Kernel Architecture And Scheduling Algorithms: A Comparative Study

Shrivishnu Mukundhan, Rafnida Rauf, Sean Martin

1. Department of Computer Science and Physics, Rider University, Lawrenceville, New Jersey, United States

Abstract

Real-time systems are critical in fields like automotive, industrial automation, aerospace, and telecom, where timing and reliability are essential. At their core is the kernel, whose architecture directly affects system latency, scheduling, and stability.

This project examines how different kernel types—Standard Linux, PREEMPT_RT, QNX, KVM, Xenomai, and eCos—perform under real-time conditions. Using tools like cyclicterst, custom schedulers, and benchmarking, we evaluate each for latency, deadline misses, and responsiveness. We also simulate RM and EDF scheduling and apply CPU isolation to enhance performance.

Results show that PREEMPT_RT offers the best latency and consistency, while QNX provides strong modularity and security. KVM, due to virtualization overhead, performs poorly for strict real-time needs.

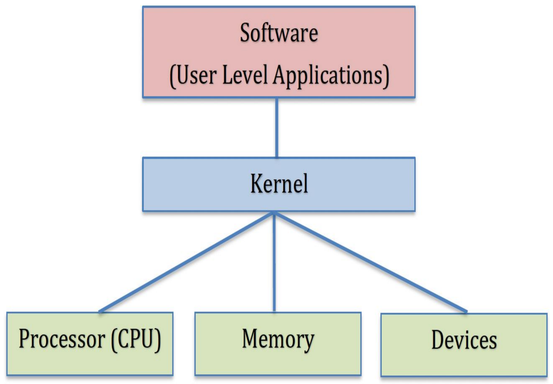
Our findings help guide the choice of real-time kernels for industrial systems that demand high precision and reliability.

Kernel Architecture

In any computer or device, the operating system is what manages all the hardware and software. At the core of this system is something called the kernel. Think of the kernel as the 'brain' of the operating system—it controls how memory is used, how tasks are scheduled, how devices like keyboards and hard drives talk to the computer, and more.

kernel architecture refers to the way this kernel is designed and structured. Different architectures offer different trade-offs in terms of performance, security, modularity, and efficiency. For example, some systems use a monolithic kernel, where everything is packed tightly together for speed. Others use a microkernel, which is split into smaller, more secure pieces that communicate with each other.

"Why does this matter? Because in real-time systems—like those in robotics, medical devices, or aerospace—you need precise timing, low latency, and high reliability. The kernel architecture directly affects whether those demands can be met."



Kernel Architecture Evaluation for Real-Time Performance

1. Measuring Preemption Latency in Linux Kernels

Kernel Type	Avg Latency
Standard Linux	> 100 µs
PREEMPT_RT	< 50 µs
KVM	150-180µs
Xenomai	65–90 µs
Embedded RTOS (eCos)	~40–50 µs

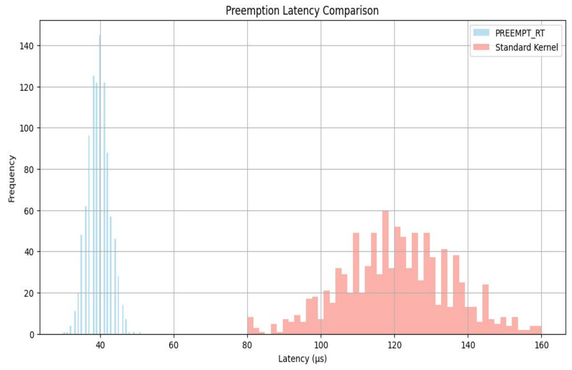
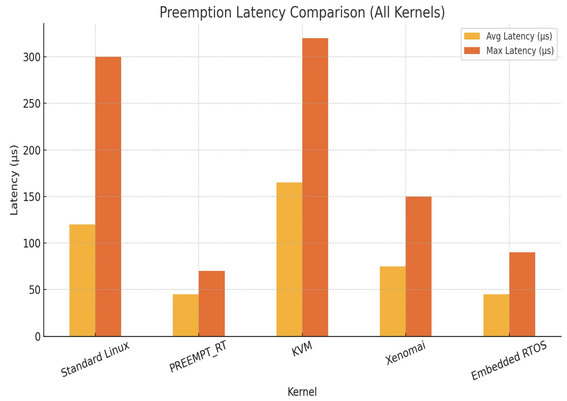


Figure 2&3 Preemption Latency Comparison (All Kernels)



2. Comparing Real-Time Schedulers: RM vs. EDF

Simulated three-task scheduling in user space using standard Linux and EDF patch, including KVM results where available.

