Memory Management Visualizer: An InteractiveTool for Real-Time System Monitoring and Historical Analysis

1st Sathvik Guptha A
Integrated Mtech-Software Engineering)
Vellore Institute Of Technology
Chennai, India
sathvikguptha.a2022@vitstudent.ac.in

2nd Mahendra Babu T
Integrated Mtech-Software Engineering
Vellore Institute Of Technology
Chennai, India
mahendrababu.t2022@vitstudent.ac.in

***Abstract*—The Memory Management Visualizer is an interactive application designed to monitor and analyze system resourceusage in real time. By leveraging the Psutil library for data collection, the application provides detailed insights into RAM usage, disk space, CPU load, and network throughput. The data is visualized through an intuitive dashboard developed in Dash, featuring alert mechanisms when system thresholds are exceeded. The application also logs historical data, allowing usersto review past performance trends and detect issues preemptively.This project is intended for users and students interested in understanding and managing system memory, making it especially valuable in educational environments and IT resource management**

I. Introduction

Efficient memory management plays a crucial role in ensuring the stability, responsiveness, and performance of modern computer systems. With the growing complexity of computational workloads and data-driven applications, the need for real-time visualization and analysis of system resources has become increasingly important. Understanding parameters such as memory utilization, CPU load, disk space, and network throughput enables users to maintain optimal performance and identify potential bottlenecks proactively. In the absence of efficient monitoring mechanisms, systems may experience performance degradation, resource starvation, or memory leaks that can severely affect both end-user experience and operational efficiency.

Traditional monitoring tools, such as the Windows Task Manager and built-in Linux utilities, provide only limited functionalities for performance evaluation. These tools generally lack advanced capabilities such as historical trend tracking, customizable alert thresholds, and integrated visualization of system metrics. As a result, users often face difficulties in diagnosing resource-related issues or predicting potential performance bottlenecks. Such constraints become more evident in environments that demand continuous system reliability, such as data centers, software testing laboratories, and real-time embedded applications [8], [9].

Recent advancements in interactive visualization and educational simulation tools have demonstrated the potential of graphical analysis in understanding complex operating system behavior. Frameworks such as WebVizOS [1], VMSim [2], CPU-Vis [3], and PARACACHE [4] provided valuable insights into the visualization of memory management and CPU scheduling, particularly in academic contexts. These tools emphasized the importance of bridging theoretical operating system concepts with hands-on experimentation through visual interfaces. Similarly, research efforts in the area of performance visualization, including memory management schemes [5], fault simulation tools [6], and traffic storage visualization systems [7], established the foundation for integrating system metrics into comprehensible graphical dashboards.

Building on these developments, the Memory Management Visualizer project introduces an integrated and interactive platform designed for real-time monitoring and analysis of system resource utilization. By leveraging the Psutil library for low-level system data collection and the Dash framework for dynamic web-based visualization, the proposed system provides a user-friendly dashboard that continuously tracks RAM usage, disk space, CPU load, and network throughput. It incorporates threshold-based alert mechanisms that notify users when predefined limits are exceeded, thereby enabling proactive identification of potential performance issues. Moreover, it stores system metrics as historical logs, allowing users to analyze performance trends and detect recurring anomalies over time.

The key objectives of this project are to:
(1) design and implement a real-time monitoring dashboard for system resources.
(2) enable user-defined threshold alerts for critical parameters.
(3) record and visualize historical usage data to support trend analysis.
(4) enhance user comprehension through intuitive graphical interfaces. Together, these objectives support both educational and practical goals by offering users the ability to study, evaluate, and optimize resource allocation dynamically.

Beyond its functional benefits, the Memory Management Visualizer also contributes significantly to the field of software testing and system performance analysis. Real-time visualization of system metrics supports testing frameworks by correlating application behavior with resource consumption patterns, allowing testers to validate efficiency and detect performance regressions early in the development lifecycle. Prior studies in system monitoring frameworks [10], [11], [13] have highlighted the importance of integrating real-time diagnostics into testing environments, particularly for cloud and distributed systems. This tool extends such frameworks by offering a lightweight, standalone solution suitable for educational and local environments, where accessibility and ease of understanding are critical.

