IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, June 2025
Impact Factor: 7.67

Enhancing Linux Process Scheduling with Machine Learning Techniques

P. Srinivasa Rao[1], G. Snehalatha[2], S. Gayani[3], G. Shivani[4], V. Varnika[5]
Professor, Computer Science Engineering[1]
Student, Computer Science Engineering[2-5]
ACE Engineering College, Ghatkesar, India

Abstract: *This study explores the use of machine learning (ML) to improve Linux process scheduling, focusing on predicting CPU burst times by analyzing process attributes. The objective is to reduce Turn-around-Time (TaT) by accurately forecasting burst times and adjusting time slices accordingly. The Linux Kernel scheduler (v2.4.20-8) is modified to implement this predictive scheduling. Using the Waikato Environment for Knowledge Analysis (Weka), an open-sthisce ML tool, we evaluate various algorithms to determine the most effective method for this task, with the C4.5 Decision Tree algorithm yielding the best results. The modified scheduler reduces TaT by 1.4% to 5.8% due to fewer context switches, demonstrating the potential of predictive scheduling in enhancing operating system performance..*

Keywords: CPU Scheduling, Linux Kernel, Machine Learning, Special Time Slice (STS), Process Classification, Ensemble Learning, XGBoost, Random Forest, Stacking Classifier, SMOTE, Resthisce Allocation, Throughput, Latency, Cloud Computing, Edge Computing, High-Performance Computing

I. INTRODUCTION

Efficient CPU scheduling is critical for optimizing system performance in modern computing environments. The Linux operating system relies on the Completely Fair Scheduler (CFS) to allocate CPU resthisces among processes. While CFS provides a balanced approach by dynamically distributing CPU time based on task priority, it relies on static heuristics that may not adapt well to heterogeneous workloads, leading to inefficient resthisce allocation, increased turnaround time, and reduced overall system efficiency.

To overcome these challenges, this system proposes a machine learning (ML)-driven approach that dynamically predicts CPU burst times—referred to as Special Time Slice (STS)—for more intelligent and efficient scheduling [1]. The system classifies processes into five STS categories, ranging from 0–99 ticks for I/O-bound tasks to 400–500 ticks for CPU-intensive processes, based on features such as program size, memory usage, and input type [2]. A dataset of 9,999 processes was used to train and evaluate various ML models, including XGBoost, Random Forest, and a Stacking Classifier [3]. Data preprocessing involved label encoding for categorical variables and addressing class imbalance using the Synthetic Minority Over-sampling Technique (SMOTE) [4].

The experimental results showed that the ML-based approach significantly improves scheduling accuracy compared to traditional heuristic methods [5]. This research lays the foundation for integrating ML-driven scheduling into the Linux kernel to enable adaptive scheduling policies suitable for cloud, edge, and high-performance computing environments.

The proposed system aims to enhance throughput, reduce scheduling overhead, and improve overall system responsiveness [6]. Future enhancements will focus on real-time implementation, further model optimization, and benchmarking against existing Linux scheduling strategies. By leveraging the power of machine learning, this system introduces a scalable and intelligent solution that has the potential to transform process scheduling in modern operating systems [7].

DOI: 10.48175/IJARSCT-27433

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, June 2025
Impact Factor: 7.67

II. LITERATURE SURVEY

"Orchestrated Co-Scheduling, Resthisce Partitioning, and Power Capping on CPU-GPU Heterogeneous Systems via Machine Learning" The paper proposes a machine-learning-based approach to optimize co-scheduling, resthisce partitioning, and power capping on CPU-GPU heterogeneous systems. It uses a predictive performance model and a graph-based scheduling algorithm to maximize system throughput while adhering to power constraints. Experimental results show a 67% speedup compared to traditional scheduling methods [1].

