Real-time Linux for controls

Industrial control performance testing with Red Hat and Intel

Table of contents

Real-time Linux for controls

Industrial control performance testing with Red Hat and Intel

Industrial control is evolving: Performance and openness are a source of change

Industrial control technology, traditionally anchored by proprietary programmable logic controllers (PLCs) and distributed control systems (DCSs), has not undergone major architectural changes since the 1980s. These devices are, and have always been, proprietary with hidden operating systems (OSes) and vendor-specific connectivity that is traditionally slow to update. The result is an architecture shaped by single-vendor management, locked-in applications, and tightly controlled hardware.

This is proving increasingly unsustainable due to demands being placed on today's industrial control systems. These systems are often characterized by:

- Interconnectivity.
- Continuous modernization requirements.
- Increased cybersecurity threats.
- AI challenges.
- Extensive enterprise data needs.

The traditional proprietary approach is creating significant challenges, including elevated system costs, complex integrations, inherent inflexibility, and increased expenses associated with the adoption of new technologies.

This reality has led to the creation of industry organizations—such as Open Process Automation™ Forum (OPAF) and the User Association of Automation Technology in Process Industries (NAMUR)—and the Industry 4.0 movement, focused on adjusting the trajectory of the industrial automation industry toward a more open, security-focused, and scalable architecture. Red Hat and Intel are collaborating to bring a solution to the market that aligns with these initiatives.

Deterministic performance is a critical consideration

The transition to modern architecture is well underway for most system components, but modernizing the control itself has been slower in this transformation due to concerns around performance, including:

- **Latency.** This is the time between an event (such as a sensor reading) and the system's response. In a typical Linux® environment, basic performance tuning can reduce this time.
- **Determinism.** This is the measure of how predictable that latency response time is. It is vital for high-priority tasks to execute within a guaranteed timeframe, regardless of system load.

For operational technology (OT) systems running critical applications—such as motion control, robotic coordination, or quality inspection—an unpredictable delay of even a few milliseconds can lead to machine crashes, defective products, or safety hazards. This is why many industrial control systems rely on specialized real-time operating systems (RTOS) on highly-tuned hardware.

However, these specialized RTOS systems are often proprietary, hard to integrate with cloud infrastructure, and can become a bottleneck for innovation and data-based insights—the exact goals industry is aiming for. This leaves a gap at the edge of the network with respect to control.

Testing industrial control on Red Hat Enterprise Linux and Intel

Understanding the terms of the test

Industrial control relies on asynchronous loops. The real-time performance of an input/output (I/O) loop within this system is measured by 2 key metrics: screw-to-screw time and jitter.

Screw-to-screw time is the total time it takes for a signal to complete a full round trip through the control system. In literal terms, this means from the physical input wiring terminal (the 1st screw) to the physical output wiring terminal (the 2nd screw).

For simplicity, we will talk about this as 3 distinct asynchronous loops. First, the input, which is when a discrete or analog signal from a field device arrives at the physical input module. The I/O subsystem digitizes this signal and routes it to the controller. Second is processing, when the controller receives the data, executes its control algorithm, and determines the necessary action. Third is output, when the controller sends this result back to the I/O subsystem, which converts it back into an analog signal and delivers it to the physical output module to trigger the field device.

Jitter is any variation or inconsistency in that screw-to-screw time. Because control systems rely on highly precise timing, high levels of jitter can severely disrupt operations. Fluctuations in response time can cause control loop instability, equipment timing failures (such as diverter misses), and downstream manufacturing and quality problems.

When creating a control strategy, the typical process is to come up with a mathematical analysis of the worst-case screw-to-screw time of all the asynchronous system loops. That screw-to-screw time is critical for safety system response calculations, motion controller loop updates, process loop closure, overall line speed of production, and other automation tasks. Once you know your worst-case screw-to-screw time, you can start writing code and understanding how your system is going to operate.

It is the control system's responsibility to make sure the entire system meets that calculated max round-trip time. Failing to do so can result in catastrophic control system failures, quality problems, machine damage, and production loss. While they often do not meet the needs of digital modernization, traditional proprietary controllers with proprietary OSes are dedicated devices and typically make sure that performance is met. On the other hand, software-based controllers—bare metal, containerized, or virtualized controllers—offer scale, cost, and management benefits to manufacturing. There is, however, concern that these open, software-based solutions cannot meet the screw-to-screw-time and jitter specifications required, as factors such as hardware interrupts, cache flushing by other applications, network card interrupts, or other delays typical in a commercial software/hardware environment all require special handling.

