



High-Performance-Computing meets OpenModelica: Achievements in the HPC-OM Project

Linköping, 02/02/2015



HPCOM

www.hpc-om.de

Rexroth
Bosch Group



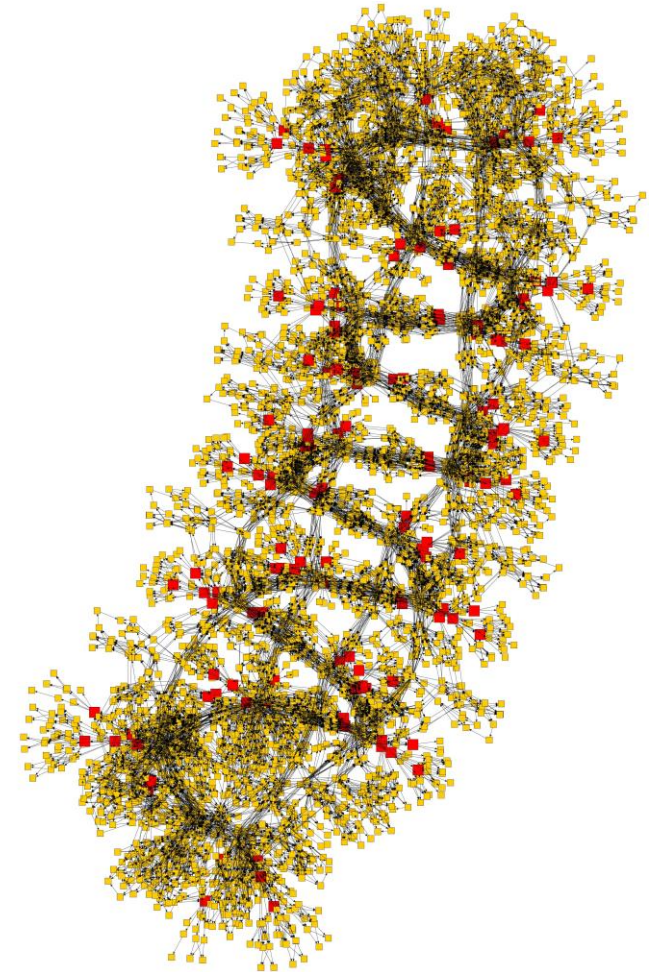
SPONSORED BY THE



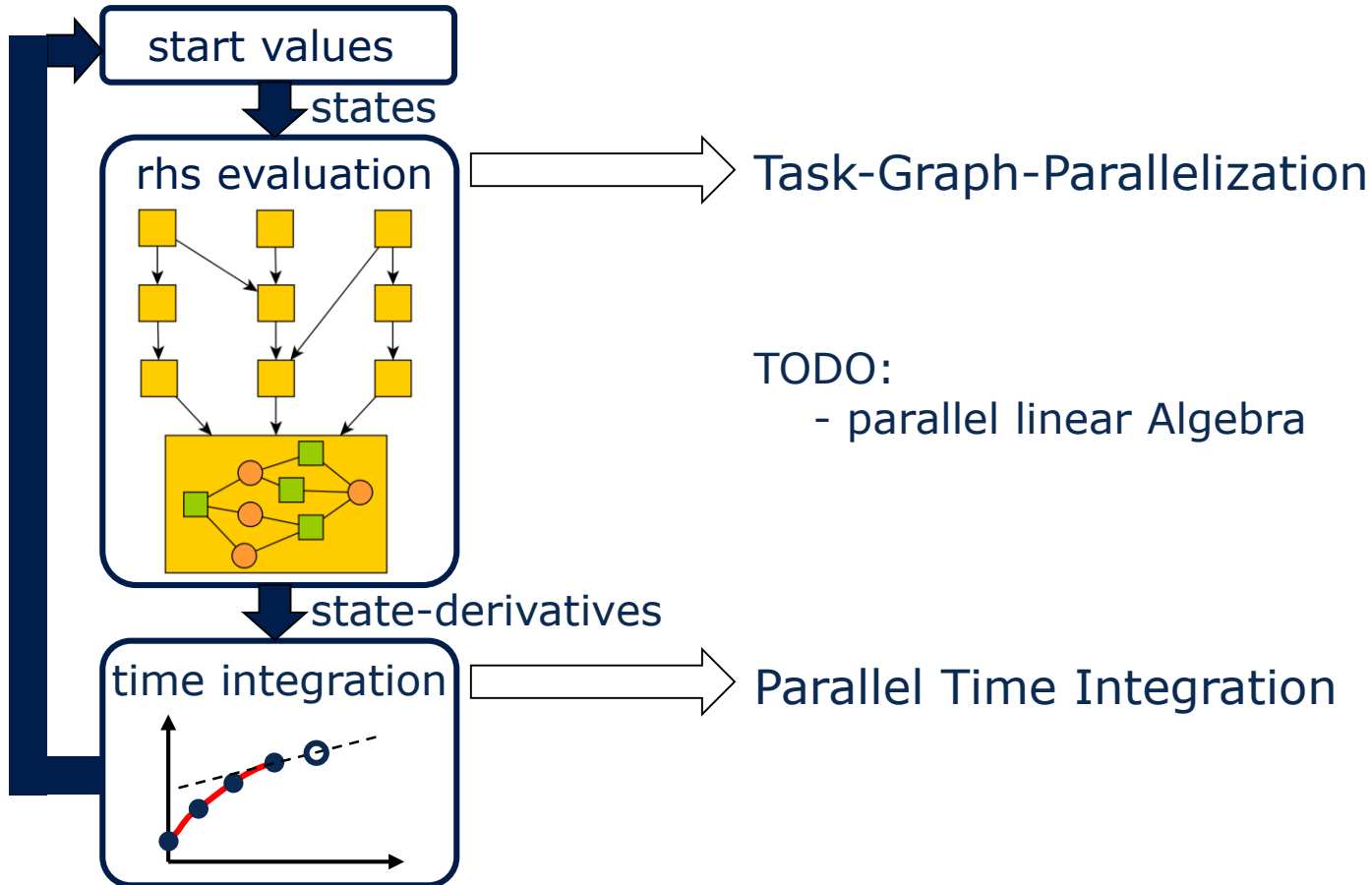
Federal Ministry
of Education
and Research

