How Real (Time) Are Virtual PLCs?

Diogenes Javier Perez
Technical University of Munich
Munich, Germany
diogenesjavier.perez@tum.de

Josef Waltl
Software Defined Automation GmbH
Garching, Germany
josef.waltl@softwaredefinedautomation.io

Laurin Prenzel, Sebastian Steinhorst
Technical University of Munich
Munich, Germany
{laurin.prenzel, sebastian.steinhorst}@tum.de

***Abstract*—Production systems continuously need to become more cost-efficient and flexible. Hardware-based programmable logic controllers, while widely used in the industry, do not offer the level of flexibility and scalability required for future applications. Each hardware-based PLC entails costs for maintenance and they cannot keep up with resource-intensive loads, such as artificial intelligence. The virtualization of PLCs promises to solve these issues. A Virtual PLC at the local edge level between cloud and industrial assets provides the flexibility and resource capacity needed for modern control applications. In this paper, the concept of virtual PLCs in a COTS server is outlined as a SoftPLC that is running within a virtual machine managed by a hypervisor. In addition, the virtual PLC is implemented and evaluated to determine whether virtual PLCs can satisfy the requirements for specific domains of industrial automation. We compare multiple virtual PLC configurations to a SoftPLC without a hypervisor. Our results indicate that the virtual PLC implementation is on par in terms of switching and response time for applications requiring response times below 10 ms and deterministic behavior is achievable. While further work is necessary, virtual PLCs may offer tremendous advantages for future industrial systems.**

***Index Terms*—Software Defined Automation, vPLC, Real-time Control, Server Virtualization, Edge Computing**

I. Introduction

Modernizing the factory automation stack requires more than an update of the actual hardware implementations, i.e., the introduction of new control paradigms for a software-defined automation [1], [2]. The design and realization of flexible and changeable manufacturing systems for individualized products are specified as crucial for competitive production systems of the future [3]–[5]. In such systems, reconfiguration or redeployment of industrial automation systems can be done for every piece, the application of machine learning and artificial intelligence (AI) algorithms is essential and full-loop feedback systems enable self-optimizing production systems.

Current programmable logic controllers (PLCs), along with the way the industry is structured, can only support future industrial requirements in a constrained way [6], [7]. Scaling the hardware is costly and often not possible. They do not support resource-demanding tasks, such as machine learning and artificial intelligence. Furthermore, hardware-based PLCs imply individual maintenance and hardware costs, as every single technical process regularly includes an individual PLC for each subsystem [7]. Modern control systems should meet requirements for operational technologies and information technologies. Therefore, PLCs require an intense transformation. To achieve this, an Internet-of-things-PLC was prototyped and evaluated in [8]. After a positive outcome, an extension of software and hardware design for industrial evaluation was suggested as future work.

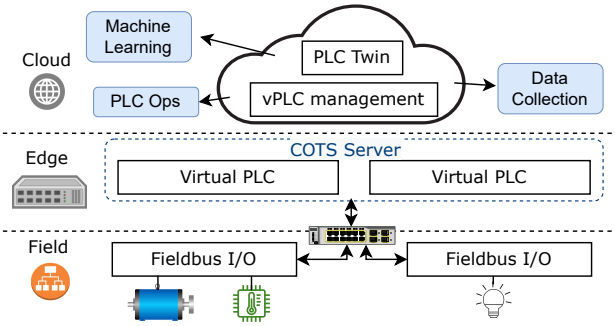

Fig. 1. In a software-defined automation virtual PLCs can be used to control industrial assets in the same way as conventional PLCs are. A COTS server located on the edge can host multiple virtual PLCs to control assets in real-time. Virtual PLCs can be managed and monitored from a cloud instance.

In this paper, we implement an architecture using virtual PLCs (vPLCs) on top of a real-time hypervisor on a commercial off-the-shelf (COTS) server with real-time industrial fieldbuses. Servers and computers can offer enough resources to fulfill the functions of PLCs, Human-Machine Interfaces (HMIs) and programming terminals together [9]. An edge server hosting virtual PLCs that communicate with the shop floor and cloud facilitates scalability and reduces the amount of hardware, vendor dependencies and maintenance costs (see Fig. 1). Coupling the cloud and shop floor further allows the implementation of software-based PLC Operations (Ops), as well as data collection and use of advanced machine learning algorithms, while still satisfying deterministic real-time requirements.

In this paper, we evaluate the performance of a virtual PLC implementation in a server for the edge, as proposed in [10]. The goal of this work is to determine if virtual PLCs have already reached the point where they can satisfy the requirements of industrial automation domains. We compare multiple vPLC configurations using a bare-metal hypervisor to a traditional SoftPLC without a hypervisor to establish differences in the real-time behavior. In multiple measurements using native monitoring tools, as well as in measurements on input/output (I/O) modules, we demonstrate that with existing solutions, industrial-grade performance and reliability are feasible.

The rest of the paper is structured as follows. Section II

summarizes the current state of the art of virtual PLCs. Section III introduces the architecture used to tackle the problem. Then, Section IV describes the method of evaluation and the results of our measurements, which are discussed in Section V. We conclude in Section VI.

