RunPHI: Enabling Mixed-criticality Containers via Partitioning Hypervisors in Industry 4.0

Marco Barletta, Marcello Cinque, Luigi De Simone, Raffaele Della Corte, Giorgio Farina, Daniele Ottaviano

DIETI - Università degli Studi di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy

{marco.barletta, macinque, luigi.desimone, raffaele.dellacorte2, giorgio.farina, daniele.ottaviano}@unina.it

Abstract—Orchestration systems are becoming a key component to automatically manage distributed computing resources in many fields with criticality requirements like Industry 4.0 (I4.0). However, they are mainly linked to OS-level virtualization, which is known to suffer from reduced isolation. In this paper, we propose *RunPHI* with the aim of integrating partitioning hypervisors, as a solution for assuring strong isolation, with OS-level orchestration systems. The purpose is to enable container orchestration in mixed-criticality systems with isolation requirements through partitioned containers.

Index Terms—Partitioning hypervisor, Orchestration, Mixed-criticality, Containers, Industry 4.0

I. INTRODUCTION

Nowadays, we are witnessing the spread of Information Technologies in several industrial domains (e.g., railways, avionic, automotive). This transforms industrial scenarios in Edge cloud environments populated by many Industrial Internet of Things (IIoT), looking towards the *Industry 4.0* (I4.0) vision [1], [2]. Thus, these systems must meet not only mandatory regulatory requirements involving functional safety and control timeliness, but also performance scalability, interoperability, low latency, and reconfigurability through fast and efficient deployment.

Virtualization is an enabling technology for I4.0 since it responds to the needs of reconfiguration, modularity, and consolidation through resource partitioning and multiplexing. It allows the execution of heterogeneous Operating Systems (OS) (real-time and general-purpose) on the same system-on-a-chip (SoC), becoming a prominent way for the industry to realize Mixed-Criticality systems due to the current COVID-19-induced silicon shortage phenomenon [3]–[5]. *Partitioning hypervisors* (e.g., *Jailhouse* [6], *Bao* [7], *Xtratum* [8]) have gained the attention of both academia and industry [9], [10] due to the strong isolation provided through static allocation, at the cost of a reduced flexibility of deployment compared to classical virtualization (recently named *consolidating hypervisors*).

However, in the I4.0 vision, the stress is on the automatic management, reconfiguration, and self-healing of IT systems. Thus, *criticality-aware orchestration systems* are paramount since they automatically place, deploy, monitor, and migrate the packaged software across the infrastructure [11]; still being aware of the isolation guarantees required by critical workloads to prevent interferences in terms of faults and attacks from non-critical jobs. Currently, containers are seamlessly integrated into orchestration systems, but ensuring their isolation is still an open issue, threatening the practicability of OS-level virtualization under strict real-time, safety, and security requirements. Unikernels seem to be a solution since they do not share the underlying host kernel, but their portability issues and real-time support are still open issues [12].

In this position paper, we propose *RunPHI*, a framework that integrates partitioning hypervisors into container orchestration systems with the aim of leveraging the strong isolation provided by partitioning solutions while taking advantage of orchestration techniques. This project advances the state of the art since *i)* it simplifies the deployment of critical workloads in edge/cloud environments, useful for maintenance, upgrades, and new deployments; *ii)* it enables failure mitigation through migration and spawning of new partitions; *iii)* it is a driving force for the full reconfigurability of I4.0 for workloads with isolation requirements.

II. RELATED WORK

In the literature, several solutions (summarized in Table I) adapt general-purpose hypervisors with the aim of providing true isolation between containers, aka *sandboxed containers*. *IBM Nabla*[1] builds containers on top of unikernels. *Google gVisor*[2] creates a dedicated guest kernel to run containers. *Amazon Firecracker*[3] is a lightweight hypervisor for sandbox applications. Both *KubeVirt*[4] and *vSphere Integrated Containers (VIC)*[5] integrate VMs and containers under a single orchestration infrastructure. *Kata Containers*[6] allows running secure container runtime with lightweight VMs. *RunX*[7] uses Xen hypervisor to run containers in multiple separate VMs, either with the provided custom-built Linux-based kernel, or with container-specific kernel/ramdisk.

