Intelligent Automation of Operating System Service Management: A Bot-Driven Approach

Daniil Kondrashov
Kyrgyz-German Institute of Applied Informatics
Bishkek, Kyrgyzstan
dnccira@gmail.com

***Abstract*—Manual administration of background processes in modern Operating Systems (OS) is a laborious task that often leads to suboptimal system performance. This study introduces an autonomous software agent designed to regulate, oversee, and streamline OS services. By utilizing real-time monitoring and algorithmic decision-making, the proposed solution ensures efficient resource utilization. Additionally, the paper examines the application of advanced code generation techniques and algorithmic optimization strategies to maximize the reliability and adaptability of the automation tool.**

***Index Terms*—Service Orchestration, System Automation, Python, Performance Tuning.**

I. Introduction

Contemporary computing environments rely on a multitude of background services to manage hardware interactions and network protocols. For administrators and advanced users, the selective enablement of these services is crucial for maintaining system responsiveness. However, manual intervention is time-consuming and inefficient.

To overcome these limitations, we propose an intelligent automation agent capable of managing OS services dynamically. The primary objective is to develop a system that adjusts the state of background processes in response to current workload demands, thereby enhancing stability and freeing up system resources.

Our development approach is influenced by recent breakthroughs in algorithmic efficiency and robust software generation. We specifically adopt resource optimization frameworks similar to those analyzed by Usupova and Khan [1] to ensure low-overhead operation. Furthermore, to ensure the safety of automated command execution, we incorporate structural logic inspired by the ablation and code generation studies of Rakimbekuulu et al. [2].

II. Methodology

The developed tool functions as an intermediary layer, bridging the gap between high-level user requirements and low-level kernel operations.

A. System Architecture

The Bot consists of three main modules:

- **Observer Module:** Continuously polls the state of system services (e.g., Active, Inactive, Suspended).
- **Logic Engine:** Analyzes collected data against resource thresholds (CPU/RAM usage) to determine necessary state transitions.
- **Command Dispatcher:** Translates decisions into system-specific directives (utilizing tools like `systemctl` or `sc.exe`) for execution.

B. Optimization and Logic

To prevent the automation tool from becoming a resource burden itself, we implement strict efficiency protocols. Drawing parallels to the equation optimization in machine learning training [1], our system evaluates the computational ”cost” of restarting a service versus keeping it idle.

Moreover, dynamic command generation carries inherent risks. To mitigate potential system failures, we employ a modular design strategy. This approach, informed by modular code generation techniques [2], ensures that a failure in the command syntax generation remains isolated, preventing critical OS crashes.

III. Implementation

The prototype is implemented in Python, leveraging native OS APIs for service enumeration and control.

- **Adaptive Profiles:** The system supports user-defined usage scenarios, such as ”Performance Mode” or ”Office Mode.” For instance, activating ”Performance Mode” triggers the immediate suspension of non-essential utilities like the Print Spooler or Telemetry services.
- **Fail-safe Mechanism:** A strict whitelist protocol is hard-coded to prevent the accidental termination of essential kernel processes, ensuring system integrity.

IV. Conclusion

This paper outlined the design and implementation of an automated agent for OS service management. By delegating routine maintenance to an intelligent script, users can experience improved system responsiveness and reduced manual workload. Future iterations of this project aims to incorporate predictive maintenance using deep learning algorithms to preemptively identify failing services.

REFERENCES

[1] E. Usupova and A. Khan, ”Optimizing ML Training with Perturbed Equations,” *2025 6th International Conference on Problems of Cybernetics and Informatics (PCI)*, Baku, Azerbaijan, 2025, pp. 1-6, doi: 10.1109/PCI66488.2025.11219819.

[2] S. Rakimbekuulu, K. Shambetaliev, G. Esenalieva and A. Khan, ”Code Generation for Ablation Technique,” *2024 IEEE East-West Design & Test Symposium (EWDTS)*, Yerevan, Armenia, 2024, pp. 1-7, doi: 10.1109/EWDTS63723.2024.10873640.