Scheduler	Task	Avg Latency	Deadline Miss Rate
Rate Monotonic (RM)	T1	38.15 ms	0.06
Rate Monotonic (RM)	T2	50.23 ms	0.00
Rate Monotonic (RM)	T3	56.66 ms	0.00
Earliest Deadline First (EDF)	T1	89.45 ms	0.39
Earliest Deadline First (EDF)	T2	149.25 ms	0.48
Earliest Deadline First (EDF)	T3	127.98 ms	0.50
KVM Environment	T1	150 ms	0.50
KVM Environment	T2	180 ms	0.60
KVM Environment	T3	200 ms	0.65

3. Kernel Optimization: CPU Isolation and Affinity

CPU isolation and task affinity to reduce interference for real-time threads.

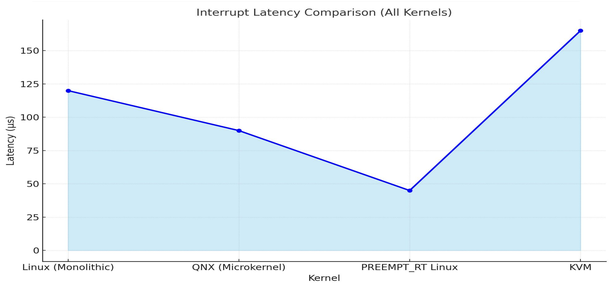
Observed Impact:

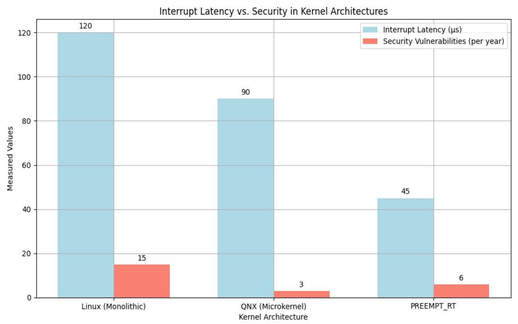
Real time tasks assigned to isolated cores reduced context switches by over 85%, improving predictability.
Execution time improved by ~20%.
Core isolation and affinity are lightweight kernel-level enhancements that do not require code changes to the kernel itself but yield measurable real-time performance benefits.

4. Interrupt Latency

Comparing how different kernel architectures — monolithic (Linux) and microkernel (QNX) handle interrupt latency and system load under stress, with implications for real-time system reliability and responsiveness

Metric	Linux (Monolithic)	QNX (Microkernel)	PREEMPT_RT Linux	KVM
Avg. Preemption Latency	120 µs	90 µs	45 µs	150–180 µs
Deadline Miss Rate	12%	8%	4%	15%
Yearly Security CVEs	15	3	N/A	Variable (depends on host kernel)





Analysis & Discussion

- PREEMPT_RT delivered the best preemption latency and real-time consistency across tests.
- QNX Microkernel provided stronger fault isolation and lower attack surface, ideal for secure domains.
- KVM performed poorly in real-time metrics due to virtualization overhead.
- CPU Isolation improved execution time by ~20% and reduced context switches by over 85%.
- RM scheduling was more predictable than EDF in a non-real-time Linux environment.

Recommendations

Use PREEMPT_RT Linux for applications requiring low latency and real-time responsiveness, such as industrial automation, robotics, and control systems. It enhances standard Linux with features like full kernel preemption and real-time interrupt handling, making it suitable for time-sensitive and flexible deployments.

Choose QNX for safety-critical systems in automotive, aerospace, and medical devices, where security, fault isolation, and system stability are crucial. Its microkernel design offers strong modularity and is well-suited for environments requiring high reliability and regulatory compliance.

References

1. **Linux Kernel RT Benchmarking (OLS 2012)** https://www.landley.net/kdocs/mirror/ols2012.pdf
2. **Microkernel vs. Monolithic Kernel Analysis** https://www.researchgate.net/publication/385879846_Research_paper_microkernel_vs_monolithic_kernel
3. **Performance Analysis of KVM Hypervisor** https://www.researchgate.net/publication/366174295_Performance_Analysis_of_KVM_Hypervisor_Using_a_Self-Driving_Developer_Kit
4. **Real-Time OS Benchmarking** – *Xenomai vs. eCos* https://www.researchgate.net/publication/261462928_On_performance_of_kernel_based_and_embedded_Real-Time_Operating_System_Benchmarking_and_analysis
5. **EDF vs RM with Linux Sched_Deadline Patch** https://www.sciencedirect.com/science/article/abs/pii/S0164121212001460X