Furthermore, efficient resource visualization has implications in energy efficiency and hardware optimization. Understanding CPU and memory workloads enables developers to fine-tune applications, reduce redundant processing, and manage power consumption effectively. In educational contexts, this project serves as a practical extension of theoretical topics in operating systems, computer architecture, and performance testing. It allows students and researchers to experiment with real-time data, fostering deeper comprehension of how resource allocation impacts overall system behavior.

As an essence, the Memory Management Visualizer aims to bridge the gap between traditional monitoring tools and modern, data-driven visualization frameworks. Through real-time tracking, threshold alerts, and historical performance analysis, it provides a comprehensive approach to understanding and managing system resources. The integration of interactive dashboards, combined with the modularity of the underlying architecture, positions this tool as a valuable contribution for both academic exploration and practical system diagnostics [12], [14], [15].

II. Related Works

Effective software testing, particularly in performance analysis, relies heavily on robust system monitoring and visualization. The Memory Management Visualizer draws upon and extends principles established in several key areas of research, including educational simulation tools, general system monitoring frameworks, and specific performance analysis techniques. This section reviews relevant works, highlighting their contributions and positioning our project within this landscape.

A. Educational and Visualization Tools

Early work focused on demystifying complex operating system concepts through interactive visualization. Papers such as [1] H. Pham et al. (2022), with their WebVizOS framework, demonstrated the efficacy of web-based, interactive simulations for OS learning. Similarly, [2] J. L. B. Garcia et al. (2019) introduced VMSim, a dedicated tool for visualizing virtual memory management, and [3] M. R. T. Santos et al. (2019) presented CPU-Vis for CPU scheduling algorithms, both aiming to improve student comprehension through visual means. [4] A. Paramita et al. (2017) developed PARACACHE, an educational simulator for cache and virtual memory, providing interactive learning experiences. These tools laid the groundwork for effective real-time data representation. Our Memory Management Visualizer extends this educational utility by providing a practical, rather than purely simulated, environment for understanding live system behavior, which is directly applicable to diagnosing application performance. [12] H. P. d. S. e Sá et al. (2010)'s OS-Circus further reinforces the value of educational tools for fundamental OS concepts, which our visualizer complements by offering a live monitoring counterpart.

B. System Performance Monitoring and Analysis Frameworks

Beyond educational simulations, research has explored comprehensive frameworks for system performance. [8] S. T. J. T. Jansen et al. (2022) proposed a real-time system monitoring framework focused on anomaly detection, a crucial aspect of identifying unexpected performance deviations. [9] W. De Pauw et al. (2015) introduced a visual analytics framework to aid in performance for IoT applications that demand real-time steering program performance analysis, emphasizing responses. the importance of visual correlation across diverse data sources. For specialized environments, [13] X. Liu et al. (2015) delved into kernel-level monitoring for I/O performance in virtualized systems, highlighting the need for deep insights in complex architectures. Our visualizer builds on these monitoring principles by offering a streamlined, accessible interface specifically tailored for application-level performance diagnostics, integrating real-time alerts that these broader frameworks might not offer at the user-friendly level.

C. C.Specialized Memory and Performance Visualization Tools

Specific visualization tools have been developed to address particular performance challenges. [5] K. Vimal and A. Trivedi (2015), for instance, focused on a memory management scheme for Android to enhance application performance, demonstrating the impact of efficient resource handling. At a lower level, [6] T. Wu et al. (2024) explored fault simulators for memory defects, underlining the critical role of diagnostic tools. [14] A. Bhatele et al. (2021) presented MemViz, a tool for visualizing complex memory-access patterns to aid in code optimization. For large- scale data, [7] Q. Dan (2023) designed a traffic storage system with open-source visualization, while [15] T. Ho et al. (2017) addressed visual performance analysis for massively parallel applications. Our Memory Management Visualizer integrates and simplifies the core benefits of these specialized tools, providing a unified view of CPU, RAM, Disk, and Network, specifically tailored for detecting application-induced performance issues relevant to software testing.

