New generation virtualized ERTMS for interoperable railway infrastructure

Domenico Uomo, Anna Lina Ruscelli, Gabriele Cecchetti, Piero Castoldi
Scuola Superiore Sant'Anna, Pisa, Italy
{domenico.uomo, annalina.ruscelli, gabriele,checchetti, piero.castoldi}@santannapisa.it

Abstract. The European Railway Traffic Management System (ERTMS) is the next-generation traffic management system designed to enhance railway interoperability among European countries. ERTMS must also address critical challenges in the coming years, including reducing the power consumption of railway systems, promoting rail transport as a sustainable mode of transportation, enhancing international connections, and leveraging the vast amount of data produced by TMS to improve service quality. In order to address all these challenges, it is crucial to deploy a new generation ERTMS that is also able to perform a wide set of tasks based on the analysis of data collected from rolling stocks and railway infrastructure; therefore, a push through the softwarization of the components is unavoidable. This paper presents preliminary results on the use of virtualization technologies and microservices as a potential solution to the mentioned challenges, along with a monitoring system to ensure the minimum standard of system safety and predictability.

Keywords: ERTMS, railway, virtualization, containers, microservices, real-time systems.

1 Introduction

The new generation of European Railway Traffic Management System (ERTMS) must provide an answer to the requests for interoperability, scalability, sustainability, and safety in the railway sector. All these requirements are fundamental for developing an ERTMS architecture that can be adopted by all the European Union countries. Legacy ERTMS is responsible for a wide set of functions, from safety-critical operations, such as those related to the European Train Control System, and allows the monitoring and management of Railways Business Services (RBS) and external services, like weather forecasts and Passenger Information System. However, the lack of standardized interfaces between different services prevents the possibility of interconnecting diverse ERTMS and implies a great effort to keep the systems up to date. Furthermore, the new generation ERTMS must involve a wide set of applications to optimize train movement and allow a higher level of automation in train operation. Particularly, four Grades of Automation (GoA) in train operations have been defined (see Fig. 1), which are currently deployed up to GoA-2. This scenario raises the need for an ERTMS architecture capable of integrating and managing a huge amount of data from different data sources and based on standardized frameworks, data structures, communication infrastructures, and real-time data management. The crucial role of ERTMS in improving the railway services is highlighted in [10]. Here, the authors also

stress the importance of digitalization in improving safety and optimizing maintenance costs for infrastructures and rolling stocks.

GoA	Train Operations	Setting the train in motion	Driving and stopping the train	Opening and closing door	Operation in case of disruptions
GoA 1	ATP with driver	Driver	Driver	Driver	Driver
GoA 2	ATP and ATO with driver	Supervised by driver	Automatic	Driver	Driver
GoA 3	Driverless	Automatic	Automatic	Supervised by attendant	Supervised by attendant
GoA 4	Unattended	Automatic	Automatic	Automatic	Automatic

Fig. 1. Grades of Automation

Focusing on ERTMS implementation, the European Union Agency for Railways has defined three levels of ERTMS implementation, based on how signaling is performed with the progressive introduction of monitoring and communication methodologies based on radio transmissions instead of physical monitoring systems (Eurobalises). However, no definition has yet been proposed for the Information Technology infrastructure that can support this evolution. Indeed, as pointed out in [9], no formal requirement has been specified, leaving the interpretation to the stakeholder.

The shift to level 2 (and beyond) of ERTMS has also highlighted the need for softwarization. Indeed, the adoption of a radio communication based on the GSM-R protocol, rather than Eurobalises, enables continuous information exchange between trains and trackside equipment, and the collection of fine-grained information about train state. This requires advanced software frameworks for filtering, processing, and utilizing the data to optimize train running, thereby improving the overall service offered. Nevertheless, so far, the digitalization process has not yet yielded the expected results [8]. Besides that, the authors in [11] pointed out some technologies that could be applied to railway TMS, such as IoT devices to collect data, and big data platform, and AI algorithms to process that data to perform predictive maintenance, optimize the train's time schedule, and enhance the safety of trains.

