bwRSE4HPC

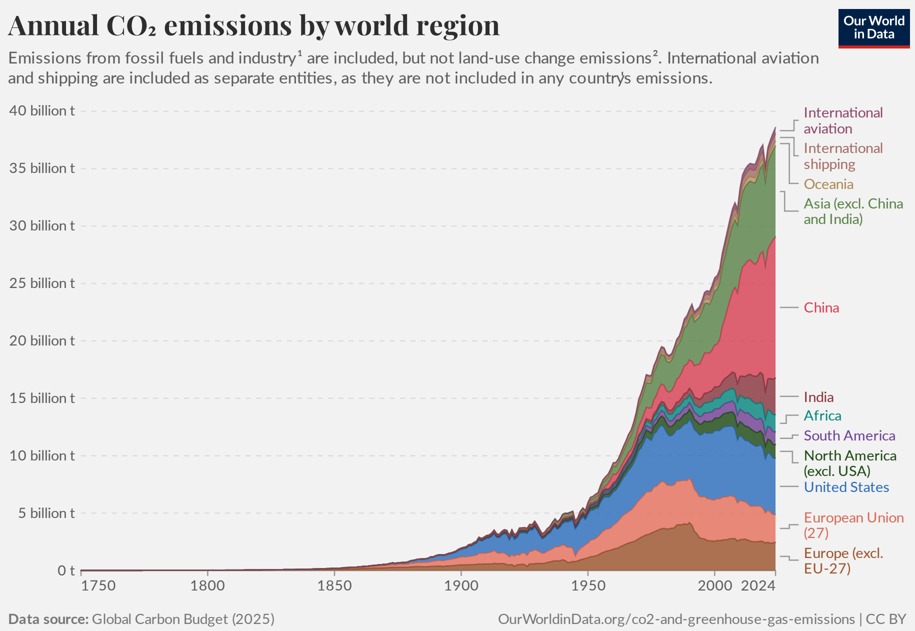
eCoSimHPC: Scaling up sector-coupled energy system analysis with RSE support

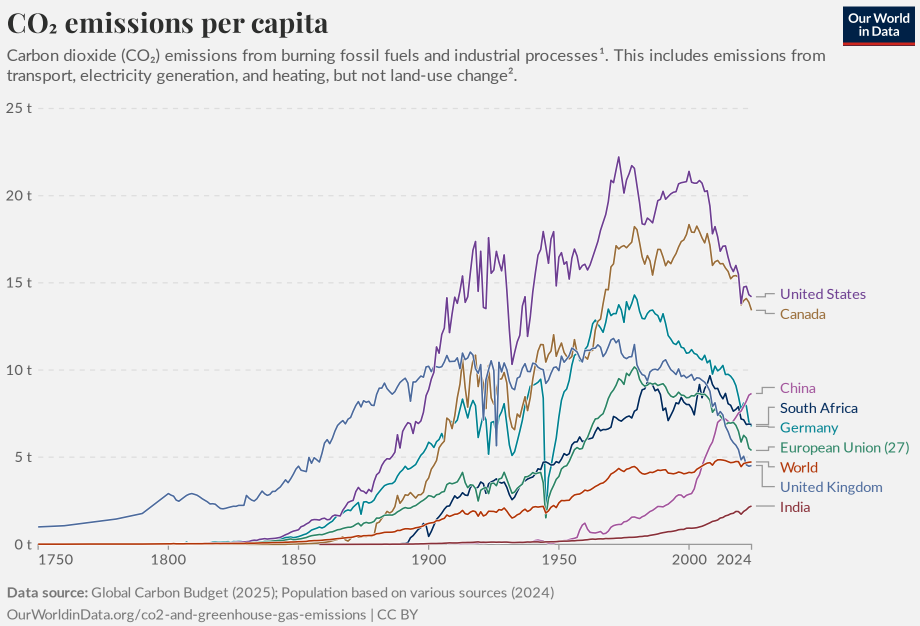
Glen Hunter, Thomas Isensee, Hüseyin Kemâl Çakmak, Alexander Kocher, Maurice Müller, Veit Hagenmeyer

SCC Scientific Computing Center
KIT Karlsruher Institut für Technologie
ESA eASiMOV eCoSim
SCIENTIFIC SOFTWARE CENTER
UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Towards carbon neutrality

Paris Agreement 2015: “hold the increase in the global average temperature to well below 2°C above pre-industrial levels and pursue efforts to limit the temperature increase to 1.5°C above pre-industrial levels.”