D. Tools and Surveys in Performance Testing

[10] S. B. P. Valadão et al. (2018) introduced an interactive and extensible tool for performance testing of web-based applications, emphasizing the need for flexible and user-friendly testing environments. Furthermore, [11] S. M. G. de B. Costa et al. (2019) provided a valuable survey on

performance monitoring and testing for cloud applications, offering a comprehensive overview of existing techniques and identifying gaps. Our Memory Management Visualizer distinguishes itself by focusing on real-time, local system monitoring with integrated, user-definable alerts and historical logging. Unlike general performance testing suites, our tool offers a granular, black-box view of an application's resource footprint, making it an indispensable asset for developers and QA engineers to diagnose specific performance regressions and memory leaks at the operating system level, which complements existing application-specific testing tools. Service Computational Resource Management Strategy Based on Edge-Cloud Collaboration. Li et al. (2023) tackle theissue of high latency in IoT by implementing an edge-cloud collaborative approach to resource management. The proposed framework enables efficient prioritization of resources across edge and cloud computing environments, improving stability with long-time efficiency

III. Methodology

The Memory Management Visualizer employs a modular architecture, which integrates data collection, storage, and visualization into distinct yet interconnected components. This design enhances the ease of tracking and monitoring system metrics in real time. Key modules include a user-friendly interface built with Dash, real-time data collection via the Psutil library, historical data storage in CSV format, and a system for alerting based on pre-set thresholds. These elements provide users with comprehensive insights into the state of system resources such as RAM, disk usage, network throughput, and CPU load.

A. Architecture Overview

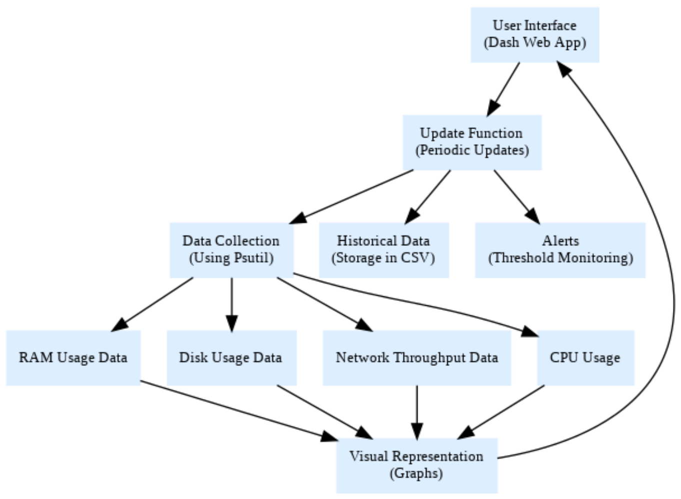

Fig. 1. Key components and modules of the Memory Management Visualizer.

- **User Interface (Dash Web App):** The primary interface for user interaction is built using Dash, a Python-based framework for creating interactive web applications. Through this interface, users can view up-to-date system metrics displayed in easily interpretable graphs and charts. The dashboard is designed to auto-refresh, ensuring that users have access to real-time data at all times. Customizable thresholds allow users to define alert levels for each metric, tailoring the visualization to specific monitoring needs.
- **Update Function (Periodic Updates):** Data collection is conducted at intervals of two seconds, balancing the need for real-time updates with efficient resource usage.Each update triggers the collection of system metrics, which are subsequently displayed on the dashboard and logged for historical reference. The periodic update system allows for continuous, unobtrusive monitoring of system health.
- **Data Collection (Psutil):** The Psutil library serves as the core of the data collection module. Psutil offers functionsto access a wide array of system metrics:
- **RAM Usage:** psutil virtual memory() retrieves memory statistics, including total, available, used, and percentage usage. This enables the visualizer to monitor memory load dynamically.
- **Disk Usage:** psutil disk usage('/') collects disk storage information, focusing on total capacity, used space, free space, and percentage utilization, offering a clear view of available disk resources.
- **Network Throughput:** psutil net io counters() tracks network activity by monitoring bytes sent and received over the network interface, allowing the user to observe network data rates.