Red Hat® Device Edge or larger scale Red Hat Enterprise Linux both have real-time support that helps solve the challenges found in the typical commercial grade OSes. Coupling Red Hat's real-time kernel with Intel's hardware technologies, such as Cache Allocation (CAT), Intel® Time Coordinated Computing Mode (Intel® TCC Mode) mode, and Intel® Speed Shift Technology, provides a performant, real-time, off-the-shelf solution for control. This allows for a more flexible, manageable, and lower cost control solution that is simpler to update and can run on commodity hardware.

In extensive testing we captured the screw-to-screw time and isolated the OS's contribution to that screw-to-screw time using Red Hat Enterprise Linux running on Intel based hardware. We pulled out jitter numbers to better understand repeatability.

We use these 3 Intel specific features:

- **Intel® Speed Shift Technology**, which prevents central processing units (CPUs) from going into power saving slower clock speeds
- **CAT**, which dedicates processor cache space for the control application and prevents it from being flushed from cache.
- **Intel® TCC Mode**, which is a system wide optimizer for real-time workloads.

What to look for in the test

What you should see in the actual test is screw-to-screw results clustered around a typical combined loop occurrence, with a few outliers that might approach the calculated worst case, and a few other outliers that might outperform the average when the asynchronous loops occur under either nonideal or ideal circumstances.

The results should never go beyond the calculated worst case. If they do, this indicates that the control system has failed to meet the required performance mark.

Most of the worst-case scenarios are calculated assuming the data event occurs just after it is too late for the current cycle of the loop to recognize the change, therefore it has to wait for that cycle to complete, and the next cycle to run, to see the data change and act on it. Each asynchronous loop may have the risk of running twice, as data may arrive to that loop a microsecond after the data was collected for transfer in the previous cycle, thus doubling each loop for worst case.

Worst-case round-trip calculation

The loops used in the test include:

I/O card A-D conversion*	.05 ms
OPC Unified Architecture (UA) pub/sub conversion**	.5 ms
Network latency	.1 ms
Controller periodic rate	5 ms
Controller application scan time	.5ms

*Not in calculation because I/O hardware was simulated.

**This number is inclusive of both conversion to OPC UA on the I/O simulator as well as conversion to usable data on the control side and happens twice per data cycle.

The round-trip worst-case calculation is:

OPC UA +	Network +	Controller rate +	Controller scan time +	Network +	OPC UA =	Worst-case
500µs +	100µs +	5000µs +	500µs +	100µs +	500µs =	6.7 ms worst case

6.7 millisecond worst-case round-trip time.

Test bed layout and components

The Intel hardware used for the test included:

	I/O Simulator	Control Node
Model	ASRock iEP-5010G	ASRock iEP-5020G
SKU	Intel® Atom x6425RE processor	Intel® Atom x7433RE processor
Network	Intel® Ethernet Controller I226-IT	Intel® Ethernet Controller I226-IT
Linux	Red Hat Enterprise Linux 9.6	Red Hat Enterprise Linux 10.0
Kernel	5.14.0-570.44.1.el9_6.x86_64+RT	6.12.0-55.40.1.el10_0.x86_64+RT

Test layout. We highlighted test data for the impact of introducing noisy neighbor stress (heavily loaded application outside of control) on test data for:

- Pure controller execution ladder scan times.
- T2–T3 times (from the container OPC UA subscriber to the container OPC UA publisher).
- T1–T4 times (from the simulated I/O OPC UA publisher to the simulated I/O OPC UA subscriber).

