SAGE-OS: A Self-Evolving, AI-Driven Operating System for Cross-Architecture Adaptability

Ashish Vasant Yesale

May 2025

Abstract

The rapid diversification of hardware architectures and the increasing complexity of computing environments demand operating systems (OS) that can autonomously adapt and optimize performance. This paper introduces SAGE-OS (Self-Adaptive, Generative, and Evolving Operating System), a novel OS designed by Ashish Vasant Yesale to leverage artificial intelligence (AI) for self-evolution and seamless adaptability across diverse hardware platforms, starting with the Raspberry Pi. By embedding AI at the kernel level, SAGE-OS autonomously generates drivers, optimizes resource allocation, and adapts to both known and emerging architectures without manual intervention. We present the conceptual framework, architectural design, implementation strategy, and evaluation methods for SAGE-OS, positioning it as a transformative solution in OS design. This work aims to bridge the gap between static OS frameworks and the dynamic needs of modern computing, offering a blueprint for future autonomous systems.

1 Introduction

The proliferation of heterogeneous computing platforms—from ARM-based IoT devices to x86 servers and specialized AI accelerators—poses significant challenges for traditional operating systems. Conventional OS designs, such as Linux and Windows, rely on manually crafted drivers and updates, limiting their ability to adapt to new hardware in real-time [1]. This rigidity results in increased development overhead and delayed support for emerging architectures. SAGE-OS, conceived by Ashish Vasant Yesale, addresses these challenges by integrating AI at the kernel level to create a self-evolving OS capable of autonomous adaptation, optimization, and hardware support. This paper outlines the vision, architecture, and implementation plan for SAGE-OS, emphasizing its potential to redefine OS design for the era of intelligent computing.

2 Background

Traditional operating systems operate on static rule-based frameworks, requiring human intervention for updates, driver development, and performance tuning [2]. Recent advancements in AI have introduced self-adaptive systems in domains like cloud computing and autonomous vehicles [3], yet their integration into OS kernels remains limited. Research highlights AI's potential to enhance OS functionalities, such as memory management, process scheduling, and intrusion detection [4]. Projects like Ubuntu AI and Tesla's proprietary AI OS demonstrate specialized AI integration, but none achieve the level of self-evolution proposed by SAGE-OS [5]. Large Language Model (LLM)-based systems, such as BabyAGI and MemGPT, automate workflows but operate at the application layer, not the kernel [6]. SAGE-OS builds on these foundations, aiming for a fully autonomous, kernel-level AI-driven system.

3 Proposed Solution: SAGE-OS Architecture

SAGE-OS is designed with a modular, AI-centric architecture to enable self-evolution and cross-architecture adaptability. The system comprises three core components:

1. **Kernel Core**: A lightweight, hardware-agnostic microkernel implemented in Rust for memory safety and performance. It handles essential tasks like memory management, process scheduling, and interrupt handling.
2. **AI Engine**: An embedded AI module utilizing lightweight frameworks (e.g., TensorFlow Lite Micro) to monitor system performance, detect hardware configurations, and generate adaptive solutions.
3. **Adaptation Layer**: A dynamic layer that synthesizes drivers and configurations on-the-fly, guided by the AI engine, using LLVM for runtime code generation.

Figure 1: Conceptual architecture of SAGE-OS, illustrating the interplay between the kernel core, AI engine, and adaptation layer.

The AI engine employs reinforcement learning (RL) and lightweight LLMs (e.g., quantized TinyLlama) to analyze telemetry data and optimize system behavior. For hardware adaptation, it uses ML classifiers to detect architecture traits and generates drivers via a custom domain-specific language (DSL) compiled through LLVM. This architecture ensures SAGE-OS can support legacy platforms (e.g., x86, ARM) and emerging architectures (e.g., RISC-V) without predefined configurations.

4 Implementation Plan

The development of SAGE-OS is structured in four phases to balance immediate functionality with long-term autonomy:

- **Phase 1: Core OS Development (0–12 months)**: Implement a minimal microkernel in Rust, targeting the Raspberry Pi (ARM architecture). Include a bootloader (C/Assembly), basic drivers (e.g., UART), and a command-line interface (CLI). Establish cryptographic foundations (e.g., secure boot using Libsodium).
- **Phase 2: AI Integration (12–24 months)**: Embed a lightweight AI engine using TensorFlow Lite Micro for tasks like dynamic scheduling and hardware detection. Develop a DSL for driver synthesis, integrated with LLVM for runtime compilation.
- **Phase 3: Self-Evolution (24–36 months)**: Enable the AI engine to generate and validate kernel patches or drivers autonomously, using RL for optimization and symbolic reasoning (e.g., Z3) for code verification. Add internet connectivity for external data retrieval (e.g., hardware specs from GitHub).
- **Phase 4: Cross-Architecture Expansion (36+ months)**: Extend support to x86, RISC-V, and other architectures via a hardware abstraction layer (HAL). Enhance the AI engine to adapt to unknown architectures using meta-learning techniques.

