Proceedings of CONF-MLA 2026 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2026.LD37589

Research on Embedded Hypervisor Technology Selection: Requirements Analysis and Capability Comparison for Extreme Environment Application Scenarios

Zhichen Wang

Pudong Foreign Language School Affiliated to Shanghai International Studies University, Shanghai, China
awzdtd@163.com

Abstract. As embedded devices grow more complex, running multiple operating systems on a single device—and deploying virtualization beyond data centers into embedded systems—has become standard. This paper focuses on the hypervisor Virtual Machine Monitor and examines its selection challenges in extreme environments: disaster rescue, space exploration, deep-sea and polar operations, and security confrontations. We propose a four-layer analytical framework—"Scenario → Threat → Requirement → Capability"—to systematically compare mainstream embedded hypervisors across key dimensions. The framework is validated through real-world cases: the Mars Pathfinder mission, AWS Snowcone edge devices, and the DARPA HACMS red-team experiment. Results show no one-size-fits-all hypervisor for extreme settings, optimal selection depends on specific application context, security needs, and system constraints. In safety-critical domains, certification standards (e.g., DO-178C, ISO 26262) and formal verification—once limited to military applications—are now spreading to critical infrastructure. The selection decision process proposed in this paper can provide a reference for embedded virtualization engineering practice.

Keywords: Embedded virtualization, Hypervisor, Extreme environment, Fault tolerance, Real-time performance

1. Introduction

The hypervisor Virtual Machine Monitor (VMM) sits between hardware and operating systems, enabling virtualization to partition a single physical device into multiple logically isolated execution environments—supporting resource multiplexing, isolation, and dynamic scheduling. Originally developed for mainframe partitioning and later scaled for data center cloud computing, this architecture struggles in embedded systems: they are resource-constrained, operate in harsh environments, and demand deterministic latency and strong fault isolation, not just high throughput or utilization. In extreme applications, such as spacecraft, rescue equipment, and deep-sea detectors, hypervisors face three hard constraints: limited resources, hostile environments, and mission-critical reliability.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.LD37589

Embedded hypervisors address these needs by prioritizing time/space partitioning, deterministic scheduling, low overhead, and high assurance. They enable heterogeneous task integration and functional safety isolation. Yet significant capability gaps remain for disaster response, space exploration, and other extreme scenarios—particularly in offline autonomy, environmental tolerance, fault self-recovery, and multi-node decentralized collaboration.

This report selects seven representative solutions—Jailhouse, seL4, Xen, ARM, ACRN, QNX, XtratuM, and PikeOS—objectively states their characteristics based on official materials and analyzes the advantages and disadvantages of each solution from multiple dimensions, provides conclusions and a general selection decision process, and clarifies the capability boundaries and adaptation paths of existing hypervisor technology systems in extreme environments. The findings provide both theoretical insight and practical guidance for designing virtualized architectures in rescue and exploration systems.

2. Technical principles of embedded virtualization

2.1. Basic concepts of virtualization and hypervisor

A Virtual Machine (VM) is a computer system simulated through software, with an independent operating environment, capable of installing and running its own operating system. The Hypervisor VMM is system software located between physical hardware and virtual machines, responsible for two core tasks: first, resource allocation, dividing CPU, memory, and peripherals among different virtual machines, and second, isolation protection, ensuring that a failure or compromise of one virtual machine does not affect others. The primary goal of virtualization in data centers is to improve resource utilization, while the primary goal of virtualization in embedded devices is functional integration and security isolation.

2.2. Classification and architecture selection of hypervisors

According to architectural position, hypervisors are divided into two types: bare-metal (Type-1) and hosted (Type-2). Bare-metal types run directly on hardware, with representative products including ESXi, KVM, Xen, PikeOS, and seL4, hosted types run as applications on general-purpose operating systems, with representative products including VMware Workstation and VirtualBox. The embedded field generally adopts bare-metal architecture for reasons including smaller footprint, faster boot, more predictable scheduling behavior, and smaller attack surface. These four points directly correspond to the characteristics of embedded devices: resource-constrained, real-time requirements, and high security demands.

