The Modern RTOS

Ready for a Cloud-Native World

By Michel Chabroux, Senior Director, Product Management, Wind River

INTRODUCTION

Today's device makers are caught between two opposing trends. On the one hand, legacy code continues to run in deployed systems. On the other hand, development tools — and developers — have continued to evolve. The challenge is how to bridge today's developers to yesterday's systems.

In this case, legacy refers to software that was written quite a few years ago. The code still does what it is supposed to do; however, silicon, development tools, libraries, RTOSes, and even programming languages have changed substantially during this time. Consider an application such as an industrial robot, with a lifecycle of eight to 10 years. The equipment and software upon which the system was developed may not even be actively supported anymore.

The problem is only magnified for more complex systems, such as airplanes. In addition to having a longer lifecycle (on the order of 30+ years), these systems are really collections of many individually developed systems that must interoperate. Further complicating even simple maintenance of these systems is the constant need for upgrades and improvement of operational capabilities. Consider the pace of change in terms of IoT security. Connected devices must receive regular security updates to keep them secure from hackers. Devices a decade or older must also maintain a sufficient level of security, even though they were not necessarily designed to do so. And this is only one aspect of the system. For example, communication technology and protocols continue to advance, and, somehow, these systems need to be able to continue to integrate with the new systems coming online, especially those in the cloud.

TABLE OF CONTENTS

CLOUD-NATIVE DEVELOPMENT

Perhaps the most difficult aspect of maintaining systems is staffing the teams that will make these updates. The current generation of developers who understand these legacy systems and know the languages they are written in are starting to retire. The engineers who will maintain these systems in the coming years may not even have been born when these systems were first deployed.

The next generation of developer is focused on very different skill sets. They are not studying compilers and assemblers; rather, they are working in the cloud and with connecting technologies. They are passionate about developments such as AI at the edge. And there is a surge in demand for these new skills, with a 21% demand growth for software engineers by 2028.[1] It should be no surprise that it will be difficult for manufacturers to attract the talent they need to maintain their legacy systems.

Today, 33% of all developers build with cloud-native architecture in mind,[2] and this is driving significant change in how software is developed. A cloud-native approach employs more scalable, adaptable, and powerful platforms than traditional development tools.

The opportunity is to make current — and legacy — systems more easily adaptable and future ready by aligning them with the cloud-native skills and thinking that new developers bring.

Traditionally, developers started writing a program; months or years later, following a waterfall approach, it was ready. Today's DevOps model is much more rapid and agile. For example, the ability to constantly update code increases the pace of development and deployment.

Rapid deployment brings many benefits. The ability to react quickly and resolve issues allows development teams to introduce new capabilities with less lead time. This increases a company's competitiveness. It also makes it easier to maintain compliance with constantly changing industry standards, protocols, and regulations. In fact, certification of software can become a part of DevOps rather than an end-of-design process.

Moving to a DevOps model is a necessity for attracting new developers to support legacy systems, as it will allow them to use the tools with which they are familiar. Doing so also increases the efficiency of system development overall. Consider that DevOps teams:

- Deploy software 46x faster
- Enjoy 440x faster lead time for changes
- Recover 96x faster, on average
- Suffer 5x fewer change failures[3]

Certainly, efficiencies will vary based on team and application, but they are representative of the magnitude of improvement that manufacturers can expect to experience. Consider that AWS deploys software every 11.7 seconds.[4] Airplanes and robots don't need to be updated at this rate, but they can certainly benefit from updates more frequently than once a year or even less often.

A cloud-native DevOps approach fundamentally changes how systems are designed. Consider the impact on testing of a safety-critical system through a battery of tests. Traditionally, there is limited compute dedicated to testing. With a cloud-native DevOps approach, the system can be containerized and run in parallel across cloud resources to complete testing in a fraction of the time.

DevOps also enables tools to adapt to both device lifecycle and development teams. The infrastructure and tools can be adjusted regardless of the number of people or teams involved. In fact, the infrastructure becomes easier to work with, even after systems are deployed.

