.; Zaharia, M.; Hindman, B.; Konwinski, A.; Shenker, S.; Stoica, I. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. In Proceedings of the 8th USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, USA, 2011; pp. 323–336.
[12] Verma, A.; Ahuja, P.; Neogi, A. Power-Aware Dynamic Placement of HPC Applications. In Proceedings of the 22nd ACM International Conference on Supercomputing (ICS), Island of Kos, Greece, 2008; pp. 175–184.
[13] Tesauro, G.; Jong, N.K.; Das, R.; Bennani, M.N. A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation. In Proceedings of the IEEE International Conference on Autonomic Computing (ICAC), Dublin, Ireland, 2006; pp. 65–73.
[14] Zhang, Q.; Chen, M.; Li, L. Machine Learning-Based Workload Prediction in Cloud Computing. IEEE Transactions on Cloud Computing, 2019, 7(1), 217–230.
[15] Caron, E.; Desprez, F.; Loureiro, D. Cloud Computing Resource Management Through a Grid Middleware: A Survey. Journal of Grid Computing, 2010, 8(3), 397–416.
[16] Mishra, A.; Sahoo, B.; Parida, P.P. Load Balancing in Cloud Computing: A Big Picture. Journal of King Saud University – Computer and Information Sciences, 2020, 32(2), 149–158.
[17] Delimitrou, C.; Kozyrakis, C. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Salt Lake City, UT, USA, 2014; pp. 127–144.
[18] Grandl, R.; Ananthanarayanan, G.; Kandula, S.; Rao, S. Multi-Resource Packing for Cluster Schedulers. In Proceedings of the ACM SIGCOMM Conference, Chicago, IL, USA, 2014; pp. 455–466.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)