II. Background

This section presents solutions that are based on three approaches as alternatives for implementing the control layer in a more up-to-date way. First, cloud-based solutions are portrayed, followed by solutions based on two different approaches from the edge, i.e., industrial control from the edge and virtualized PLCs on real-time hypervisors on the field. The solution approaches are portrayed in a sequence where the succeeding approach considers challenges faced by the previous approach. The solution proposed in this paper considers the challenges experienced in the three approaches. Finally, previous work on real-time hypervisors is presented as it is a fundamental component for this work and the previously done ones mentioned in this section.

The terms SoftPLC, i.e., a software application that emulates a standard PLC in a computer or server and virtual PLC, i.e., a SoftPLC that is running within a virtual machine (VM) managed by a hypervisor, are used throughout this paper, respectively, as SoftPLC and virtual PLC (vPLC).

A. PLCs in the cloud

Cloud computing offers scalability, efficiency and cost reduction that can improve the actual control hardware implementations. A question that arises is to what extent implementations from the cloud could be used in automation systems. Two works based on cloud computing are presented in this subsection.

A vPLC based on cloud computing was analyzed in [11] to assess its performance. The PLCs were implemented in Microsoft Windows 7 32-Bit virtual machines within a VMware vCloud suite private cloud architecture and their performance was compared with a Phoenix Contact Hardware PLC (ILC 350 PN) located in the same Profinet RT network.

To evaluate its performance, an IEC 61131-3 project was used. The project read an externally generated signal and wrote the inverted value to the output connected to an oscilloscope. The cloud-based vPLC showed lower performance than the hardware PLC but it was still acceptable for soft real-time applications. Their implementation did not use a real-time hypervisor.

In [12] a cloud-based SoftPLC was implemented in a cloud environment for building automation, to analyze its round-trip times. Their SoftPLC achieved round-trip times below 1000 ms in 99.72% of the cases, while still reducing the implementation costs compared to a traditional hardware PLC implementation. One downside of this implementation is its vulnerability to internet connection failures.

After analyzing the works presented in this subsection, it can be inferred that by locating the control unit on the edge, as proposed in this paper and in [13], crucial real-time tasks could be fulfilled faster and also during a loss of internet connection. It can be further inferred that using a real-time hypervisor, as proposed in [10] and in this work, could help improve the performance of the vPLC.

B. Industrial control from the edge

As the edge is located physically near to the asset to be controlled, reacting to events in real-time becomes possible, while still offering flexibility and more resource capacity than classical industrial controls. In addition, systems on the edge are less susceptible to internet network failures than cloud implementations are. Two works proposing industrial control from the edge are portrayed in this subsection.

In [13] a concept for linking control and cloud technologies was presented and implemented using the edge as a mediator between field-level and the cloud. This opens possibilities for cloud-assisted control for resource-intensive data processing and feedback control loops from the shop floor.

In their implementation, User Datagram Protocol (UDP) and serial communication were used for real-time communication between the edge device and the IIoT devices controlling the robots. MQTT was used for non-real-time communication between the edge device, the cloud and the IIoT devices controlling the robots.

Edge computing technologies were used to execute IEC-61131-3 applications in a vPLC from the edge in [14]. Their implementation demonstrated that deterministic IP networking can be used for field-level communications, e.g., Profinet RT over DetNet. In their architecture, the virtual PLC was hosted in an Ubuntu virtual machine and connected over a virtual switch in the hypervisor environment to a deterministic IP network.

In a word, the edge device used in [13] could be swapped by the server with a real-time hypervisor portrayed in this paper and in [14], to host virtualized PLCs and other applications. This would offer more standardized programming options, e.g., IEC 61131-3. In addition, real-time communication could implement several industrial communication protocols and flexibly switch between them. In this work, the virtual PLC will be hosted as in [14] but the virtual switch will be instead directly connected to the real-time fieldbus.

C. Virtualized PLCs on a real-time hypervisor on the field

The works presented in this subsection propose using a real-time hypervisor to manage the resources of the virtual machines that host the vPLCs.

The concept and architecture of a virtual PLC are presented in [10]. In their architecture, the dedicated PLC I/O bus was replaced with a SDN-based networking infrastructure where both virtualized PLCs, running within a VM on a real-time hypervisor, and non-real-time workloads (HMI, SCADA Master Stations (MS) and Historian Database servers (HDB)), running within a VM on a COTS hypervisor, interacted. In [15] an evaluation was conducted to determine to which extent partitioning techniques can improve real-time hypervisor environments using Cyclic Test [16]. Experiments were performed

with varying kernel configurations and amount of processors assigned to a single VM. Evaluation results showed that the virtual PLC is achievable from the systems virtualization perspective.

Considering all of these, in this paper, a virtual PLC implementation is put under test, not only from a hypervisor performance perspective but also including the software performance of the vPLC and the hardware I/O performance.

D. Real-time hypervisors

Virtualization entails benefits such as improved resource utilization, cost reduction and simpler hardware management. However, its overhead due to additional layers can increase latency if not managed properly. Two works presented in this subsection evaluate the real-time performance of two broadly used hypervisors, Xen and VMware vSphere.