Outline

1. Parallelization Approaches in HPC-OM
2. Current Benchmarks
3. Parallel Time Integration
4. Performance Analysis

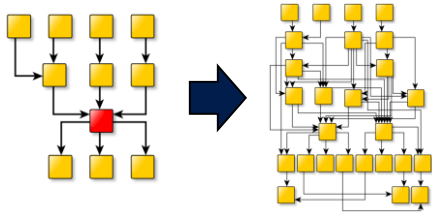


Parallelization Approaches in HPC-OM

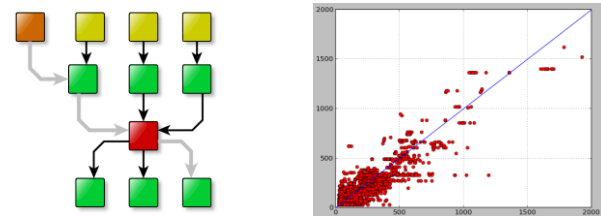


Task-Graph-Parallelization in HPC-OM

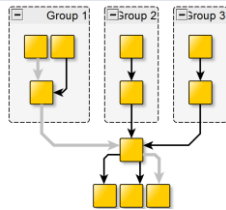
Symbolic Task-Graph Conditioning



Cost-Benchmarking & Estimation



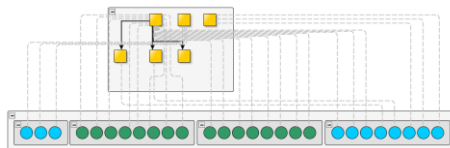
Task-Merging & Clustering