Several European and national research projects have investigated the next generation of ERTMS, yielding promising results that can serve as a foundation for validating future ERTMS applications. In particular, the Shit2Rail H2020 OPTIMA (cOmmunication Platform for TraffIc ManAgement demonstrator) project [2] designed and implemented a new architecture with a standardized communication system, the Integration Layer (IL), which uses standardized and interoperable interfaces between different services and applications and a Common Data Model (CDM) [6]. The IL uses the publish-subscribe paradigm to provide access to persistent data, real-time status data from heterogeneous sources, high availability, and a configurable Quality of Service (QoS). On the other hand, in [7], the introduction of network softwarization and resource virtualization is proposed as a viable solution to promote and ease interoperable and efficient ERTMS implementations. Furthermore, a promising microservices approach is introduced to deploy ERTMS applications. Indeed, microservices are lightweight processes that can be easily managed and scheduled, improving the efficiency of the system and the isolation of different applications. In this paper, the main goal is to explore the current technologies to create a framework suitable for hosting the next generation

ERTMS. The preliminary results about the introduction of virtualization techniques in the implementation of new-generation ERTMS are illustrated. This will be one of the aims of the PhDs EU-Rail project under the Horizon Europe framework funded by Europe's Rail Joint Undertaking (EU-Rail), the successor of Shift2Rail. In particular, the first evaluation results of a sample architecture that reproduces the coexistence of different railway applications with different requirements are described. The architecture under test is based on a hybrid microservices-native applications approach, coupled with a monitoring and protection mechanism for system safety and predictability.

The remaining part of the paper is organized as follows: the proposed architecture is introduced in Sec. 2, along with a brief introduction to resource virtualization technologies to find which ones are more suitable for the railway environment. Sec. 3 describes the implemented testbed, the experimental results and a proposal for a Rail Business Service Monitor, which can be configured in order to optimize the execution of critical components for the railway applications. Finally, Sec. 4 draws some conclusions.

2 A new generation ERTMS architecture

By specification, the ERTMS, is based on three main elements [1]:

- European Train Control System (ETCS): in-cab equipment that supervises train movement, providing Automatic Train Protection;
- ATO: the component that automates train operation. At the current state of the art, it allows for automation up to the GoA-2;
- GSM-R: the protocol that allows radio communication between the train and the trackside equipment.

In ERTMS level 2, it is also crucial the Radio Block Center (RBC), which receives the information from the train through a radio signal and computes the Movement Authority (MA), the permission for a train to move from one point to a specific location, taking into account the infrastructure constraints and speed supervision. In Fig. 2, the interactions among the components of the ERTMS is shown.

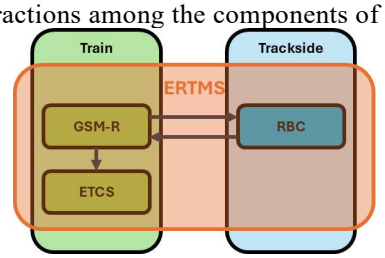

Fig. 2. Interaction among ERTMS components.

The next generation ERTMS should include more modules and components, able to perform more advanced computations, besides the simple MA and speed limits. For instance, new modules for predictive maintenance, computation of optimal speed and braking curve to save energy, a more accurate estimation of delay for the Passenger Information System (PIS) could be added into the legacy architecture. All these applications could be well integrated with the OPTIMA demonstrator [2], schematized in Fig. 3, for which a new environment suitable for the Application Framework, where the Traffic Management tasks run and are monitored, is proposed and tested in this paper.

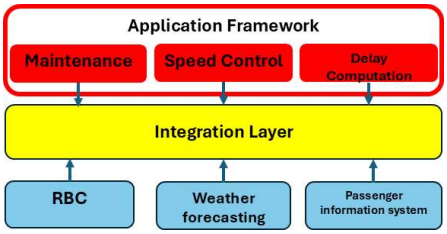

Fig. 3. Schema of the new applications environment for the OPTIMA demonstrator.

