Profiling of Modelica Real-Time Models

Christian Schulze[1] Michaela Huhn[1] Martin Schüler[2]

[1]Technische Universität Clausthal, Institut für Informatik, Julius-Albert-Str. 4,
38678 Clausthal-Zellerfeld, Deutschland
{Christian.Schulze | Michaela.Huhn}@tu-clausthal.de

[2]TLK-Thermo GmbH, Hans-Sommer-Str. 5, 38106 Braunschweig, Deutschland
M.Schueler@tlk-thermo.de

Overview

1. Introduction
2. Profiling of Modelica Models
3. Implementation
4. Case Study
5. Conclusion

Introduction

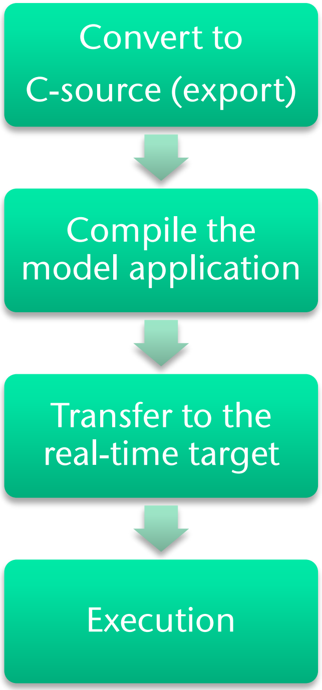
- Usages of RT simulation:
  - Rapid Control Prototyping (RCP)
  - Hardware-in-the-Loop (HiL)
  - Model Predictive Control (MPC)
- Hard real-time (execution time per global solver step <= output step size)
- Overruns due to events and algebraic loops
- Model has to be improved manually
- Profiling helps to trace back the cause of an overrun
- There is no applicable profiling tool
- We want to profile on the RT-Target:
  - Calls to external functions (libraries)
  - Algebraic loops
- We want minimize the overhead caused by profiling

Overview

1. Introduction
2. Profiling of Modelica Models
3. Implementation
4. Case Study
5. Conclusion

Simulation on Real-Time Targets

- To execute a model on a Real-Time target one has to:


Convert to C-source (export)
Compile the model application
Transfer to the real-time target
Execution

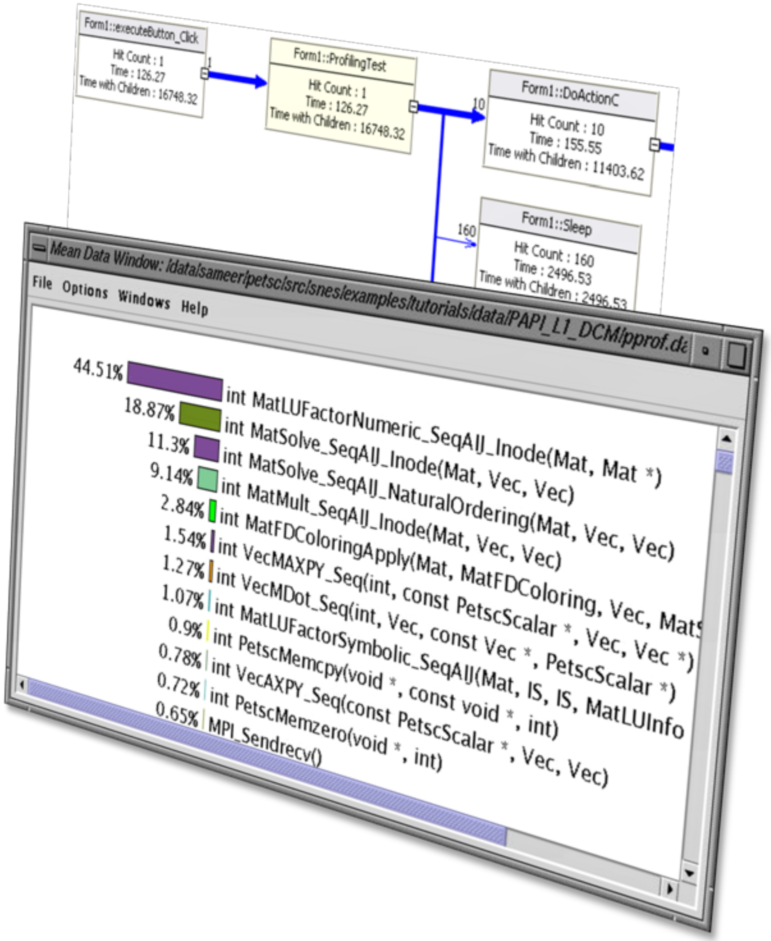
Profiling

- Tracing vs. Profiling
  - Call-Graph
  - Flat-Profile

- Instrumentation must be added but causes additional work load

- We are profiling by measurement (just like Tau) but are not logging the callee or point in time