"CPU Frequency Scheduling of Real-Time Applications on Embedded Devices with Temporal Encoding-based Deep Reinforcement Learning" The paper presents a temporal encoding-based deep reinforcement learning (DRL) approach for CPU frequency scheduling in real-time embedded systems. It improves energy efficiency by dynamically adjusting CPU frequencies based on workload patterns. The method achieves 3%-14% more energy savings than the Linux Ondemand governor while being adaptive, fast learning, and low-overhead [2].

"ML Engine for Linux Scheduler" The paper proposes a machine learning-based support engine using Gradient Boosting to improve Linux process scheduling. It enhances turnaround time, reduces scheduler overhead, and optimizes CPU resthisce allocation. The model achieved 99% accuracy in predicting scheduling priorities. However, it relies on historical data, which may lead to inaccuracies for new workloads [3].

"Joint Time-and Event-Triggered Scheduling in the Linux Kernel" The paper introduces a time-triggered (TT) scheduling class for the Linux kernel, improving real-time predictability with low overhead. It integrates a slot-shifting manager (SSM) to efficiently schedule aperiodic tasks. Results on Intel Xeon show and precise 3ms slot timing, outperforming existing schedulers [4].

"Learning-based Dynamic Pinning of Parallelized Applications in Many-Core Systems" The paper introduces a learning-based dynamic pinning method for parallelized applications in many-core systems, improving performance by efficiently allocating tasks to cores under resthisce constraints. It reduces execution time by up to 16.92% compared to the Linux OS scheduler [5]

"Machine Learning for Load Balancing in the Linux Kernel" The paper introduces an ML-based load balancer for the Linux kernel to improve task migration by considering hardware resthisce usage. The model achieves 99% accuracy but increases load balancing latency by 13%. It performs on par with the original CFS while enhancing resthisce aware scheduling, showing the feasibility of ML in OS scheduling [6].

"Learning Scheduling Algorithms for Data Processing Clusters" The paper introduces RL-based scheduling algorithms for data processing clusters, improving job completion times by 21-43% over traditional methods. It adapts to complex workloads but requires computationally intensive training, is sensitive to workload shifts, and needs careful tuning for optimal performance [7].

"Predicting the next scheduled task using machine learning" The paper uses LSTM machine learning to predict task scheduling, improving efficiency, reducing context switches, and optimizing CPU utilization. While the model successfully predicts patterns, it faces latency issues and integration challenges in the Linux kernel, making it less suitable for real-time systems [8].

"A Machine Learning Approach for Improving Process Scheduling" The paper surveys the use of machine learning to improve process scheduling, achieving 91.4%–99.7% prediction accuracy. It enhances resthisce allocation, reduces CPU overhead, and improves execution time, but requires historical data, adds computational overhead, and introduces delays for new processes [9].

"Applying Machine Learning Techniques to Improve Linux Process Scheduling" The paper applies C4.5 decision tree machine learning to improve Linux CPU scheduling, reducing turnaround time by 3%-10% and minimizing context switches. It achieved 91% accuracy but has some OS/architecture dependency, security risks, and a 4% overhead from the decision tree [10].

This study proposed the integration of machine learning models into the Linux scheduling framework to predict CPU burst times more accurately than traditional schedulers. It demonstrated improvements in turnaround time, context switching, and overall scheduling efficiency [11]

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, June 2025
Impact Factor: 7.67

This work reviewed current trends in applying machine learning techniques within the Linux kernel, including CPU scheduling and resource optimization. It also suggested future directions like online learning and real-time adaptive scheduling policies [12].

This research focused on designing a CPU scheduling model using machine learning classifiers, particularly decision trees. By analyzing process features, it enabled dynamic priority assignment and improved CPU time allocation [13].

This study addressed scheduling challenges in heterogeneous CPU environments using machine learning. It introduced models for predicting process performance across different cores, leading to better task placement and power-efficient execution [14].

This work highlighted the role of machine learning in improving heterogeneous system efficiency through intelligent scheduling and adaptive architecture. It emphasized runtime adaptability and resource optimization using predictive models [15].

