Streamlined Deployment of a Distributed and Scalable Co-Simulation Platform for Integrated Energy Systems

Streamlined Deployment of a Distributed and Scalable Co-Simulation Platform for Integrated Energy Systems

Fabio Chini
DEIB
Politecnico di Milano
Milan, Italy
fabio.chini@polimi.it

Davide Canali
DEIB
Politecnico di Milano
Milan, Italy
davide.canali@polimi.it

Luca Barbierato
DAUIN and Energy Center LAB
Politecnico di Torino
Turin, Italy
luca.barbierato@polito.it

Daniele Salvatore Schiera
DENERG and Energy Center LAB
Politecnico di Torino
Turin, Italy
daniele.schiera@polito.it

Lorenzo Bottaccioli
DIST and Energy Center LAB
Politecnico di Torino
Turin, Italy
lorenzo.bottaccioli@polito.it

Alessandro Margara
DEIB
Politecnico di Milano
Milan, Italy
alessandro.margara@polimi.it

Edoardo Patti
DAUIN and Energy Center LAB
Politecnico di Torino
Turin, Italy
edoardo.patti@polito.it

***Abstract*—To reach decarbonisation goals, energy systems are becoming increasingly complex and deeply integrated with other sectors and activities through digital technologies such as Big Data analytics and Digital Twins. In recent years, co-simulation has become a powerful technique to study complex cyber-physical integrated energy systems. Indeed, co-simulation enables the coupling of diverse simulators, each specialised in modelling a specific aspect of the system, to provide a comprehensive understanding of its behaviour.**

As the simulation scenarios grow in scale and complexity, they require more extensive compute infrastructures to scale horizontally. Yet, deploying and operating co-simulation platforms in distributed computing infrastructures remains a difficult task. This paper explores the challenges faced in achieving scalable co-simulation through horizontal distribution using cluster/cloud deployments, both in terms of engineering complexity for deploying and operating co-simulation platforms and in terms of absolute performance and scalability. It proposes a novel framework to streamline the process of deploying and operating complex co-simulation scenarios in cloud environments. The framework adopts established methodologies in engineering software solutions in the cloud to deploy and operate co-simulations. Specifically, it adopts containerisation technologies to decouple the definition of the simulation scenario from the execution infrastructure, automating and abstracting away distribution and deployment concerns and allowing seamless migration to different computing infrastructures. An initial assessment of absolute performance and scalability of realistic co-simulation scenarios within the proposed framework shows the benefits of horizontal scaling but also the performance bottlenecks introduced by current technologies for orchestrating simulators.

***Index Terms*—Co-simulation, Integrated Energy Systems, Distributed computing, Performance Analysis, Cloud, Software Engineering**

I. Introduction

A more flexible decentralised energy system is needed to pursue the energy transition pathway towards climate neutrality. However, future energy systems shall be more complex and deeply integrated with other sectors and activities to reach the decarbonisation goals. This is especially true in future smart cities, where different city assets, such as transportation, buildings, renewables, and so on, will be integrated by employing digital technologies [1] such as Big Data analytics for decision making [2] and Digital Twins [3].

This vision demands coordinated planning and operations of these complex cyber-physical integrated energy systems, as suggested by [4]. Towards this goal, in recent years, co-simulation has become a powerful technique for integrating multiple simulation tools to analyse complex systems [5]. Co-simulation enables the coupling of diverse simulators, each specialised in modelling a specific aspect of the system, to provide a comprehensive understanding of its behaviour.

Scalability is a critical factor in co-simulation, which refers to efficiently handling more extensive and complex simulation scenarios. It encompasses the performance of individual simulators and the intricacy of their interconnections. As the number of simulators and the complexity of the system under study grow, the performance, efficiency, and scalability of co-simulation frameworks become paramount.

