Service Oriented Computing and Applications (2023) 17:73–75
https://doi.org/10.1007/s11761-023-00360-0

EDITORIAL

Real-Time FaaS: serverless computing for Industry 4.0

Marcello Cinque[1]

Published online: 3 April 2023

1 Introduction

In recent years, we are witnessing the advent of service computing and cloud technologies in industrial applications, with intriguing innovations and novel compelling challenges. For instance, in the automotive, there are initiatives for consolidating electronic control units (ECUs) as virtual machines on the same board [1]. Or, in the Industry 4.0 (I4.0), researchers and practitioners are dealing with the challenge of making the factory floor *programmable* by *softwarizing* hardware elements with edge cloud-native components [2,3].

Virtualization technologies typical of cloud settings are hence starting to be adopted in such industrial scenarios, to consolidate multiple applications on the same hardware, mainly to reduce Size, Weight, Power and Cost (SWAP-C). Virtualization also serves the need of running, on the same device, applications with differentiated criticality levels, to realize so-called Mixed-Criticality Systems (MCS) [4]. This can enable, for instance, to run on the same board a Drive-by-wire ECU side by side with the infotainment.

To fully embrace the cloud paradigm in the industry, in terms of also application development, deployment and operations (the so-called *devops*), it is also emerging the idea of adopting cloud-native solutions, such as containerization and serverless computing, in industrial settings [5].

Serverless computing is a recent paradigm for the deployment of cloud applications, due to the large adoption of containers and microservices for the development of enterprise applications [6]. It is a term coined by software industry to describe a programming model where code snippets can run in the cloud without any control or knowledge on the resources on which the code run. It does not mean that servers are removed, rather they "disappear" from the user perspective, who is no longer billed to rent a server, but to run his code as a service. When functions are used as deployment (and billing) units, the paradigm is called Function-as-a-Service (FaaS).

We see tremendous potential in the application of the FaaS paradigm in industrial settings, in terms of flexibility, compatibility and re-use of know-how. The paper discusses this trend, that we name *Real-Time FaaS*, along with its benefits and requirements. It then reviews the recent evolution of virtualization solutions, enabling such view, along with the many open challenges ahead to be faced to make Real-Time FaaS a reality for the future industry.

2 Real-Time FaaS: vision and requirements

Imagine to write a function that controls the maneuver of a robot in a factory floor in real-time. The developer, after that automated tests are passed, can deploy the function in the "industrial cloud" without actually knowing (and caring) if it will be run on the robot's microprocessor or the robot's closest edge server. So, for instance, the function is initially deployed on the robot. And, over time, it is transparently migrated or scaled on the infrastructure, depending on its non-functional requirements, e.g., latency, availability, etc.). Let's imagine that a fault in the control software is detected trough real-time comparisons with the predictions made by a digital twin function, running elsewhere on a cloud server. As response, a diverse replica of the function is spawn in real time and run on the edge, to guarantee the correct continuity of the robot and bring it to a safe state. Developers, after analyzing the monitoring data collected by a different function, running side by side with the control function, uncover and fix the robot's control function, run automated test routines, and, when passed, deploy the new version of the function on the industrial cloud, in the same way and using the same tools they would adopt in a serverless environment.

Realizing such vision requires to rethink the traditional cloud infrastructure, in order to face a number of fundamental requirements typical of real-time industrial systems:

- *Computing timeliness*: industrial functions usually have stringent timing constraints, in terms of deadlines and/or