The performance of virtual environments has been analyzed and aimed to be improved in [17] and [18]. However, I/O virtualization can face challenges in latency-sensitive domains. The solution to these issues relies on the scheduler of the hypervisor. In [19], RT-Xen, a real-time hypervisor scheduling framework for the hypervisor Xen, was implemented and evaluated. The obtained results in their evaluation suggest that, when using this framework, Xen can provide effective real-time scheduling to guest Linux operating systems. In [20], enhancements to be applied on the Xen scheduler, e.g., management of caches and scheduling latencies, were presented and their experiments indicate that with their modifications Xen can host soft real-time guests without affecting non-real-time applications. In their experiments, they were able to obtain good quality audio running an IP telephony workload.

VMware vSphere aims to maintain this overhead as small as possible to fulfill latencies within milliseconds for applications such as voice over Internet Protocol (VoIP) streaming, where packets need to arrive in intervals of 20 ms [21]. Latency sensitivity improves response time, jitter and determinism by reducing sources of extra overhead and latency introduced by virtualization [22]. Techniques such as tuning of the virtualization layers, bypassing virtualization layers and giving private access to hardware resources help to decrease the latency of virtualization. An evaluation performed in [21] showed that latency sensitivity combined with other features attained an almost native performance on response time and jitter in a server with a hypervisor. For instance, the median of a ping was 20 µs, which is only 2 µs more than a native setup.

Based on the findings of the works considered in this subsection, it can be inferred that hypervisors can be already suitable for hosting real-time workloads, e.g., a vPLC, as proposed in this work and in [10].

III. Methodology

The virtual PLC architecture we propose consists of several interacting hardware and software layers. In this section, the architecture used for a generic virtual PLC implementation in a COTS server is portrayed, without referring any to specific hardware.

A. Architecture

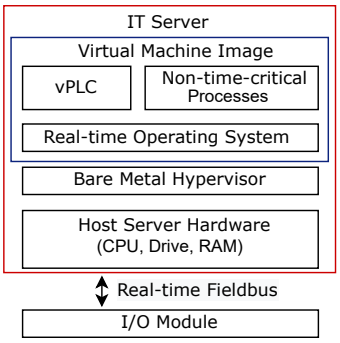

Fig. 2. Architecture of the virtual PLC in a server hosting one virtual PLC

Figure 2 shows the fundamental architecture followed for the implementation of the virtual PLC. As proposed in [10], the virtual PLC is running besides other processes within an independent virtual machine with an operating system optimized for real-time workloads.

To connect the input and output module to the server, an Ethernet interface is required on the server and the module should be able to connect to this Ethernet interface, either directly with a normal Ethernet cable or using a converter. As the purpose of using this architecture is to evaluate the input and output performance of the vPLC, the I/O module should have input and output terminals and their dynamic behavior, i.e., rising and falling times should be considered.

The following layer is the host server hardware, on which the vPLCs run and communicate with devices on the field. The host server hardware must have processing, storing and Ethernet connectivity capabilities. Unlike in [10] and [14], the I/O module was directly connected to the network interface card operating with a real-time fieldbus protocol instead of connecting the devices over a network. The aim of connecting the devices directly to the network interface card is to assess the performance of the vPLC without being affected by external factors arising from the network infrastructure.

This layer is followed by the bare metal hypervisor. The hypervisor manages the virtual machines hosting the vPLCs and the available resources from the host server hardware. A hypervisor is used to individually assign CPU cores, RAM and disk space to the virtual machines hosting the vPLCs. The hypervisor should be configured for hosting real-time workloads to assure determinism and low latencies on virtual machines. It is recommended to leave one of the CPU cores free, i.e., not assigned to any virtual machine, for the hypervisor.

The next layer is the real-time operating system contained inside the virtual machine. The operating system will host the applications running within the virtual machine, i.e., PLC runtime and other processes. To achieve determinism and low latencies, the operating system must be optimized for real-time workloads as well. As the PLC runtime is the most relevant process running within the operating system, the operating system should be chosen based on the compatibility of the PLC runtime. The operating system will also initialize the

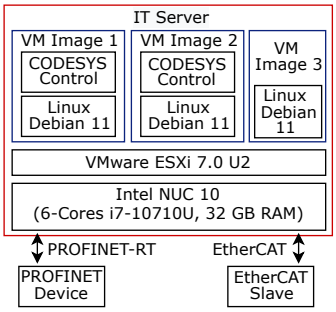

Fig. 3. Specific software and hardware components used to implement the third configuration under test (2vPLC-B) of the virtual PLC.

Ethernet interfaces whenever the system is started to allow the vPLC to be reached from the network and reach other devices over the fieldbus network. The virtual machine should be given access to a network interface card (NIC) either through a virtual switch or directly connecting the virtual machine to the NIC.

The uppermost layer is composed of the virtual PLC or PLC runtime and other non-real-time tasks. The PLC runtime is responsible for running a process that simulates a PLC in a computer-based environment, runs a cyclic program and communicates with the inputs and outputs through a fieldbus protocol. A commercial PLC runtime is recommended as they offer better support of I/O interfaces and fieldbuses. Non-time-critical processes comprehend tasks, such as human-machine interfaces, system interrupts or retrieving and sending status of the vPLC.

