Co-simulation of FMUs and Distributed Applications with

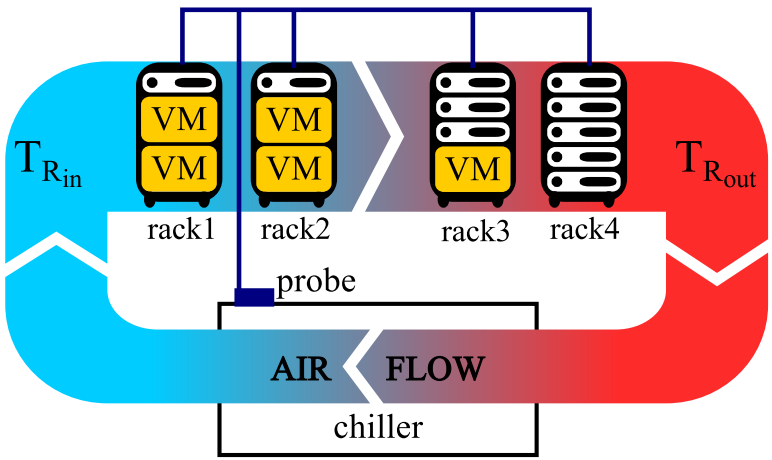
B. Camus, A.-C. Orgerie, Martin Quinson
(IRISA, Rennes, France)

PADS'18
Roma, May 28. 2018.

Simulating Distributed Cyber-Physical Systems

Goal: Co-simulation of physical systems with IT systems

Motivating Example: A Data Center with a chilling facility



- Servers host VMs, which load heats the air
- The chiller cools the air flow, up to a point
- Above a given load threshold, the chiller stops
- Above a given temp., the DC shuts down
- Q: How to migrate the VMs before shutdown?

M&S Challenges

- Two Continuous Systems: heat produced by CPU load, cooling system
- Two Discrete Systems: IT infrastructure (scheduling; emergency response)

Co-simulating (Cyber-) Physical Systems

One Classical Answer: Functional Mock-up Interface (FMI)

- Generic way to exchange models designed with different tools
- Software interface to manipulate equation-based models and their solvers
- Large consortium from the industry: over 100 compliant tools

FMI Model Exchange

- Contains only equations
- User must implement the solver
- Access: state vector, derivatives, I/O

FMI Co-Simulation

- Contains equations and solver
- Comes with a dedicated solver
- Access only I/O ports of the model

In this work, we use FMI Co-Simulation

- FMI CS makes it easier to interface with non FMI tools
- FMI ME: exchange equations between FMI tools (having their solver)

Modeling Distributed IT Systems

DEVS seem unavoidable for such Discrete-Event Systems

- Then leverage existing solutions to co-simulate FMI and DEVS systems

Classical Tradoff: *Domain-specific Expertise vs. M&S Expertise*

- DEVS != prog. languages; Code-based $\rightsquigarrow$ more intuitive to domain specialists
- Formalism-based $\rightsquigarrow$ allow classical M&S methodology, easier to reason about
- Geographically Distributed Systems are particularly challenging to model
- Most IT simulators are code-based: NS3, SimGrid, PeerSim, SST, BigSim

Modeling Distributed IT Systems

DEVS seem unavoidable for such Discrete-Event Systems

- Then leverage existing solutions to co-simulate FMI and DEVS systems

Classical Tradoff: *Domain-specific Expertise vs. M&S Expertise*

- DEVS != prog. languages; Code-based $\rightsquigarrow$ more intuitive to domain specialists
- Formalism-based $\rightsquigarrow$ allow classical M&S methodology, easier to reason about
- Geographically Distributed Systems are particularly challenging to model
- Most IT simulators are code-based: NS3, SimGrid, PeerSim, SST, BigSim

Our proposal: Bridge a code-based IT simulator with FMI-CS models

- We want to reuse validated IT performance models (network, CPU, disk)
- We use SimGrid, which models are (in)validated continuously since 15 years
- SimGrid already bridged with NS3