In particular, this implementation leverages containers, a lightweight virtualization technology, to deploy the applications needed for the ERTMS: a process can be scheduled either on the host Operating System (OS) or in a container, based on the application requirements. Furthermore, a monitoring system to improve the efficiency and dependability of the software tasks is proposed. This approach enables the implementation of a complex architecture with low coupling and high cohesion, and thus, highly extensible and scalable, minimizing interference between processes and maximizing system efficiency. Moreover, the used modular approach is crucial to ease the deployment of standardized software and improve its maintainability. From the efficiency point of view, the proposed approach can result in a more effective utilization of the growing volume of data generated by ERTMS to boost data analytics and, consequently, improve the efficiency and dependability of railway systems that are a source of heterogeneous big data not yet fully exploited [11].

2.1 ERTMS and resource virtualization

Virtualization is a recent standard approach for deploying different services with different requirements since it guarantees:

- Consolidation and energy efficiency: several processes can share the same hardware, overcoming the problem of having a huge number of underutilized machines.
- High availability: a Virtual Machine (VM) that fails for some reason can be rescheduled on the same node or on another node.
- Flexibility: It is possible to add new virtual components without the need for new hardware, thus providing an extensible architecture.

Currently, different kinds of virtualization frameworks are arising. Besides the classical Type 1 (or bare-metal) and Type 2 hypervisors, nowadays containers are being extensively exploited. Containers can be seen as lightweight virtual machines. They share the kernel with the host OS, allowing the deployment of extremely light virtual environments, carrying the bare minimum needed to execute specific applications. Containers can be allocated quickly and with low resources compared with traditional VMs. One of the main challenges when dealing with software sharing the same hardware is the lack of total isolation, which can bring unexpected delays for critical tasks. In an ERTMS, this is a critical issue that needs to be addressed, given the very heterogeneous applications it can run. For instance, an ERTMS can run a Passenger Information

System application, which is a soft real-time component, and, at the same time, the ETCS component, responsible for Automatic Train Protection, which has strict real-time requirements. Such diverse components must coexist in the same environment, and it is crucial to ensure that safety-critical tasks are not interrupted by soft-critical tasks. In this regard, [4] shows the performance differences among the mentioned programming approaches, concluding that container and native applications are mostly equivalent. Another topic often underestimated is the choice of the guest OS. [5] evaluates different solutions, showing that the best performances are obtained with containers running on Real-Time OSs, but VMs with a Real-Time guest OS can be a valid alternative.

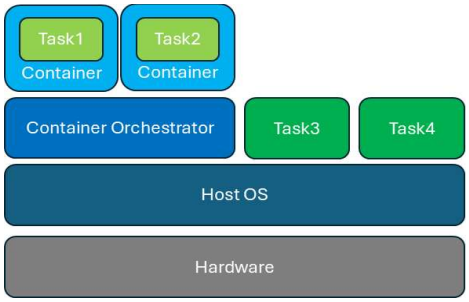

Fig. 4. Tested architecture.

3 Experimental results on ERTMS applications virtualization

In this preliminary work, the focus is on how microservices based on containers and native applications perform when they are deployed together, and whether they are suitable for ERTMS implementation. For this purpose, the architecture shown in Fig. 4 was deployed and tested. In particular, two native tasks are deployed together with two containerized applications in order to check the feasibility of such a solution for a softwareized ERTMS.

The hardware for the testbed is the following: 112 cores Intel(R) Xeon(R) Gold 6238R CPU @ 2.20GHz, 4.4Tb of storage and 251 Gb of memory.

In the experimental deployment, based on the literature review illustrated in Sec. 2.1, Linux as the host OS and Kubernetes as the container orchestrator were used. In a Kubernetes cluster, the smallest deployable unit, i.e. a group of one or more containers, is called POD. In the proposed implementation of ERTMS architectures, the single microservices are deployed as a POD.