IV. Evaluation

The aim of evaluating the implementation of the virtual PLC is to establish how deterministic the implementation is, how well it performs in comparison to a conventional SoftPLC and to determine if the performance fluctuates when additional processing load is introduced into the server where the vPLCs are running. The tests were performed on the (soft-) PLC that communicates over EtherCAT.

A. Implementation

The architecture outlined in Section III was implemented using the technologies presented in Figure 3. This implementation hosts up to two virtual PLCs in the server. Each of them communicates independently with the sensors and actuators through its fieldbus.

Host Server A Intel NUC with a 6-Core Intel Core i7 and 32 GB RAM was used for the implementation. Hyper-threading was deactivated in the basic input/output system (BIOS) of the server to increase determinism.

Hypervisor The bare-metal hypervisor VMware vSphere ESXi 7.0 U2 was directly installed on the solid-state drive. Within the hypervisor environment, the virtual machines were connected to the physical network interface cards over a dedicated virtual switch (vSwitch). In the hypervisor configuration, latency sensitivity was set to high. The CPU cores and RAM that were assigned to each of the VMs were reserved.

Virtual machines Along the implementation, up to three virtual machines were hosted within the server: Two containing vPLCs and one used to generate extra load for testing purposes (see Fig. 3). Each of the VMs hosting a vPLC was assigned with two CPU cores. One core was isolated for running the PLC runtime and the other core deals with other tasks and interrupts. The third virtual machine was assigned one CPU core and the remaining CPU core was left for the hypervisor.

Guest OS and workloads Linux Debian 11 was used along with the PREEMPT_RT patch. The performance of the traditional Linux kernel is not enough for real-time applications due to its lack of determinism. The PREEMPT_RT patch reduces the part of the kernel code that is non-preemptible to increase its predictability and reduce latencies characterizing the system [23]. This is achieved by modifying the symmetrical multiprocessing (SMP) capabilities of the Linux kernel without rewriting it completely, turning Linux into an operating system that meets soft real-time requirements.

Additional tuning was done on the operating system using the Daemon "Tuned" with the real-time profile. For the VM images one and two, the CODESYS Control for Linux SL version 4.2.0.0 was the runtime used on top of the operating system. For VM image three, a command was run to stress the CPU assigned to this virtual machine up to 100%.

Fieldbus Two I/O modules were connected to the Intel NUC over Ethernet interfaces. For this implementation EtherCAT and PROFINET I/O modules were used. The EtherCAT slave (Beckhoff EK1100 with EL1809 and EL2809) was connected through the Ethernet port of the server. The Profinet device (ifm AL1100) was connected over a USB/Ethernet adapter (Tp-Link UE300).

B. Configuration under Test

Four configurations were evaluated. Table I summarizes the properties of each configuration. In configurations 1vPLC, 2vPLC and 2vPLC+L, vPLCs were installed on the server. The aim of implementing these three configurations is to determine whether adding more processing workload to the server where the vPLCs were hosted affects the performance of the vPLC. In configuration 2vPLC+L, additionally, a VM to load the server was created. Configuration SoftPLC resembles a traditional SoftPLC implementation, where the operating system was installed directly on the SSD without a hypervisor. In this configuration, the operating system was tuned as well, as described in the implementation subsection and only two of the physical CPU cores were activated in the BIOS. The goal of implementing this configuration is to find out whether the vPLC can perform as well as a traditional SoftPLC implementation would or not.

C. Tests

All four configurations were evaluated using three tests to compare the implemented configurations from a software and

TABLE I
PARAMETERS OF THE CONFIGURATIONS UNDER TEST FOR THE EVALUATION OF THE VIRTUAL PLC IN A SERVER.

Configu-ration	PLCs in Server	Overload VM	Hyper-visor	Fieldbus	Active CPU cores
1vPLC	1	0	Yes	EtherCAT	3
2vPLC	2	0	Yes	EtherCAT/ PROFINET	5
2vPLC+L	2	1	Yes	EtherCAT/ PROFINET	6
SoftPLC	1	0	No	EtherCAT	2

hardware perspective. Cyclic Test was performed to evaluate the performance of the hypervisor and operating system, without considering the performance of the SoftPLC or vPLC. The CODESYS Monitoring test was carried out to assess the performance of the SoftPLC or vPLC from a software perspective. The input/output tests were performed to evaluate the performance of the SoftPLC or vPLC from a software and hardware perspective, having a full loop that encompasses processing software and physical hardware inputs/outputs.

1) Cyclic Test: The aim of this test is to assess the performance of the tuned operating system by measuring its latency. This is achieved by measuring the difference between the time a thread should wake up and the time at which it does in reality [16]. The Cyclic Test was run for 12 hours. For the configurations 1vPLC, 2vPLC and 2vPLC+L the test was run in the VM hosting the SoftPLC communicating over EtherCAT. For the configuration SoftPLC, the test was run directly on the host operating system.