Wind River® has experienced the benefits of this digital transformation for itself. Automation has allowed us to find issues sooner in the development cycle. For example, 99.3% of the top test cases are automated, and 86% of defects are found during feature testing. The end result is that Wind River has been able to find and resolve issues in one-tenth the time, compared to its pre-2015, pre–digital transformation experiences.

THE MODERN RTOS

A key technology in bridging legacy embedded systems with cloud-native DevOps is the modern RTOS. This is an RTOS that can support both modern and legacy tools and languages. For example, many applications utilize C, but there are other options that are better at performing specific tasks, such as C++ augmented with Boost, Rust, or Python. A modern RTOS enables developers to leverage the best tools available today while supporting the tools of yesterday to maintain legacy systems.

With everything being connected, a modern RTOS needs to connect edge systems to the cloud — even those systems that were designed and deployed before there was a cloud. Hand in hand with connectivity is cybersecurity. The RTOS needs to provide vulnerability protection, monitoring, information assurance, and hardening of systems to minimize security risks. This includes capabilities such as secure boot, protected storage, and communication, as well as application of least privilege throughout the system, access control, security event monitoring, etc.

A modern RTOS must also streamline development. Cloud-native tooling needs to be integrated with emulation and simulation platforms. The RTOS must also support advanced deployment constructs and open frameworks such as OpenCV and TensorFlow. It must also allow developers to test logic on the desktop but run and test in the cloud for speed and scale.

CONTAINERIZATION

Secure boot prevents unapproved code from being launched. The process of securing the code starts in the production environment when the firmware is encrypted (optional), then digitally signed. Once deployed, the secure boot loader (trusted and for the most part unmodifiable) loads the firmware, verifies the digital signature, and decrypts (if needed) before launching. In turn, the firmware would verify the authenticity of the applications it launches. When an application has been packaged as a container, both the authenticity of the container registry and the packaged applications must be verified to continue the chain of trust established from boot-up.

With 50% year-over-year growth,[5] container technology is seeing broad adoption, and Wind River has been investing in bringing containers to embedded IoT development. In simple terms, a container has whatever software is necessary for an application to function in an environment. Containers allow software to be packaged with isolated dependencies, allowing for a simplified develop, test, and deploy workflow. Additionally, leveraging container registries, it is much easier to deploy updates once devices have been fielded leveraging existing IT infrastructures.

To put this in context, the VxWorks® container engine, including all dependencies, is 500 KB. The runtime itself is under 100 KB and maintains the real-time performance and determinism that the industry has come to expect from VxWorks. The container engine is also safety certified, making it ready for applications such as avionics.

The VxWorks container engine has been designed to provide compatible workflows with runc and with support for Docker Registry HTTP API V2, enabling support for registries from Amazon, Microsoft®, and Google. The VxWorks container engine makes it possible to work through container registries such as Amazon, Microsoft®, and Google. It's another step toward a tactical edge cloud that gives connectivity to real-time embedded devices.

CONCLUSION

The embedded IoT industry faces the challenge of maintaining legacy systems in a manner that attracts next-generation engineers and leverages their cloud-native skill sets. This is a complex and time-consuming issue for the industry to solve. However, if we embrace the change that is coming, we will see tremendous benefits.

REFERENCES

1. Learn.org, "Software Engineer: Career Definition, Occupational Outlook, and Education Prerequisites"
2. IoT Developer Survey 2019 Results, Eclipse.org, April 2019
3. State of DevOps Report, Splunk
4. Akash Lomas, "Why it's Vital to Release Software to Production Multiple Times a Day," Net Solutions, July 5, 2021
5. BusinessWire, "New Survey by Denodo Reveals a Steep Increase in Hybrid & Multi-Cloud Architectures as Organizations Look to Save Costs and Build Resiliency," April 28, 2020

WNDRVR