Vibhoosh Gupta, Emerson

CONTROL ENGINEERING
ELECTRONICALLY REPRINTED FROM DECEMBER 2019

Virtualization for the plant floor

End users are comfortable with virtualization in information technology server rooms and desktops, and many similar benefits now can be realized with programmable logic controllers and programmable automation controllers.

Commercially available technologies are almost always applied at a quicker pace in the consumer and information technology (IT) space than their industrial counterparts. For instance, virtualization technologies have been commonplace within IT environments, most often in server-based applications, for many years. On the other hand, industrial automation operations technology (OT) applications lagged by several years in adopting virtualization.

Today, virtualization has become mainstream for almost all OT products, practices, and applications, though it is still often used in computer-room environments. It is now common for OT system servers to host multiple virtual machines (VMs) for visualization, historian, redundancy and other uses.

Automation engineers use desktop-based virtualization to quickly create instances of development and testing systems. Virtualization provides benefits for quickly deploying systems, optimizing resource usage and backing up configurations.

The concepts and advantages of virtualization are commonly associated with PCs and servers, but they can be employed elsewhere. Recently, virtualization capability has been extended into more specialized and robust industrial programmable logic controllers (PLCs) and programmable automation controllers (PACs) used for process and machine automation.

This creates more options for end users, such as enabling analytics closer to the source of data. It also provides other benefits including enhanced productivity, efficiency and security.

Virtualization concepts

A basic definition of virtualization is providing the ability to run more than one VM software operating system (OS) on one hardware platform, allowing one physical computer to be more utilized as many virtual computers. Each VM must operate independently.

There are two types of virtualization, Type 1 and Type 2, depending on where the hypervisor is located. A hypervisor is the combination of hardware, firmware, and software running on the host machine and managing guest VMs.

Type 2 virtualization, which may be called "hosted," is used for desktop and server PCs, with the hypervisor running on top of a traditional host OS already operating on the hardware. This creates virtual "sandboxes" where multiple OSs can run simultaneously, but it adds latency due to the underlying OS.

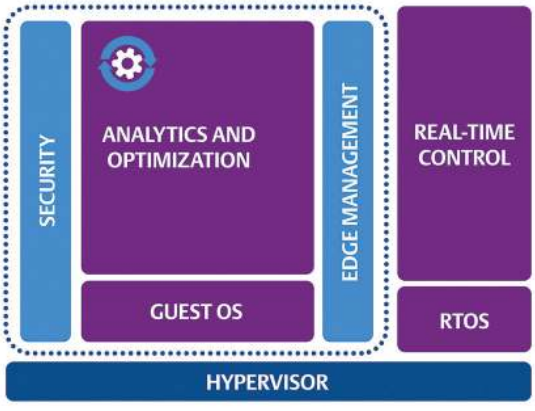

Figure 1: This figure depicts how a single Type 1 hypervisor manages two virtual OSs on an industrial controller, a RTOS for real-time control and a guest OS for edge processing. Images courtesy: Emerson

Type 1 virtualization, sometimes called "native," utilizes a hypervisor running directly on the bare metal hardware without an underlying OS. The hypervisor partitions the hardware itself to each OS. This results in very low latency and jitter, which is ideal for real-time deterministic or time-sensitive applications. Type 1 offers greater performance than Type 2 because it has direct access to the hardware without delays due to a host OS system.

Until recently, virtualization has not been practical or possible at the plant floor level. Now, the development of a new class of PLCs and PACs leveraging multiple processor cores and virtualization offers the ability to extend the same virtualization concepts down into the industrial controller, providing an integrated approach.

A common class of industrial controller for typical automation applications is the PLC, which has used a dedicated processor and specific real-time OS (RTOS) to provide high-speed deterministic control. The challenge to virtualizing PLC functions is maintaining high-speed determinism.

Today there are hardware advances common to the commercial PC world such as multicore processors and large memory. By using multicore technology and Type 1 virtualization, industrial controller platforms can run multiple OSs on the same proces-

Figure 2: Industrial controllers capable of virtualization, such as Emerson's Outcome Optimizing Controller (OOC), include multiple processing cores, digital communication ports, and support for parallel operation of a control RTOS and a general-purpose OS.