2) CODESYS Monitoring: The goal of this test was to assess the performance of the SoftPLC or vPLC communicating over EtherCAT from a software perspective. In this context, a task refers to a time-based flow unit of an IEC program that calls one or more Program Organization Units (POUs). For the configurations implemented in this work a project containing two tasks was used: The main task, which contains the PLC program and the EtherCAT task, which handles the EtherCAT frame to communicate with the I/O module. The main and EtherCAT tasks called one POU. The POU called by the main task copied the value of a digital input into a digital output using Structured Text (ST). This POU had two variables that were respectively linked to the physical input and output bits of the I/O module. The POU called by the EtherCAT task was not examined. CODESYS Monitoring is a native tool integrated into the CODESYS Development System version 3.5.17.0 that provides, in real-time, metrics from a task running in a PLC, including:

Cycle period The period with which the task is cyclically executed.

Cycle time This metric represents how long a task takes to execute. It measures elapsed time between the time at which a task starts and when it finishes.

Jitter The difference between the time a task should start and the time at which it started. It should be noted that negative jitter values can also be expected in this metric.

CODESYS Monitoring also provides internal details about the tasks running in the PLC. In particular, for the EtherCAT task, it shows the number of EtherCAT frames sent and how many frames would go missing during the transmission. The CODESYS Development System was installed on a local computer in the same network as the server hosting the virtual PLCs. This test was run for 2 hours. Cyclic interval was chosen as the condition to trigger the start of the tasks. The set cycle period for the main and EtherCAT task was 500 µs and both execution priorities were set to 1.

3) Input/Output test: This test aims to assess the behavior of the vPLC from a software and hardware perspective. The oscilloscope used for the tests was the PicoScope 2205A MSO. The main and EtherCAT tasks called one POU, the set cycle period for the main and EtherCAT task was 500 µs and both execution priorities were set to 1 as well. Three input/output tests were performed on each of the configurations:

Period Jitter In this test, the period jitter of a 100 Hz PWM signal generated by the PLC was externally measured. The aim of this test is determining whether the vPLC would be suitable for applications where a PWM signal needs to be generated, e.g. motion control without a driver. For this test 1394 cycles were considered.

Switching time In this test, the switching time of the PLC was externally measured. To achieve this, a program that toggled an output of the PLC was run and the pulse width of the signal was measured with the oscilloscope. The POU called by the main task toggled the value of a digital output using Structured Text (ST). This POU had one variable linked to the physical output bit of the I/O module. This test aims to determine how long the execution of one cycle of a PLC task would last and how much jitter could be observed at the PLC output. For this test 4175 cycles were considered.

Response time from input to output The experimental response time of the PLC was externally measured with the oscilloscope. To attain this, a +24v squared signal was externally generated and connected to the input module of the PLC. The POU called by the main task copied the value of a digital input into a digital output using Structured Text (ST). This POU had two variables that were respectively linked to the physical input and output bits of the I/O module. Then, the time that it took for the PLC to change the status of an output based on this input was measured. This metric was named experimental response time from input to output. This would offer an evaluation from a full loop perspective that encompasses input hardware, processing software and output hardware. For this test 1200 of each rising and falling edges at the PLC input were considered.

D. Results

The results of the previously described tests are shown in this section.

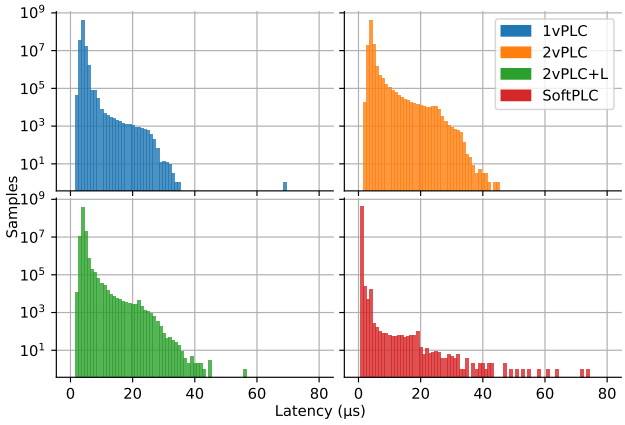

Fig. 4. Latencies recorded while running Cyclic Test during 12 hours on each of the configurations under test.

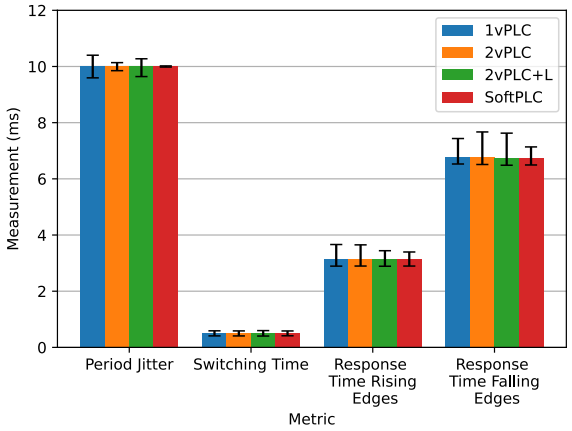

Fig. 5. Average values of the metrics retrieved in the input/output tests performed on each of the configurations under test for the vPLC communicating over EtherCAT. The vertical black error bars represent the minimum and maximum recorded values.