✉ Marcello Cinque
macinque@unina.it

[1] Federico II University of Naples, Via Claudio 21, 80125 Napoli, Italy

cyclic behavior, needed to control a process in real time. A function cannot be deployed on a computing node that has not enough remaining computing power to accommodate the required load.
- *Network timeliness*: similarly, functions need to interact each other often in a deterministic way, to exchange commands or to detect anomalies and start timely responses. The deployment must take into account networking timing constraints, other than computing nodes capabilities.
- *Isolation*: multiple functions can be deployed on the same node, and multiple data flows can use the same network channel. Nodes and networks have to *assure* isolation guarantees: the timing behavior of a function must not by affected by other functions running on the same node or using the same network, also in case of accidental faults or malicious attacks.
- *Mixed-criticality*: different functions might have different levels of criticality that can influence deployment choices. Placement decisions of functions over the infrastructure (made by the so-called *orchestrator*) have to take into account the level of assurance that individual nodes and/or networks can guarantee. In this way, a highly critical function will be deployed on a node able to assure the required level of isolation and timeliness, while a non-critical function will be deployed using best effort heuristics.
- *Transparency*: all deployment and/or migration choices are transparent to the developer that will only specify the desired assurance or criticality level, along with timing requirements, in a standard way (i.e., editing a yaml configuration file processed by the orchestrator, as it would do for whatever function to be run in a serverless environment).

3 Current solutions enabling Real-Time FaaS

The consolidation of isolated and mixed-criticality functions in industrial settings can benefit from the large body of solutions on real-time virtualization.

Partitioning hypervisors are among the main virtualization solutions used in industrial environments. An example is *Jailhouse* [7], a Linux-based partitioning hypervisor developed by Siemens. It introduces the notion of *cells*, with statically assigned resources that are exclusively mapped to one guest operating system (OS) and its applications called *inmates*, that, in our vision, can host the functions of a Real-Time FaaS. *Bao* [8] is a lightweight bare-metal hypervisor for mixed-criticality IoT systems, focusing on security and safety requirements by providing strong isolation, fault-containment and real-time features. *Xtratum* [9] is a paravirtualized partitioning hypervisor certifiable for the avionic domain according to the ARINC 653 standard.

Lightweight virtualization approaches, e.g., *containers*, are starting to be explored as well in industrial domains. For instance, VxWorks by WindRiver, a popular real-time OS, now features an OCI (Open Container Initiative) compliant container engine. Also many open-source solutions are available, exploring the use of containers to run real-time tasks on real-time patched Linux kernels (using the PREEMPT_RT patch) [10,11]. Other open-source solutions are based on the *real-time group scheduling* (rt-cgroups), provided by Linux to host real-time containers. Or, they adopt real-time co-kernels, as RTAI or Xenomai, to schedule hard real-time tasks within containers [12].

Orchestration systems are another key element for the Real-Time FaaS vision, as they are used to automatically place, deploy, monitor and migrate the packaged software across the infrastructure, behaving as cloud operating systems [13]. Recently, several studies emerged to adapt existing orchestration systems to real-time needs, many of them based on the popular Kubernetes platform [14,15].

4 Open issues

Despite the great deal of solutions available for real-time virtualization and orchestration, further research is needed toward the realization of the Real-Time FaaS, at least along the following directions:

- *Orchestration*: current conventions to specify functions' requirements and node capabilities need to be extended. Orchestrators for an industrial serverless environment need to take into account the criticality and timeless demands of the functions, along with the assurance level of the nodes and networks. This enables orchestrators to perform optimal deployment choices, beyond the current solutions (based mainly on available CPU/memory on nodes), so to be able to meet the novel requirements of industrial settings.
- *Device and network Heterogeneity*: differently from cloud platforms, where uniform server machines are used, industrial settings see a proliferation of heterogeneous devices and networks. A standardization effort is needed, as already done in some sectors (such as the AutoSAR platform for the automotive). Interoperable hardware/software and networking solutions need to be devised, to abstract underlying details.
- *Virtualization of accelerators*: to answer stringent performance requirements, advanced platforms are used in the industrial edge [16], such as multi-processor systems on chip (MPSoC), featuring Graphical Processing Units (GPUs) Field-Programmable Gate Arrays (FPGAs) and Real-Time Processors. In server environments, accelerator virtualization can rely on some form of support, but

no solutions are directly supported at the hardware level by embedded devices at present. In addition, no common abstractions are defined to make such resources easily accessible and shareable by functions.