SimGrid: Versatile Simulator of Distributed Apps

Install a Scientific Instrument on your Laptop

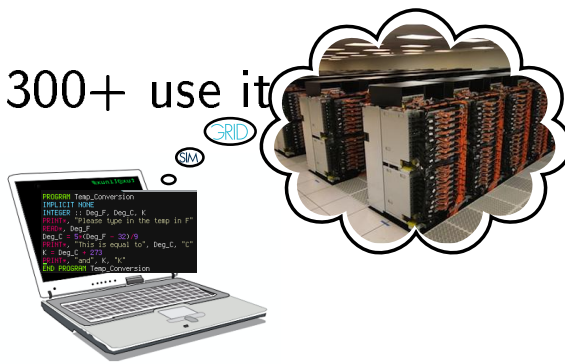
- Joint Project since 1998, mostly from French institutions
- Open Project, contributors in the USA (UHawaii, ISI, NEU), UK, Austria, Cern

Key Strengths

- Performance Models validated with Open Science $\rightsquigarrow$ Predictive Power
- Architectured as an OS $\rightsquigarrow$ Efficiency; Performance & Correction co-evaluation
- Versatility: Advances in Clouds modeling reused by DataGrid users
- Usability: Fast, Reliable, User-oriented APIs, Visualization

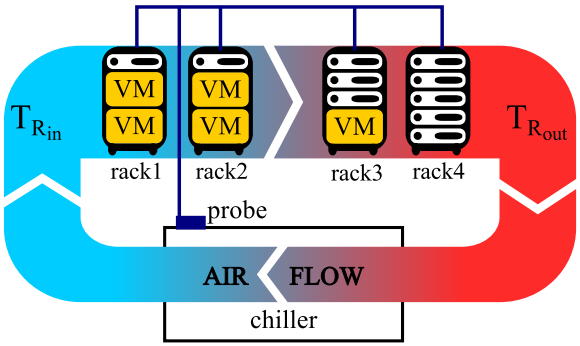
Community

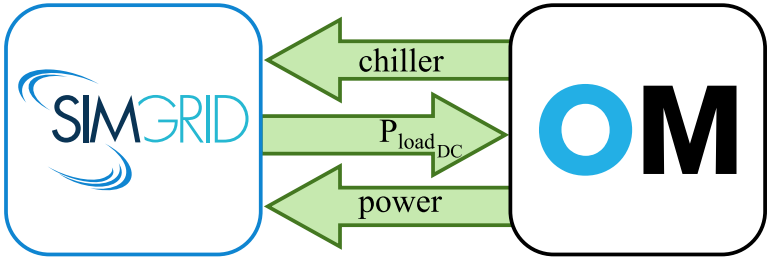
- Scientists: 500+ publications only cite it, 60 extend it, 300+ use it
- Co-development of Application and its Model
- Some industrial users, several pedagogical resources
- Open Source: several unaffiliated contributors



Back to our example

Co-simulation of 2 equational models and 2 concurrent processes





- Pure OM model: chiller failure model only depends on temperature
- Pure SimGrid process: Scheduler puts VMs on servers
- Mixed models: Server load (SimGrid → OM) and Temp Prob (OM → SimGrid)

