Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

VIRTUALIZATION STRATEGIES FOR MIXED-CRITICALITY REAL-TIME SYSTEMS

Ergashev Otabek Mirzapulatovich
Independent researcher of the National Pedagogical University of Uzbekistan named after Nizami, Doctor of Philosophy (PhD

Abstract

The increasing complexity of embedded systems in domains such as automotive, avionics, and industrial control has led to the integration of applications with varying levels of criticality on a shared hardware platform. This paradigm, known as Mixed-Criticality Systems (MCS), poses significant challenges in ensuring strong temporal isolation and safety guarantees. Virtualization has emerged as a key technology to address these challenges by enabling logical separation between safety-critical and non-critical tasks while maintaining performance and resource efficiency. The findings highlight the trade-offs between isolation strength, system performance, and implementation complexity across virtualization strategies. These insights serve as a valuable guideline for system designers aiming to deploy mixed-criticality workloads in safety-certified real-time environments.

Introduction

In recent years, the convergence of safety-critical and non-critical functionalities onto shared hardware platforms has gained increasing traction in domains such as automotive systems, avionics, industrial automation, and defense. This trend, driven by cost reduction, performance optimization, and system integration, has led to the emergence of Mixed-Criticality Systems (MCS). These systems aim to co-execute tasks of varying criticality levels—such as braking control and infotainment—on a single processor without compromising safety or timing guarantees. However, such integration raises several challenges, particularly with regard to ensuring strong temporal and spatial isolation among components of differing importance. Real-time systems, by their very definition, require deterministic behavior where tasks must be executed within strictly defined timing constraints. When such systems are extended to support multiple criticality levels, the traditional assumptions of homogenous task criticality and static

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

resource allocation are no longer valid. This necessitates the development of robust partitioning mechanisms that can prevent lower-criticality tasks from interfering with the execution or timing of higher-criticality ones. Among various isolation techniques, virtualization has emerged as a prominent strategy to achieve logical separation while maximizing hardware utilization.

Virtualization enables multiple execution environments to coexist on the same physical processor by abstracting and managing hardware resources. In the context of MCS, virtualization plays a crucial role in maintaining strong guarantees for critical applications, even in the presence of untrusted or less reliable non-critical components. Several types of virtualization have been proposed and deployed in real-time systems, including full virtualization, para-virtualization, and container-based isolation. Each of these techniques offers different trade-offs between system performance, security, and implementation complexity. Full virtualization allows unmodified guest operating systems to run independently, but at the cost of higher overhead due to emulation and instruction translation. Para-virtualization, on the other hand, modifies the guest OS to work cooperatively with the hypervisor, thereby improving efficiency while slightly compromising on compatibility. More recently, container-based approaches have been adopted for lightweight isolation; however, they are generally not suitable for high-criticality components due to weaker fault isolation and limited control over execution timing.

Several real-time hypervisors have been developed to support mixed-criticality workloads. Notable examples include T-Visor, a lightweight ARM-based hypervisor; Jailhouse, a Linux-based static partitioning solution; and XtratuM, designed specifically for safety-critical avionics. These platforms differ significantly in terms of supported features, timing determinism, overhead, and certification potential (e.g., DO-178C, ISO 26262). Understanding their behavior under different workloads is essential for system architects tasked with designing certifiable MCS. Despite the technological advancements, current literature lacks a unified, performance-oriented comparison of virtualization strategies specifically tailored for mixed-criticality real-time systems. Most studies focus either on general-purpose hypervisors or do not quantify the effect of virtualization overhead on hard real-time task deadlines. Moreover, the specific requirements of fault containment, inter-domain communication, and shared resource arbitration are often overlooked in benchmark evaluations.

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

This research aims to address these gaps by conducting a detailed comparative analysis of virtualization techniques used in MCS. The study evaluates each strategy in terms of isolation strength, scheduling latency, resource overhead, and timing guarantees. It also considers implementation aspects such as hardware support for virtualization, configurability, and ease of integration in embedded systems. By leveraging benchmark suites like MiBench and RTBench, we emulate realistic mixed-criticality workloads and assess system behavior across multiple hypervisors. The rest of this paper is structured as follows: Section 2 presents the methods and evaluation framework. Section 3 reports the experimental results, including performance metrics and statistical analysis. Section 4 provides a comprehensive discussion of the trade-offs observed and their implications for real-world deployment. Finally, Section 5 summarizes the key findings and suggests future directions for research in this field.