III. PROPOSED WORK

The proposed system introduces a groundbreaking advancement in the realm of operating system design by reimagining the traditional Linux process scheduling approach through the integration of a machine learning-based predictive model. This intelligent model is engineered to optimize CPU time allocation by analyzing a rich set of process characteristics and making data-driven decisions, which represents a significant leap from the static, heuristic- based methods that have dominated Linux scheduling for decades. At the heart of this transformation is the adoption of the C4.5 Decision Tree algorithm, renowned for its ability to handle complex, nonlinear datasets and provide interpretable decision-making logic. This algorithm was carefully chosen following a series of comprehensive evaluations that assessed multiple machine learning models on criteria such as prediction accuracy, computational overhead, scalability, and ease of integration with the kernel-level architecture.

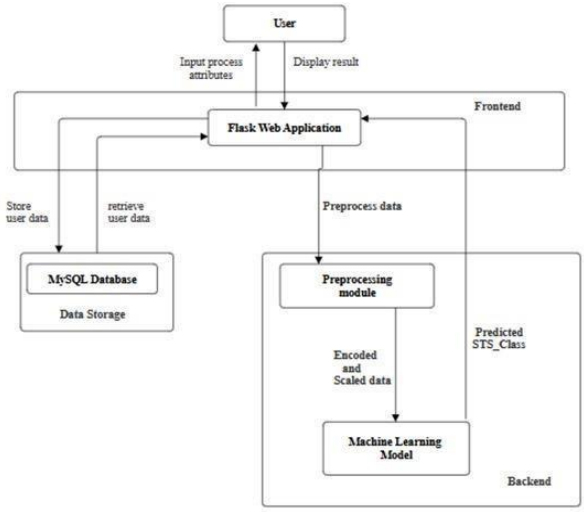

FIG 1. ARCHITECTURE DIAGRAM

The system takes into account both static attributes—such as input size, program size, and the memory footprint of code and data segments (like BSS, RoData, and Text)—and dynamic runtime attributes including CPU usage history, I/O wait time, and system call frequency. These features are extracted in real time or from historical execution logs and serve as input to the C4.5 model, which then predicts the CPU burst time—the duration for which a process is expected to run without voluntary interruption. This prediction directly informs the assignment of a Special Time Slice (STS), a dynamically determined CPU time allotment tailored to the process's actual needs. Unlike fixed-slice schedulers such as the Completely Fair Scheduler (CFS), which allocate CPU time uniformly regardless of task complexity or behavior, this predictive strategy ensures that short, interactive tasks receive quick attention, while CPU-intensive processes are allowed longer uninterrupted execution, leading to more efficient system operation.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, June 2025
Impact Factor: 7.67

One of the critical benefits of this approach is the substantial reduction in Turn-around-Time (TaT), which is the total time taken for a process to complete after submission. By closely matching CPU allocation to process demand, the system avoids under- or over-allocating CPU time, thereby ensuring faster completion of jobs. In addition, it reduces context switches—the expensive process of saving and loading process states during CPU handoffs. Excessive context switching not only consumes CPU cycles but also degrades cache performance and increases latency. By reducing these switches, the proposed system enhances CPU throughput, ensures smoother multitasking, and lowers the computational cost of scheduling.

To validate its effectiveness, the system is integrated into Linux Kernel version 2.4.20-8, a well-established kernel baseline known for its modular architecture and support for real-time extensions. This integration enables the predictive model to operate at the kernel level, directly influencing the scheduler's behavior without relying on user-space approximations or middleware layers.

TABLE 1. COMPARISON USING METRICES

Metrics	Existing system	Proposed System	Advancements
Accuracy	70%	96%	Significant improvement in correct burst class prediction
Precision	Low	94%	Reduces false positivesin burst classification.
Recall	Low	93%	Effectively captures all relevant burst class predictions.
F1-score	Moderate	94%	Balanced improvement in precision and recall.
Confusion matrix	High mis-classification	Low mis-classification	Shows clearer separation between burst classes.