2.3. Key mechanisms of embedded virtualization

Embedded virtualization relies on three key mechanisms. The first is the isolation mechanism: the hypervisor uses hardware virtualization extensions (such as ARM Virtualization Extensions) to strictly separate memory and peripheral access of each virtual machine. The second is the real-time determinism mechanism: tasks must be completed within specified time limits, and scheduling behavior must be statically analyzable. The third is the fault tolerance mechanism: when hardware errors occur, the system should be able to self-heal or operate in degraded mode rather than failing completely.

DOI: 10.54254/2755-2721/2026.LD37589

3. Overview of mainstream embedded hypervisor solutions

3.1. Partitioning-type solutions: Jailhouse and XtratuM

Jailhouse is an open-source partitioning hypervisor led by Jan Kiszka and published in the Linux kernel community. Its design goal is static partitioning, that is, hardware resources are fixedly divided among different guests during the configuration phase, and dynamic migration is not supported [1, 2]. Compared with full-featured virtualization solutions such as KVM, Jailhouse adopts a lightweight design with a relatively small core code scale, which can reduce system complexity and improve isolation efficiency. Jailhouse supports Linux and bare-metal applications as Guests and currently holds no safety certification.

XtratuM is a partitioning hypervisor maintained by the Spanish company fentISS, designed for aviation and space scenarios. Multiple missions of the European Space Agency (ESA) have adopted this solution. This report did not find publicly available data on XtratuM code scale, and its detailed performance data is also not publicly available.

3.2. Microkernel-type solutions: seL4, PikeOS, and QNX

seL4 is a microkernel developed by the Australian NICTA institution. It completed formal verification in 2009 and was open-sourced in 2014 [3]. seL4 is one of the earlier operating system kernels to complete formal verification, proving its key security properties satisfy predefined specifications through mathematical methods [4]. seL4 supports running Linux and other guest systems through its built-in Virtual Machine Monitor (VMM). According to 2024 seL4 summit information, institutions such as Collins Aerospace, Kry10, and the UAE Technology Innovation Institute (TII) continue to participate in its ecosystem construction, among which Collins Aerospace participated in the DARPA HACMS project (seL4 Project).

PikeOS is a commercial microkernel hypervisor developed by the German company SYSGO, adopting a partitioning architecture. According to SYSGO official materials, PikeOS supports compliance with aviation domain DO-178C DAL A, Common Criteria EAL5+, and space domain ECSS standards [5]. Airbus Defence and Space has used PikeOS for satellite onboard computer projects (SYSGO).

QNX is a commercial real-time operating system and hypervisor under BlackBerry and is a mainstream solution in the automotive domain. QNX has passed the automotive functional safety standard ISO 26262 ASIL D requirement, and its hypervisor supports guest types such as QNX, Linux, and Android. QNX code scale is not publicly available.

3.3. Full-featured solutions: Xen ARM and ACRN

Xen is an open-source hypervisor originating from Cambridge University, widely used in cloud computing in its early stages. Xen ARM is its version for embedded ARM platforms. Xen ARM supports Linux, Zephyr, and various real-time operating systems as guests, and provides strong device sharing capabilities through virtio and PV drivers, its Dom0less (Hyperlaunch) boot mode enables fast boot. In November 2024, AMD and the Xen community jointly promoted the Xen functional safety certification program, aiming to meet ISO 26262 ASIL D and IEC 61508 SIL 3 levels, but formal certification has not yet been completed (Xen Project) [5, 6].

ACRN is an open-source hypervisor initiated by Intel and hosted by the Linux Foundation, released in 2018. ACRN targets IoT and industrial edge scenarios, characterized by strong device-

DOI: 10.54254/2755-2721/2026.LD37589

sharing capabilities, supporting graphics, audio, USB, and other devices to be shared among multiple guests. Supported guest types include Linux, Zephyr, and Windows. In 2020, ACRN's functional safety concept received approval from TÜV SÜD, confirming that its design can meet IEC 61508 SIL 3 level requirements [7].