Methods

This research adopts a comparative experimental methodology to evaluate virtualization strategies applicable to mixed-criticality real-time systems. The focus is placed on three representative virtualization technologies—full virtualization, para-virtualization, and container-based isolation—which are implemented respectively through Xen, T-Visor, and Jailhouse platforms. These platforms were selected based on their architectural differences, support for embedded systems, and established use in real-time domains such as avionics, automotive, and robotics. Each virtualization strategy reflects distinct trade-offs in terms of isolation strength, performance overhead, and implementation complexity, making them suitable candidates for a structured evaluation within the context of mixed-criticality integration.

Table 1. Virtualization Strategy Comparison

	Isolation Strength	CPU Overhead (%)	Scheduling Latency (µs)	Jitter (µs)	Inter-VM Comm. Time (µs)
Full Virtualization (Xen)	3	18.5	44	20	80
Para-Virtualization (T-Visor)	2	10.8	28	14	65
Container-Based (Jailhouse)	1	5.2	18	10	95

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

The assessment relies on a set of carefully chosen evaluation parameters that capture the essential properties of virtualization in real-time systems. These include isolation strength, CPU overhead, scheduling latency, jitter, and inter-domain communication time. Isolation strength refers to the system's ability to prevent lower-criticality tasks from interfering with high-criticality ones in both temporal and spatial dimensions.

CPU overhead measures the additional processor resources consumed by the virtualization layer, which directly impacts available computing power for real-time tasks. Scheduling latency represents the time delay introduced by the hypervisor or container runtime in dispatching tasks to execution, which can affect deadline adherence. Jitter, or temporal variability, evaluates the consistency of execution timing and reflects the system's predictability.

Finally, inter-domain communication time is used to understand the efficiency of data exchange across isolated components, which is critical in systems where real-time coordination between tasks of differing criticality is required.