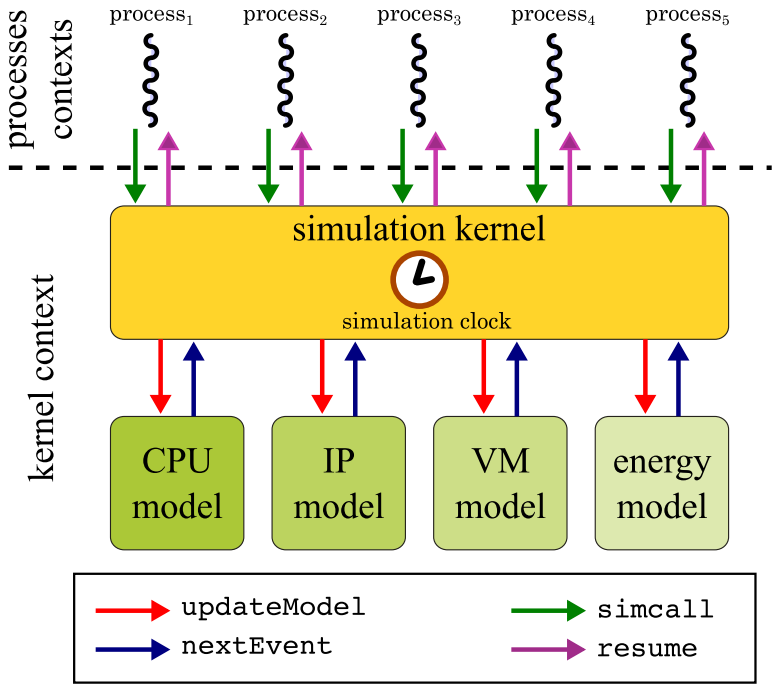
Challenges

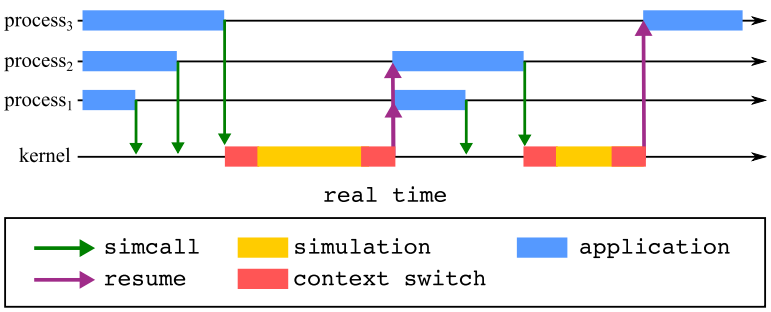
- Co-evolution of processes and physical system models (passive/active loops)
- Manage interactions computing processes ↔ physical models (get/set)
- Detect state changes in the continuous systems that trigger discrete events

Understanding SimGrid Internals

Architectured as an OS

- Processes are fully isolated (CSP) similar to threads exchanging messages
- Interactions mediated through the kernel strictly enforced for McSimGrid
- Highly scalable, up to 2M processes



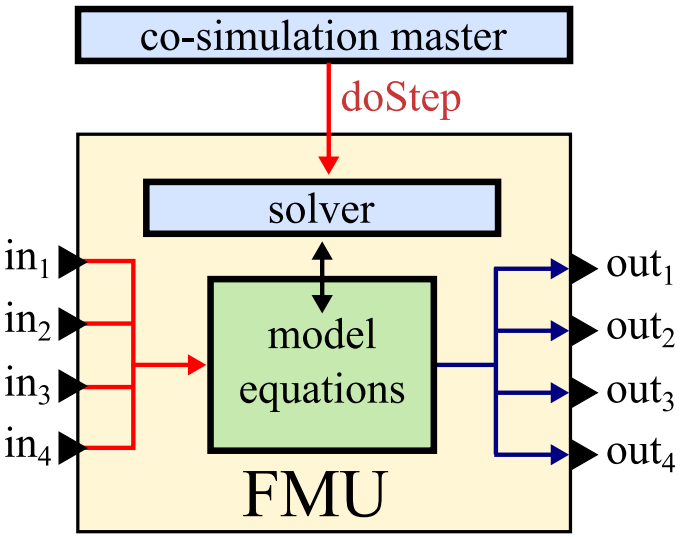


Execution Model: Scheduling Rounds

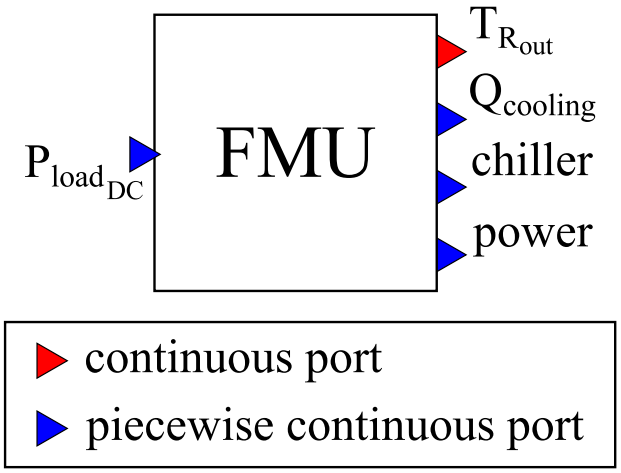
- Run all ready-to-run processes (in parallel)
- Change users' requests in resource consumptions
- nextEvent = min{nextEvent of each model}
- Update all models up to that time
- Some processes become ready-to-run again

