Smart Grid Simulation

3rd Colloquium of the Munich School of Engineering:
„Research Towards Innovative Energy Systems and Materials“
Garching, 04.07.2013

Christoph Doblander
Joint work with: Christoph Goebel, Hans-Arno Jacobsen

Department of Computer Science,
Chair for Application and Middleware Systems (I13)
Smart grid, balancing both sides
Smart grid vision

Smart grid applications
- Power system operators’ apps
- Aggregator A’s apps
- Aggregator B’s apps
- End users’ apps

Smart grid middleware
- Information Aggregators
- Distributed computation
- Overlay network

Communication network
- Wide area network
- Local network access points

Power grid
- Distributed data storage
- Distributed computation

Continuous co-simulation of generation, communication, shiftable demand, market conditions, optimization
Imagine the case ...

• The mobile phone network of a specific operator fails. **Does uncontrolled battery charging overload the grid?**

• High latencies in the GSM network occur. **How far from the optimal is the distributed optimization performing?**

• Two aggregators with competing optimization algorithms control car charging. **Could this scenario lead to an overheated transformer?**
Motivation for smart grid simulation

• Many smart grid stakeholders
  – Aggregators with competing objectives can interfere
  – Fluctuating energy sources and controllable demand
  – Increased volatility on energy markets

• Many communication strategies
  – Low latency e.g., responding to frequency changes
  – High latency e.g., GSM network

• Fault scenario
  – N-1, N-2, N-3 …..
  – Grid infrastructure, communication infrastructure
  – Byzantine behavior

→ Explore scenarios through simulation
Smart grid simulation requirements

- Discrete time
  - Fluctuating demand/generation
  - Powerflow analysis

- Agent modeling
  - Markets and control
  - Distributed optimization
State of the art

- Research and Open Source
  - GridLab-D
  - Issues
    - Hard to extend
    - Many hard-coded U.S. grid assumptions

- Commercial
  - Digsilent PowerFactory
  - Issues
    - Most components black boxes
    - Focused on current operator needs
    - Integration
Recent approach: Mosaik

• Schütte et al. „Mosaik – Smart Grid Simulation API“
  – Python, Matlab
  – Reference data model for smart grid simulation
  – Integration of COTS, e.g.: Digsilent PowerFactory

• Issues
  – Step size must be set initially
  – Discrete time: communication between models could be realised by setting and getting properties after step
  – Accuracy $\Leftrightarrow$ step size $\Leftrightarrow$ performance
Recent approach: FMI

• FMI „Functional Mock-up Interface 2.0“
  – Targets interoperability between modeling tools (Modelica)
  – Development initiated by Daimler
  – Usage for micro grid simulation has been shown

• Issues
  – API definition very low level, C/C++
  – Not a good fit for large scale distributed simulation
    • Generated code grows with number of model instances
  – Accuracy $\Leftrightarrow$ step size $\Leftrightarrow$ performance
Proposed approach

- Parallelism by optimistic simulation of next steps
- In case of communication, rollback and breakup into smaller intervals
- No tradeoff between accuracy and simulation step size

**Diagram**

- Parallel simulation of discrete step with communication
- Rollback and breakup in multiple steps
- Simulate next step

**Graphs**

- Energy Usage: Without rollback
- Heat Pump: Accurate response
- Aggregator: Step without communication
Proposed interface

- Simulation and receiving messages can be composed
- State can be recomputed by replaying communication events and steps

```scala
case class NextTimeslot(from: DateTime, to: DateTime)

trait GridAgent[TState] {
  def simulate(timeslot: NextTimeslot, beginState: TState) : (GridUtilization, TState)
}

trait CommunicatingAgent[TState, TMsg] {
  def receiveMessage(beginState: TState, message: TMsg) : TState
}
```
Our approach compared to Mosaik and FMI

• Advantages
  – Accurate representation of communication
  – Speedups through lookups and larger discrete time steps
  – Interactive simulation

• Disadvantages
  – Needs more memory than mutable approaches
  – Implementation has to be side effect free
    • e.g.: Random generators?
  – More coordination, latencies over network

But we can work around 😊
16GB ~ 140 €
Batch persistence
Testing, DSL
Random seed part of the state
Clustering chatty models, prioritisation
Proposed architecture

• Distribution across multiple machine by using actors

• Models should run on the JVM
  – Expressed in any JVM language (JRuby, Clojure, JPython, …)
  – Scala DSL
  – „Bridging the Communication Gap“ with Modim (Modelica → Java)
    • Dymola, Simulation X, Wolfram SystemModeler ...

• Cluster deployment, resource management with Apache YARN

• Respect software engineering principles
  – Modularity, reusability, tests and documentation
Thank you!

• Questions?

• References
  – Schütte et. al. Mosaik – Smart Grid Simulation API, SmartGreens 2012
  – Akka, http://akka.io/
  – Höger – Modelica on the Java Virtual Machine, EOOLT 2013