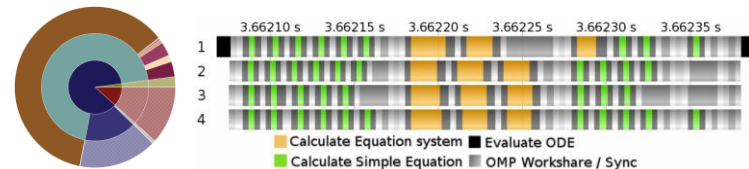
Scheduling & Parallel Codegeneration



Memory Optimization



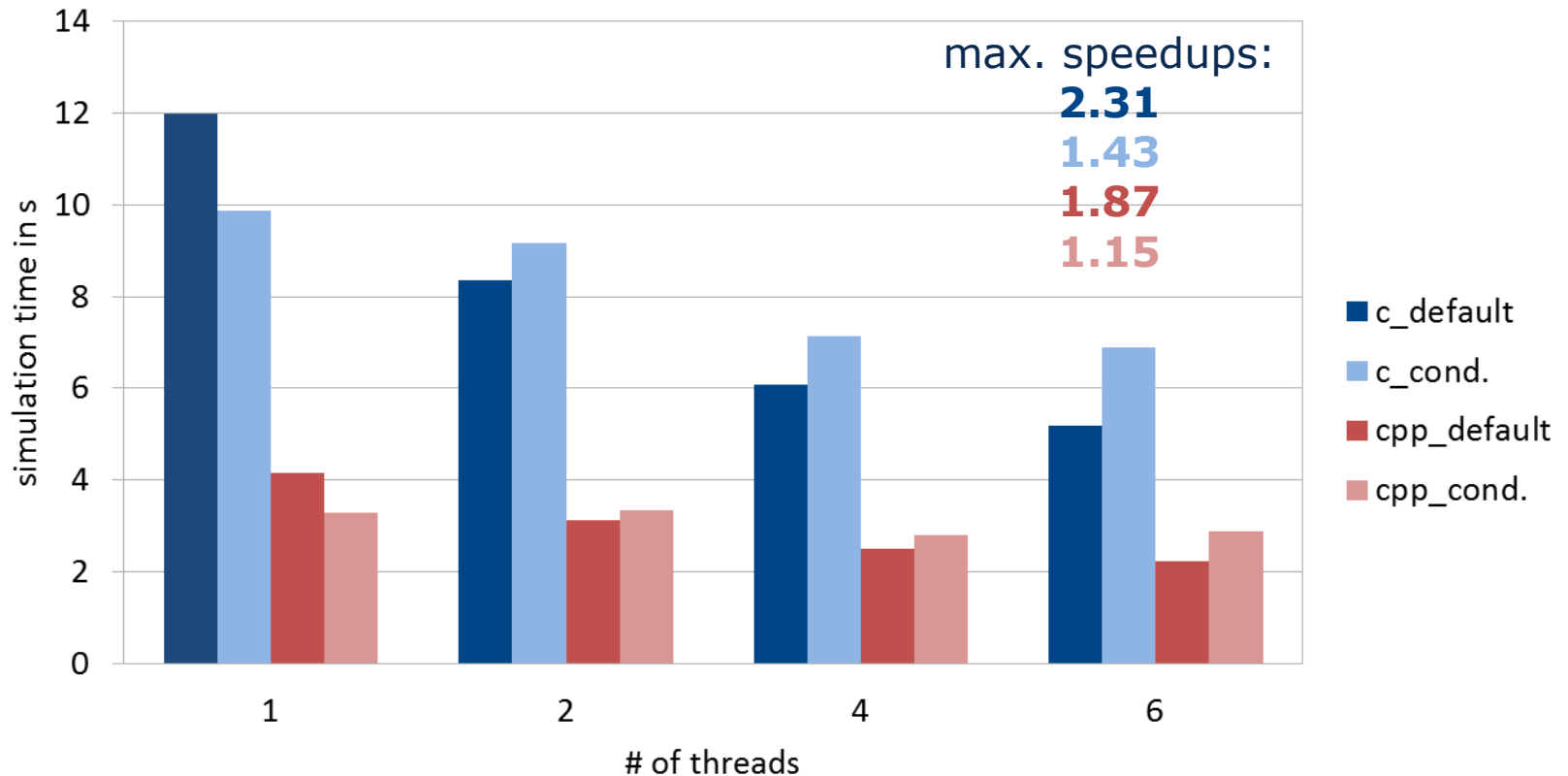
Profiling & Tracing



Current Benchmarks

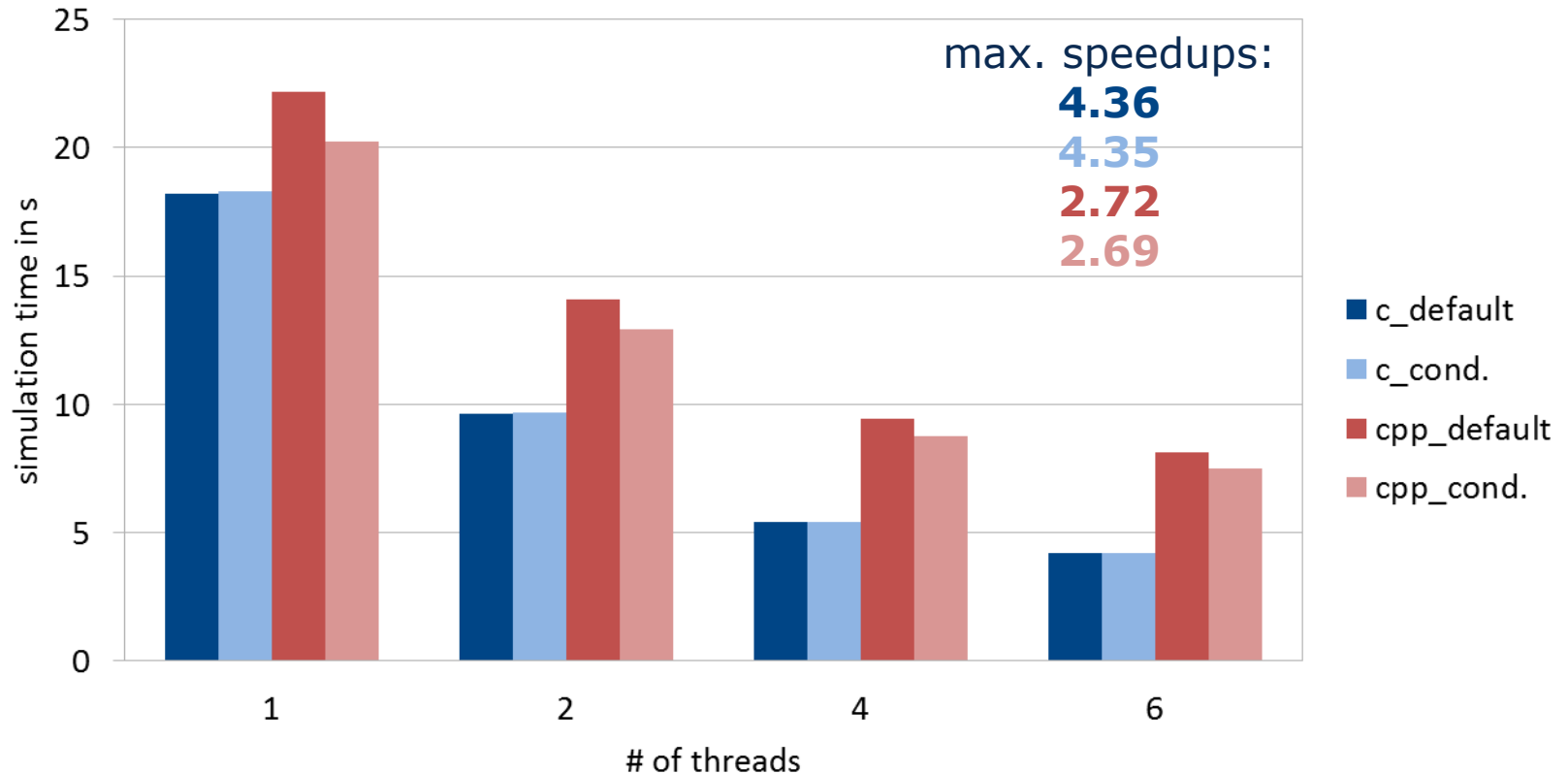
- Compare: **C runtime** & **C++ runtime** using HPC-OM
- Compare: **with task-graph conditioning and without** (i.e. partitioning of linear torn systems)
- System: i7-3930K, 6 cores @ 3.20 GHz, linux, GCC- 4.8.2
- „Dynamic Scheduling“ using OpenMP with level scheduler