Understanding FMI-CS

Component Architecture



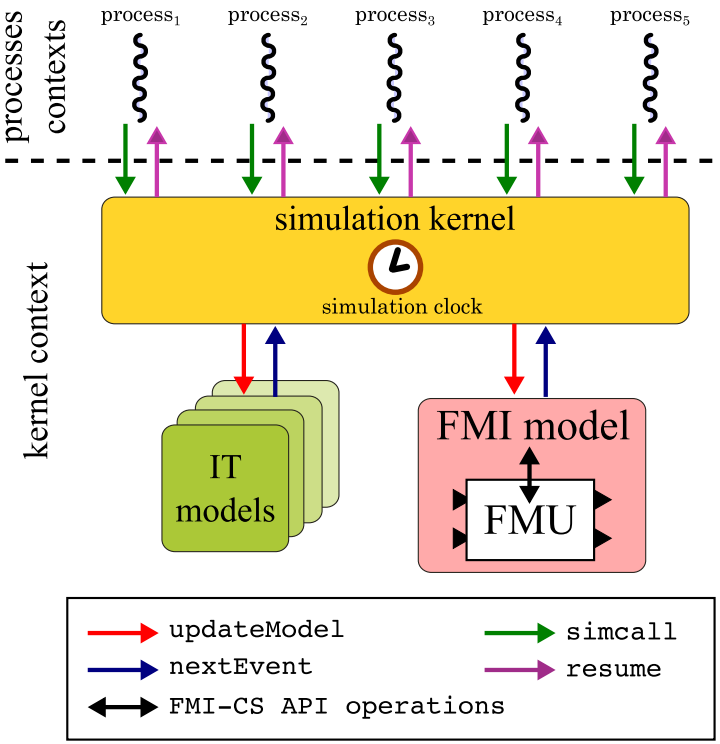
Chiller's Failure Model



Execution Model

- doStep(): proceed for a given period (until next *communication point)*
  No way in FMI 2.0 to proceed until next internal event :(
- FMU exposes input and output *ports*. get/set at comm points only
- FMU can also be fully checkpointed/restaured