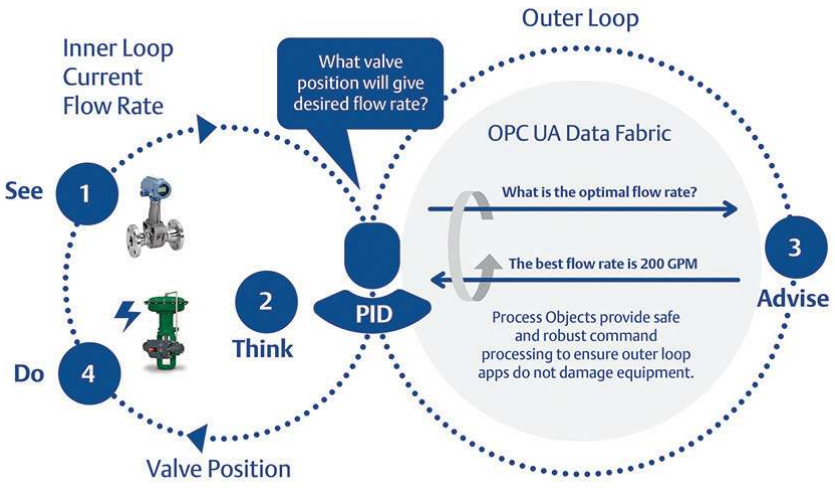

Figure 3: Using virtualization, Emerson's OOC uses a real-time "inner" loop for direct control, which can be advised by a general purpose "outer" loop where advanced optimization can be performed.

sor, including a RTOS for control which will experience little to no effect on the determinism and speed (Figure 1). A second Linux guest OS can be used for other edge processing. As the number of available cores increases, even more OSs can be deployed.

For edge-located controllers, the possibility of running multiple OSs is a dramatic shift (Figure 2). The primary concern is maintaining a robust RTOS for control functionality, as with PLCs, so the automation functionality is not compromised. A secondary OS, operating as supplemental to the RTOS, provides more computing options. The two OSs must be independent and maintain ability to interact.

The concept of running two OSs in an edge-located industrial controller can be further explained in terms of an "inner" and an "outer" loop, which may be familiar to users of cascaded process control loops. In this case, the inner loop is the RTOS VM for control, while the outer loop is a standard OS VM for added functions. The inner loop could monitor a process flow input, perform the proportional-integral-derivative (PID) calculations, and command the control valve output. The outer loop can advise the inner loop regarding the optimal flow rate, but it will not impact inner loop operation otherwise.

Another way of explaining the inner and outer loop concept is an analogy related to navigating a car. The inner loop is represented by the driver's direct attention to drive the vehicle to its destination, while the outer loop could be the car's dashboard navigation system providing supplementary information.

The inner loop is mission critical and must carry on operations without fail, even if the outer loop has a problem. On the other hand, the outer loop is valuable, but not essential, to basic system operation.

Control remains the inner loop

PLCs provide the specialized functionality, robust packaging, and input/output (I/O) connectivity necessary to automate equipment and processes. These devices have gained processing power and networking capabilities to interact better with higher-level systems, with more advanced versions often called PACs. PLCs and PACs still have dedicated roles.

PLCs and PACs use many types of programming languages; ladder logic is the most popular. A basic measurement of PLC performance is how fast the controller can scan through the ladder logic, typically measured in milliseconds. All other overhead tasks must be handled so a deterministic scan time is preserved. The OS from a PC is not a good candidate for millisecond control because it must handle many overhead tasks such as graphics and user interface.

A controller-based VM with PLC or PAC functionality requires using an RTOS to provide features needed for inner-loop PLC functions without performance-sapping features.

In the outer loop

Since control functionality remains essentially the same, the real advantage to controller virtualization is the addition of an outer-loop Linux VM carefully integrated in a cooperative manner. This VM can do anything a dedicated PC could, but at a lower cost, and packaged in a more compact form factor, with no need to integrate a PC to the controller. It is unnecessary for industrial users to take advantage of the extra OS right away, as they could use a virtualized controller for the basic PLC functionality. Many users are finding a general-purpose Linux OS at the edge can enhance applications by running machine learning elements; performing analytics; communicating to the cloud; using messaging queuing telemetry transport (MQTT) or some other publish-subscribe model to exchange information; executing optimization calculations to inform the controller VM; driving a local display; and serving web pages.

These functions have required upstream computing resources. Users benefit from the efficiency of implementing these functions out at an edge-located OS, because they are taking advantage of locally available processing power and acting on the data as close to the source as possible. This removes layers of computing and streamlines networking usage. The ability to drive a local display or directly serve web pages out of the controller are examples of this.

When implemented properly, an outer loop VM lets users safely perform processing at the edge closer to the source of data, which unloads upstream networking and processing requirements. **ce**

Vibhoosh Gupta *is a portfolio leader, Emerson's Machine Automation Solutions business unit. Edited by Chris Vavra, production editor,* Control Engineering, *CFE Media and Technology, cvavra@cfemedia.com.*