Basic Concepts of the Energy Lab 2.0 Co-Simulation Platform

Jianlei Liu
Outline

- Introduction
- Overview
- Web User Interface
- Usage of Apache Kafka and Docker
- Role of Adapters
- Simulation of a Wind Turbine as an Example of a Co-Simulation
- Conclusion
- Outlook
Introduction: Role in Energy Lab 2.0

- SenSSiC (Smart Energy System Simulation and Control Center)

[Diagram of various energy sources and systems, including Wind Power, Solar Power, Storage (HP Gas), Thermal Storage (DLR), Electrolysis Facility (FZ Jülich), and Consumer (Buildings).]
Introduction: Role in Energy Lab 2.0

- SenSSiC (Smart Energy System Simulation and Control Center)
Introduction: Role in Energy Lab 2.0

- SenSSiC (Smart Energy System Simulation and Control Center)
Introduction: Role in Energy Lab 2.0

SenSSiC (Smart Energy System Simulation and Control Center)
Introduction: Role in Energy Lab 2.0

- IT Infrastructure of Control, Monitoring and Visualization Center
Introduction: Role in Energy Lab 2.0

Simulation and Analysis Service Infrastructure – Co-Simulation Platform

- is an important application service and planning tool of the IT infrastructure for Energy Lab 2.0 to model intelligent multi-domain energy system solutions
Introduction: Role in Energy Lab 2.0

- Simulation and Analysis Service Infrastructure – Co-Simulation Platform
  - is an important application service and planning tool of the IT infrastructure for Energy Lab 2.0 to model intelligent multi-domain energy system solutions
Introduction of Co-Simulation Platform

- **Aim of Work**
  - configurable co-simulation platform for linking heterogeneous simulators
  - integration with different data sources and real hardware nodes

- **Basic Approaches**
  - generic message communication infrastructure for data and event exchange
  - container virtualization (Docker) and microservices (REST API) as an automated runtime environment
  - web-based user interface
Overview

- Web User Interface
- Asset Manager to save model components
  - Docker instance containing an executable model
  - Metadata
- Co-Simulation Service
- Communication Infrastructure for exchanging data
- Adapters connecting simulators with the Comm. Infrastructure
Web User Interface

- Model Component Editor: upload and integrate model components
- Co-Simulation Editor: create co-simulations
- Operator Interface: control co-simulations
- Visualization: display and analyze results of co-simulations
Usage of Apache Kafka and Docker

- Cluster computing infrastructure as runtime environment for high scalability
- Apache Kafka as scalable message server environment for exchanging data
- Docker as an automated runtime environment to operate and manage simulation nodes
The Role of Adapters

Sending data
- 1. A Matlab matrix object to be sent is converted into a JSON object.
- 2. Using the Matlab function `webwrite()`, the JSON object is sent from Matlab to the REST API via an HTTP call.
- 3. The REST API then sends the JSON object to a specific Kafka topic.

Receiving data
- 4. The Matlab model requests data by using the Matlab function `webread()` to call the REST API of the adapter via an HTTP call.
- 5. The adapter retrieves a corresponding JSON object from the Kafka topic queue.
- 6. The Matlab simulator receives the JSON object which can then be transformed into more appropriate Matlab type(s) (e.g. a matrix object) for further use.
Simulation of a Wind Turbine as an Example of a Co-Simulation

- Two simulators
  - weather data source node that generates or retrieves input data for the wind power turbine
  - Python script to simulate a wind power turbine
- Visualization of simulator output
A Visualization of Output of the Wind Turbine

Weather Data

Kafka Server

Wind Power Plants

![Weather Data Diagram](image)

![Kafka Server Diagram](image)

![Wind Power Plants Diagram](image)

![Wind Turbine Power Output](image)

![Graph of Enercon E126](image)
Conclusion

- a new scalable and very generic system architecture for a co-simulation platform
  - models, controls and analyzes co-simulations
  - provides an easy to use web user interface
  - manages and executes „Software-Stack“ as simulation nodes that can represent simulation models, data sources or software control and monitor interfaces of real hardware nodes
  - implements the communication between different individual simulation nodes and the Co-Simulation Service using Apache Kafka
  - implements the execution of different independent simulation nodes on the computing cluster using container virtualization and microservices
Outlook

- integrate other simulation environments, such as Modelica, Simulink, OpenDSS etc. into the platform
- enhance the user interface allowing users to dynamically adjust the value of parameters in order to obtain different co-simulation results
- implement a universal access for simulator to the microservice-based data management
Thank you for your attention
Microservice Architektur

- Energy Management
- LUPO Bayern
- Dashboard

Gateway

Message Channel

- Search Service
- Link Service
- Master Data Service
- Time Series Service
- Digital Asset Service

NoSQL DB
Graph DB
NoSQL DB
OpenTSDB
HDFS
Integration mit Co-Simulationsplattform

Sim 1
Sim 2
Sim 3

Sim 1
Sim 2
Sim 3

Energy Management
LUPO Bayern
Dashboard
Co-Sim Frontend

Gateway

Search Service
Co-Simulation Service
TS Cache Service
TS Streaming Service

Schema Service
Link Service
Master Data Service
Time Series Service

NoSQL DB
Graph DB
NoSQL DB
OpenTSDB

Digital Asset Service
HDFS

Message Channel

Jianlei Liu – Basic Concepts of the Energy Lab 2.0 Co-Simulation Platform

20.02.2017
Underlying IT Concepts: REST API

The co-simulation platform uses Representational State Transfer (REST) based interfaces as lightweight communication mechanism between different components of the platform. REST is a web-based software architecture style rather than a standard and designed to provide interoperability between computer software applications or programs across a network, such as services and web applications provided on the Internet. REST interfaces are usually based on the use of the Hypertext Transfer Protocol (HTTP) which provides the methods GET, POST, PUT and DELETE to enable communications between clients and servers.
Underlying IT Concepts: Apache Kafka

Apache Kafka is a message oriented distributed data streaming platform which is used via a Java API in Java applications for building real-time data pipelines and data processing applications. Using the consumer group concept, Kafka generalizes the two traditional models of messaging: queuing and publish-subscribe. With the use of a queue the processing of data by the consumer group is distributed to multiple consumer instances (the different simulation members of the consumer group) to scale the processing. By instrumenting publish-subscribe, Kafka can broadcast messages to multiple consumer groups, which can process data in parallel for different purposes.
Underlying IT Concepts: Docker

Docker is the world's leading software containerization platform which can be used to package and isolate applications in containers by using operating system virtualization. Docker uses the separation of resources in the linux kernel, such as cgroups and the linux kernel namespace to create independent software containers. Since Docker containers can contain all necessary software dependencies to execute applications, e.g. operating system and framework libraries and the application binaries, and Docker images containing all these application components and dependencies can be easily transported and installed as files, the deployment of applications could be simplified and easily automated.