Key technologies include: - **Rust**: For kernel and system components, ensuring safety and portability. - **TensorFlow Lite Micro**: For embedded AI inference. - **LLVM**: For dynamic code generation and cross-architecture compilation. - **Libsodium**: For cryptographic security (e.g., secure boot, code signing).

The initial prototype in Phase 1 is expected to yield a bootable OS within 12 months, with subsequent phases building incrementally toward full autonomy.

5 Evaluation

SAGE-OS will be evaluated using quantitative and qualitative metrics to assess its performance, adaptability, and autonomy:

Table 1: Evaluation Metrics for SAGE-OS

Metric	Description
Boot Time	Time from power-on to CLI readiness, targeting <5 seconds on Raspberry Pi.
Resource Usage	CPU, memory, and energy consumption compared to Linux and QNX.
Adaptability	Success rate of autonomous driver generation for new hardware (e.g., simulated USB devices).
Autonomy	Percentage of system updates performed without human intervention.
Security	Resilience against AI-driven exploits, measured via penetration testing.

Evaluation methods include: - **Benchmarking**: Compare SAGE-OS against Linux and QNX on identical hardware (e.g., Raspberry Pi 4) for boot time and resource usage. - **Adaptability Testing**: Simulate hardware changes (e.g., new peripherals in QEMU) to assess the AI engine's driver synthesis accuracy. - **Security Analysis**: Conduct fuzzing and penetration tests to evaluate the robustness of AI-generated code and cryptographic safeguards. - **Longitudinal Testing**: Run SAGE-OS continuously for 24+ hours to measure stability and self-healing capabilities.

6 Challenges and Mitigation Strategies

SAGE-OS faces several challenges, including: - **Resource Constraints**: Running AI at the kernel level on resource-limited devices like Raspberry Pi requires optimization. Mitigation includes using quantized models and efficient frameworks like uTensor. - **Stability Risks**: AI-generated code may introduce bugs. A sandboxed testing environment and rollback mechanisms will ensure safe updates. - **Security Concerns**: Autonomous updates increase the attack surface. Cryptographic verification (e.g., ECDSA signatures) and secure boot will mitigate risks. - **Scalability**: Adapting to unknown architectures is complex. Meta-learning and internet-sourced data (e.g., GitHub repositories) will enhance adaptability.

7 Conclusion

SAGE-OS, developed by Ashish Vasant Yesale, represents a paradigm shift in operating system design, harnessing AI to create a self-evolving, hardware-agnostic platform. Its potential to reduce development overhead, enhance system longevity, and adapt to diverse computing environments positions it as a pioneering solution for future computing. While challenges like resource constraints and security remain, the proposed modular architecture and phased implementation plan provide a viable path forward. Future work will focus on real-world deployment, scalability to embedded systems, and collaboration with the open-source community to refine its capabilities. SAGE-OS could inspire a new generation of autonomous, intelligent operating systems, reshaping the landscape of computing.

8 Acknowledgments

The author acknowledges the open-source community and research in AI-driven systems for providing foundational insights and tools that inform the development of SAGE-OS. The project repository is available at `https://github.com/AshishYesale7/SAGE-OS`.

References

[1] Tanenbaum, A. S., & Bos, H. (2006). *Modern Operating Systems*. Prentice Hall.

[2] Love, R. (2010). *Linux Kernel Development*. Addison-Wesley.

[3] Buyya, R., Srirama, S. N., Casale, G., et al. (2019). A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade. *ACM Computing Surveys*, 51(5), 1–38.

[4] Operating System And Artificial Intelligence: A Systematic Review. (2024). *arXiv:2407.14567v1*. `https://arxiv.org/html/2407.14567v1`.

[5] Best AI Operating Systems: A Comprehensive Overview. (2025). *Walturn Insights*. `https://www.walturn.com/insights/best-ai-operating-systems-a-comprehensive-overview`.

[6] LLM OS Guide: Understanding AI Operating Systems. (2024). *DataCamp Blog*. `https://www.datacamp.com/blog/llm-os`.