- *Safety and Security*: the use of node and network orchestration solutions and virtualization in the industrial field may open to novel risks and attacks. The Real-Time FaaS must consider industrial safety and security standards, such as the IEC 61508 (for safety-related systems) or the ISO/IEC 15408 (common criteria), requiring to satisfy stringent requirements in terms of the provided partitioning level, the degree of resource isolation, complete control over the network channels, and the adoption of monitoring and auditing mechanisms.

5 Conclusion

This paper discussed the Real-Time FaaS vision, along its main benefits and challenges. There is potential in the adoption of cloud technology in the industry, in terms of reuse of know-how and streamlined devops. However, the path ahead is still disseminated with unprecedented challenges, requiring further research efforts and academia–industry cooperation.

Acknowledgements This work is partially supported by the Italian Plan for Recovery and Resilience, PE11 (circular and sustainable Made in Italy), Spoke 1: "Digital Advanced Design: Technology, Processes and Tools".

References

1. Intel, ECU Consolidation Reduces Vehicle Cost, Weight, and Testing, accessed 2023/03/10 08:41:26(2022). https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ecu-consolidation-white-paper.pdf
2. Gogouvitis SV, Mueller H, Premnadh S, Seitz A, Bruegge B (2020) Seamless computing in industrial systems using container orchestration, Elsevier Future Generation Computer Systems
3. Bortolini M, Galizia FG, Mora C (2018) Reconfigurable manufacturing systems: literature review and research trend. J Manuf Syst 49:93–106
4. Burns A, Davis RI (2022) Mixed criticality systems-a review, York
5. Gil G, Corujo D, Pedreiras P (2021) Cloud native computing for industry 4.0: Challenges and opportunities, In: 2021 26th IEEE international conference on emerging technologies and factory automation (ETFA), pp 01–04. https://doi.org/10.1109/ETFA45728.2021.9613386
6. Baldini I, et al. (2017) Serverless computing: Current trends and open problems, in: Research Advances in Cloud Computing, Springer, pp 1–20. https://doi.org/10.1007/978-981-10-5026-8-1
7. Ramsauer R, Kiszka J, Lohmann D, Mauerer W (2017) Look mum, no vm exits!(almost), arXiv preprint arXiv:1705.06932
8. Martins J, Tavares A, Solieri M, Bertogna M, Pinto S (2020) Bao: A lightweight static partitioning hypervisor for modern multi-core embedded systems, In: Proc. NG-RES, Schloss Dagstuhl - LZI
9. Crespo A, Ripoll I, Masmano M (2010) Partitioned embedded architecture based on hypervisor: The XtratuM approach, In: Proc. EDCC, IEEE, pp 67–72
10. Mao C-N, et al (2015) Minimizing latency of real-time container cloud for software radio access networks, in: Proc. CLOUDCOM, pp 611–616
11. Masek P, Thulin M, de Andrade HS, Berger C, Benderius O (2016) Systematic evaluation of sandboxed software deployment for real-time software on the example of a self-driving heavy vehicle, CoRR abs/1608.06759. arXiv:1608.06759
12. Barletta M, Cinque M, De Simone L, Della Corte R (2022) Achieving isolation in mixed-criticality industrial edge systems with real-time containers, In: Proc. ECRTS, Schloss Dagstuhl - LZI
13. Casalicchio E (2019) Container orchestration: a survey. Methodologies and Tools, Springer Systems Modeling
14. Fiori S, Abeni L, Cucinotta T (2022) Rt-kubernetes-containerized real-time cloud computing, In: Proc. SAC
15. Johansson B, Rågberger M, Nolte T, Papadopoulos AV (2022) Kubernetes orchestration of high availability distributed control systems, In: Proc. ICIT
16. Cilardo A, Cinque M, De Simone L, Mazzocca N (2021) Virtualization over Multiprocessor System-on-Chip: an Enabling Paradigm for Industrial IoT. IEEE Comput. https://doi.org/10.1109/MC.2022.3140896