Different co-simulation frameworks have been proposed in the literature to address scalable complex scenarios. Mosaik has garnered significant attention in the research community as a powerful tool for integrating and simulating diverse components within complex cyber-physical systems [6]. By enabling the integration of heterogeneous simulation models, Mosaik allows researchers to analyse the behaviour and performance of complex systems comprehensively. In addition, Mosaik provides extensive capabilities for scenario generation, data visualisation, and analysis, empowering researchers to gain valuable insights into system dynamics and optimise system

designs.

In [7], the authors have introduced a distributed multi-model platform designed for scalable multi-energy system scenarios. This platform utilises the Mosaik framework and incorporates the Functional Mock-up Interface (FMI) [8] standard to connect and synchronise different simulators and models. This platform allows models and simulators to be easily integrated plug-and-play, seamlessly replacing one or more models without impacting the entire simulation engine. This feature enhances the capability to set up more extensive and complex simulation scenarios than the Mosaik framework.

In [9], the authors have introduced HELICS, designed to enable scalable and efficient co-simulation across multiple domains by providing a flexible and modular architecture. HELICS allows for the integration of diverse simulators and models, each specialised in a specific domain while facilitating data exchange and synchronisation. The authors have demonstrated the capabilities of HELICS through several case studies, including power system simulations and co-simulation of power grids with communication networks. The results show that the framework can effectively handle large-scale simulations, providing improved scalability and performance compared to traditional co-simulation approaches. In [10], the authors present a comparative analysis between the scalability performance of Mosaik and HELICS.

However, as the simulation scenarios grow in scale and complexity, they require more and more compute capabilities, thus demanding the adoption of more significant compute infrastructures to scale horizontally. Yet, despite the promising results presented above, deploying and operating co-simulation platforms in distributed computing infrastructures remains a difficult task. Moving from these observations, this paper explores the challenges faced in achieving scalable co-simulation through horizontal distribution using cluster/cloud deployments.

Specifically, the paper addresses two main research challenges: (**RC1**) how to reduce the complexity involved in deploying and operating co-simulation frameworks in a cloud environment, and (**RC2**) how to enhance the scalability that may be achieved through horizontal distribution in cloud environments.

RC1 refers to the setup, deployment, and operation of co-simulation frameworks, which often involve complex tasks that require manual intervention from co-simulation experts. Configuring and integrating multiple simulators, ensuring their proper interoperability, and managing their data exchange demands significant expertise and effort. Existing co-simulation frameworks often lack streamlined deployment processes, resulting in time-consuming and error-prone setups.

RC2 studies how co-simulation platforms can exploit large-scale compute infrastructures to scale the size of the simulation scenario by distributing it horizontally across multiple machines. Ideally, the size of the co-simulation should scale linearly with the addition of new resources, but the inter-communication between simulators may introduce a bottleneck and reduce the benefits.

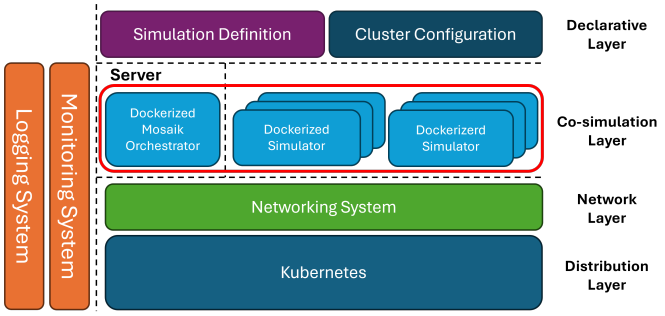

Fig. 1. Architecture of the distributed co-simulation platform

To address the above challenges, the paper builds upon the CO-simulation for Energy Systems Integration (COESI) platform presented in [7]. It introduces novel solutions to streamline the process of deploying and operating complex co-simulation scenarios in cloud environments.

Concerning **RC1**, the paper proposes the adoption of established methodologies in engineering software solutions in the cloud to deploy and operate co-simulations. Specifically, it adopts containerisation technologies to decouple the definition of the simulation scenario from the execution infrastructure, automating and abstracting away distribution and deployment concerns and allowing seamless migration to different computing infrastructures.