1) Cyclic Test: Figure 4 shows the recorded latencies for every configuration. By comparing the maximum observed latency of configuration SoftPLC (without hypervisor) with the maximum observed latency of the other configurations (with hypervisor), it can be observed that the usage of the hypervisor did not increase the maximum registered latency. The average latency was lower without a hypervisor.

2) CODESYS Monitoring: Table II summarizes the results obtained for all configurations with CODESYS Monitoring. It can be inferred that between configuration SoftPLC (without hypervisor) and configurations 1vPLC, 2vPLC and 2vPLC+L (with hypervisor), a slight increase in the cycle time and jitter arises for both main and EtherCAT tasks. The larger differences were observed in the average cycle time of the EtherCAT task as this metric was lower for SoftPLC, i.e., the execution of the EtherCAT task was faster in the SoftPLC configuration.

It was noted in all the configurations that EtherCAT frames were going missing. It should be also mentioned that the average of the values is directly retrieved from CODESYS Monitoring and no decimal positions beyond 1 ms were shown in these values.

3) Input/Output Tests: The following results were obtained for each of the input/output tests:

Period Jitter The period jitter bars in Figure 5 portray the measured average period and period jitter for all configurations. All configurations showed similar period average values. However, significant differences can be observed when considering the jitter of the signal, i.e., the black error bars on top of the average bars. The lowest jitter was obtained with configuration SoftPLC. Configurations 1vPLC, 2vPLC and 2vPLC+L showed considerably higher jitter than the SoftPLC configuration.

Switching time The switching time bars in Figure 5 represent the average switching time for all configurations. No significant differences were observed in terms of average switching time or switching time jitter, i.e., the black error bar on top of the average bars.

Response time from input to output The response time for rising edges and response time for falling edges bars in Figure 5 represent the average experimental response time from input to output for rising and falling edges at the input. No significant differences could be identified between the average values of the response times for rising and falling edges when comparing the implemented configurations with each other. However, in terms of response time jitter values, i.e., the black error bar on top of the average bars, it can be inferred that lower experimental maximum response times, for both rising and falling edges, were observed on configuration SoftPLC. Additionally, significant differences were observed in the response time depending on whether the triggering change at the PLC input would be a rising or falling edge. It can be observed that the response time for falling edges would take on average longer than for rising edges. Figure 6 shows the input and output signal behavior of the PLC when the response time from input to output was being measured by connecting a generated squared signal into the input of the PLC and connecting the output into an oscilloscope. The horizontal error bar represents the minimum and maximum recorded values for the response times from input to output.

V. Discussion

In this section, the results presented in subsection IV-D are addressed to assess the performance of the configurations under test.

The maximum latencies observed with Cyclic Test indicate that the server with a real-time hypervisor would deliver a similar performance, in terms of maximum latency, to a server without a hypervisor. The real-time tuning done for the virtual machines hosted in the hypervisor ensures that no other tasks use the cores assigned to each virtual machine and another core is responsible for the hypervisor-level tasks, this contributes

TABLE II
METRICS OF THE PLC PROJECT OBTAINED WITH CODESYS MONITORING FROM THE CODESYS IDE FOR THE EVALUATION OF THE VIRTUAL PLC

Metric (µs) / Configuration	**1vPLC**		**2vPLC**		**2vPLC+L**		**SoftPLC**	
Task	Main	EtherCAT	Main	EtherCAT	Main	EtherCAT	Main	EtherCAT
Set Cycle Period	500	500	500	500	500	500	500	500
Min. Cycle Time	1	5	1	4	1	4	1	3
Avg. Cycle Time	2	9	1	10	1	14	1	4
Max. Cycle Time	28	45	32	51	36	77	18	21
Min. Jitter	-31	-30	-38	-42	-28	-35	-26	-20
Max. Jitter	30	30	38	41	29	37	26	20

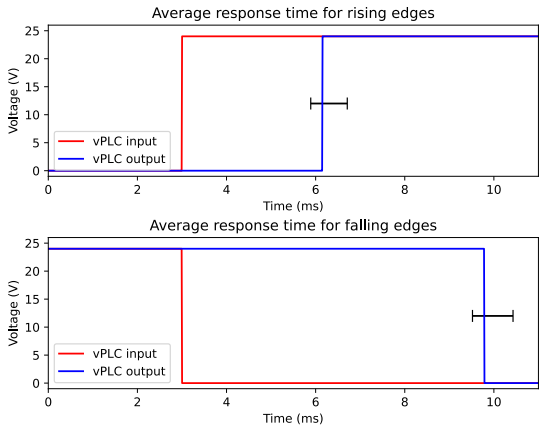

Fig. 6. Average response times from input to output for rising and falling edges at the input measured in configuration 1vPLC. The horizontal error bars represent the minimum and maximum recorded values.

to having similar maximum latencies when compared to a server without a hypervisor. However, the average registered latency was lower without a hypervisor due to the additional hypervisor layer. Similar average and maximum latency values registered between configurations 1vPLC, 2vPLC and 2vPLC+L suggest that hosting multiple vPLCs or loading the server with additional activities will not influence the latency behavior. It is important to note that, when compared to the average latency, the maximum observed latency gives a closer estimate of the worst-case latency. Worst-case latencies are a significant factor to consider for safety-critical applications [24].