Modelica.Electrical.Analog.Examples.CauerLowPassSC



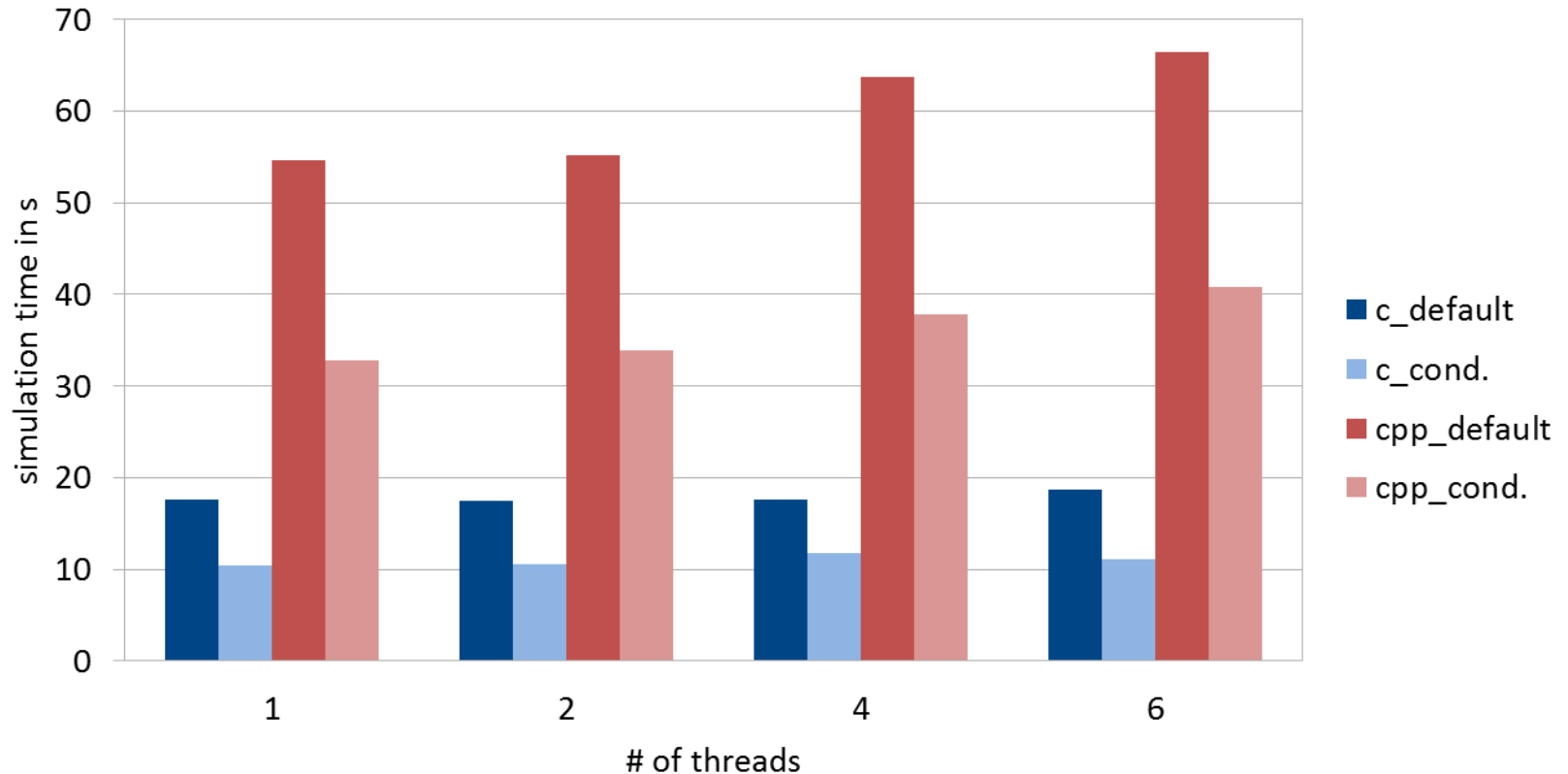
- Task graph conditioning leads to serial speedup, but less parallel speedup

Modelica.Fluid.Examples.BranchingDynamicPipes



- C runtime has higher speedups than C++ runtime

Modelica.Mechanics.MultiBody.Examples.Loops.EngineV6



- Dominating linear systems in mechanical models are a bottleneck
- Serial conditioning speedup **1.69** and **1.66**