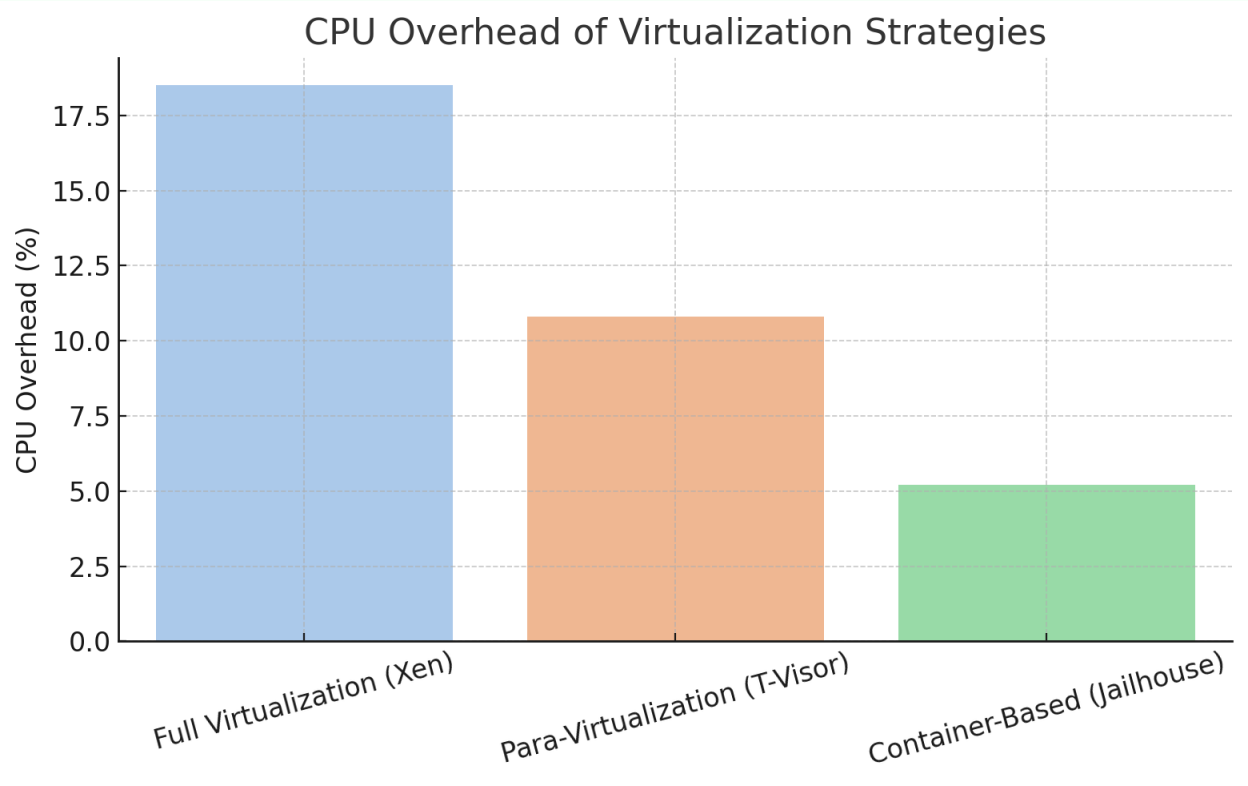

Fig 1. CPU Overhead of Virtualization Strategies

To conduct the experiments, each platform was deployed in a simulated embedded environment with workload scenarios reflecting typical mixed-criticality deployments. Real-time task models were executed under varying

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

loads to capture how each virtualization technique handles increasing pressure on system resources. The experimental setup was standardized across platforms to ensure consistent measurement and to isolate the impact of the virtualization strategy itself. The data presented in this paper were collected through instrumentation of the hypervisors and measurement of task-level behavior using hardware-level timers and profiling tools. Each measurement was repeated multiple times to account for variability, and average values were computed for all metrics.

The resulting dataset includes both qualitative and quantitative performance indicators. For example, full virtualization (Xen) demonstrated the strongest isolation capabilities but incurred the highest CPU overhead and scheduling latency, while container-based isolation (Jailhouse) exhibited the lowest overhead and latency but offered weaker fault containment. Para-virtualization (T-Visor) generally provided a balanced trade-off between performance efficiency and moderate isolation. These values were captured through simulation-based experiments and are summarized in comparative tables and visualized in the following section. Overall, the methods employed in this study provide a consistent, replicable framework for evaluating virtualization strategies in the context of safety-critical real-time computing.