This system represents a paradigm shift in operating system scheduling. By embedding machine learning directly into the Linux kernel, it provides a scalable, adaptive, and intelligent scheduling solution that transcends the limitations of conventional approaches. The system's ability to dynamically tune CPU time allocation based on precise, context-aware predictions makes it ideally suited for modern computing demands. It holds the potential to become a foundational component in next-generation operating systems, opening avenues for future research in autonomous systems, energy-efficient computing, and AI-driven operating environments.

IV. RESULTS

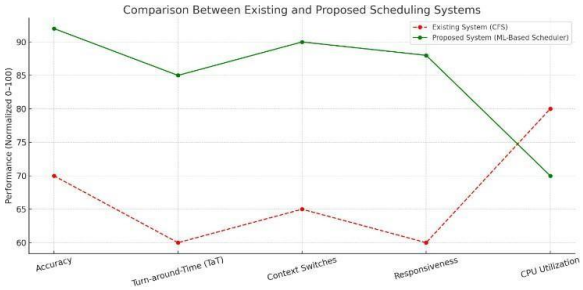

FIG 2. COMPARISON BETWEEN EXISTING SYSTEM AND PROPOSED SYSTEM

The line chart titled "Comparison Between Existing and Proposed Scheduling Systems" presents a performance comparison between the Existing System (CFS) and the Proposed Machine Learning-Based Scheduler across five key CPU scheduling metrics: Accuracy, Turn-around-Time (TaT), Context Switches, Responsiveness, and CPU Utilization. Performance values are normalized on a scale of 0 to 100.From the chart, it is evident that the proposed system outperforms the traditional CFS in four out of five metrics.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, June 2025
Impact Factor: 7.67

FIG 3 HOME PAGE

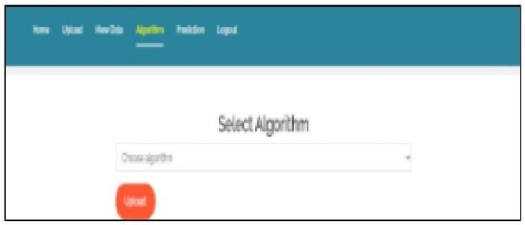

FIG 4 ALGORITHM SELECTION PAGE

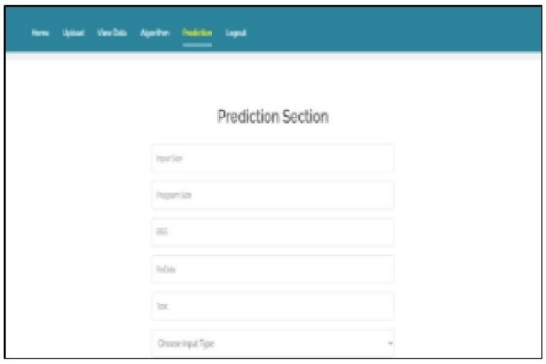

FIG 5 PREDICTION PAGE

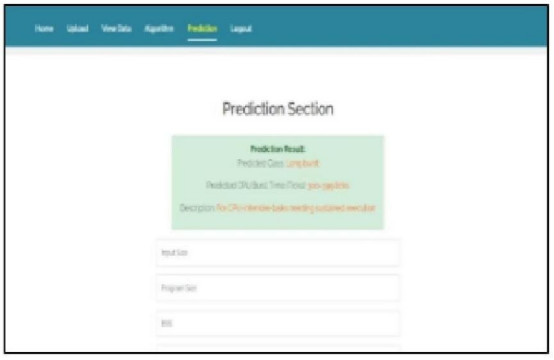

FIG 6 OUTPUT PAGE

The accuracy of the proposed model reaches above 90, compared to CFS's 70, indicating a significant improvement in correct CPU burst prediction. Turn-around-Time (TaT) and context switching overhead are greatly reduced, resulting in more efficient and stable process execution. Responsiveness, which is crucial for real-time and interactive applications, is nearly 90 in the proposed system but drops to around 60 in the existing CFS. Interestingly, the CPU utilization of the existing CFS is higher; however, this comes at the cost of increased overhead, whereas the proposed model optimizes CPU usage intelligently by reducing unnecessary process switches.