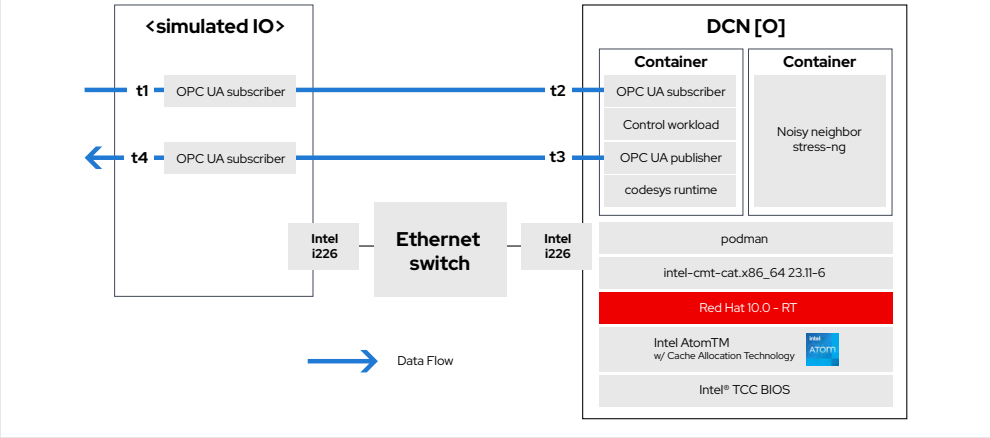

Figure 1. The layout of the test.

Software and OS:

- Codesys OPC UA PubSub stack
- Intel® TCC Mode in BIOS
- CAT
- Intel Speed Shift Technology Disabled
- Codesys Virtual Control for Linux
- Stress-ng to introduce noisy neighbor

Stress-NG background load (noisy neighbor)

One of the key advantages of modernizing the control stack is the ability to run additional peripheral workloads side-by-side with control. While this is 1 of the key value propositions, there is zero tolerance for these noncritical peripheral workloads to impact critical control tasks (product quality, equipment and personal safety, and yield). We used stress-ng to simulate heavy background loading as follows:

- Best-effort cores:
  - 1 CPU (80%)
  - 1 memory
  - 1 virtual memory
  - 5 LL2 cache
  - 5 LLC cache
- Real-time cores:
  - 1 CPU (80%)

Test scenarios and results

To evaluate how well the real-time kernel feature within Red Hat Enterprise Linux supports industrial control workloads, we ran 2 sets of tests. The 1st measured the OS's raw real-time performance, and the 2nd measured full screw-to-screw round-trip time across the entire control loop, including network travel, data conversions, and PLC scan cycles. Control engineers care about both tests, but especially the 2nd, because it reflects how systems behave in the real world.

Test 1 (T2 to T3) : OS real-time performance

This subset of the overall screw-to-screw test highlights the collaboration of Intel and Red Hat. This test isolates the Intel control hardware and Red Hat Enterprise Linux for Real Time to show latency and jitter values without the additional loops of a control solution, such as network latency, pub/sub protocol conversions, or analog-to-digital conversion.

Tests were using 6 different permutations, with each running for a minimum of 1 hour. Permutations were a software controller installed directly on Red Hat Enterprise Linux for Real Time (in blue and labeled just PLC), and again containerizing the PLC using Podman (in orange labeled vPLC). We then added a noisy neighbor for each test, and later implemented CAT.

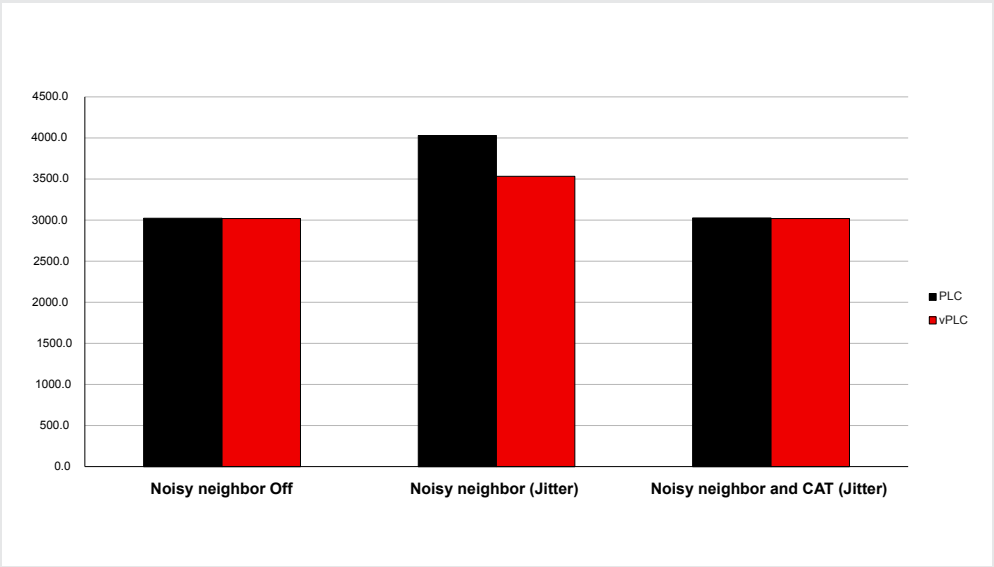

Figure 2. The worst case jitter on 5 millisecond PLC interrupt (in microseconds).