Contributions



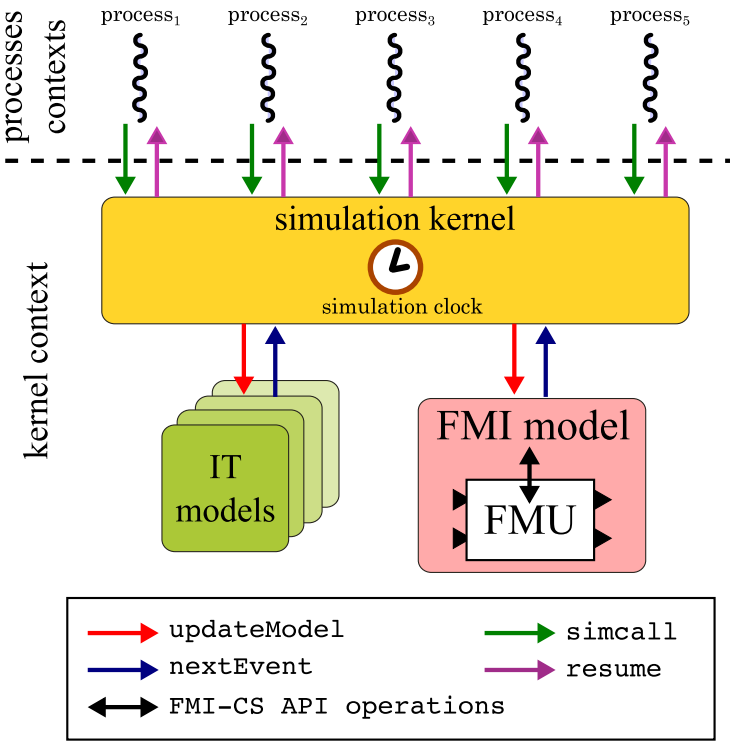
SimGrid Model that hosts FMUs

- Integrates into SimGrid active loop
- API for processes to get/set the ports' values
- Control when to recompute downstream models
- Implemented as a SimGrid plugin, with FMI++

In our example

- Scheduling agent changes *Temp*, input of Chiller failure, after each change

Contributions



SimGrid Model that hosts FMUs

- Integrates into SimGrid active loop
- API for processes to get/set the ports' values
- Control when to recompute downstream models
- Implemented as a SimGrid plugin, with FMI++
- How to detect discrete changes with FMI 2.0?

In our example

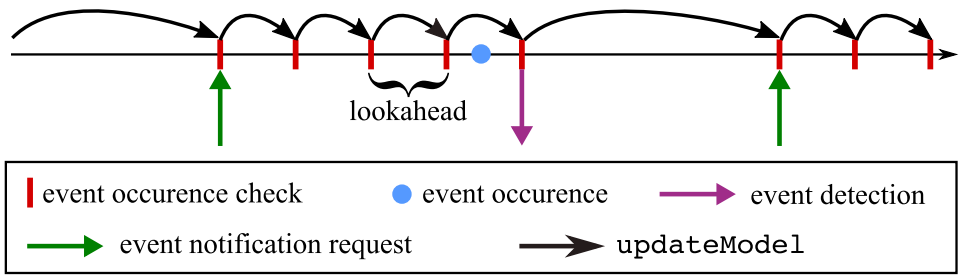
- Scheduling agent changes *Temp*, input of Chiller failure, after each change

Discrete Events Interactions in SimGrid-FMI

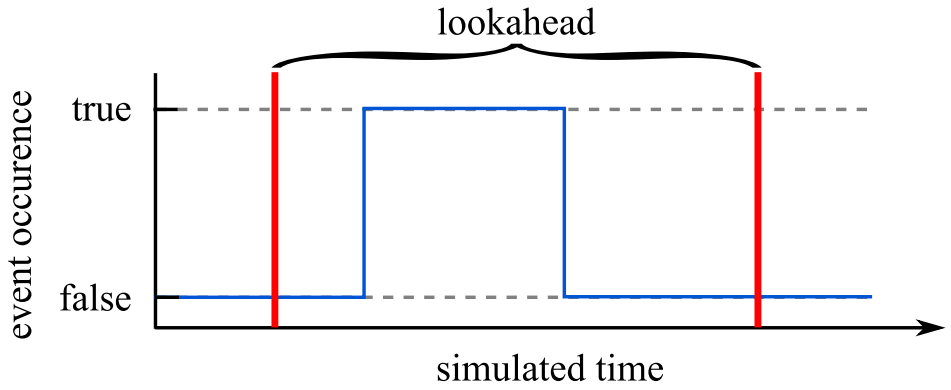
- Provide a generic API to step-wise search, that processes can use at will

```
registerEvent(bool (*condition)(),
              void (*callback)(vector<string>),
              vector<std::string> parameters)
```