- One can draw this information from the general structure of those models


Mean Data Window: /data/sameer/petsc/src/snes/examples/tutorials/data/PAPI_L1_DCM/pprof.da
File
Options
Windows
Help
1
10
160

```
Form1::executeButton_Click
Hit Count : 1
Time : 126.27
Time with Children : 16748.32

Form1::ProfilingTest
Hit Count : 1
Time : 126.27
Time with Children : 16748.32

Form1::DoActionC
Hit Count : 10
Time : 155.55
Time with Children : 11403.62

Form1::Sleep
Hit Count : 160
Time : 2496.53
Time with Children : 2496.53
```

```
44.51% int MatLUFactorNumeric_SeqAIJ_Inode(Mat, Mat *)
18.87% int MatSolve_SeqAIJ_Inode(Mat, Vec, Vec)
 11.3% int MatSolve_SeqAIJ_NaturalOrdering(Mat, Vec, Vec)
 9.14% int MatMult_SeqAIJ_Inode(Mat, Vec, Vec)
 2.84% int MatFDColoringApply(Mat, MatFDColoring, Vec, Mat
 1.54% int VecMAXPY_Seq(int, const PetscScalar *, Vec, Vec *)
 1.27% int VecMDot_Seq(int, Vec, const Vec *, PetscScalar *)
 1.07% int MatLUFactorSymbolic_SeqAIJ(Mat, IS, IS, MatLUInfo
  0.9% int PetscMemcpy(void *, const void *, int)
 0.78% int VecAXPY_Seq(const PetscScalar *, Vec, Vec)
 0.72% int PetscMemzero(void *, int)
 0.65% MPI_Sendrecv()
```

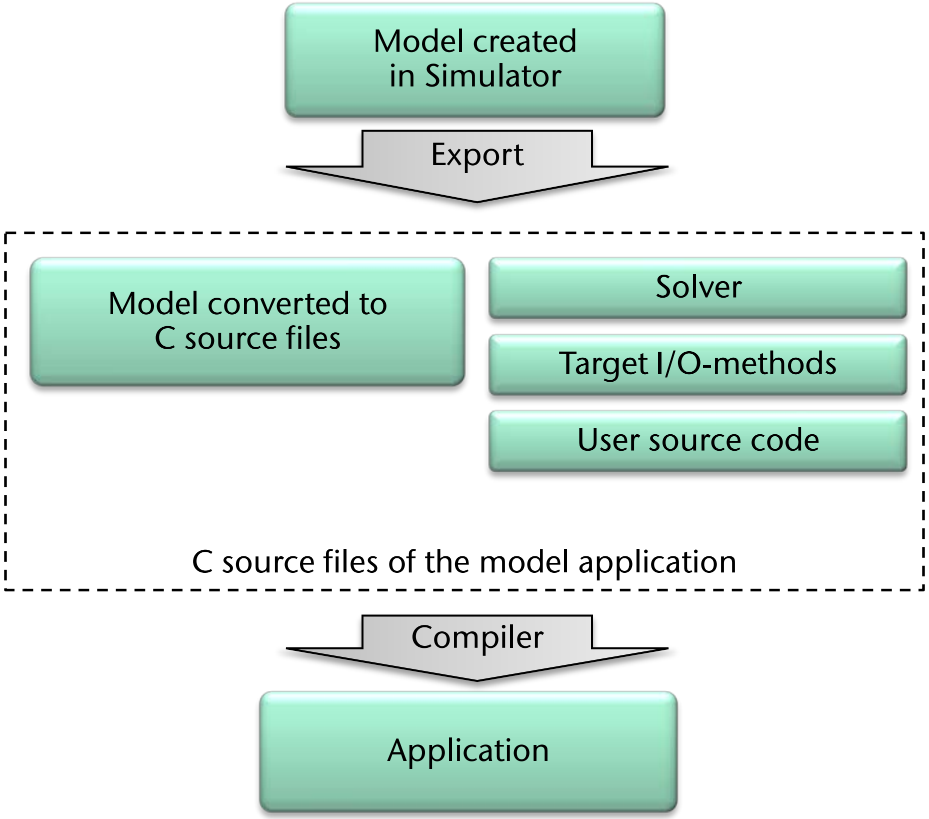
Source code of Modelica models

- Separated into Initialization and Simulation
- Solver step:
  - n_I integration steps
    - e_I external function calls
    - c_I additional calculations
    - a_I (non-)linear blocks
      - e_{aI} external function calls
      - c_{aI} additional calculations
  - 1 output of variables
    - e_O external function calls
    - a_O additional calculations
- flat profiling for each section
- profiling is done separately for each solver step