The lower cycle times observed in the SoftPLC configuration for the EtherCAT task and not the main task indicate that the higher latencies observed in the configurations with a hypervisor arise from the hypervisor's networking infrastructure because the EtherCAT task accesses the devices through an Ethernet interface, whereas the main task does only processing.

The results observed in the input/output test for period jitter in the signal revealed that this specific implementation of the vPLC should not be used for applications requiring low jitter at the output, e.g., motion control. However, it would still be possible to do motion control using a driver between the vPLC and the device to be controlled, e.g. a servo motor or robotic arm.

The results obtained in the input/output test for the switching time test were similar with the ones obtained in the CODESYS Monitoring test as the period time of both main and EtherCAT task were set to 0.5 ms and their execution times were within tens of microseconds. It should be noted that the rising and falling times of the output module also add jitter to the measured times. This test shows as well that the vPLC can nearly perform as well as the SoftPLC running on a server without a hypervisor, i.e., a traditional SoftPLC implementation.

The differences observed in the input/output test for the experimental maximum response times between rising and falling edges can be attributed to the switching times for the digital output terminal Beckhoff EL2809, which are 60 µs for switching on and 300 µs for switching off. A further reason is the asymmetry of the 3ms input filter time for rising and falling edges of the digital input terminal of Beckhoff EL1809. Two further factors that increased the experimental maximum response times were the loss of EtherCAT packets while being transmitted and the time overhead added by the virtual switch in the VMware hypervisor connecting the virtual machine with the physical network interface card.

The measured maximum response times suggest that the vPLC can be used in areas of industrial automation without hard real-time requirements that can deal with a slightly longer response and switching time than the SoftPLC in a server without a hypervisor has. For instance, in factory automation, where response times between 5-10 ms are acceptable [25].

The results presented in this work and observations done throughout the implementation indicate that there are aspects that could enhance the performance observed in the configurations under test. This work can be improved with a deeper analysis of the networking infrastructure within the hypervisor. Two indicators highlight the relevance of this matter. First, the average cycle time for the EtherCAT task, i.e., the task dealing with the transmission of the EtherCAT frame, was larger for those configurations using a hypervisor. Second, EtherCAT packets were lost during the transmission, these had to be resent and, therefore, the task would take longer to execute.

A further pending issue is the hardware properties of the EtherCAT input/output module, which were not considered as part of this work. The input/output module could be replaced by a faster module, i.e., with a lower input filter and faster rising and falling times for the output. This would lead to shorter response times from input to output and less jitter on average. Other PLC runtimes could be tested to compare its

performance with CODESYS Control for Linux. CODESYS Control for Linux was chosen for these experiments because of the broad fieldbus support and monitoring tools.

VI. Conclusion

In this paper the concept of virtual PLC was explored, implemented and evaluated. This notion can help overcome the limitations of hardware-based PLCs by offering more flexibility, better resource usage, scalability and lower costs. In addition, it can fulfill the functions of PLC, HMI and programming terminal.

Tests were performed from a software and hardware perspective encompassing host software performance, hypervisor behavior, PLC and input/output module performance. Based on the results obtained in the experimental configuration, it has been seen that using a real-time hypervisor increased the average system latency while maintaining the maximum observed latency consistent. This suggests that the usage of a real-time hypervisor would not decrease the performance in terms of determinism. The evaluated configurations of the virtual PLC in a server exhibited, on average, a maximum experimental response time from input to output of 3.67 ms, being 0.274 ms larger than a SoftPLC running in the same hardware without hypervisor. The test performed also revealed that the virtual PLC can deliver similar performance in terms of switching time while having an increased period jitter. It can also be deduced from the evaluation tests that whether the server's resources are used to their maximum capacity or not does not influence the performance of the individual virtual machines or virtual PLCs.

This experimental evaluation can be followed up with longer and more exhaustive studies on the real-time performance of the hypervisor, behavior of the input/output modules, networking and hardware elements. For instance, implementations with other virtual network topologies, instead of using a virtual switch or different setups within the PLC programming environment in CODESYS. The results indicate that virtual PLCs in servers with a hypervisor is currently a viable option for important domains of factory automation, where response times between 5-10 ms are acceptable. The observations made in this work indicate that networking and I/O module enhancements are needed and these would improve the response times measured in the configurations implemented. This points out that virtual PLCs in COTS servers could become a vital component of future industrial automation systems to control a broad spectrum of processes.

References

[1] J. Waltl, *Unchain the ShopFloor through Software-Defined Automation*, May 2018. [Online]. Available: https://www.engineersrule.com/unchain-shopfloor-software-defined-automation/ (visited on 04/05/2022).

[2] H. Forbes, *The End of Industrial Automation (As We Know It)*, Dec. 2018. [Online]. Available: https://www.arcweb.com/blog/end-industrial-automation-we-know-it (visited on 04/28/2022).