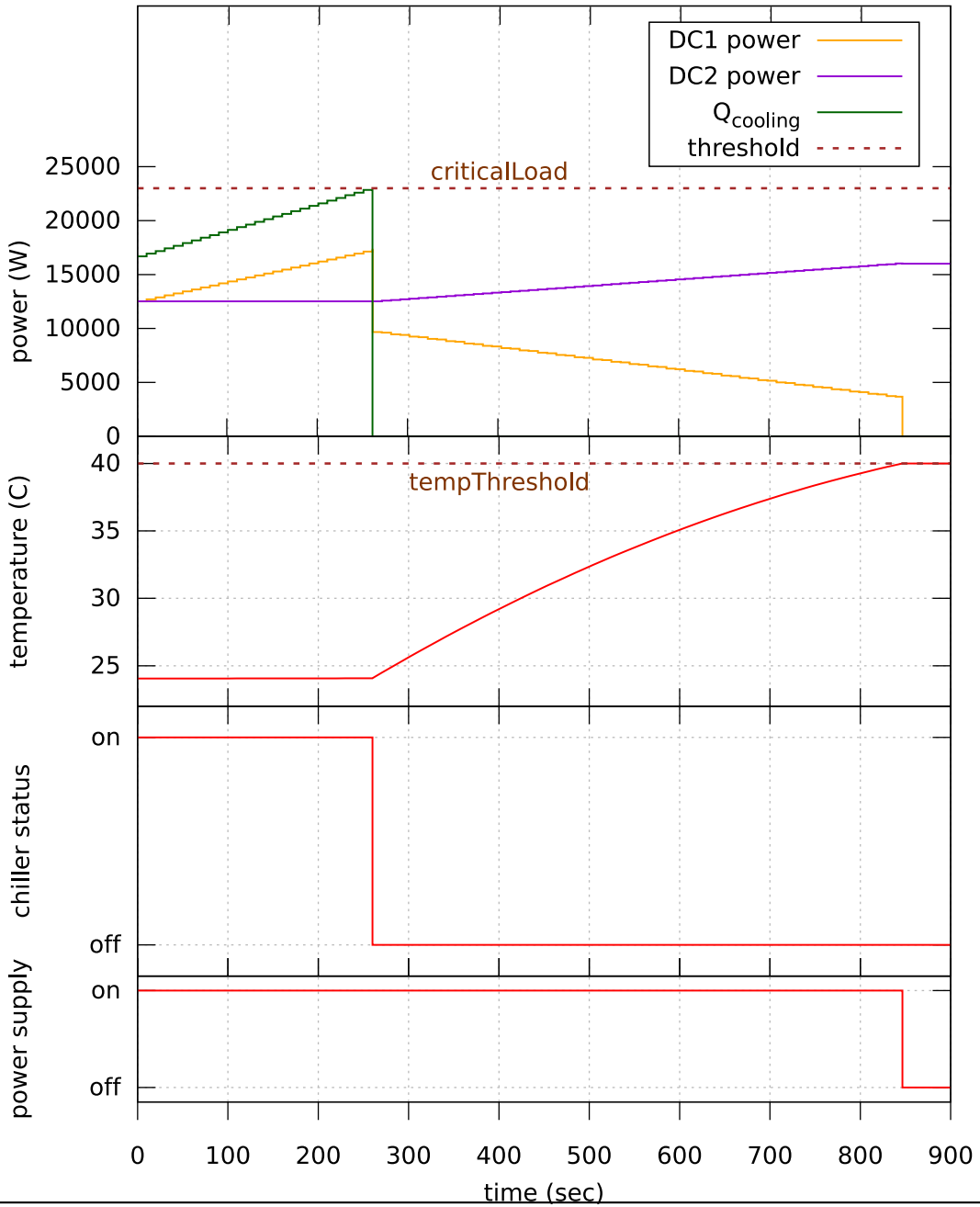
- Lookahead strategy to detect events (only when at least one is registered)



- Still possible to miss short state changes (but nothing better until FMI 2.1)