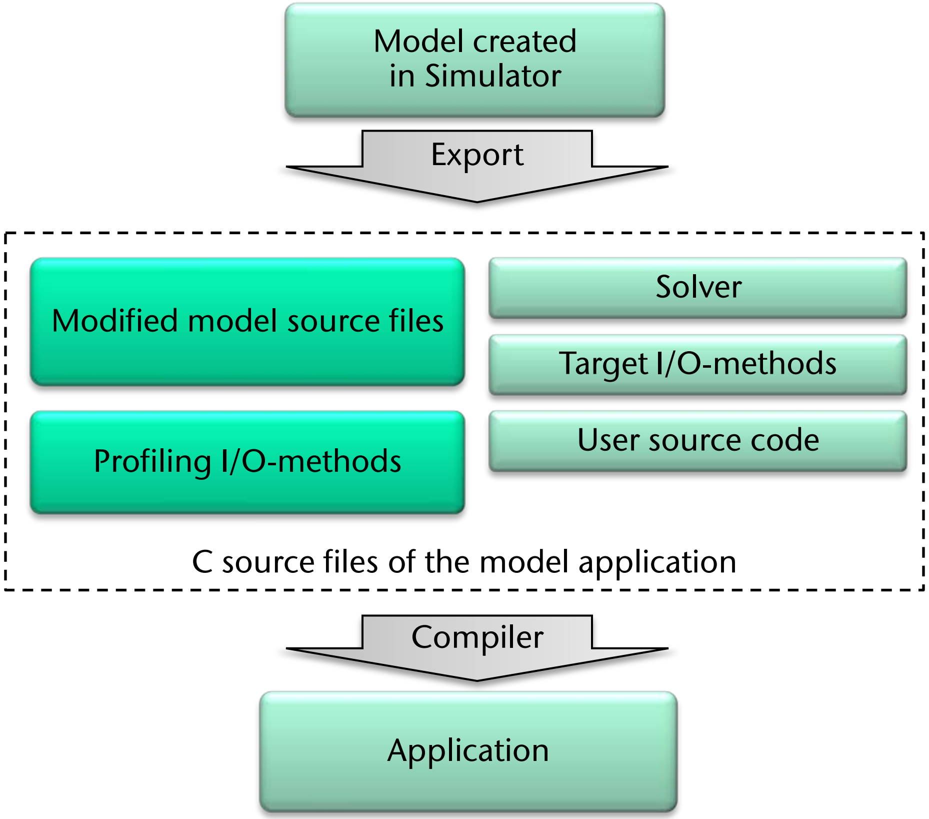
Overview

1. Introduction
2. Profiling of Modelica Models
3. Implementation
4. Case Study
5. Conclusion

Export of models


Model created in Simulator
Export
Model converted to C source files
Solver
Target I/O-methods
User source code
C source files of the model application
Compiler
Application

Modified Export of models



Modified Source Code

- Major impact on the overhead:
  - Saving profiling data (access to HDD)

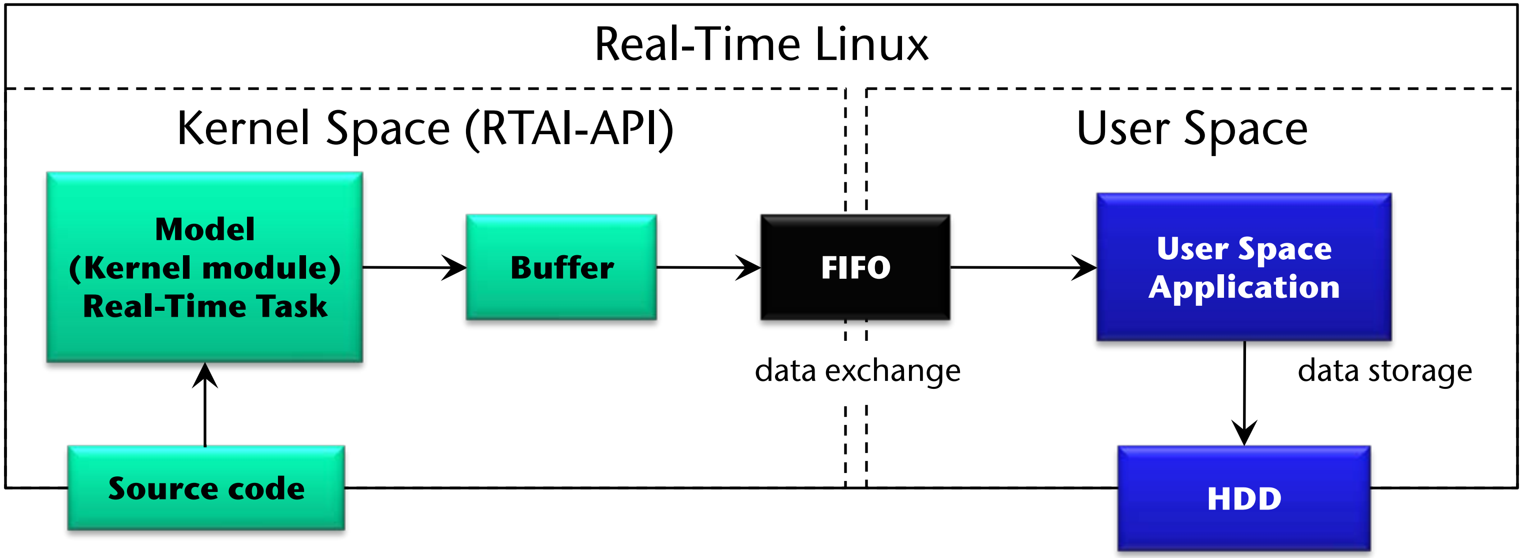
- Access to HDD is outsourced to secondary application