[3] Y. Koren, *The Global Manufacturing Revolution: Product-Process-Business Integration and Reconfigurable Systems*. New Jersey: John Wiley & Sons, Ltd, 2010, pp. 227–252.

[4] I. Garbie and A. Garbie, "Outlook of requirements of manufacturing systems for industry 4.0," in *2020 Advances in Science and Engineering Technology International Conferences (ASET)*, Dubai, Feb. 2020, pp. 1–6.

[5] S. Vaidya, P. M. Ambad, and S. M. Bhosle, "Industry 4.0 a glimpse," *Procedia Manufacturing*, vol. 20, pp. 233–238, 2018.

[6] R. Langmann and M. Stiller, "Cloud-based industrial control services: The next generation PLC," in *Online Engineering & Internet of Things. Lecture Notes in Networks and Systems*. Cham: Springer International Publishing, 2018, vol. 22, pp. 3–18.

[7] P. Gaj, J. Jasperneite, and M. Felser, "Computer communication within industrial distributed environment a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 182–189, Jul. 2013.

[8] J. Mellado and F. Nuñez, "A container-based iot-oriented programmable logical controller," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1, Tampere, Jun. 2020, pp. 55–61.

[9] E. R. Alphonsus and M. O. Abdullah, "A review on the applications of programmable logic controllers (PLCs)," *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 1185–1205, Feb. 2016.

[10] T. Cruz, R. Queiroz, P. Simoes, and E. Monteiro, "Security implications of SCADA ICS virtualization: Survey and future trends," in *ECCWS 2016 - 15th European Conference on Cyber Warfare and Security*, Munich, Jun. 2016.

[11] O. Givehchi, J. Imtiaz, H. Trsek, and J. Jasperneite, "Control-as-a-service from the cloud: A case study for using virtualized PLCs," in *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, Toulouse, May 2014, pp. 1–4.

[12] T. Goldschmidt, M. K. Murugaiah, C. Sonntag, B. Schlich, S. Biallas, and P. Weber, "Cloud-based control: A multi-tenant, horizontally scalable Soft-PLC," in *2015 IEEE 8th International Conference on Cloud Computing*, New York, Jun. 2015, pp. 909–916.

[13] C. Pallasch, S. Wein, N. Hoffmann, M. Obdenbusch, T. Buchner, J. Waltl, and C. Brecher, "Edge powered industrial control: Concept for combining cloud and automation technologies," in *2018 IEEE International Conference on Edge Computing (EDGE)*, San Francisco, CA, Jul. 2018, pp. 130–134.

[14] A. Badar, D. L. Zhe, U. Graf, C. Barth, and C. Stich, "Intelligent edge control with deterministic-ip based industrial communication in process automation," in *2019 15th International Conference on Network and Service Management (CNSM)*, Halifax, Oct. 2019, pp. 1–7.

[15] T. Cruz, P. Simoes, and E. Monteiro, "Virtualizing programmable logic controllers: Toward a convergent approach," *IEEE Embedded Systems Letters*, vol. 8, no. 4, pp. 69–72, Sep. 2016.

[16] *Cyclictest*, The Linux Foundation, Aug. 2018. [Online]. Available: https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cyclictest/start (visited on 11/03/2021).

[17] A. Menon, A. L. Cox, and W. Zwaenepoel, "Optimizing network virtualization in Xen," in *USENIX 2006 Annual Technical Conference*, Boston, Jun. 2006.

[18] Y. Dong, D. Xu, Y. Zhang, and G. Liao, "Optimizing network I/O virtualization with efficient interrupt coalescing and virtual receive side scaling," in *IEEE International Conference on Cluster Computing, ICCC*, Austin, Sep. 2011.

[19] S. Xi, J. Wilson, C. Lu, and C. Gill, "RT-Xen: Towards real-time hypervisor scheduling in Xen," in *Embedded Systems Week 2011, ESWEEK 2011 - Proceedings of the 9th ACM International Conference on Embedded Software, EMSOFT'11*, Taipei, Oct. 2011.

[20] M. Lee, A. S. Krishnakumar, P. Krishnan, N. Singh, and S. Yajnik, "Supporting soft real-time tasks in the Xen hypervisor," in *VEE 2010 - Proceedings of the 2010 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, New York, Mar. 2010.

[21] J. Heo, "Voice over IP (VoIP) performance evaluation on VMware vSphere 5," VMware Inc., Palo Alto, CA, Tech. Rep., Dec. 2011.

[22] J. Heo and L. Singaravelu, "Performance study: Deploying extremely latency-sensitive applications in VMware vSphere 5.5," VMware Inc., Palo Alto, CA, Tech. Rep., Sep. 2015.

[23] F. Reghenzani, G. Massari, and W. Fornaciari, "The real-time linux kernel: A survey on PREEMPT_RT," *ACM Comput. Surv.*, vol. 52, no. 1, Feb. 2019.

[24] L. Zhao, P. Pop, and S. S. Craciunas, "Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41 803–41 815, 2018.

[25] PROFIBUS International, "PROFINET Real-Time Communication," Tech. Rep., 2013.