Concerning **RC2**, it provides an initial assessment of absolute performance and scalability of realistic co-simulation scenarios within the proposed framework, showing the benefits of horizontal scaling but also the performance bottlenecks introduced by Mosaik, the co-simulation platform we adopt in our platform, which affects the performance profile of large-scale simulations.

The paper is structured as follows. Section II presents our distributed co-simulation platform, comprehensively describing its internal architecture. Section III showcases the scenarios deployed to test the platform's scalability. Section IV presents the performance results we achieved with our platform. Finally, Section V provides concluding remarks and outlines future work.

II. Distributed Co-Simulation Platform

This paper introduces an enhancement of the COESI [7] co-simulation platform that exploits software engineering methodologies and tools developed for cloud environments to enable seamless and effective distribution of the simulation load for horizontal scalability in clusters of machines. The platform builds on three design principles: (i) declarative specification of simulation scenario and deployment environment; (ii) modularity and re-use of simulators; (iii) separation of scenario modelling and simulation deployment/execution concerns.

This translates to the high-level architecture presented in Figure 1.

a) Declarative layer: Users interact with the platform using a declarative interface, as shown on top of Figure 1).

They specify the co-simulation scenario by providing a *simulation definition* in the form of a YAML file that identifies the individual simulators to be adopted, their configuration parameters, and their connections. We envision the availability of a catalogue of simulators that the platform users can compose to build complex scenarios: individual simulators can be publicly available or private to a given company to accommodate a fusion of community and proprietary simulators. The interested reader may find more details on the definition syntax and semantics in the paper describing COESI [7].

Orthogonally, users define the computing infrastructure to be adopted in a *cluster configuration* file. As better discussed in the following, we adopt containerisation technologies, where simulators are packaged into independent units of deployment and execution known as containers. Accordingly, the deployment definition file maps simulator containers onto the physical resources made available by the distributed computing infrastructure. For instance, users may decide to run a certain simulator on a specific node due to its hardware characteristics.

The above approach promotes the decoupling of scenario definition and deployment strategies, enabling porting an existing scenario onto a new computing infrastructure by simply changing the deployment definition.

b) Co-simulation layer: Based on the information provided in the declarative layer, the co-simulation layer instantiates (i) the individual *simulators* that compose the co-simulation scenario, and (ii) the *orchestrator* that governs their interactions at runtime. The platform fully automates the creation, deployment, and execution of the simulators and the orchestrator. This automation streamlines the deployment process, ensuring all components are correctly configured and ready to operate. Both the simulators and the orchestrator are provided as Docker containers[1]. Docker is the containerisation technology used to create custom environments for all the services and simulators running inside a computing infrastructure. This approach ensures that each simulator operates within its own isolated environment, meeting the minimum requirements necessary for correct functionality while providing the flexibility to customize and adapt the system as needed. By leveraging Docker, the platform can maintain consistent and reproducible environments, enhancing reliability and simplifying the deployment process.

Currently, our platform supports a variety of simulators, including Modelica, EnergyPlus, and Matlab, all adhering to the FMI standard. Moreover, it supports custom simulators written in Python. All the simulators have their containerised environment to allow easy integration with the cluster and the services created to monitor them. The architecture offers flexibility and adaptability by supporting a diverse range of simulators and accommodating various simulation scenarios.