- Model application is a Real-Time Kernel Task(SCALE-RT)
  - Prioritized execution

- Model runtime increases at maximum by 4%

Implementation

- Communication between user task and model task

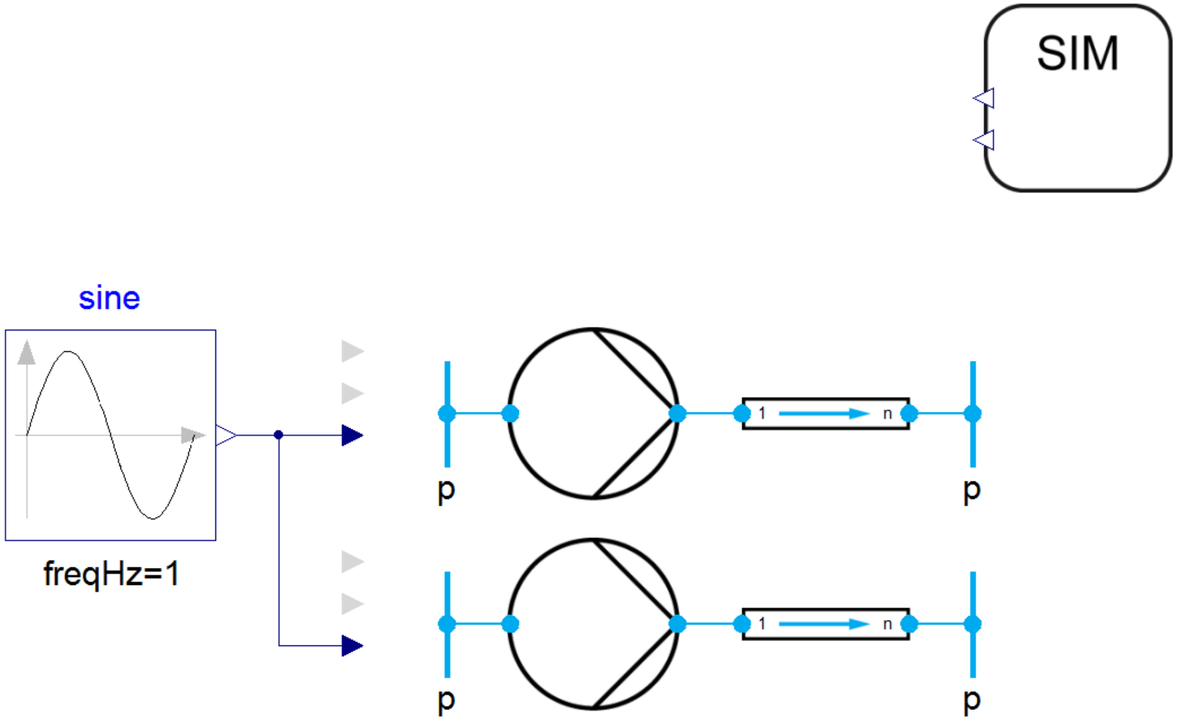


Overview

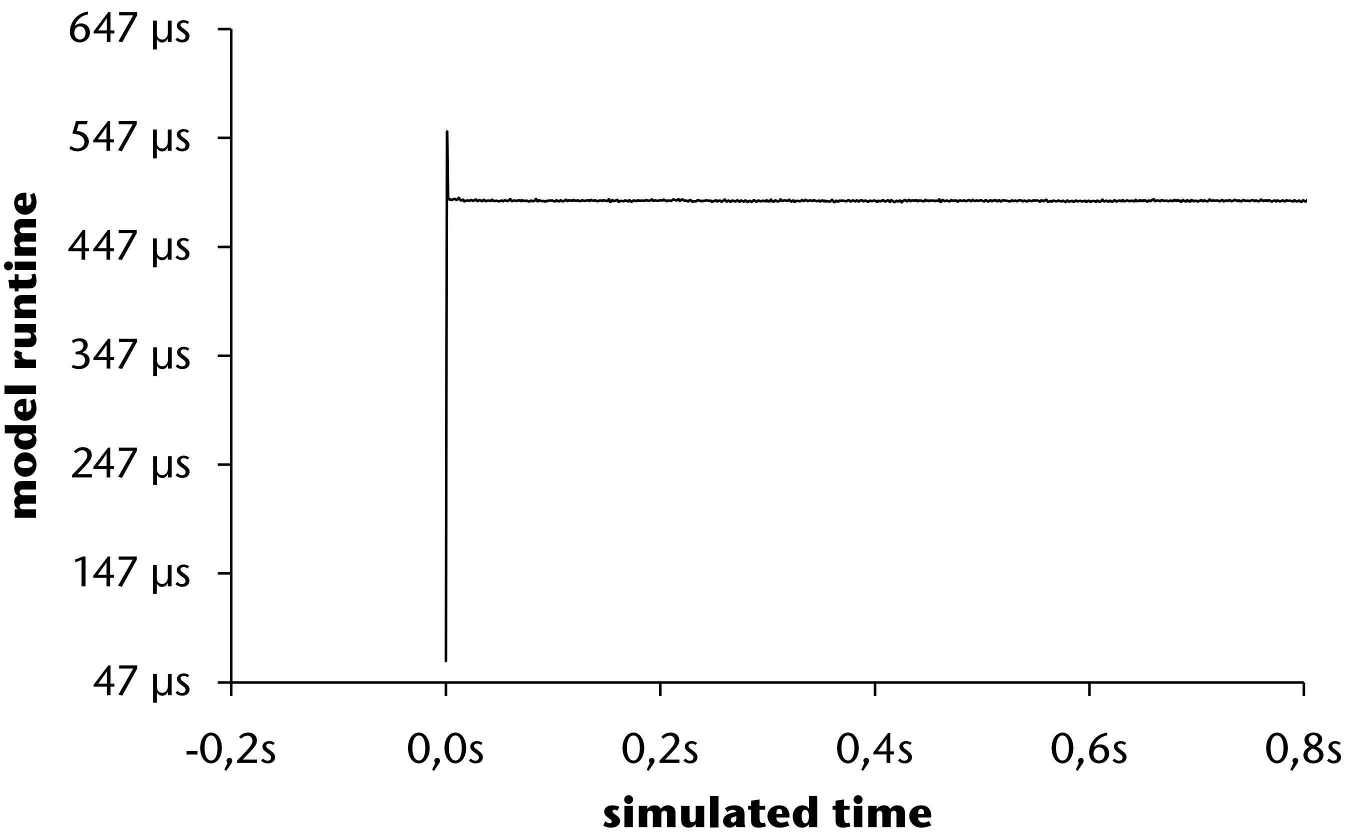
1. Introduction
2. Profiling of Modelica Models
3. Implementation
4. Case Study
5. Conclusion