  CPU Load: psutil cpu percent() tracks CPU utilization on a real-time basis, providing an overview of CPU workload.
- **Historical Data (CSV Storage):** Collected data points are stored in a CSV file format to facilitate long-term storage and trend analysis. This file serves as a log for each five- second interval, capturing the state of each metric over time. By maintaining a continuous record,the visualizer enables users to review historical trends, perform system analyses, and identify periods of high resource demand.
- **Alerts (Threshold Monitoring):** Threshold monitoring provides an immediate alert mechanism for potential system performance issues. Pre-defined threshold values, such as 80 percent for memory or CPU usage, are set to trigger alerts if these values are exceeded. When an alert condition is met, visual indicators appear on the dashboard, signaling the need for user attention. This-proactive feature supports preventive maintenance and helps mitigate performance bottlenecks before they affect system operations.
- **Visualization (Graphs and Charts):** The real-time data is presented through line charts and gauges, offering a quick visual snapshot of system resource usage. Each-graph is updated with every five-second interval to display the latest metric values, making it easy to spot spikes in usage or sustained trends at a glance. The graphs cover

individual resource areas (RAM, disk, CPU, network) to allow for targeted analysis.

B. Data Collection and Storage Process

The Psutil library is used to collect system metrics, which are then processed and stored. Specific functions are called to retrieve the following:

- **RAM Usage:** Using psutil virtual memory(), the program accesses both overall memory statistics and specific usage values. These details help assess how much of the system's memory is in use at any given time.
- **Disk Usage:** psutil disk usage('/') provides a breakdown of disk utilization, allowing users to gauge available storage capacity.
- **Network Throughput:** Network data, collected through psutil.net io counters(), gives insight into the volume of data transmitted and received, which can highlightnetwork activity trends.
- **CPU Load:** With psutil cpu percent(), the program captures the processor's workload in real-time, useful for monitoring performance stability under varying demands. These metrics are processed and appended to a CSV file every five seconds, which accumulates data for both real-time visualization and historical review. This process is efficient and lightweight, minimizing impact on system resources.

C. Workflow

- **Initialization:** Upon startup, the Dash-based application initializes with default threshold values, such as 80 percent for CPU and memory alerts. These thresholds are modifiable by users based on their specific monitoring needs. The interface components, including the graphs and alert indicators, are set up, preparing for incoming data.
- **Data Update:** Every two seconds, system metrics are refreshed, collected via Psutil, and updated in both the CSV file and the Dash dashboard. This periodic update system provides real-time data access, making the visualizer useful for monitoring current system status.
- **Visualization Update:** The dashboard interface refreshes with each data update, displaying the latest figures for each system metric. The line charts and gauges allow users to interpret data intuitively, supporting the quick identification of trends or resource-intensive processes.
- **Alerts:** If resource usage exceeds user-defined thresholds, alerts are displayed on the dashboard, drawing attention to potential issues. For example, if RAM usage crosses 80percent, an alert signals the need to investigate memory- intensive processes. This early warning system is keyfor performance management and helps prevent excessiveresource strain.
- **Historical Data Logging:** Data points are logged into the CSV file under a structured format that includes timestamps for each update cycle. This log supports both trend analysis and historical performance reviews, allowing users to examine data over time. Historical data provides valuable insights, such as patterns of peak usage or extended periods of low demand, enabling a more detailed understanding of system behavior. The modular setup and periodic update design ofthe Memory Management Visualizer ensure continuous, real- time access to essential system metrics. With historical data logging and alerts, the application serves as a comprehensive tool for understanding and managing resource usage, allowing for both proactive maintenance and retrospective analysis. This methodology providesa robust foundation for system monitoring, supporting both immediate insights and long-term trend evaluation.