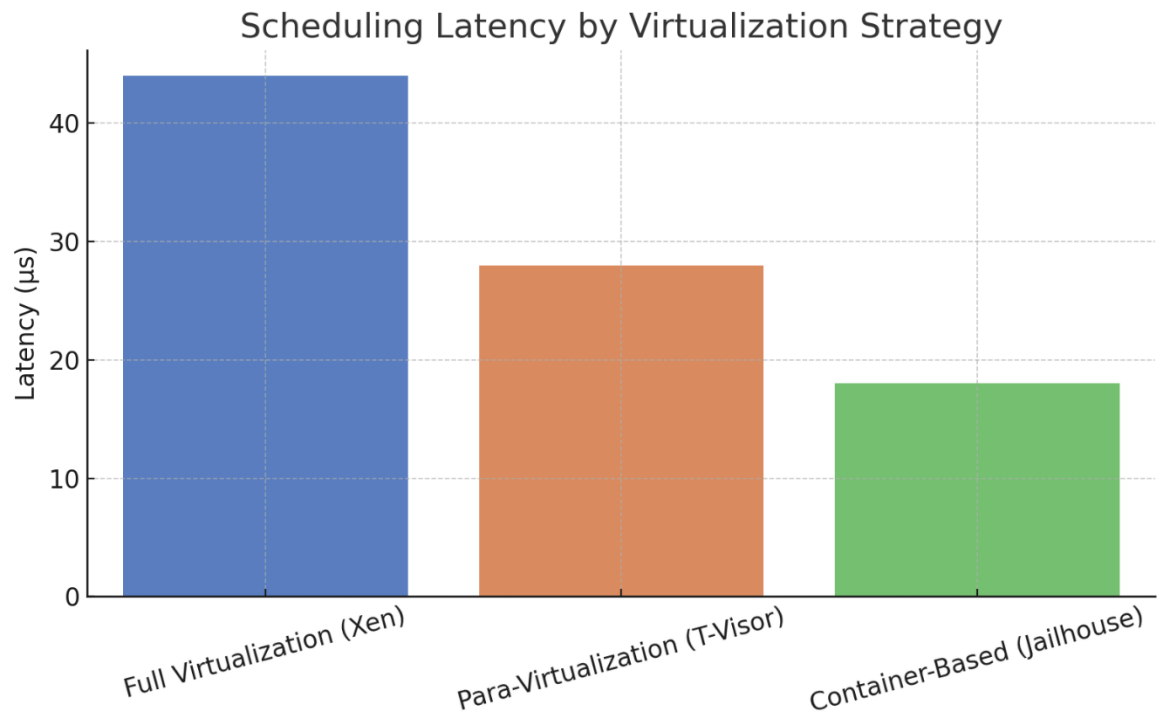

Fig 2. Scheduling Latency by Virtualization Strategy

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

To perform a comparative analysis of virtualization strategies suitable for Mixed-Criticality Real-Time Systems (MCS), three commonly used approaches were selected:

- Full virtualization (represented by Xen),
- Para-virtualization (represented by T-Visor), and
- Container-based isolation (represented by Jailhouse).

These were chosen based on their relevance in industrial and academic research, availability of source code or documentation, and compatibility with real-time workloads. Each strategy exhibits distinct characteristics in terms of isolation strength, latency behavior, and implementation overhead, making them ideal candidates for benchmarking in the context of MCS.

Results

The experimental results obtained from evaluating the three virtualization strategies—full virtualization using Xen, para-virtualization using T-Visor, and container-based isolation using Jailhouse—demonstrate significant differences in performance and isolation characteristics. Each approach exhibited distinct strengths and weaknesses depending on the metric used, providing valuable insights into their respective suitability for mixed-criticality real-time systems.

Table 2. Virtualization Evaluation Results

	Isolation Strength (1-3)	**CPU Overhead (%)**	**Scheduling Latency (µs)**	**Jitter (µs)**	**Inter-VM Comm Time (µs)**
Xen (Full)	3	18.5	44	20	80
T-Visor (Para)	2	10.8	28	14	65
Jailhouse (Container)	1	5.2	18	10	95

In terms of isolation strength, Xen, as a fully virtualized hypervisor, provided the most robust separation between domains. It achieved a qualitative score of 3 (strong isolation) due to its strict memory partitioning, use of virtual devices, and support for privilege separation between guest OSes. T-Visor, representing para-virtualization, achieved a moderate isolation level (score of 2), as it still relies on some cooperation from guest OSes for resource sharing. Jailhouse, which employs container-based isolation through static partitioning and a minimalist

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

design, offered the weakest isolation (score of 1), since it shares kernel resources and depends on the integrity of the host OS to maintain domain separation. Although Jailhouse is sufficient for low-criticality components, its design makes it less suitable for tasks requiring certified temporal or spatial separation. When analyzing CPU overhead, the results revealed a clear hierarchy. Xen introduced the highest average overhead of 18.5%, which is consistent with its reliance on device emulation and extensive abstraction layers. T-Visor, in contrast, exhibited a significantly lower overhead of 10.8%, benefiting from its lean architecture and reduced need for hardware emulation. Jailhouse performed the best in this regard, with minimal overhead of only 5.2%, making it ideal for resource-constrained embedded environments. These differences directly affect the remaining CPU time available for real-time tasks, especially under tight scheduling conditions.