3.4. Practical case analysis

To analyze the role of different hypervisor capabilities in actual environments, this paper selects three representative cases, corresponding to real-time determinism, security isolation, and edge deployment requirements.

Mars Pathfinder Priority Inversion Incident (1997) [8]: After landing, NASA's Mars Pathfinder experienced periodic system reset problems. Post-incident analysis showed that the cause was a priority inversion problem in the real-time operating system: a low-priority task held a mutual exclusion lock when interrupted by a medium-priority task, causing the high-priority task to time out, and the watchdog mistakenly judged the system as dead and triggered a reset. Engineers finally solved the problem by remotely uploading a patch and enabling priority inheritance. This incident is classic evidence of the importance of real-time determinism.

DARPA HACMS Red Team Experiment (2012-2015) [9]: The U.S. Defense Advanced Research Projects Agency organized a professional red team to attack an unmanned helicopter. When attacking the original system, the red team took over flight control within minutes, when attacking the same system retrofitted with seL4, the red team failed to break the isolation within six weeks. Even when the project team directly granted the red team root privileges on a Linux virtual machine, the red team still could not affect flight control functions. This experiment is one of the most convincing public comparative data points for virtualization isolation capabilities.

AWS Snow Edge Device Deployment [10]: The AWS Snow series is edge computing equipment for harsh environments. According to AWS official specifications, the Snowcone weighs 2.1 kilograms. After configuration in a networked environment, the device can operate completely offline. Multiple devices can form a cluster to ensure data is not lost during single-point failures. Virtual machine instances can run locally, with underlying Linux and KVM virtualization technology. This product is a typical case of general-purpose virtualization solutions in offline scenarios such as disaster rescue.

4. Capability dimension analysis and scenario requirements analysis

4.1. Code size and lightweight analysis

Code size is an important indicator for embedded hypervisors. The smaller the code size, the lower the cost of manual audit and formal verification and the relatively lower the potential defect risk. Figure 1 summarizes the core code sizes of five solutions with publicly available data. seL4 is approximately 10,000 lines, and Jailhouse is approximately 12,000 lines, both belonging to lightweight solutions, ACRN is approximately 25,000 lines, at an intermediate level, Xen's full code is approximately 100,000 lines, with its ARM version approximately 47,000 lines, having relatively higher complexity. XtratuM, QNX, and PikeOS code sizes are not publicly available and thus cannot be included in the comparison.

DOI: 10.54254/2755-2721/2026.LD37589

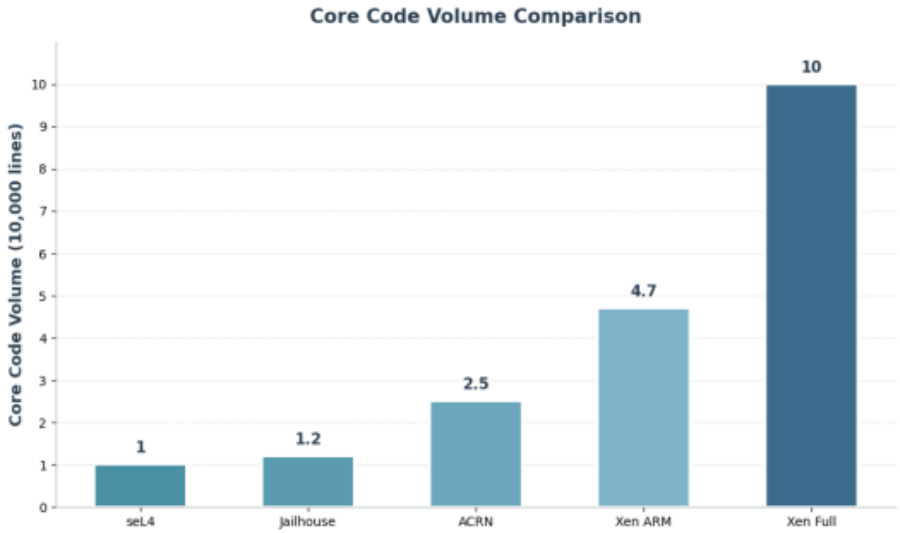

Figure 1. Comparison of core code size of mainstream embedded hypervisors [1, 2, 6, 7]