Integrating renewable sources

- Power required for a Stuttgart-sized city : ~180MW
- Large wind turbines can produce a few megawatts each.
- Connecting a wind farm to the electrical grid can overload the grid unless proper preparations are made.
- Simulations are required to demonstrate the impact renewables have on the power grid and identify where changes are needed.

Components of Coupled Energy System Simulations

- Structural components
  - Electrical, heat and gas grids
  - Sector coupling: P2X/X2P
  - Buildings (PV, batteries, heat pumps)
  - Transportation: electric vehicles, charging stations
- Variable components
  - Occupancy
  - Time of day
  - Weather


Photo by Brett Sayles

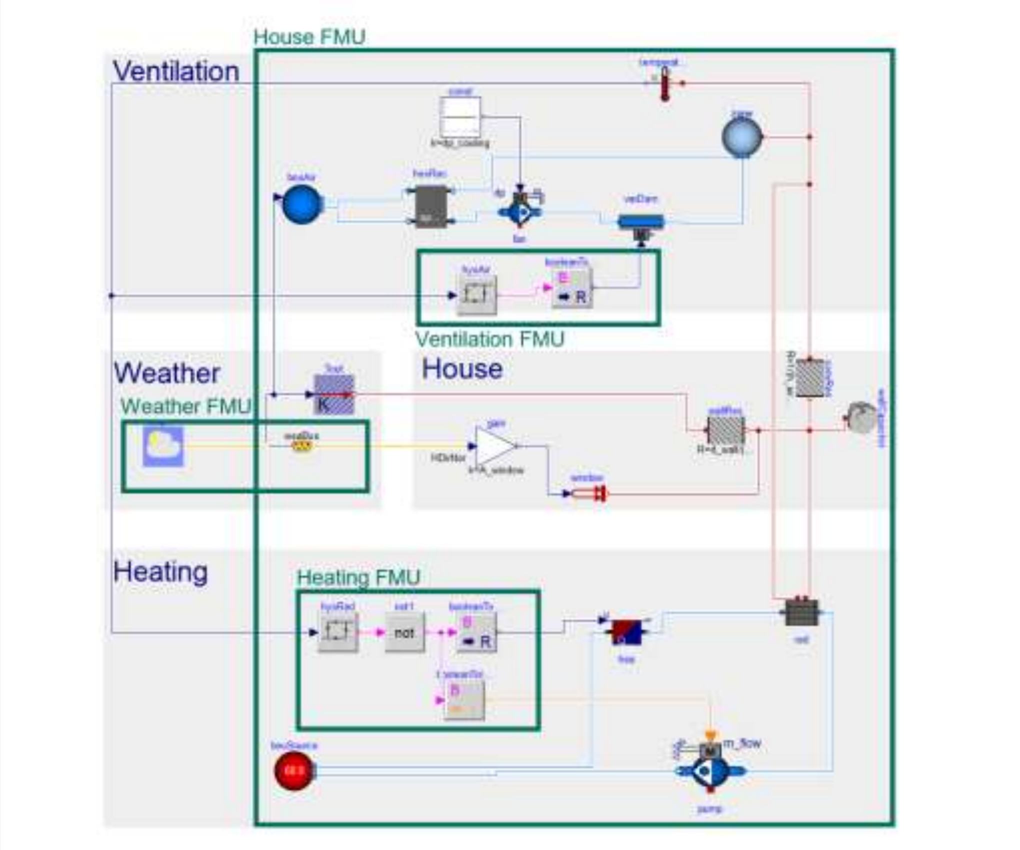

Photo by Pixabay


Photo by Pixabay


Photo by Jonathan Petersson

eCoSim: Energy System Co-simulation

- A co-simulation framework for controlling and executing simulators. (Çakmak et al. 2019).
- Enables data communication between different simulators.
- Integrates into existing data visualisation software – *e*ASiMOV (Kyesswa et al. 2017).