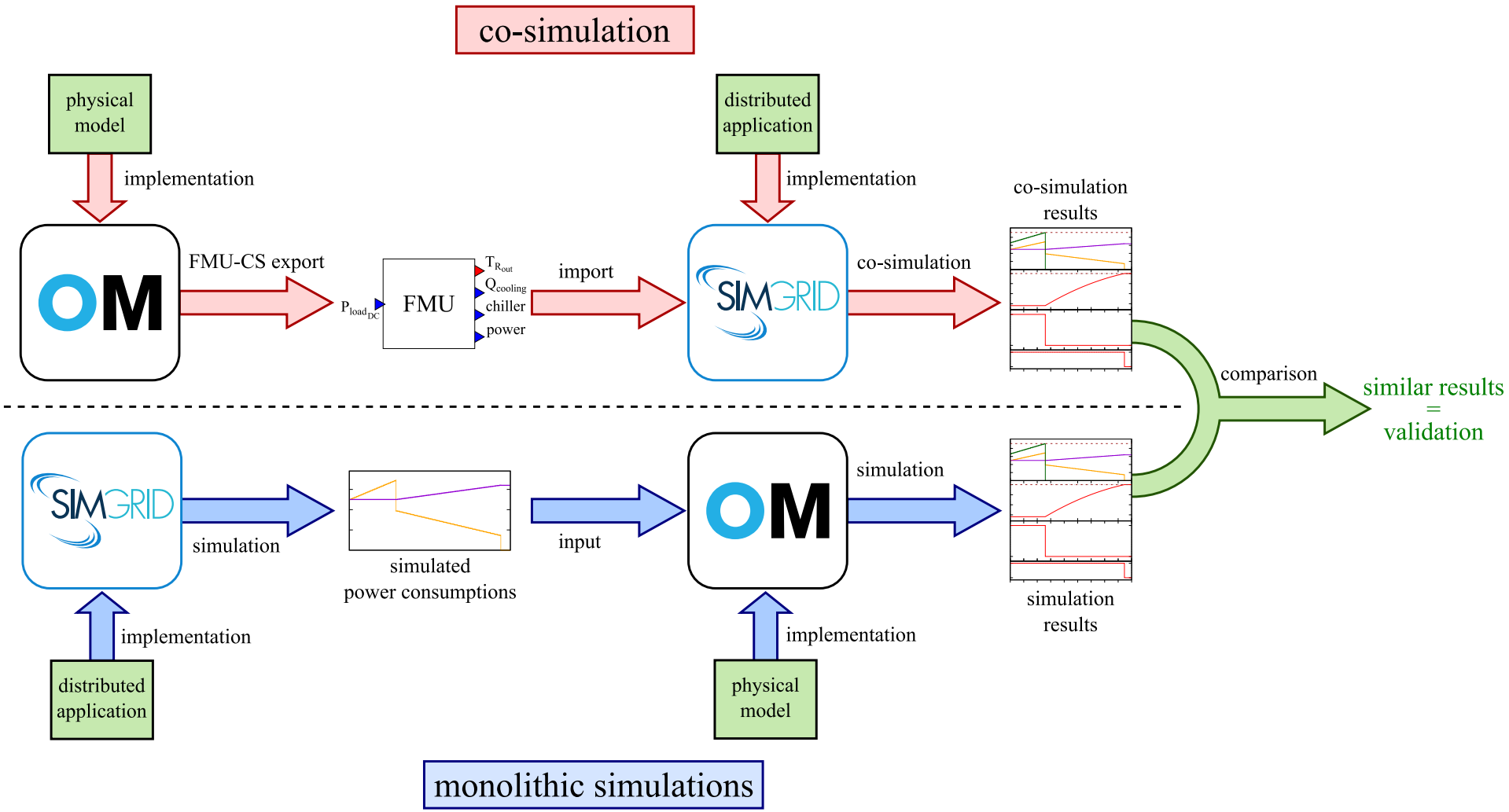
Evaluation: Result interpretation



Conform to expectations

- Chiller load increases with Server load until threshold
- After chiller failure, DC1's temp increases slowly
- But so does the load on DC2: migration in progress!
- When temp=thres, shutdown

Validation against Monolithic Simulations



Conclusion

SimGrid-FMI: Bridging concurrent IT processes with CPS

- Co-simulation of code-based models with equation-based models
- Developed as a SimGrid plugin, using FMI++ library
- Best of Both Worlds: Versatile (FMI ecosystem), Reproducible, Scalable.
- Available on GitHub, along with SimGrid

Future Work

- Don't rely on lookahead to detect events (FMI 2.1 or FMI Model Exchange)
- Better support of multi-FMU, in particular multi-scale or multi-paradigm
- Use it: Green IT, SmartGrid distributed control, advanced IoT studies, ...