4.2. Safety certification and isolation capability analysis

Safety certification is a mandatory threshold for entering safety-critical fields such as aviation and automotive. Table 1 summarizes the certification status of the seven solutions. PikeOS has the most comprehensive certification coverage, QNX holds the highest-level certification in the automotive domain, seL4 provides the strongest isolation guarantee through formal verification, and ACRN has received SIL 3 concept approval. Xen's ASIL D certification is still in progress. Jailhouse and XtratuM do not hold publicly available safety certifications [11].

Table 1. Safety certification status of the seven hypervisors [3, 5-7, 11]

Solution	Verification Status
seL4	Formal verification (mathematical proof of functional correctness, world's first)
PikeOS	DO-178C DAL A (highest level in aviation), Common Criteria EAL5+
QNX	ISO 26262 ASIL D (highest level in automotive functional safety)
ACRN	IEC 61508 SIL 3 Concept Approval (2020, TÜV SÜD)
Xen ARM	ASIL D certification in progress (first phase completed November 2024)
Jailhouse	No publicly available certification records
XtratuM	No publicly available certification records

4.3. Device sharing and ecosystem capability analysis

Device sharing capability determines whether multiple virtual machines can share peripherals such as GPU, network, and storage. Xen ARM and ACRN perform strongest in this dimension: Xen supports extensive device virtualization through virtio and PV drivers, and ACRN supports multi-guest sharing of graphics, audio, and USB. In contrast, Jailhouse and XtratuM adopt static resource partitioning methods with weak device-sharing capabilities.

In terms of open-source ecosystems, Xen and ACRN are hosted by large foundations with active communities, Jailhouse and seL4 have smaller but stable communities, QNX, PikeOS, and XtratuM are commercial or semi-closed solutions with ecosystems dependent on vendor support.

DOI: 10.54254/2755-2721/2026.LD37589

4.4. Evaluation summary table and capability radar chart

Table 2 summarizes the performance of the seven solutions across dimensions such as code size, boot time, guest types, device sharing, and safety capabilities. Among them, code size, guest types, device sharing, and certification status mainly come from publicly available materials, due to the lack of unified test conditions across different platforms, boot time and resource overhead data vary significantly, so some indicators are not quantitatively compared.

Table 2. Five-dimensional evaluation of the seven hypervisors [1, 2, 5-7]

Solution	Code Size	Boot Type	Guest Type	Device Sharing	Safety Certification
Jailhouse	~12,000 lines	/	Linux, bare-metal apps	Weak	None
seL4	~10,000 lines	/	Linux, etc. (VMM)	Weak	Formal verification
ACRN	~25,000 lines	/	Linux, Zephyr, Windows	Strong	SIL 3 concept approval
Xen ARM	~47,000 lines	<1 second	Linux, Zephyr, RTOS	Strong	ASIL D in progress
QNX	/	/	QNX, Linux, Android	Medium	ASIL D
PikeOS	/	/	Linux, AUTOSAR,etc.	Medium	DO-178C、EAL5+
XtratuM	/	/	RTOS, Linux	Weak	No public records