IV. Results and Discussion

RESULTS The Memory Management Visualizer successfully tracked and displayed real-time system metrics including RAM usage, diskspace utilization, network throughput, and CPU load. Below are the findings based on the functionality and outputs of the system:

A. RAM Usage Monitoring

The application accurately monitored RAM usage, displaying the percentage of used and free RAM with a gauge. The following metrics were observed during testing:

- **Normal System Load:** RAM usage ranged between 30-50 percent, with no alerts triggered
- **High System Load:** When system load increased, the RAM usage gauge reflected values exceeding 80 percent, and alerts were triggered accordingly.

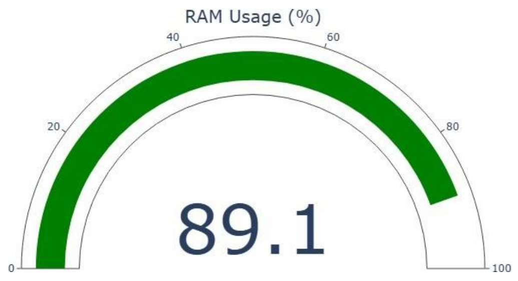

Fig. 2. Gauge chart for RAM - usage real-time gauge displaying the percentage of RAM utilization with threshold-based alerts.

B. Disk Usage Monitoring

Disk usage was effectively monitored, with results displayed as a percentage of used disk space. The following trends were observed:

- **Initial Disk Usage:** Disk utilization started at approximately 50 percent on average.
- **Threshold Alert:** When a threshold of 80 percent was set, the application displayed alerts upon exceeding this limit, allowing immediate attention to disk space constraints.

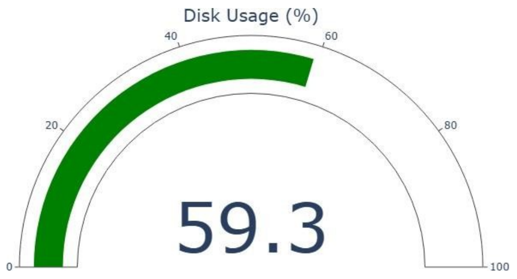

Fig. 3. Disk Usage Gauge – Illustrates the percentage of disk space used and triggers alerts if usage exceeds the set threshold

These visuals helped present the data objectively and allowed users to quickly assess resource consumption. Each graph was refreshed in real-time to keep the information displayed up to date.

C. Network Throughput Monitoring

Network I/O was tracked with line graphs that displayed data as bytes sent and received. During high data transfer activities (e.g., file uploads or downloads):

- **Bytes Sent:** Increased linearly with network-intensive tasks.
- **Bytes Received:** Also increased predictably with download activities, providing real-time insights into data flow.

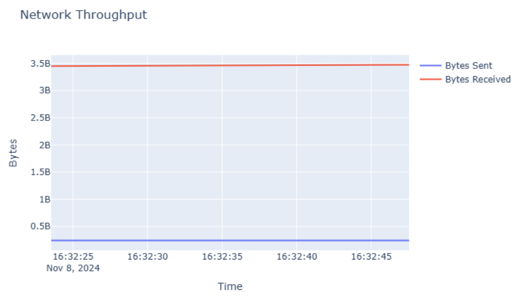

Fig. 4. Network Throughput Line Graph – Shows bytes sent and received,with peaks indicating high data transfer activity

D. CPU Load Monitoring

The application tracked CPU usage in real-time, and significant findings included:

- **Normal Conditions:** CPU usage remained below 50 percent on average.
- **Intensive Tasks:** During processor-heavy operations, CPU usage rose above 80 percent, and alerts were generated, giving a visual and textual cue.