Case Study

- Steady State Continuity Equation

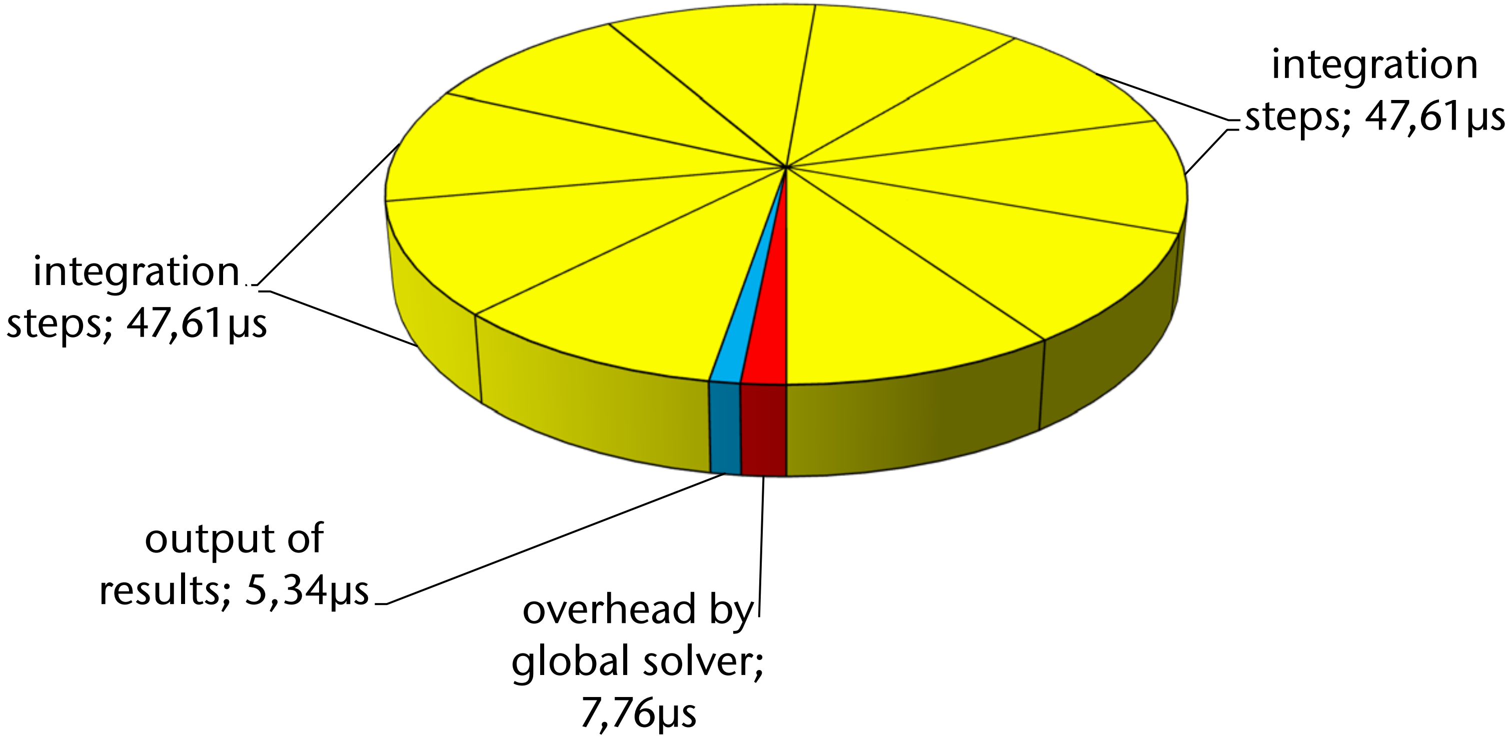


Model runtime over simulated time



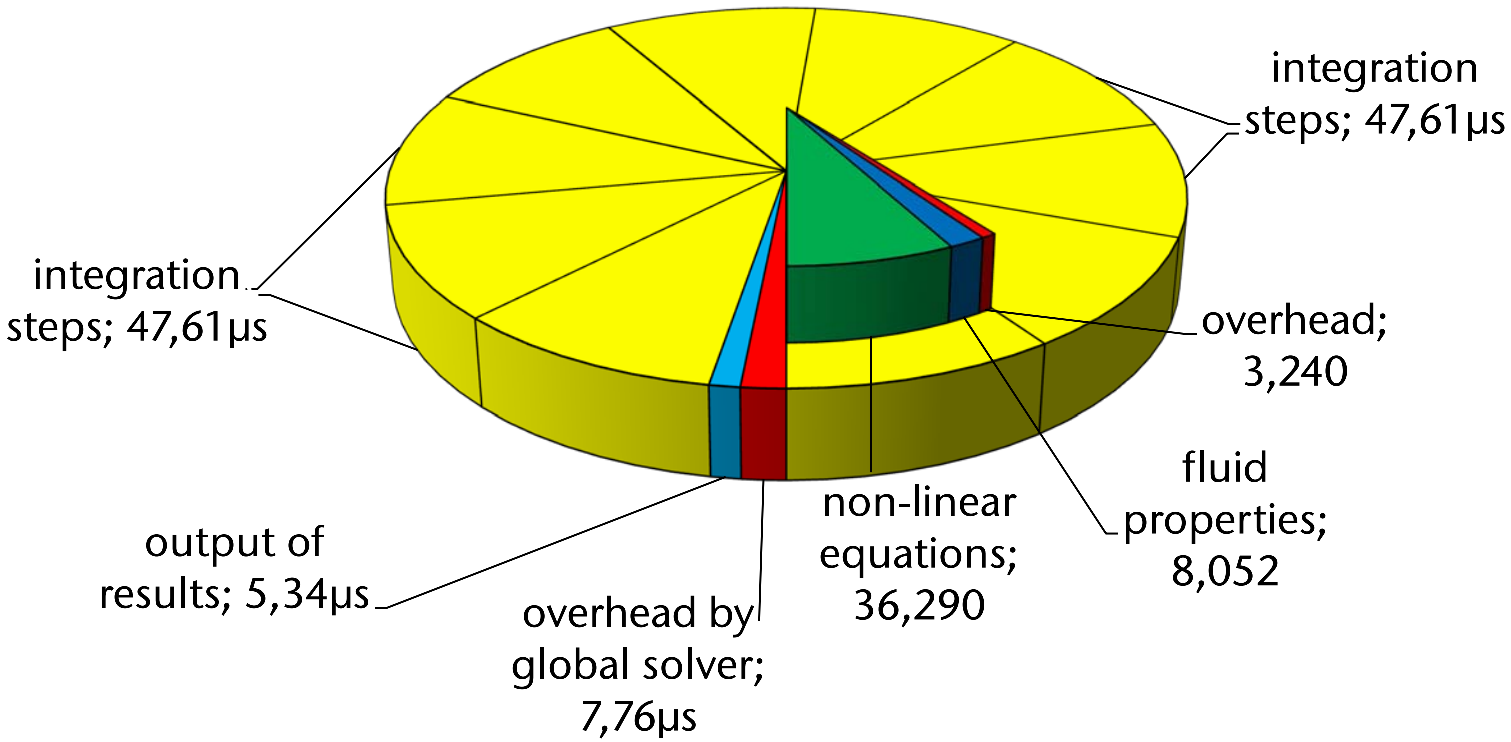
Results of Profiling