The platform relies on the Mosaik orchestrator, which manages the state of all the simulators required to complete the simulation scenario. The platform generates a specific orchestrator equipped with all the required information for each new simulation, ensuring that the simulation environment is accurately configured and ready to run. A common server handles the initial configuration of the simulation. It ensures that all necessary parameters and settings are correctly applied before the simulation begins. Once all the relevant containers are up and running, the server sends a starting command to the orchestrator to initiate the simulation.

c) Network layer: The network layer provides fine-grained control over the communication, enforcing network policies that protect sensitive data and segment traffic as needed and allowing a better sectorization of the system when multiple simulations are running concurrently, ensuring that each simulation remains insulated from potential disruptions or security breaches. This is implemented using the Calico[2] networking and security solution. It allows the implementation of custom security policies by adding NetworkPolicy pods in the deployment definition.

d) Distribution layer: To distribute containers across multiple machines and enable horizontal scaling, the platform builds upon Kubernetes[3], a robust framework designed for managing containerised applications. Kubernetes automates the deployment of application containers across clusters of hosts, providing container-centric infrastructure. It allows us to handle crucial tasks such as scaling, failover and balancing while ensuring that all the components are running in the desired state.

e) Monitoring system: Monitoring is an orthogonal concern that spans multiple layers. We rely on Prometheus[4] as a service that monitors and collects different metrics (CPU, Memory and Network usage) about nodes and containers that are inside the cluster, providing real-time insights into the performance of the simulation infrastructure.

f) Logging system: Another orthogonal concern involves gathering log data from the simulators and analyzing them. We rely on Loki[5] as a service to aggregate log files, enabling the retrieval and analysis of custom metrics, such as the execution time of the simulation or runtime errors that may occur related to different simulators and their states.

III. Scenario

The scalability of our distributed co-simulation platform was evaluated through the simulation of a urban energy scenario. This scenario comprises multiple instances of a detailed building energy model, each representing a distinct structure within a urban environment. The complexity of these models stems from their integration of various interconnected simulators, each designed to accurately depict the physical components of a building's energy system. In particular, each building encompasses several key elements:

- The *Building Energy Model* (BEM) is simulated with EnergyPlus, integrated into the platform via a Functional

[1]https://www.docker.com

[2]https://docs.tigera.io/calico/

[3]https://kubernetes.io

[4]https://prometheus.io

[5]https://grafana.com/oss/loki/

Mock-up Unit (FMU) to evaluate the thermodynamics of the building structure.

- The *Household Behaviour Model* (HBM) simulates occupancy patterns and energy consumption from lighting and appliances based on individual inhabitant actions.
- The *Rooftop PV System* model (PV) provides a high-resolution photovoltaic profile using detailed spatiotemporal data and specific technical information about the building's rooftop solar potential.
- The *Electric Heat Pump* (EHP) and *EHP Controller* models simulate the heating system's operations to maintain desired indoor temperatures. Both models are simulated with Modelica encapsulated in FMUs.
- The *Electrical Energy Storage System* (BT) is developed in MATLAB Simulink using the Simscape Library's generic battery model and encapsulated into an FMU. This model can simulate the dynamic behaviour of various popular rechargeable battery types, allowing for full parameterisation using commercial battery datasheets.
- Additional Python-based standalone simulators, such as the *Weather Module* (WEA) providing weather data for the Rooftop PV System, EHP and Household Behaviour models; a *Scheduler Module* (SCH) providing the schedule of the indoor temperature setpoint per each building; and the *power nodes* (PN) and *smart meters* (SM) that manage and measure the electric and thermal power flows among the simulators.

Finally, an HDF5 database simulator is used to collect the relevant data from each building. For a more in-depth exploration of the individual simulators and their integration within our co-simulation framework, readers are encouraged to refer to our previous work, as detailed in [7].

The integration of these elements allows for a nuanced and realistic representation of each building's energy dynamics within the larger urban context and helps to test the platform's capability to handle complex, multi-building scenarios.

We integrated various simulators into our Kubernetes infrastructure to evaluate the platform's scalability. We encapsulated each simulator in a Docker container and associated it with a pod (the unit of deployment in Kubernetes), enabling fine-grained distribution across multiple physical machines.