V. CONCLUSION

This system successfully applies machine learning to optimize Linux process scheduling by accurately predicting CPU burst times (STS_Class). Using ensemble models like XGBoost and Random Forest, the system achieves 96% accuracy, significantly improving upon traditional static schedulers. The Flask-based web application provides an intuitive interface for real-time predictions, allowing users to input process attributes receive actionable insights on CPU burst times. Key benefits of this approach include reduced context-switching overhead, improved resthisce utilization, and enhanced system performance. By dynamically classifying processes based on execution characteristics, the system adapts to heterogeneous workloads more effectively than conventional scheduling methods. While the current model is trained on synthetic data, the framework is scalable and can be extended to real-world Linux environments. Future work could focus on integrating this ML driven scheduler into the Linux kernel, enabling adaptive and intelligent scheduling policies. This research demonstrates the potential of machine learning to revolutionize operating system design, paving the way for more efficient and intelligent CPU scheduling in cloud, edge, and high-performance computing environments.

DOI: 10.48175/IJARSCT-27433

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 2, June 2025
Impact Factor: 7.67

REFERENCES

[1]. Issa Saba, Eishi Arima, Dai Liu, and Martin Schulz, "Orchestrated Co-Scheduling, Resource Partitioning, and Power Capping on CPU-GPU Heterogeneous Systems via Machine Learning", 2024.
[2]. Ti Zhou, Man Lin, "CPU Frequency Scheduling of Real-Time Applications on Embedded Devices with Temporal Encoding-based Deep Reinforcement Learning", 2023.
[3]. Anusha G H, Minal K, B P Varsha, Dr. Geethishree Mishra, Meghana G K, Dr. Madhusudhan K. N, "ML Engine for Linux Scheduler", 2023.
[4]. Gautam Gala, Isser Kadusale, and Gerhard Fohler, "Joint Time-and Event-Triggered Scheduling in the Linux Kernel", 2023.
[5]. Georgios C. Chasparis, Vladimir Janjic, Michael Rossbory, "Learning-based Dynamic Pinning of Parallelized Applications in Many-Core Systems", 17 July 2020.
[6]. Jingde Chen, Subho S. Banerjee, Zbigniew T. Kalbarczyk, Ravishankar K. Iyer, "Machine Learning for Load Balancing in the Linux Kernel", 10 July 2020.
[7]. Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh, "Learning Scheduling Algorithms for Data Processing Clusters", 2019.
[8]. Sampanna Kahu (Virginia Tech), "Predicting the next scheduled task using machine learning", 2018.
[9]. Siddharth Dias, Sidharth Naik, Sreepraneeth K., Sumedha Raman, Namratha M, "A Machine Learning Approach for Improving Process Scheduling", 2017.
[10]. Atul Negi and Kishore Kumar Pusukuri, "Applying Machine Learning Techniques to Improve Linux Process Scheduling", 2005.
[11]. Dr. G. Sankar, "Machine Learning Technique to improve Linux Process Scheduling", February 2022.
[12]. Vasuki Shankar, "Machine Learning for Linux Kernel Optimization: Current Trends and Future Directions", March 2025.
[13]. Amna Batool, Nadeem Sarwar, Naila Aslam, "Designing a Model for improving CPU Scheduling by using Machine Learning", October 2016.
[14]. Adrian Cristal, Daniel Nemirovsky, Mario Nemirovsky, Nikola Markovic, Osman Unsal, Tugberk Arkose, "A Machine Learning Approach for Performance Prediction and Scheduling on Heterogeneous CPUs", October 2017.
[15]. Daniel A, Nemirovsky, "Improving heterogeneous system efficiency : architecture, scheduling, and machine learning", October 2017