- Global solver step separated into integration steps and output of results



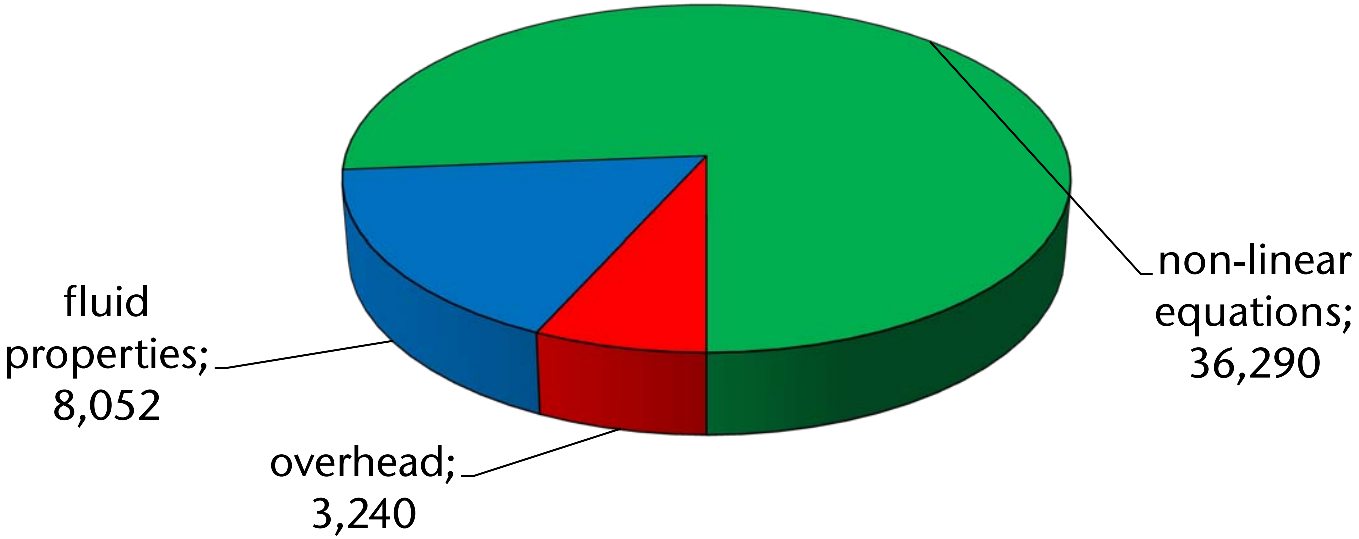
Results of Profiling

- Integration steps separated into external fluid property calculations, non-linear equations and overhead