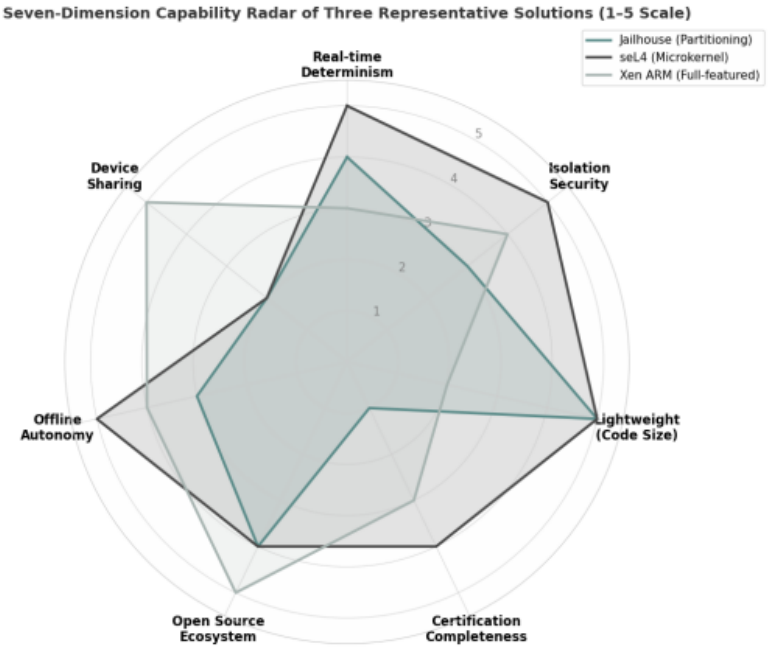

Figure 2. Seven-dimension capability radar of three representative solutions [1, 2, 5-7]

Comprehensive comparison reveals that the profiles of the three solutions differ significantly. seL4 leads in isolation security and being lightweight, but has weak device-sharing capabilities, Xen ARM leads in device sharing and open-source ecosystems, but is the least lightweight, Jailhouse is between the two, with good real-time performance but no safety certification [1]. No solution leads in all dimensions.

DOI: 10.54254/2755-2721/2026.LD37589

4.5. Scenario requirements analysis and conclusions

Different extreme environments have different priority requirements for hypervisors. Based on the aforementioned capability analysis, this paper discusses four typical application scenarios.

Disaster rescue scenario: The core constraints of this scenario are network disconnection and power outage. As shown in Section 4.3, full-featured solutions have advantages in device support and ecosystem, the AWS Snow case also shows that general virtualization technology can support offline edge computing [10]. Therefore, the disaster rescue scenario prioritizes recovery capability and offline autonomy, recommending KVM-like edge solutions or Xen, ACRN.

Space exploration scenario: The core constraints of this scenario are radiation-induced random hardware errors and non-repairability. The Mars Pathfinder case shows the importance of real-time determinism, and Table 1 shows that PikeOS has relatively comprehensive aviation safety certification support, while seL4 provides formal verification capabilities [3, 5, 9]. Avionics partitioning scheduling design adopts a two-level task model to ensure its determinism [12]. Therefore, the space exploration scenario recommends PikeOS or seL4, with certification and verifiability prioritized over other indicators.

Deep-sea and polar exploration scenario: The core constraints of this scenario are energy limitation and long-term unmanned operation. As shown in Section 4.1, seL4 and Jailhouse have lower code complexity, which helps reduce system maintenance costs. Therefore, the deep-sea and polar scenario prioritizes lightweight and long-term stability, recommending seL4, Jailhouse, or QNX.

Security confrontation scenario: The core constraint of this scenario is malicious human attacks. The HACMS case shows that seL4's isolation mechanism has strong security value, and PikeOS has a high security evaluation level [5, 9]. Therefore, the security confrontation scenario recommends seL4 or PikeOS, with isolation security as the first priority.