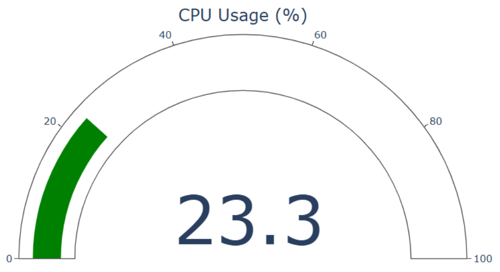

Fig. 5. CPU Usage Gauge – Monitors and alerts based on the user-definedCPU threshold

E. Historical Data Logging

The system stored historical data in the system monitoring history.csv file, capturing trends in RAM, disk, network, and CPU usage over time. This feature allowed users to view usage trends, helping identify patterns related to system performance.

F. Summary of Key Findings

- **Data Visualization**: The above figures were presenting the data clearly. These visuals helped present the data objectively and allowed users to quickly assess resource consumption. Each graph was refreshed in real-time to keep the information displayed up to date.
- **Alert Accuracy:** The application consistently triggered alerts based on pre-set thresholds for each monitored metric.
- **Trend Analysis:** Historical data storage enabled long-term tracking of system metrics, useful for understanding memory trends.
- **Real-time Responsiveness:** Each component's real-time updates provided users with immediate feedback on system performance, supporting responsive resource management.

Discussion

The results indicate that the Memory Management Visualizer meets its objective of providing real-time insights into system performance. The key findings contribute significantly to the field by offering both real-time alerts and historical trend analysis, addressing limitations identified in traditional system monitors like Task Manager.

1. Significance Of Findings:

- **Improved Monitoring Accuracy:** The tool's capability to provide real-time, threshold-based alerts supports immediate response to system constraints, which is crucial in educational and IT management contexts.
- **Enhanced Data Visibility:** Visualization of network, RAM, and CPU usage aids in a better understanding of resource demands and allocation, especially valuable for users learning about operating system resource management.

2.Comparison with Previous Studies: Unlike traditional tools, this application:

- Allows user-defined alert thresholds, a feature not typically available in standard monitoring tools.

- Tracks historical usage trends in CSV format, which is advantageous for identifying resource usage patterns overtime.
- Integrates interactive visualizations that are more accessible and informative for users, especially compared to static resource managers.

3. Unexpected Findings:

- While the application performed effectively under most conditions, it occasionally recorded brief spikes in CPU and RAM usage that could not be immediately explained by active processes. This could potentially indicate transient background activities within the operating system, which merits further investigation to improve accuracy.

4. Limitations:

- **Limited to Single Machine Monitoring:** The application currently only monitors the system it runs on and cannot support networked or multi-system monitoring.
- **Resource Overhead:** Running the visualization tool, especially in a web-based format, slightly increased system resource usage, which could impact performance on low-end systems.
- **Dependency on Psutil:** The Psutil library may not capture certain types of data accurately on all operating systems, which could limit the application's generalizability.

Future Research Directions

- Enhanced Data Storage: Consider storing historical data in a more robust database (e.g., SQLite) to support more complex queries and analyses.
- Multi-System Monitoring: Extending the application to monitor multiple devices within a network would increase its applicability in enterprise environments.
- User Customization: Adding more customization options for alerts and visual themes could make the application more user-friendly and adaptable to different needs.

As an overall, the Memory Management Visualizer effectively fulfills its role in real-time monitoring and visualization of system metrics, contributing to both educational purposes and practical system diagnostics. By addressing the identified limitations and expanding its capabilities, future versions of the tool could provide even greater value to users across various fields.

V. Conclusion

The Memory Management Visualizer effectively meets its objective of providing an interactive, real-time tool for monitoring critical system metrics such as RAM usage, disk space, network throughput, and CPU load. Through customiz- able threshold-based alerts and historical data logging, theapplication empowers users to proactively manage system resources and understand usage trends over time. Key findingsindicate that the tool's real-time responsiveness and detailed visualizations offer significant improvements over traditional monitoring solutions, making it particularly useful in educational settings and IT diagnostics. The broader implications of this research lie in its potential to foster deeper insights into operating system resource man- agement. By allowing users to detect performance bottlenecks and observe historical trends, this tool serves as a valuable resource for system administrators, students, and educators. Practical applications extend to real-time diagnostics in en- terprise environments and as an educational tool for learning about system memory dynamics. This project's contribution to the field is underscored by its ability to combine real-time monitoring, trend analysis, and alert mechanisms within a single platform, offering a comprehensive approach to memory management visualization. Future developments could include expanding the tool to support multi-system monitoring and enhanced customization, which would further increase its utility and impact in diverse settings.