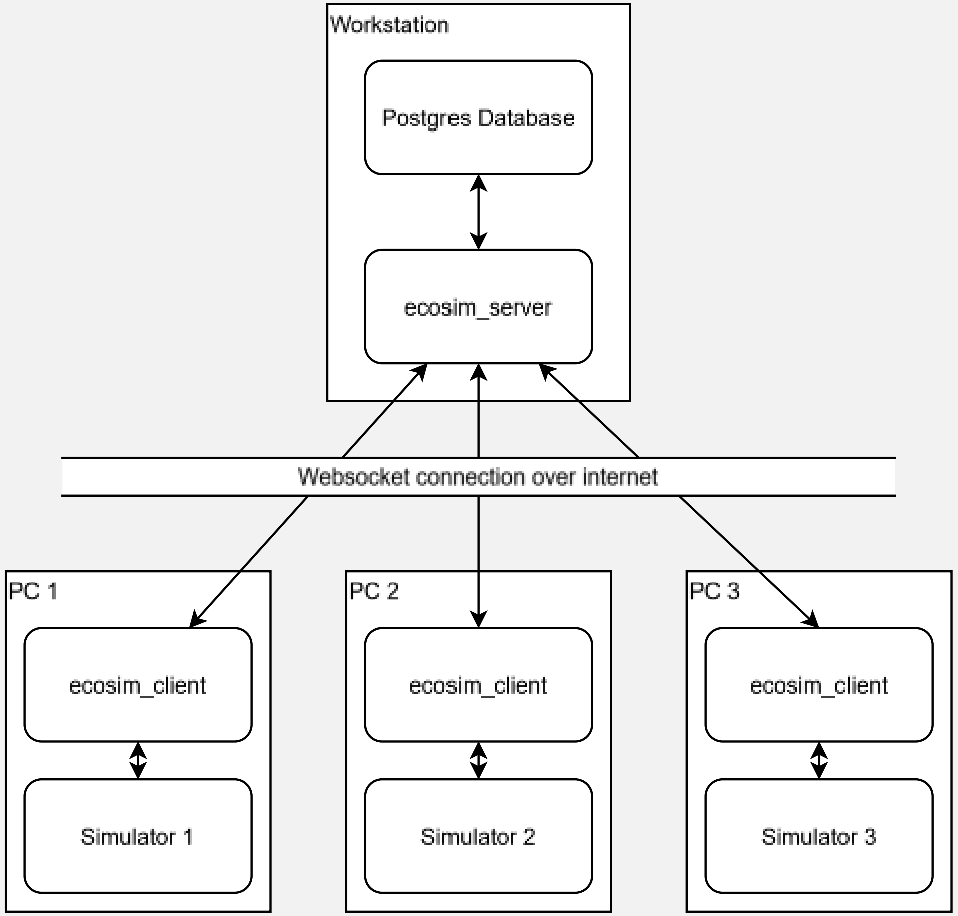


Kocher et al. 2024

Software stack

- Written primarily in Python.
- Data transfer through TCP communication with Python’s native socket package.
- Data storage with PostreSQL and sqlalchemy.
- Simulators can be created in Python, MatLab or compiled into a Functional Mock-up Unit (FMU).

eCoSim design:



eCoSim RSE history

eCoSim: Java based

(2019)

(2024)
Converted to Python

Python simulator compatibility

(2024/25)

(2025/26)

eCoSimHPC:

Dask backend

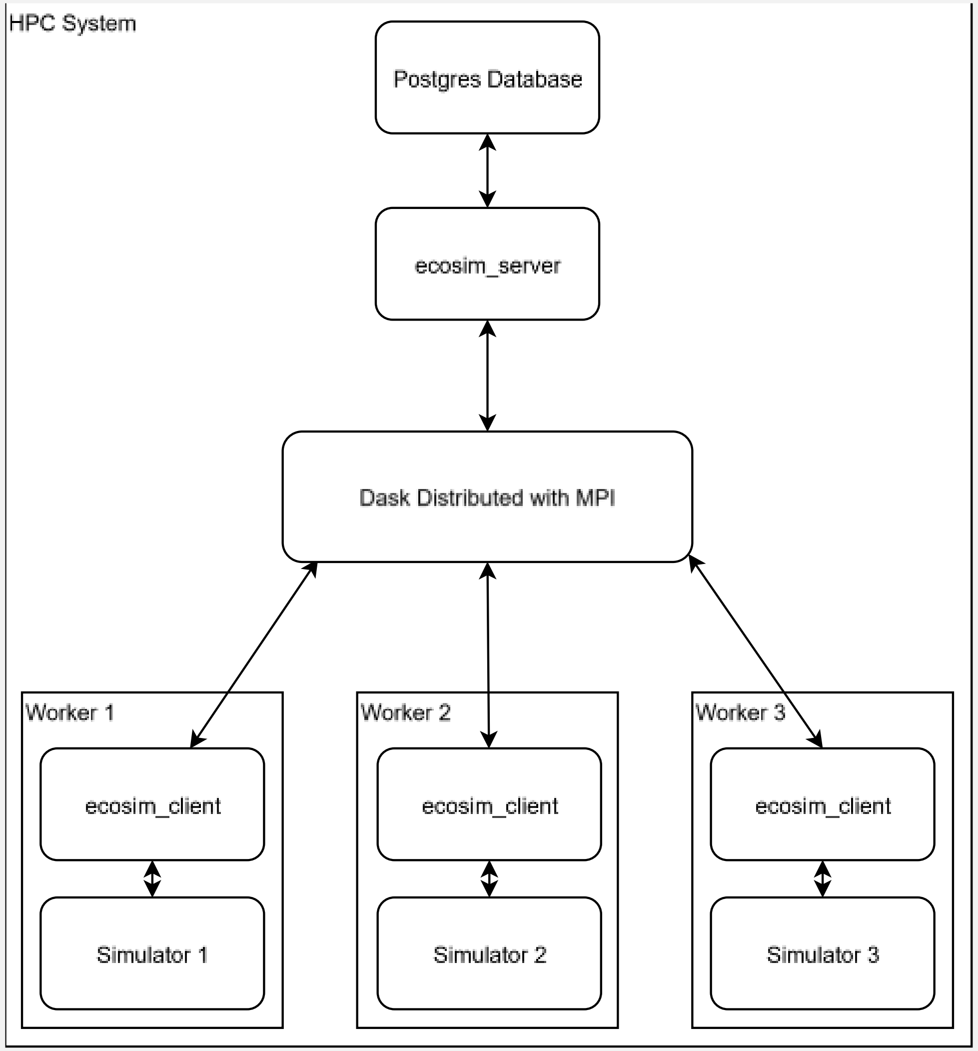
From eCoSim to eCoSimHPC

- In order to simulate a city’s energy grid, > 50000 simulators are needed! -> HPC Systems required.
- Need a way to distribute the simulators across multiple nodes whilst maintaining compatibility with existing software structure.
- Single executable for simpler user experience.

Development

1. Make use of abstraction methods to maintain compatibility.
2. Make use of Dask Distributed and Dask-MPI for distribution and communication.
3. Adapting existing tests to the new Dask-based back end.

eCoSimHPC design



Runtime comparison

Using the IceLake dev nodes on bwUniCluster:

Test	eCoSim runtime (s)	eCoSimHPC runtime (s)
Single Client	10.19	7.46
Three Clients	10.07	2.13
Matrix Transfer	10.39	17.94
Codependent	10.6	25.98
Batch steps (5 test range)	10.03 – 10.05	1.5-1.91

- eCosim contains unneeded sleep statements when running on a single machine

Larger scale testing

- 43 simulators (42 FMUs and 1 Python)

eCoSim:

- System: 12 core workstation
- Runtime for 44638 timeteps: 2hrs 13mins
- Runtime per timestep: 0.17s

eCoSimHPC:

- bwUniCluster3 Compute Node using 96 cores
- Runtime for 100 timesteps: 11mins 20sec.
- Runtime per timestep: 6.8s

=> Processing of messages on ecosim-server currently runs sequentially.

Next steps

- Resolve the concurrent message processing issue.
- Investigate and solve context switching issues (reduce the number of cores needed).
- Test other kinds of larger scale simulations to better gauge scalability and performance, including ones that require more computing resources than one workstation or node.
- Publish eCoSim and eCoSimHPC as open source software.

Summary

- eCoSim is a Python based co-simulation program designed for simulating energy systems.
- With the implementation of Dask, simulators can be distributed whilst maintaining compatibility with existing code.
- RSE support has enabled the code to expand, supporting more simulators and running larger co-simulations.