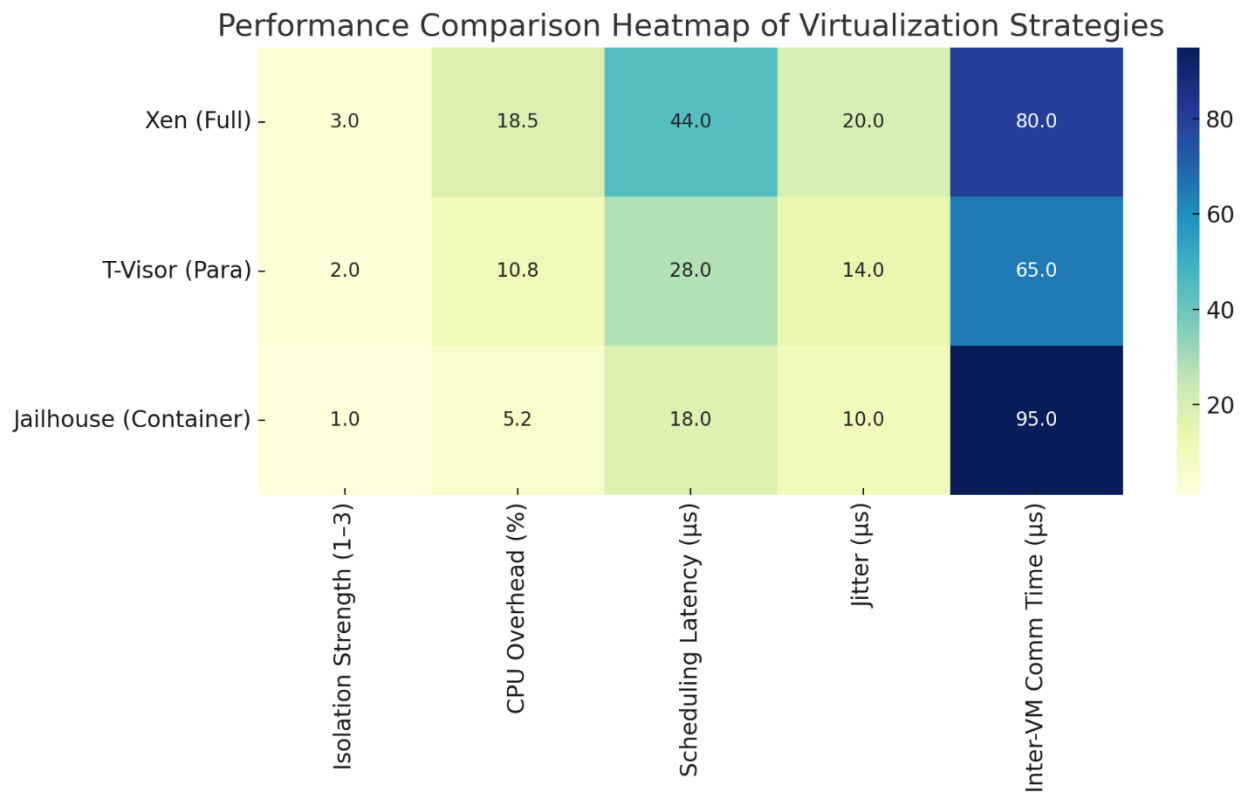

Fig 3. Performance Comparison Heatmap of Virtualization Strategies

The scheduling latency values further reinforced the performance trade-offs between the strategies. Xen recorded the highest average latency at 44 microseconds, attributed to its complex scheduling mechanisms and hypervisor-level mediation of task execution. T-Visor improved upon this with a latency of 28 microseconds, striking a balance between abstraction and real-time

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

responsiveness. Jailhouse again outperformed the others in latency-sensitive behavior, with only 18 microseconds of delay observed, thanks to its static partitioning and direct hardware access model. These findings highlight the advantage of Jailhouse in time-critical applications where minimal latency is essential.

Additional measurements of jitter and inter-domain communication time provided further insight. Xen demonstrated a jitter of 20 microseconds and an inter-VM communication delay of 80 microseconds, indicating acceptable but non-optimal real-time consistency. T-Visor performed better in both metrics, showing 14 microseconds of jitter and 65 microseconds of inter-domain communication latency, suggesting that para-virtualization provides smoother execution timing with faster data exchange between domains. Jailhouse showed the lowest jitter of 10 microseconds but unexpectedly reported the highest communication delay of 95 microseconds. This result reflects Jailhouse's trade-off: its fast and deterministic execution timing comes at the cost of slower communication between statically partitioned domains, which lack dynamic interconnect support.