References

[1] H. Pham, H. N. Le, T. N. D. Nguyen, and K. T. Tran, "A Web-based Interactive and Visualized Approach to Simulations of Operating Systems," in *Proc. 2022 IEEE International Conference on Information and Education Technology (ICIET)*, 2022, pp. 245–251.

[2] J. L. B. Garcia, D. G. de M. Salgado, and P. S. L. de Souza, "VMSim: A Visualization Tool for Teaching Virtual Memory Management," *IEEE Transactions on Education*, vol. 62, no. 4, pp. 276–284, Nov. 2019.

[3] M. R. T. Santos, G. C. D. Santos, and R. de M. M. Vale, "CPU-Vis: A Visualization Tool for CPU Scheduling Algorithms," in *Proc. 2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, 2019, pp. 1–6.

[4] A. Paramita, D. P. K. W., and A. A. de Fretes, "PARACACHE: Educational Simulator for Cache and Virtual Memory," in *Proc. 2017 International Conference on Information Technology Systems and Innovation (ICITSI)*, 2017, pp. 416–420.

[5] K. Vimal and A. Trivedi, "A Memory Management Scheme for Enhancing Performance of Applications on Android," in *Proc. 2015 IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 1916–1921.

[6] T. Wu, H. Fan, J. Gu, and J. He, "A New Test Algorithm and Fault Simulator of Simplified Three-Cell Coupling Faults," *IEEE Transactions on Device and Materials Reliability*, vol. 24, no. 2, pp. 283–292, Jun. 2024.

[7] Q. Dan, "Design of Traffic Storage System Based on Hybrid Storage and Open-Source Visualization Software," in *Proc. 2023 IEEE International Conference on Computer, Information and Communication Technology (CICT)*, 2023, pp. 43–47.

[8] S. T. J. T. Jansen, R. S. M. de R. S. Torres, and A. C. B. Garcia, "A Real-Time System Monitoring Framework for Performance Anomaly Detection," in *Proc. 2022 IEEE International Conference on Cloud Computing (CLOUD)*, 2022, pp. 21–30.

[9] W. De Pauw, B. R. de Supinski, and M. Schulz, "A Visual Analytics Framework for Steering Program Performance Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 1, pp. 116–129, Jan. 2015.

[10] S. B. P. Valadão, R. S. M. de Souza, and F. S. F. Pereira, "An Interactive and Extensible Tool for Performance Testing of Web-Based Applications," in *Proc. 2018 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 2018, pp. 464–469.

[11] S. M. G. de B. Costa, M. A. S. Netto, and P. R. F. Cunha, "A Survey on Performance Monitoring and Testing for Cloud Applications," in *Proc. 2019 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2019, pp. 408–415.

[12] H. P. d. S. e Sá, A. d. A. B. da Silva, and F. M. Quintão, "OS-Circus: A New Educational Tool for Operating Systems," in *Proc. 2010 IEEE International Conference on Education and Management Technology*, 2010, pp. 22–26.

[13] X. Liu, J. Mellor-Crummey, and M. Fagan, "Kernel-Level Monitoring and Analysis of I/O Performance in Virtualized Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1147–1159, Apr. 2015.

[14] A. Bhatele, T. Gamblin, and K. E. Isaacs, ”MemViz: A Tool for Visualizing Memory-Access Patterns,” in *Proc. 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2021, pp. 22–32.

[15] T. Ho, Y. Chen, and K. Chen, ”Visual Performance Analysis of Massively Parallel Applications,” in *Proc. 2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 868–875.