The table in figure 2 displays the maximum jitter recorded between PLC interrupt requests, which were scheduled at 5 millisecond intervals. Conducted over a 60-minute period, this test demonstrates the OS's ability to handle interrupts consistently. Notably, with CAT turned on, the worst-case jitter stayed under 30 microseconds for both the direct Red Hat Enterprise Linux and Podman installations.

It is interesting to note the containerized Podman solution provided more consistent performance than bare metal PLC installations when CAT was turned on.

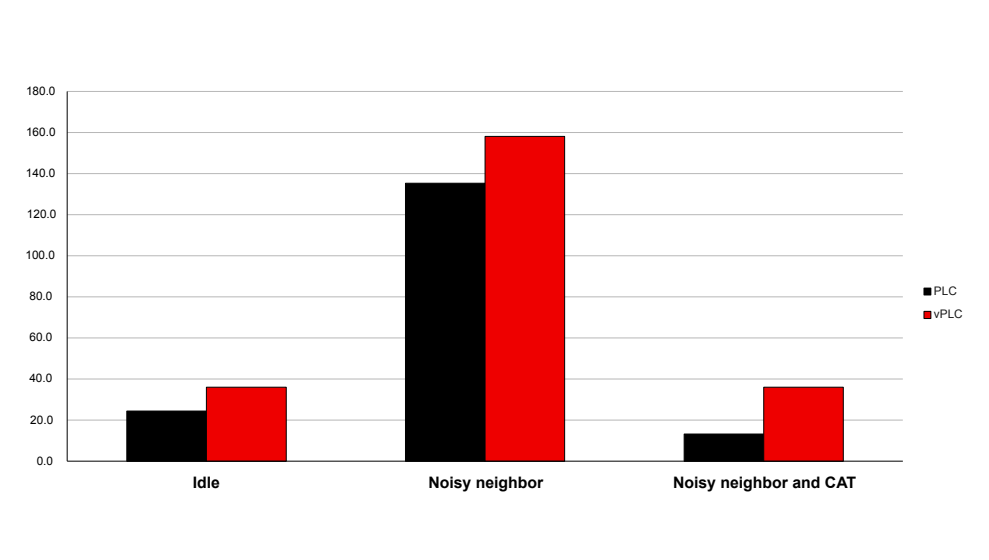

Figure 3. Standard deviation (distribution) of PLC logic execution time in microseconds.

Figure 3 shows the standard deviation of variation in all the control workload cycles over the 1 hour test. Contrasting this with figure 2, which shows the outliers of the system handling interrupts, figure 3 shows the most likely execution variation of the execution of the actual PLC workload. So for our PLC task that runs every 5 millisecond and a best case execution time around 1 millisecond, this table shows in the noisy neighbor with CAT on scenario, that the expected variation in that workload's execution time is less than 32 microseconds for the Podman case.

Test 2 (T1 to T4): Screw-to-screw times

While the 1st test highlights the true OS performance, this 2nd test provides a more real-world scenario incorporating the software stacks typically used for control. A screw-to-screw test simulates the full journey of a control event, including asynchronous delays from open platform communications UA (OPC UA) conversions, network travel, PLC scan time, and application processing. Based on all these asynchronous loop components, the theoretical worst-case round-trip time is 6.7 milliseconds (for more information on how that theoretical worst-case round-trip was calculated, see the worst-case round trip section).

While the actual OS has little impact on that theoretical number, this test highlights that when the system is stressed and heavily loaded, the OS still delivers its deterministic behavior by never letting the control loops, network packet builds, and other elements in the OS's realm of control, exceed the calculated max when configured correctly using CAT technology.

Red Hat ran this test across 2 architectures (bare metal PLC and containerized PLC) with and without noisy-neighbor load and with and without CAT.