The preliminary reported tests focus on concurrency, i.e., on the resulting performance of the tasks when simultaneously accessing shared computational resources. Indeed, in ERTMS systems, it is fundamental to verify how different processes interfere with each other, considering both critical and non-critical tasks. Interferences may cause unexpected delays in running, negatively impacting on the delivery of critical results, such as the Movement Authority. In particular, in these preliminary tests, the concurrency between containers and between processes separately, and the concurrency

between containers and processes are evaluated, considering as a metric the number of operations per second on the CPU. The CPU-bound workload test is based on a series of mathematical operations that have the purpose of "filling" the CPU's capacity and forcing the applications to alternate its usage. The first test, in Fig. 5, aims to check the behavior of two processes (i.e. two tasks running in parallel on the same host OS) and compare it with two PODs running in concurrency, so comparing the resulting performance of the system when the virtualization based on microservices is adopted. These tests show that PODs for CPU-bound workloads perform better than processes in terms of number of operations, keeping higher priority for the first run task (in blue). On the other hand, processes run on the host OS perform a lower number of operations in total, but they are treated in a fairer way in the distribution of resources between concurrent tasks.

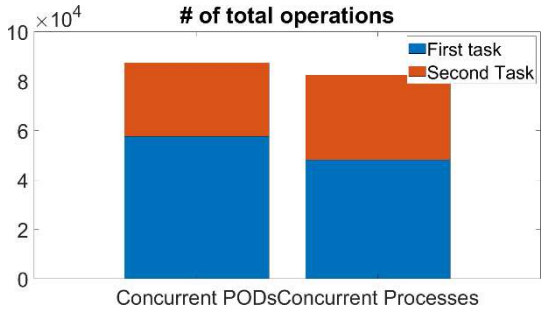

Fig. 5. CPU-bound workload with separate concurrency between PODS and processes.

To confirm what was said, the same test was repeated 30 times. The distribution of the number of operations shown in Fig. 6 highlights a tendency of PODs to perform more operations. The statistical test shows that the null hypothesis is rejected for the total number of operations with a p-value of 1.0325e-09.

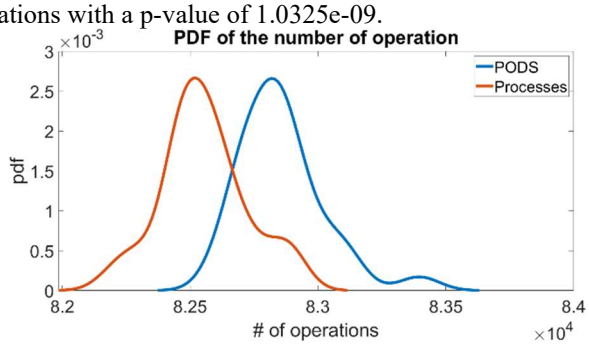

Fig. 6. Distribution of the number of operations of PODs and Processes

A second set of tests considers PODs and processes running together, in concurrency. The results are shown in Fig. 7. Here, it is very interesting to see that POD preempts the resources, and the process manages to execute very few operations. Similarly to the other case, a POD is able to perform more operations than a regular process. In a hard real-time scenario, such as the one considered for ERTMS, these results suggest that a container is very robust in its execution since it is not impacted by other processes. Differently, in the case of the competing PODs, it is

crucial to assign the tailored priority to the different tasks based on their criticality and provide a mechanism for monitoring and management of processes in order to enhance the safety of the critical Rail Business Services.

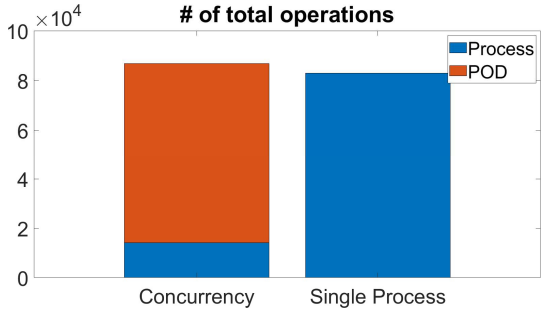

Fig. 7. CPU-bound workload with processes and PODs in concurrency.

3.1 Rail Business Services management and monitoring

The tests illustrated in the previous section show that microservices can be a valid approach to create a software ERTMS. However, as deduced above, such a solution must be enhanced through the use of a real-time mechanism for monitoring the processes, acting quickly whenever a non-critical task takes too many resources, managing unexpected system behavior, and applying different policies based on the criticality of the running tasks. Hence, a Rail Business Services Monitor is proposed together with the provided microservices solution. An RBS monitor, thus, can be implemented ad-hoc for the railway applications, in order to address the specific requirements of this particular domain. The monitor must be lightweight and fast, in order to take as few resources as possible and not become a source of delay itself. Moreover, in the case it detects some applications taking too many resources, it must be able to understand whether it is a critical application, or it is a process that can be killed without affecting the safety of the running trains. To reach those goals, the Monitor was implemented as a process written in C/C++, a de facto standard when it comes to deploying low-latency applications. It constantly checks the usage of the CPU and detects when it is above a given threshold. Then it proceeds to analyze some characteristics of the processes and decides whether they are killable or not.