We developed an automatic configurations generator to streamline the deployment of simulators and simulations on the infrastructure. This system generates the necessary simulation definitions (i.e. simulators, connections, metrics) and cluster configurations (i.e. pods, nodes, network), which are then applied to Kubernetes. This system manages all connections and pod creation, significantly reducing setup errors. Additionally, it allows users to specify the number of buildings to simulate and select the simulators to be involved. Finally, this architecture enables the creation of diverse test scenarios with a variable number of buildings, each simulated using different quantities of simulators. This flexibility allows for comprehensive testing of the platform's performance and scalability under various conditions.

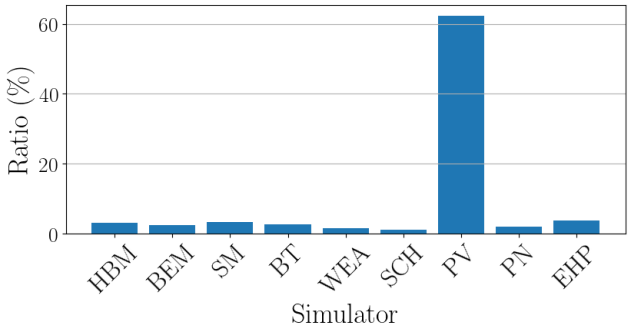

Fig. 2. Execution time of each simulator. Ratio (%) with respect to the total co-simulation time.

IV. Evaluation

Our evaluation aims to assess the absolute performance and scalability of the proposed platform by deploying the scenarios presented in Section III in a cluster environment.

The cluster comprises five machines, one leader node and four worker nodes. The leader node is equipped with a 16-core 2GHz Xeon CPU with 64GiB of memory, and each worker node has a 32-core 2GHz Xeon CPU with 128GiB of memory. Each host lives as a virtual cloud machine provided by OpenStack. The leader node hosts the *Kubernetes Control Plane*, which administers the cluster by scheduling pods (i.e. units of deployment in the Kubernetes jargon) and detecting and responding to cluster events. Each worker node hosts a *Kubernetes Node*, representing the runtime environment where pods are executed. In the following, we present the execution time of various simulations, where we change the scale and complexity of the simulation and the number of workers involved in the deployment. Each experiment simulates one week. We repeat each experiment 4 times, and we plot the average and the standard deviation of the measurements.

The first test we performed studies the scalability of the framework when increasing the number of simulators involved in the co-simulation scenario. To do so, we rely on the scenario presented in Section III, and we increase the number of buildings involved in the simulation. Each building contributes nine simulators. Figure 2 shows the computational requirements of each simulator within a building, computed as the amount of time each simulator was active during the simulation with respect to the total run time of the co-simulation. Figure 2 shows, the nine simulators have vastly different computational requirements, with the PV simulator being the most demanding.

Beside the simulators, the platform needs to instantiate the Mosaik orchestrator, a database to collect the simulation results and the monitoring and logging systems. For this first test, we adopt 4 worker nodes for a total of 128 CPU cores.

Figure 3 shows the execution time we measure while increasing the scale of the simulation by adding more buildings (and hence more simulators). The performances of the simulations are compared to a theoretical model that scales linearly with the number of buildings. We observe that the

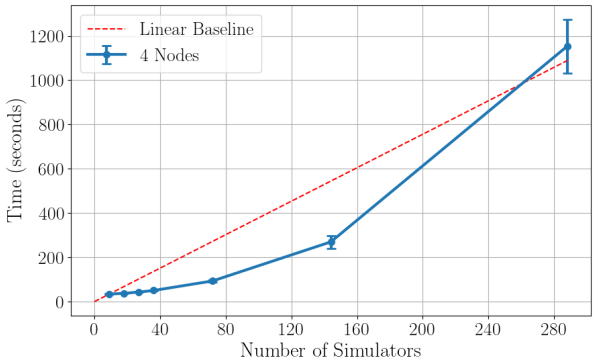

Fig. 3. Execution time with increasing number of simulators.