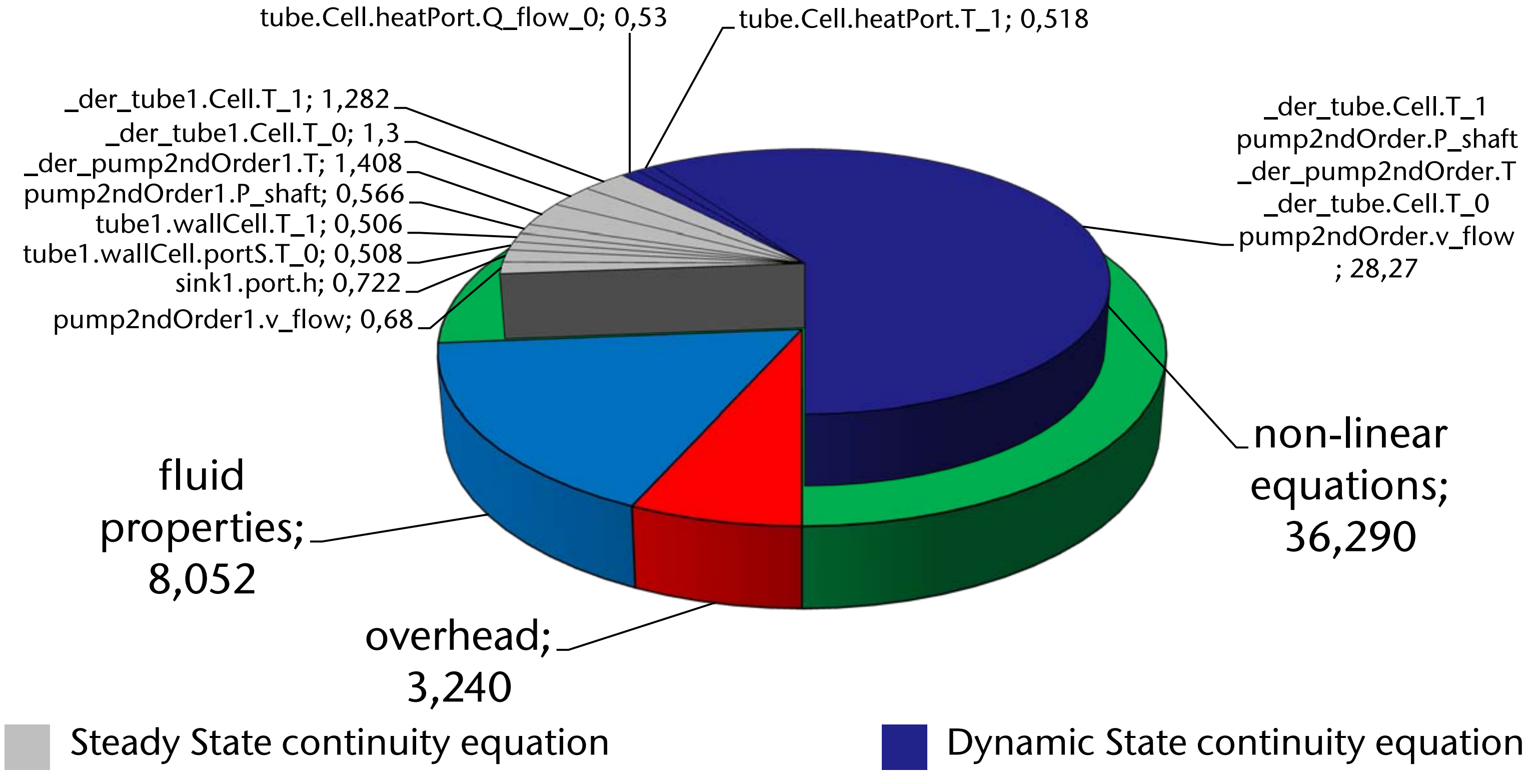
Algebraic loops

- Closer look on integration steps



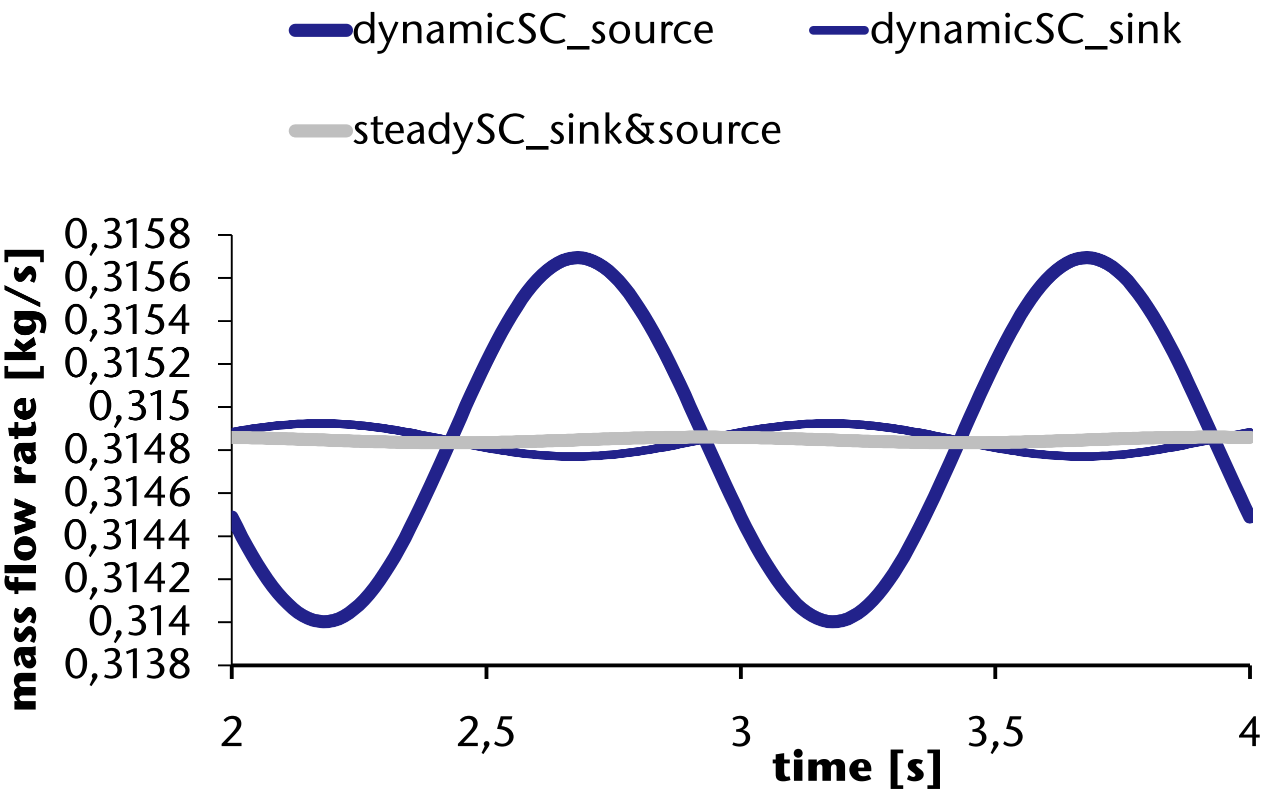
Algebraic loops

- Non-linear equations assinged to submodels



Mass flow of inlet and outlet

- Error due to thermal expansion of liquid



Overview

1. Introduction
2. Profiling of Modelica Models
3. Implementation
4. Case Study
5. Conclusion

Conclusion

- Saving results to HDD should be outsourced to a secondary non-Real-Time application
- Flat profiling by measurement for each step gives more than enough information
- Profiling helps identifying the work load contributions and aids the user optimizing the model

Thank you for your attention!