In this first implementation, the threshold is set at 80% of CPU usage. The features checked by the monitor are the owner of the process, the used CPU per process, and the name and priority. Those features are retrieved using Linux built-in functionalities to check the current system status. The Monitor periodically reads the operations that the processors are running using the register /proc/stat and computes the load of the CPU. If it is above the set threshold, then it starts looking for a process to kill. In particular, the Monitor is able to discern non-critical and critical processes based on the mentioned features, and decide which ones are killable.

For instance, the RBS monitor can decide not to kill a process that is taking a great amount of resources if it has a high priority, or if it is a Real-Time process. In the first implementation proposed here, the RBS monitor searches for processes in execution and sorts them for resource usage, see Fig. 8.

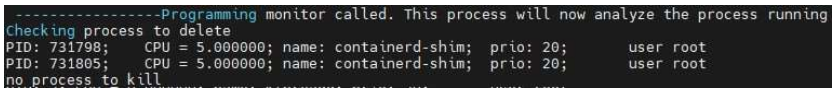

Fig. 8. RBS monitor when it finds no suitable candidate to kill.

In the shown example, the processes are using the same amount of CPU and have the same priority, but since the usage is below the threshold, it decides not to kill them. Then, a process with low priority that stresses the architecture is executed (Fig. 9). In this case, the process called "myHeavyTask" is preempting 8 CPUs almost entirely. The process is executing with a priority of 20 (average priority in Linux), therefore, since it's not a high-priority task, the RBS monitor decided that it can be safely shut down (Fig. 10), restoring a normal state of the server.

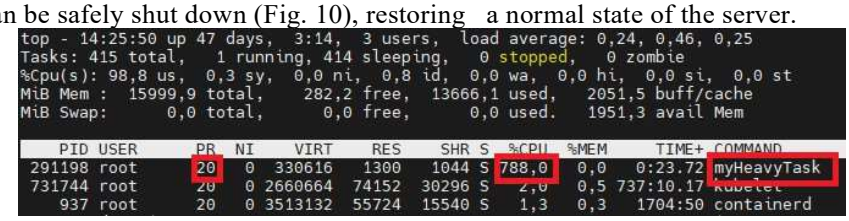

Fig. 9. Resource utilization of user process.

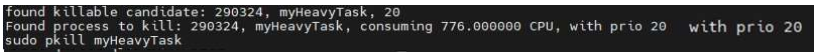

Fig. 10. The RBS monitor decides to kill the process taking too many resources.

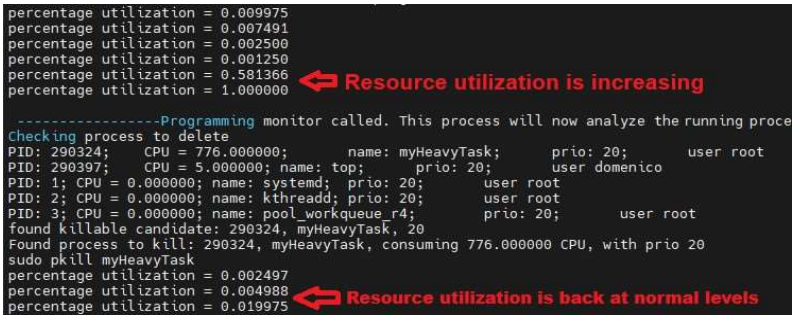

Fig. 11. The RBS Monitor detects an unusual use of resources and restores normal conditions.

Fig. 11 shows the entire output of the RBS monitor. It checks the percentage of used CPU, and when it detects an increase in resource utilization, it searches for a killable process that is taking too many resources, as illustrated above. Then, it comes back to the normal execution.