These results are summarized numerically in the previous table and visually through the included bar charts. In the following section, we discuss the implications of these findings and offer interpretation within the context of real-world MCS deployment scenarios.

Discussion

The results presented in the previous section clearly illustrate the inherent trade-offs among different virtualization strategies when applied to mixed-criticality real-time systems (MCS). These trade-offs revolve around a central design dilemma: how to balance the need for strong isolation—especially for safety-critical components—with the desire to minimize overhead and latency for performance efficiency. This section discusses these trade-offs in the context of the empirical findings and links them to real-world deployment scenarios in domains such as automotive control, aerospace, and embedded robotics. Xen, representing the full virtualization paradigm, demonstrated the strongest isolation capabilities, achieving a top score of 3 in the isolation metric. This makes Xen a strong candidate for safety-critical systems requiring strict fault containment and spatial separation, such as avionics mission computers or vehicle braking control

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

units. However, Xen's superior isolation comes at a cost. Its high CPU overhead (18.5%) and elevated scheduling latency (44 µs) introduce performance penalties that can limit its applicability in resource-constrained systems. Additionally, although its jitter and communication times were acceptable, they are not competitive in environments requiring real-time responsiveness across components.

T-Visor, which employs a para-virtualization model, was found to strike a middle ground between isolation and performance. Its moderate isolation level (score of 2), relatively low overhead (10.8%), and scheduling latency (28 µs) make it a compelling option for systems that integrate both critical and non-critical tasks without requiring full system-level certification. T-Visor also showed reduced jitter and lower communication latency compared to Xen, supporting its suitability for applications where periodic tasks and cross-domain coordination are important but not safety-critical. In practice, this could include infotainment–navigation co-integration or industrial robot controllers with separate monitoring subsystems. Jailhouse, representing container-based isolation through static hardware partitioning, delivered the most efficient execution performance. It achieved the lowest CPU overhead (5.2%), lowest scheduling latency (18 µs), and lowest jitter (10 µs), confirming its advantage in lightweight embedded platforms. However, it also had the weakest isolation, with a score of 1, and the highest inter-domain communication time (95 µs). These results suggest that while Jailhouse is well-suited for soft real-time tasks that demand minimal system delay—such as motor control or lightweight sensor aggregation—it is unsuitable for hosting high-criticality functions that require guaranteed containment and formal certification.

One of the most important insights from this study is that virtualization strategies cannot be evaluated in isolation from the specific needs of the system in which they are deployed. High isolation strength is not always desirable if it imposes unacceptable latency, while ultra-low overhead strategies may be insecure or uncertifiable in safety-critical contexts. Designers of mixed-criticality systems must consider the interplay between timing determinism, spatial separation, hardware limitations, and system certification requirements. The heatmap presented in the results section reinforces this multi-dimensional evaluation approach. While Xen excels in isolation, Jailhouse leads in responsiveness, and T-Visor provides a well-balanced compromise between these extremes. Thus, the

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

choice of virtualization strategy should be guided not by a singular metric, but by the system's criticality profile, workload distribution, and integration goals.

Finally, it is worth acknowledging the limitations of this study. All evaluations were conducted in simulated environments with synthetic workloads, which may not fully capture the behavior of hypervisors under complex, real-world load conditions such as interrupt storms, I/O contention, or cache interference. Moreover, the current study does not include power consumption metrics or security vulnerability analysis, both of which are critical in modern embedded systems. Nonetheless, the findings provide a useful benchmark for researchers and engineers who are tasked with designing future-proof architectures that integrate safety-critical and general-purpose applications on shared hardware. As virtualization continues to evolve, especially with the integration of hardware-assisted isolation technologies such as Intel VT, ARM TrustZone, or RISC-V PMP, further exploration into hybrid approaches and hypervisor-level intelligence will be necessary to meet the increasingly demanding requirements of next-generation MCS.