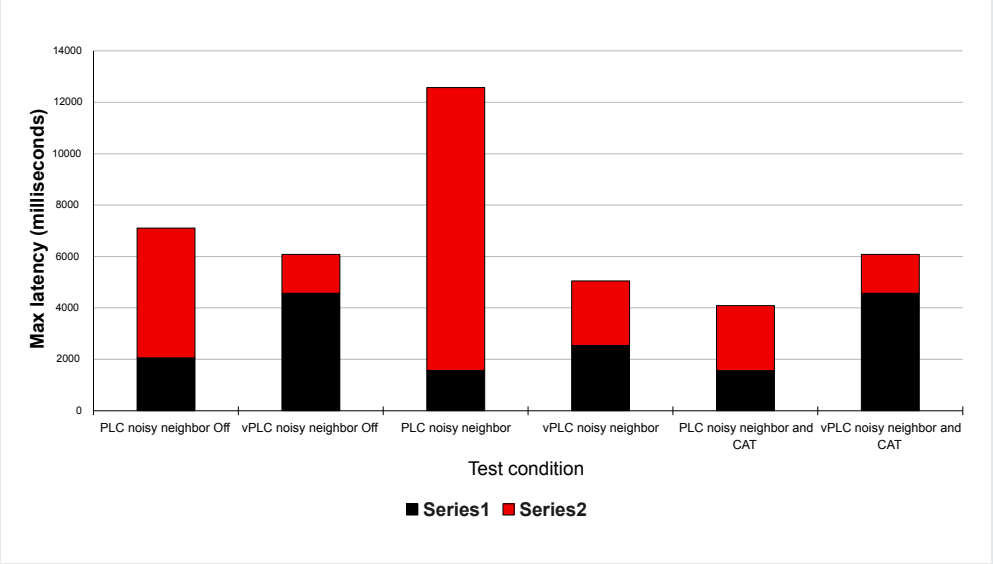

Figure 4. Screw-to-screw total round-trip times.

The results were highly deterministic. Across all tests using CAT, none of the samples exceeded the calculated 6.7 millisecond worst-case threshold once CAT technology was turned on. Even with over 80% CPU power consumed by background tasks, the OS consistently prioritized control workloads over noncritical processes. Recall that there is a 5 millisecond base PLC cycle time plus other cyclic events impacting this number beyond the RTOS.

This demonstrates the real-time capabilities of Red Hat Enterprise Linux to maintain predictable control-loop performance under realistic industrial conditions with many other industrial workloads beyond control running on the Industrial PC (IPC).

Achieve deterministic industrial control performance with Red Hat and Intel

The modern industrial edge demands a powerful, proven platform that merges the strict determinism of the OT environment with the open standards and enterprise management of IT. Together, Red Hat and Intel deliver this through a validated solution using the Red Hat Enterprise Linux 14.1 real-time kernel on Intel® Atom™ x7000RE Series hardware.

Achieving this deterministic performance is the foundational requirement for control, but it is also the gateway to a system that is simpler to manage, modernize, and adapt. Whether deploying Red Hat Enterprise Linux or Red Hat Device Edge, you are not just meeting your control layer needs by qualifying the basics of control, you are opening the door to true industrial digitization.

By using a single, open platform, your industrial edge becomes:

- **Consistently managed and protected.** Extend the same security policies, patch management, and monitoring tools used in the datacenter directly to your edge devices.
- **Flexible for innovation.** Run containers and modern applications—such as real-time analytics or AI models—directly on the same deterministic hardware. This eliminates the cost of additional hardware and the latency of moving massive datasets to the cloud for processing.
- **Future ready.** Deploy a platform that is consistently validated and evolved within the upstream open source community, providing the control to scale your industrial edge for whatever comes next.

The modern industrial edge requires a platform that merges OT precision with IT scale. Red Hat Enterprise Linux delivers that and our latest testing data shows it.

Get started

Connect with your Red Hat or Intel account executive to start transforming your industrial edge infrastructure.

About Red Hat

Red Hat is the world's leading provider of enterprise open source software solutions, using a community-powered approach to deliver reliable and high-performing Linux, hybrid cloud, container, and Kubernetes technologies. Red Hat helps customers develop cloud-native applications, integrate existing and new IT applications, and automate and manage complex environments. A trusted adviser to the Fortune 500, Red Hat provides award-winning support, training, and consulting services that bring the benefits of open innovation to any industry. Red Hat is a connective hub in a global network of enterprises, partners, and communities, helping organizations grow, transform, and prepare for the digital future.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/Redhat
x.com/RedHat
linkedin.com/company/red-hat

redhat.com

Copyright © 2026 Red Hat, LLC. Red Hat and the Red Hat logo are trademarks or registered trademarks of Red Hat, LLC or its subsidiaries in the United States and other countries. Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries. All other trademarks are the property of their respective owners.