4 Conclusions

In this preliminary work, an overview of a feasibility study about a microservices-based virtualized architecture coupled with a Rail Business Services Monitor for the implementation of an extensible, safe, and flexible ERTMS, taking into account its critical characteristics, is provided.

The reported preliminary results show a very peculiar behavior in the execution of CPU-bound workloads for PODs and processes, where PODs result in being highly robust and isolated, at the expense of the native applications. The experimental results demonstrate the robustness of containers in running applications without suffering from resource competition with processes, making them a suitable technology for critical applications. However, despite the highlighted advantages of virtualization, it is necessary to take into account the peculiarities of the railway architectures, and specifically, Railway-TMS. For this reason, it is crucial to perform continuous monitoring of the software architecture in order to detect abnormal behavior and deploy a component able to take action whenever an unexpected use of resources is detected. This monitoring system must be created ad-hoc to run on a virtualized ERTMS. Indeed, it has to be able to discern critical RBS applications from other non-critical applications, allowing correct decisions when it comes to choosing which process to shut down to ensure the safety of the railway applications. In future works, the tests run so far need to be extended, including also the efficiency of the networking and I/O operations. Moreover, different technologies and approaches to run virtualized microservices might be evaluated and compared with the results obtained so far. For instance, a candidate technology for such a comparison could be the Unikernel.

In the continuation of the PhD Eu-Rail project, starting from these considerations, a more complete architecture will be built, deploying different safety mechanisms to provide a dependable framework where safety-critical applications can run and taking into account different kinds of interactions among tasks

Acknowledgement: This work has been supported by Europe's Rail Joint Undertaking through Project 101175856 — PhDs EU-Rail HORIZON-JU-ER- 2023-01.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Europe's Rail Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them.

References

1. ERTMS in brief - ERTMS — ertms.net. https://www.ertms.net/about-ertms/ ertms-in-brief/. [Accessed 29-05-2025].
2. Shit2Rail OPTIMA project, 2020. https://www.optima-project.eu/home.aspx.
3. Cecchetti, G., Ruscelli, A. L., Cristian Ulianov, P. C., Hyde, P., Oneto, L., & Márton, P. (2022). Communication platform concept for virtual testing of novel applications for railway traffic management systems. Transportation Research Procedia, 62, 832-839.
4. Kominos, C. G., Seyvet, N., & Vandikas, K. (2017, March). Bare-metal, virtual machines and containers in OpenStack. In 2017 20th conference on innovations in clouds, internet and networks (icin) (pp. 36-43). IEEE.
5. Li, Y., Zhang, J., Jiang, C., Wan, J., & Ren, Z. (2019). PINE: Optimizing performance isolation in container environments. IEEE Access, 7, 30410-30422.
6. Magnien, A., Cecchetti, G., Ruscelli, A. L., Hyde, P., Liu, J., & Wegele, S. (2022, May). Formalization and processing of data requirements for the development of next generation railway traffic management systems. In International Conference on Reliability, Safety, and Security of Railway Systems (pp. 35-45). Cham: Springer International Publishing.

7. Ruscelli, A. L., Fichera, S., Paolucci, F., Giorgetti, A., Castoldi, P., & Cugini, F. (2019, June). Introducing network softwarization in next-generation railway control systems. In 2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS) (pp. 1-7). IEEE.
8. Thaduri, A., Galar, D., & Kumar, U. (2015). Railway assets: A potential domain for big data analytics. Procedia Computer Science, 53, 457-467.
9. Rajabalinejad, M., & Schuitemaker, K. (2017). ERTMS challenges for a safe and interoperable European railway system. In 7th International Conference on Performance, Safety and Robustness in Complex Systems and Applications 2017 (pp. 17-22). IARIA/Thinkmind.
10. Arnaudov, B. (2022). Digitalization in Railway Transport as a Factor for Improving the Quality of the Offered Railway Service. Годишник на УНСС, 2(2), 145-156.
11. Sarp, S., Kuzlu, M., Jovanovic, V., Polat, Z., & Guler, O. (2024). Digitalization of railway transportation through AI-powered services: digital twin trains. European Transport Research Review, 16(1), 58.