Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

CONTAINER-BASED SIMULATION

Daniel Seufferth[1], Falk Stefan Pappert[1], Heiderose Stein[1], and Oliver Rose[1]

[1]Dept. of Computer Science, University of the Bundeswehr Munich, Neubiberg, BY, DEU

ABSTRACT

Popular methods like machine learning, simulation-based optimization, and data-farming require a simulation environment that supports scalable simulation workloads and provides access to distributed computational resources. Containerization and container orchestration are promising methods for creating such a simulation execution platform. Therefore, we provide a first concept for a hardware-agnostic, scalable, container-based simulation environment tailored to the future needs of various simulation and optimization methods.

1 INTRODUCTION AND MOTIVATION

The popularity of methods like machine learning, simulation-based optimization, or data-farming is rising. Similarly, these methods have an increasing demand for computational power, as large-scale experiments of complex simulation models require large amounts of performant hardware. Yet these resources are constrained, given that regular office PCs have limited computing capabilities.

Based on these observations, we see an increasing demand for a scalable and dynamic simulation platform that easily scales and distributes simulation workloads on different infrastructures, ranging from high-performance computers to combining multiple office PCs as small computing clusters. Anagnostou et al. (2019) evaluated technological approaches with their work on simulation experimentation frameworks, applying a micro-services auto-scaling approach utilizing MiCADO, which is tailored to server, cloud and edge infrastructures. Yet, MiCADO's architecture is less versatile, as it extends the already complex Kubernetes Application Programming Interface (API) with its Application Description Templates (see MiCADO Project (2017 - 2019) (2023)). Consequently, MiCADO requires users to familiarize themselves with the MiCADO Templates in addition to the Kubernetes API, necessitating even more expert knowledge for implementation and use. Moreover, combining simulation and a container-based environment also brings certain considerations in terms of how simulation software and its use can be effectively managed. In Seufferth et al. (2023), we discussed the requirements and limitations of simulation software in such environments. Still, we see significant opportunities for these technologies to make large-scale simulations readily available. Therefore, we want to introduce our concept for a container-based simulation architecture that supports different experiment setups, e.g., simulation-based optimization, data-farming, or machine-learning-based approaches.

2 CONCEPT FOR A SCALEABLE SIMULATION ENVIRONMENT

Our concept consists of four major components, all built in a service-based design approach. The *Design of Experiment Services (DoESs)* are the main entry points to the remaining components of the platform and represents the scenario-creating elements. These can, for example, be a data-farming service that implements a specific design of experiment. Another example would be an optimization algorithm that creates solution candidates sent off for evaluation. The scenario data provided by *DoESs* is stored in a database managed by the *Scenario Manager (SM)*, which is the management component of our concept. Simulation scenarios defined by *DoESs* get forwarded to the *Translators*, which are responsible for generating executable simulation models from the provided scenario data. The generated models get executed in the

Simulation Runners (Runners), which are simulator-specific base containers that can easily be scaled up and down based on demand.

Figure 1 visualizes our concept from the user's point of view. An architectural overview of our concept will be shown in future research. *DoESs*, *Translators*, and *Runners* are project-specific components; therefore, the implementation effort can be tailored to the complexity of the given project. Additionally, the *Scenario Detail Databases* and *Result Databases* are project-specific. In contrast, *SM*, *SM-API*, *Project Database* and *Scenario Status Database* are generic components, meaning they are used universally and are independent of the specifics and constraints set by projects. The difference in project-specificity is also indicated by color coding, showing project-specific elements in blue and generic elements in yellow. Following this mixed implementation approach allows for creating a flexible and easy-to-use framework.

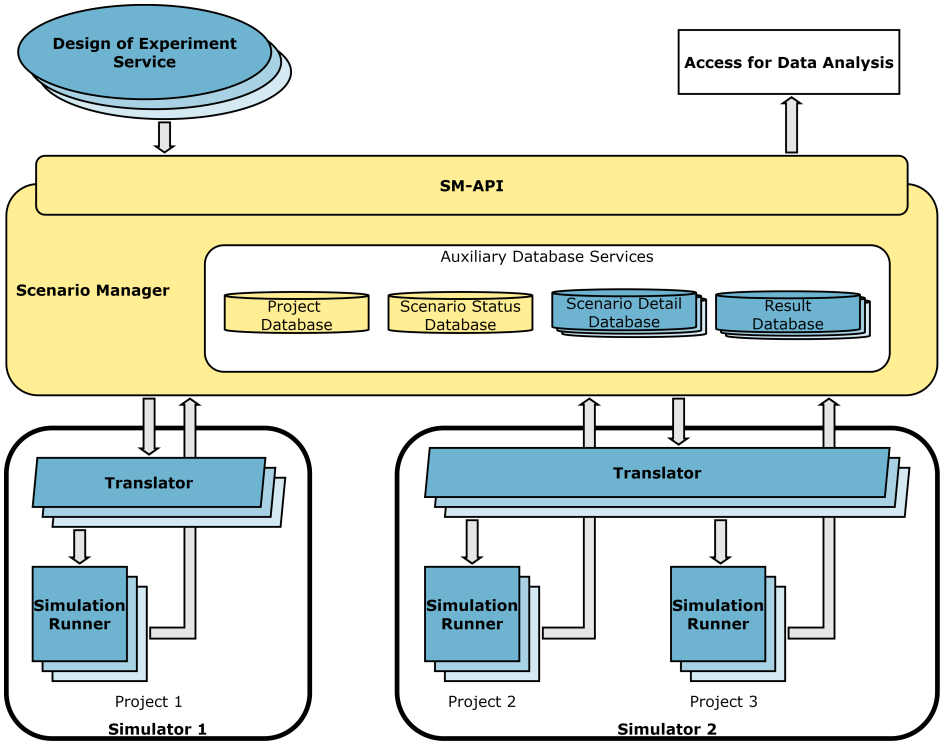

Figure 1: Concept of a container-based simulation environment, consisting of four main components: *Design of Experiment Services*, *Scenario Manager*, *Translators* and *Simulation Runners*.

ACKNOWLEDGEMENT

This research is funded by dtec.bw - Center for Digitization and Technology Research of the Bundeswehr. dtec.bw is funded by the European Union -NextGenerationEU.

REFERENCES

MiCADO Project (2017 - 2019) 2023. "Application Description Template - MiCADO". https://micado-scale.github.io/adt/.

Anagnostou, A., S. J. E. Taylor, N. T. Abubakar, T. Kiss, J. DesLauriers, G. Gesmier *et al.* 2019. "Towards a Deadline-Based Simulation Experimentation Framework Using Micro-Services Auto-Scaling Approach". In *Proceedings of the 2019 Winter Simulation Conference*, 2749–2758 https://doi.org/10.1109/WSC40007.2019.9004882.

Seufferth, D., H. Stein, F. Pappert, and O. Rose. 2023. "On the Usage of Container and Container Orchestrators as a Computational Infrastructure for Simulation Experiments". In *20. ASIM Fachtagung Simulation in Produktion und Logistik 2023*, 393–401 https://doi.org/10.22032/dbt.57789.