TABLE I
STATE-OF-THE-ART SOLUTIONS FOR PARTITIONED CONTAINERS.

Solution	Guest Type	Used Hypervisor	Orchestration Support
Nabla Container	Unikernel	*Nabla Tender*	Docker
gVisor	Container + user-space kernel	KVM	Kubernetes, Docker
Firecracker	Light VM	KVM	OCI compliant
KubeVirt	VMs and Containers	KVM	Kubernetes
vSphere Integrated Containers (VIC)	VMs and Containers	VMware ESXi	VMware Orchestrator, Docker
KataContainer	Light VM	QEMU/KVM	Kubernetes, Docker
RunX	Light VM	Xen	Kubernetes, Docker
RunPHI	Light VM	Partitioning Hypervisors	Kubernetes, Docker, OCI compliant

[1]https://nabla-containers.github.io/

[2]https://gvisor.dev/

[3]https://firecracker-microvm.github.io/

[4]https://kubevirt.io/

[5]https://vmware.github.io/vic-product/

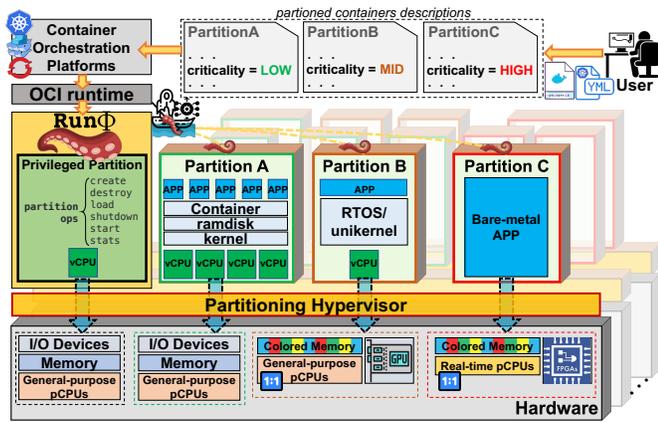[6]https://katacontainers.io/

[7]https://github.com/Xilinx/runx

1

Fig. 1. Proposed *RunPHI* Architecture.

These solutions are mainly based on general-purpose hypervisors, which do not fit well with mixed-criticality real-time requirements. In contrast, current partitioning hypervisor solutions seem to provide enough guarantees about both safety and security isolation. To the best of our knowledge, there are no solutions that support sandboxed containers in conjunction with partitioning hypervisors. The objective of *RunPHI* is to provide both isolation requirements and flexible orchestration capabilities for next-generation I4.0 scenarios.

III. PROPOSAL

Figure 1 shows a first design of *RunPHI*. Users provide *partition descriptions* via classical tools inherited from container-based orchestration (e.g., Dockerfiles, Kubernetes manifests). *Partitioned container* descriptions can be extended with requirements related to physical resources, criticality levels (e.g., low, mid, or high), real-time constraints, etc. *RunPHI* leverages a partitioning hypervisor to provide strong isolation between containers. In particular, according to partition descriptions, *RunPHI*, implemented in the privileged partition, tries to allocate physical resources in line with free resources within the host node. The *inference engine* fills the gaps with predicted values for resources not specified in the *partitioned container* description, according to requests from the orchestration platforms and current usage of hardware. *RunPHI* manages the lifecycle of partitioned containers and is designed to be highly flexible with the aim of orchestrating partitioned containers with a: i) *low-level criticality* (e.g., partition A in Figure 1) that includes a classical container abstraction with several applications running on top of it, a number of virtual CPUs (vCPUs) with no specific affinity on physical CPUs (pCPUs), without specific real-time guarantees; ii) *mid-level criticality* (e.g., partition B in Figure 1) that includes running a RTOS/unikernel single-app with strict temporal and memory isolation requirements (e.g., by using 1-to-1 vCPU-pCPU mapping and cache/RAM coloring mechanisms respectively) and use of GPUs accelerators for running machine learning algorithms; iii) *high-level criticality* (e.g., partition C in Figure 1) that includes same mechanism for mid-level criticality with the addition of running bare-metal tasks on real-time CPUs and use of programmable logic blocks like FPGAs.