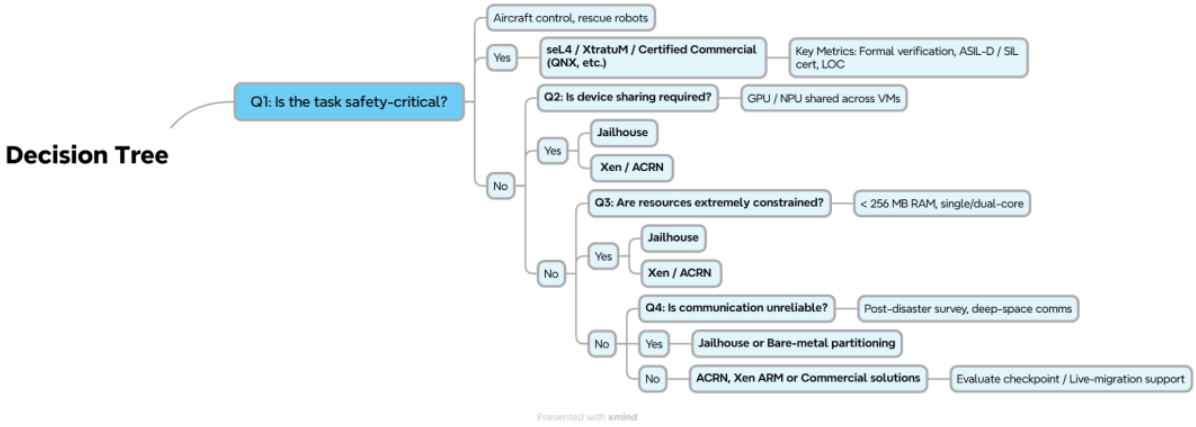

Figure 3. Decision flow for embedded hypervisor selection in extreme environments

Based on the aforementioned capability dimension analysis and scenario requirements analysis, this paper further summarizes a set of hypervisor selection processes for extreme environment applications (see Figure 3). Users can narrow the candidate range by sequentially answering four yes/no questions.

DOI: 10.54254/2755-2721/2026.LD37589

5. Development trends

Trend 1: Lightweight advancement continues. Figure 4 shows that between 2009 and 2019, emerging solutions were controlled at 10,000 to 25,000 lines from the beginning of design, while established solutions were also continuously streamlined, with Xen compressing its ARM version from approximately 100,000 lines to approximately 47,000 lines.

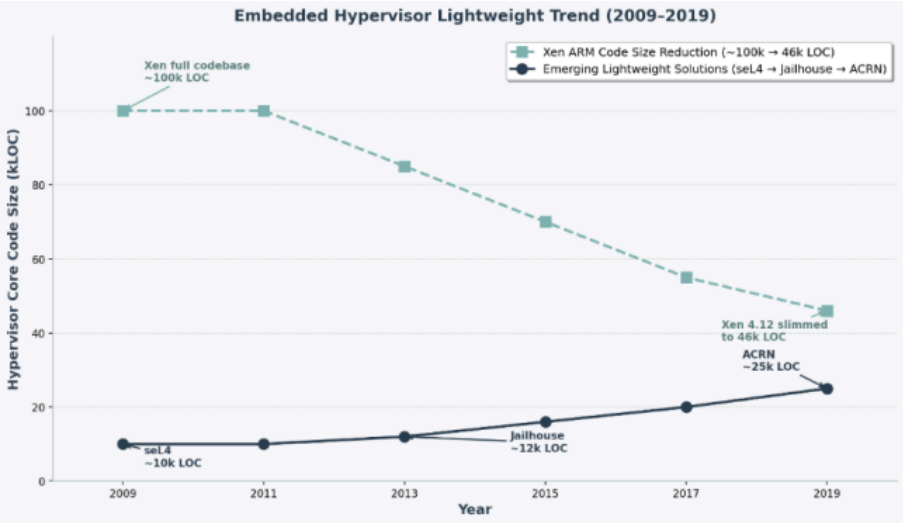

Figure 4. Lightweight development trend of embedded hypervisors (2009—2019) [1, 2, 6, 7]

Trend 2: Safety certification diffuses to open-source communities. Public information from the past two years shows that AMD and the Xen community completed the first phase of the Xen functional safety certification program in November 2024, and vehicle manufacturers such as Honda and Ford have also joined the Xen project [3]. This means that automotive-level safety certification is no longer the exclusive domain of commercial solutions.