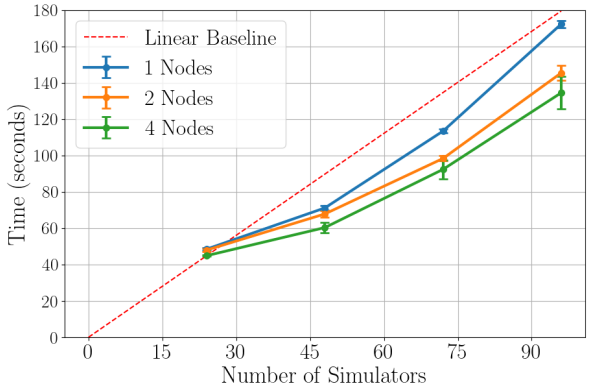

Fig. 4. Execution time when increasing the number of simulators. 16 compute-intensive simulators per building.

platform scales better than the linear baseline until 16 buildings (144 simulators). This indicates that the cluster has enough resources to run the simulators in parallel, which translates into a processing time that grows less than linearly with the number of simulators. Still, the execution time is not constant, as it would be if all simulators were free to run simultaneously. Instead, adding new simulators introduces some overhead, which we suspect is due to orchestration and communication. With 32 buildings (i.e. 288 simulators), the platform has saturated all the resources available in the cluster, so simulators cannot be executed in parallel anymore, and the overall execution time grows more than linearly.

In summary, the first experiment shows that using distributed deployment is indeed extremely beneficial, and execution time grows less than linearly with respect to the number of simulators involved if the computing infrastructure has enough resources to run simulators in parallel. Even in the presence of sufficient resources, the simulation time is not constant with respect to the number of simulators due to overhead that we attribute to orchestration and communication. Finally, the execution time increases more than linearly when the computing infrastructure becomes saturated, and simulators cannot be executed simultaneously.

To gain further insights into the scalability of the platform, we modified the scenario in Section III to increase the computational complexity required to simulate each individual building. We alredy observed that the model of the photovoltaic system (PV) was the most demanding in terms of computation (see again Figure 2). Accordingly, we artificially increased the number of PV systems for each building from 1 to 16. In this new configuration, each building consists of 24 simulators in total.

Figure 4 shows the execution time in this scenario when moving from 1 simulated building (i.e. 24 simulators) to 4 simulated buildings (i.e. 96 simulators). Figure 4 compares the scalability using 1, 2, and 4 worker nodes (32, 64, and 128 CPU cores, respectively). As in the previous test, the execution time grows less than linearly. In this case, the main bottleneck is due to the complexity of the compute-intensive PV simulators, making the relative cost of orchestration and communication smaller and allowing for a sub-linear increase of time even with a single worker node. However, the advantages of using more worker nodes become more and more evident as the simulation scenario becomes more complex. With a single building (i.e. 24 simulators), the performance with 1, 2, and 4 worker nodes is almost identical as the compute capabilities of a single node are sufficient to run all simulators in parallel. With more buildings, the benefits of adding more computing power become more and more evident, resulting in lower execution times when using 2 and 4-worker nodes.

In summary, this second experiment shows that the distributed co-simulation platform can effectively exploit increasing computational resources to scale simulation scenarios when they require significant computing capabilities.

V. Conclusion

This paper focuses on large-scale co-simulation and addresses the challenges that derive from deploying and operating co-simulation platforms in distributed environments. It enhances the COESI [7] platform by exploiting software engineering methodologies and tools developed for cloud environments to enable seamless and effective distribution of the simulation load for horizontal scalability in clusters of machines. We presented the architecture of the distributed co-simulation platform and we performed an initial assessment of its performance using simulation of realistic energy systems.

The adoption of containerisation techniques separates the definition of the simulation scenario from the specification of the deployment strategy, enabling seamless migration of the same simulation to different compute infrastructures. This makes it effortless to move a simulation from a development setup to a local cluster or to a public cloud infrastructure at need.