IV. RESEARCH QUESTIONS AND OBJECTIVES

In the following, we delineate research questions to be considered in the next steps of our project.

RQ1. How I4.0 mixed-criticality systems can be deployed via *RunPHI*?

Objectives:
▷ To support partitioned containers run at different criticality with real-time constraints
▷ To support running bare-metal applications
▷ To support accelerator devices like FPGAs and GPUs
▷ To induce minimal overhead in terms of CPU, memory, and I/O

RQ2. How to quantify the isolation between partitioned containers provided by *RunPHI*?

Objectives:
▷ To support temporal, memory, and fault isolation assessment (e.g., fault injection testing)
▷ To support security isolation assessment (e.g., fuzzing)

RQ3. How to support orchestration for partitioned containers in *RunPHI*?

Objectives:
▷ To support different runtime containers and OCI-compliance
▷ To support partitioned containers description via existing runtime containers API and existing tools for configuration file (e.g., Dockerfile, Kubernetes manifest)
▷ To implement an inference engine to determine (sub)optimal resource allocation for partitioned containers
▷ To support migration, checkpointing, and high-availability mechanisms for partitioned containers

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE TII*, 2018.

[2] A. Stavdas, "Networked Intelligence: A Wider Fusion of Technologies That Spurs the Fourth Industrial Revolution—Part I: Foundations," *Pluto Journals World Review of Political Economy*, 2022.

[3] Bloomberg. (2022) Automotive Chip-Shortage Cost Estimate Surges to $110 Billion. [Online]. Available: https://tinyurl.com/2p9akbhu

[4] M. Cinque, D. Cotroneo, L. De Simone, and S. Rosiello, "Virtualizing mixed-criticality systems: A survey on industrial trends and issues," *Elsevier FGCS*, 2021.

[5] A. Cilardo, M. Cinque, L. De Simone, and N. Mazzocca, "Virtualization over Multiprocessor System-on-Chip: an Enabling Paradigm for Industrial IoT," *IEEE Computer*, 2021, DOI: 10.1109/MC.2022.3140896.

[6] R. Ramsauer, J. Kiszka, D. Lohmann, and W. Mauerer, "Look mum, no vm exits!(almost)," *arXiv preprint arXiv:1705.06932*, 2017.

[7] J. Martins, A. Tavares, M. Solieri, M. Bertogna, and S. Pinto, "Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems," in *Proc. NG-RES*. Schloss Dagstuhl - LZI, 2020.

[8] A. Crespo, I. Ripoll, and M. Masmano, "Partitioned embedded architecture based on hypervisor: The XtratuM approach," in *Proc. EDCC*. IEEE, 2010, pp. 67–72.

[9] HERMES2020, "qualification of High pErformance pRogrammable Microprocessor and dEvelopment of Software ecosystem." [Online]. Available: https://cordis.europa.eu/project/id/101004203

[10] SELENE, "SELENE: Self-monitored Dependable platform for High-Performance Safety-Critical Systems." [Online]. Available: https://cordis.europa.eu/project/id/871467

[11] M. Barletta, M. Cinque, L. De Simone, and R. Della Corte, "Achieving isolation in mixed-criticality industrial edge systems with real-time containers," in *Proc. ECRTS*. Schloss Dagstuhl - LZI, 2022.

[12] K.-H. Chen, M. Günzel, B. Jablkowski, M. Buschhoff, and J.-J. Chen, "Unikernel-based real-time virtualization under deferrable servers: Analysis and realization," in *Proc. ECRTS*. Schloss Dagstuhl - LZI, 2022.