Trend 3: Formal verification ecosystem expands. According to the seL4 official website 2024 updates, Collins Aerospace continues to promote seL4-based verification frameworks in subsequent DARPA projects (CASE, PROVERS), and institutions such as the UAE Technology Innovation Institute are also using seL4 for drones and secure communication equipment. Verified code maintains a 15-year zero-defect record, enabling this method to move from military research to broader industrial applications [3].

6. Conclusion

This study investigates embedded hypervisor selection for extreme environments, focusing on disaster rescue, space exploration, deep-sea and polar exploration, and security confrontation. It establishes a comparative framework based on seven representative solutions—Jailhouse, seL4, Xen ARM, ACRN, QNX, XtratuM, and PikeOS—and evaluates them across code size, safety certification, device sharing, boot time, guest types, ecosystem capabilities, and isolation strength. Literature analysis, data comparison, and three representative cases—the Mars Pathfinder priority inversion incident, the DARPA HACMS red-team experiment, and AWS Snow deployment—are combined to assess the applicability of the framework.

The results show that no single hypervisor is optimal for all extreme environments. Selection should be determined by application requirements, security levels, resource constraints, and task

DOI: 10.54254/2755-2721/2026.LD37589

criticality. Disaster rescue emphasizes recovery capability and offline autonomy, space exploration prioritizes certification and verifiability, deep-sea and polar applications favor lightweight architecture and long-term stability, and security confrontation requires strong isolation and security assurance. The analysis further indicates that safety certification and formal verification are becoming increasingly important in safety-critical embedded virtualization.

This paper provides a practical decision-making basis for embedded system architecture design in extreme environments. However, the capability assessment relies mainly on publicly available information and lacks unified-hardware measurements, while standardized benchmarks for boot time and memory overhead remain limited. Future research should conduct comparative experiments on standardized hardware and further investigate hardware hardening and multi-node decentralized collaboration.

References

[1] Martins, J., & Pinto, S. (2023). Shedding light on static partitioning hypervisors for Arm-based mixed-criticality systems. *arXiv*. arxiv.org/abs/2303.11186
[2] Kiszka, J. (2013, November 19). Jailhouse: A Linux-based partitioning hypervisor. *Linux Kernel Mailing List*.
[3] Klein, G., Andronick, J., Elphinstone, K., et al. (2014). Comprehensive formal verification of an OS microkernel. *ACM Trans Comput Syst, 32*(1), 1-70.
[4] Heiser, G. (2020). *The seL4 microkernel: An introduction*. The seL4 Foundation.
[5] SYSGO. (n.d.). PikeOS certification summary. *SYSGO FAQ*. www.sysgo.com/faq
[6] Xen Project. (2025, June). The Xen safety concept. *Xen Critical Software Summit*.
[7] Linux Foundation. (2020, July 21). The ACRN open source hypervisor for IoT development announces ACRN v2.0 and functional safety certification concept approval. *PR Newswire*.
[8] Reeves, G. E. (1997). What really happened on Mars? *NASA Jet Propulsion Laboratory*. https://www.cs.cornell.edu/courses/cs614/1999sp/papers/pathfinder.html
[9] Fisher, K., Launchbury, J., & Richards, R. (2017). The HACMS program: Using formal methods to eliminate exploitable bugs. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 375*(2104), 20150401.
[10] Amazon Web Services. AWS snow family. *AWS*. aws.amazon.com/snowball/.
[11] Valvano, J. W. (2017). *Embedded systems: Real-time operating systems for ARM Cortex-M microcontrollers* (4th ed.). CreateSpace.
[12] He, F., Song, L. R., & Xiong, H. G. (2008). Two-level task partition scheduling design in integrated modular avionics. *J Beijing Univ Aeronaut Astronaut, 34*(11), 1364-1368.