[16] J. H. Boockmann and G. Lüttgen, ”Shape-Analysis Driven Memory Graph Visualization,” in *Proc. 30th IEEE/ACM International Conference on Program Comprehension (ICPC)*, 2022.

[17] R. Patriarca, G. Di Gravio, and F. Costantino, ”myFRAM: An Open Tool Support for the Functional Resonance Analysis Method,” in *Proc. 2017 2nd International Conference on System Reliability and Safety*, 2017.

[18] A. Magdy, L. Alarabi, S. Al-Harthi, M. Musleh, T. M. Ghanem, S. Ghani, S. Basalamah, and M. F. Mokbel, ”Demonstration of Taghreed: A System for Querying, Analyzing, and Visualizing Geotagged Microblogs,” in *Proc. 31st IEEE International Conference on Data Engineering (ICDE)*, 2015, pp. 1416.

[19] P. Amontamavut, Y. Nakagawa, and E. Hayakawa, ”Separated Linux Process Logging Mechanism for Embedded Systems,” in *Proc. 2012 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2012.

[20] T. Dobravec, ”Java Virtual Machine Educational Tools,” in *Proc. Informatics 2019 IEEE 15th International Scientific Conference on Informatics*, 2019.

[21] A. Barbosa, H. Mubarak, F. Mohammadi, and M. J. Sanjari, ”A Real-Time IoT-Based Data Acquisition and Monitoring System for Photovoltaic Applications,” in *Proc. 2024 3rd International Conference on Power Systems and Electrical Technology (PSET)*, 2024.

[22] H. Li, B. Zhang, Z. Wu, C. Yang, and S. Du, ”A Study on Real-time Monitoring of Meter-driven Data in Power Systems Based on OpenHarmony,” in *Proc. 2024 International Conference on Artificial Intelligence and Power Systems (AIPS)*, 2024.

[23] W. Zhong and H. Zhong, ”Real-Time Processing System of Vineyard Environmental Monitoring Data Based on FPGA,” in *Proc. 2024 International Conference on Internet of Things, Robotics and Distributed Computing (ICIRDC)*, 2024.

[24] J. Zhu, D. Hwang, and A. Sadjadpour, ”Real Time Congestion Monitoring and Management of Power Systems,” in *Proc. 2005 IEEE/PES Transmission and Distribution Conference & Exhibition: Asia and Pacific*, 2005.

[25] Q. Li and X. Guan, ”Architecture of Real-time Dynamic Stability Monitoring and Control System in Northwest Grid Based on Universal Platform,” in *Proc. 7th World Congress on Intelligent Control and Automation*, 2008.

[26] H.-W. Hsiao, M.-H. Shih, S.-H. Tung, and W.-P. Sung, ”Using Botnet Structure to Construct the Communication System of a Real-time Monitoring Platform,” in *Proc. 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2017.

[27] S. Jain, L. R. Welch, D. M. Chelberg, Z. Tan, D. Fleeman, D. Parrott, B. Pfarr, M. C. Liu, and C. Shuler, ”Collaborative Problem Solving Agent for On-Board Real-Time Systems,” in *Proc. 16th International Parallel and Distributed Processing Symposium*, 2002.

[28] H. T. Yew, S. K. Chung, M. F. Ng, and A. Chekima, ”IoT Based Real-Time Remote Patient Monitoring System,” in *Proc. 2020 16th IEEE International Colloquium on Signal Processing & its Applications (CSPA)*, 2020.

[29] L. Wang and K.-H. Liu, ”Implementation of a Web-Based Real-Time Monitoring and Control System for a Hybrid Wind-PV-Battery Renewable Energy System,” in *Proc. 2007 International Power Engineering Conference (IPEC)*, 2007.

[30] J. Zhou, H. Huang, S. Huang, W. Chen, Z. Jiang, and S. Liu, ”Remote Real-time Health Monitoring and Evaluation System for Long Bridge Structure,” in *Proc. IMACS Multiconference on Computational Engineering in Systems Applications (CESA)*, 2006.