The results of our performance evaluation let us draw the following conclusions. The platform effectively enables horizontal scalability: when enough computational resources are available, the time required to execute a simulation grows

less than linearly with respect to the number of simulators involved. The platform effectively exploits the availability of more computational resources to run compute-intensive simulators in parallel: heavy simulations reduce their execution time when given more resources. Still, increasing the size of the simulation introduces some performance overhead with respect to ideal scalability, which we attribute to the cost of orchestration and communication.

Encouraged by these promising results, we plan to further investigate the scalability of the platform to shed light on the main causes of potential bottlenecks. A possible research direction is to compare different orchestration frameworks, such as HELICS [10]. We will also study advanced scheduling policies to better distribute the simulation load across machines and to provide more resources to compute-demanding simulators. In addition, we are working on tools to simplify the definition of the simulation scenario, including visual tools and analysis tools to check the compliance with correctness conditions.

ACKNOWLEDGMENTS

This work is supported by the following projects.
COMET - Co-simulation of Multi-energy systems for Energy Transition, Italian National funded project under PNRR M4C2, Investimento 1.4 - Avviso n. 3138 del 16/12/2021 - CN00000013 National Centre for HPC, Big Data and Quantum Computing (HPC).
NEST - "Network 4 Energy Sustainable Transition – NEST", Project code PE0000021, Concession Decree No. 1561 of 11.10.2022 adopted by Ministero dell'Università e della Ricerca (MUR), CUP E13C22001890001. Project funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 341 of 15.03.2022 of MUR; funded by the European Union – NextGenerationEU

REFERENCES

[1] C. Calvillo, A. Sánchez-Miralles, and J. Villar, "Energy management and planning in smart cities," *Renewable and Sustainable Energy Reviews*, vol. 55, pp. 273–287, 2016.

[2] A. Margara, G. Cugola, N. Felicioni, and S. Cilloni, "A model and survey of distributed data-intensive systems," *ACM Comput. Surv.*, vol. 56, no. 1, aug 2023.

[3] A. E. Onile, R. Machlev, E. Petlenkov, Y. Levron, and J. Belikov, "Uses of the digital twins concept for energy services, intelligent recommendation systems, and demand side management: A review," *Energy Reports*, vol. 7, pp. 997–1015, 2021.

[4] European Commission, "Communication from the commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. Powering a climate-neutral economy: An EU Strategy for Energy System Integration, COM/2020/299 final," https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2020:299:FIN, 2020.

[5] P. Palensky, A. A. Van Der Meer, C. D. Lopez, A. Joseph, and K. Pan, "Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, pp. 34–50, 2017.

[6] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, S. Holly, B. Lüers, M. Nebel-Wenner, R. P. Ramírez Acosta, T. Raub, J. S. Schwarz, S. Stark, A. Nieße, and S. Lehnhoff, "Cpes testing with mosaik: Co-simulation planning, execution and analysis," *Applied Sciences*, vol. 9, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/923

[7] D. S. Schiera, L. Barbierato, A. Lanzini, R. Borchiellini, E. Pons, E. Bompard, E. Patti, E. Macii, and L. Bottaccioli, "A distributed multi-model platform to cosimulate multienergy systems in smart buildings," *IEEE Transactions on Industry Applications*, vol. 57, no. 5, pp. 4428–4440, 2021.

[8] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold *et al.*, "The functional mockup interface for tool independent exchange of simulation models," in *Proceedings of the 8th international Modelica conference*, 2011, pp. 105–114.

[9] T. D. Hardy, B. Palmintier, P. L. Top, D. Krishnamurthy, and J. C. Fuller, "Helics: A co-simulation framework for scalable multi-domain modeling and analysis," *IEEE Access*, vol. 12, pp. 24 325–24 347, 2024.

[10] L. Barbierato, P. Rando Mazzarino, M. Montarolo, A. Macii, E. Patti, and L. Bottaccioli, "A comparison study of co-simulation frameworks for multi-energy systems: the scalability problem," *Energy Informatics*, vol. 5, no. 4, p. 53, Dec. 2022. [Online]. Available: https://doi.org/10.1186/s42162-022-00231-6