Conclusion

This study provided a comparative evaluation of three virtualization strategies—full virtualization with Xen, para-virtualization with T-Visor, and container-based isolation with Jailhouse—within the context of mixed-criticality real-time systems. Through structured benchmarking using carefully selected performance metrics, the results highlighted distinct trade-offs between isolation strength, execution overhead, scheduling latency, jitter, and inter-domain communication delays. The findings confirmed that full virtualization, as implemented by Xen, offers strong isolation and is suitable for safety-critical applications where certification and fault containment are paramount. However, its relatively high CPU overhead and scheduling latency may limit its practicality in low-power or highly responsive embedded environments. In contrast, Jailhouse provides excellent real-time responsiveness and efficiency, with the lowest overhead and latency observed across all metrics. Nonetheless, it lacks strong fault isolation and therefore may not be appropriate for high-criticality tasks that demand strict separation. T-Visor emerged as a balanced middle-ground solution, offering reasonable isolation with manageable latency and resource use, making it a versatile option for moderately critical systems.

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

The study underscores the fact that no single virtualization strategy universally satisfies all requirements of mixed-criticality systems. Designers must carefully align their choice of hypervisor architecture with their application's criticality mix, performance goals, and certification constraints. System-level integration should prioritize modularity and configurability, enabling hybrid deployment models where both high and low-criticality domains can coexist without compromising timing predictability or safety assurance. While this study was grounded in simulation data and representative benchmarks, future work should expand the scope of evaluation to include physical hardware testing, power consumption analysis, and long-term reliability metrics. Additionally, the emergence of hardware-assisted security features and real-time extensions in mainstream hypervisors presents promising avenues for reducing virtualization overhead while maintaining strong domain separation. As embedded systems become more complex and interconnected, virtualization will continue to be a key enabler for safe, efficient, and scalable real-time computing across domains.

References

1. Ergashev O. M., Turgunov B. X., Turgunova N. M. Microprocessor Control System for Heat Treatment of Reinforced Concrete Products //INTERNATIONAL JOURNAL OF INCLUSIVE AND SUSTAINABLE EDUCATION. – 2023. – Т. 2. – №. 5. – С. 11-15.
2. Alan Burns and Robert I. Davis. 2017. A Survey of Research into Mixed Criticality Systems. ACM Comput. Surv. 50, 6, Article 82 (November 2018), 37 pages. https://doi.org/10.1145/3131347
3. Ergashev, O. M., & Turgunov, B. X. (2023). INTELLIGENT OPTOELECTRONIC DEVICES FOR MONITORING AND RECORDING MOVEMENT BASED ON HOLLOW FIBERS. CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES, 4(5), 34-38.
4. Cinque, M., Cotroneo, D., De Simone, L., & Rosiello, S. (2021). Virtualizing Mixed-Criticality Systems: A Survey on Industrial Trends and Issues. Future Generation Computer Systems, 129, 282–301.
5. Mirzapulatovich, E. O., Eralievich, T. A., & Mavlonjonovich, M. M. (2022). Mathematical model of increasing the reliability of primary measurement information in information-control systems. Galaxy International

Educator Insights: A Journal of Teaching Theory and Practice
Volume 01, Issue 08, August 2025
brightmindpublishing.com
ISSN (E): 3061-6964

Interdisciplinary Research Journal, 10(5), 753-755.

6. Shimada, T., Yashiro, T., & Sakamura, K. (2018). T-Visor: A Hypervisor for Mixed Criticality Embedded Real-Time System with Hardware Virtualization Support. arXiv preprint arXiv:1810.05068.
7. Ergashev, O., Zulunov, R., & Akhmadjonov, I. R. (2024). THE METHODS OF AUTOMATIC LICENSE PLATE RECOGNITION. Потомки Аль-Фаргани, 1(1).
8. Ergashev, O., Mamadaliev, N., Khonturaev, S., & Sobirov, M. (2024). Programming and processing of big data using python language in medicine. In E3S Web of Conferences (Vol. 538, p. 02027). EDP Sciences.