Summary

- Static scheduling → speedups between 2 and 3
- „Dynamic Scheduling“ (openmp) → speedups up to 4.5 (6 threads)
- Higher parallel speedup when using C runtime
- Task graph conditioning leads basically to serial speedup
- Mechanical models → no speedup, dominated by linear system



TECHNISCHE
UNIVERSITÄT
DRESDEN

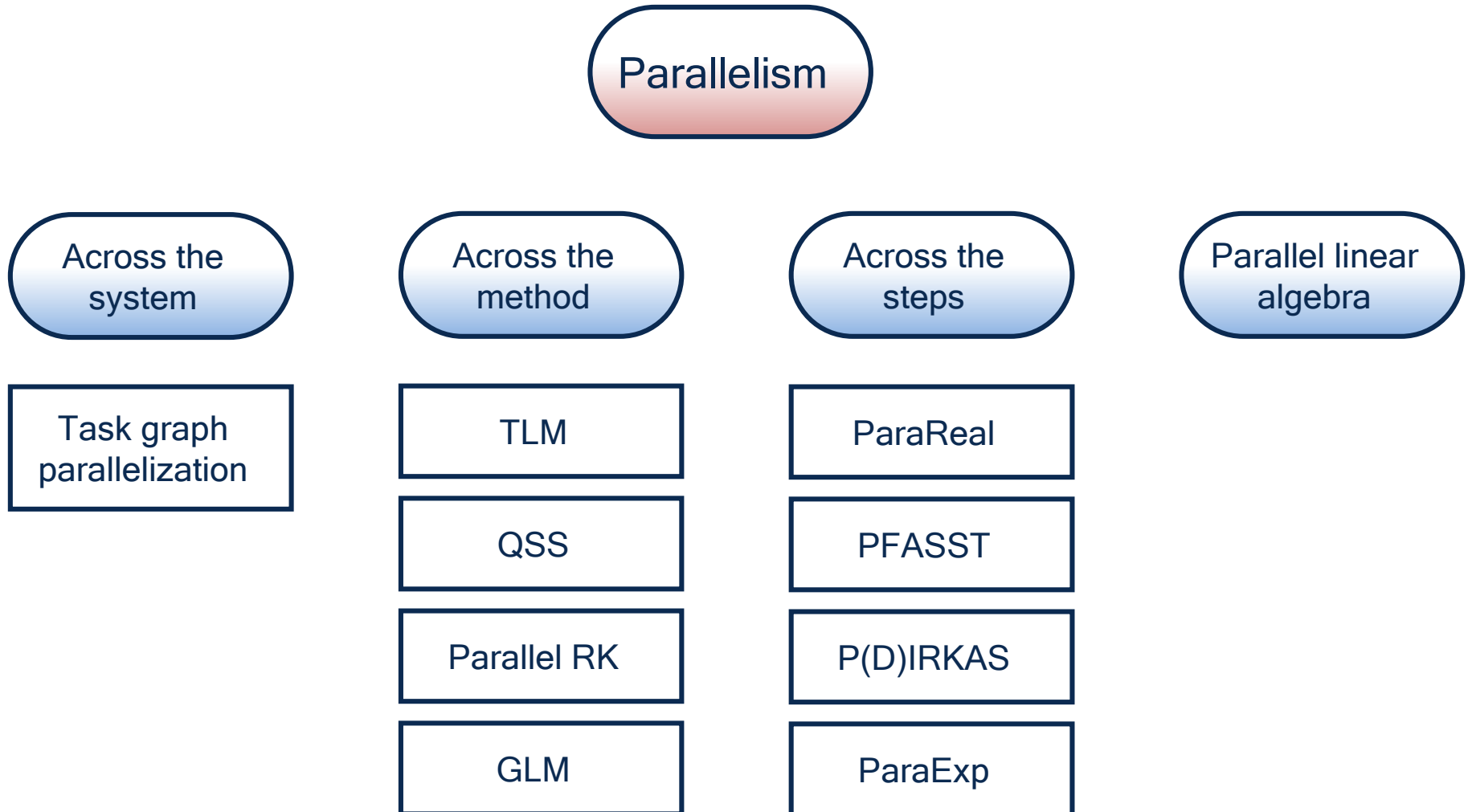
OpenModelica
Workshop 2015

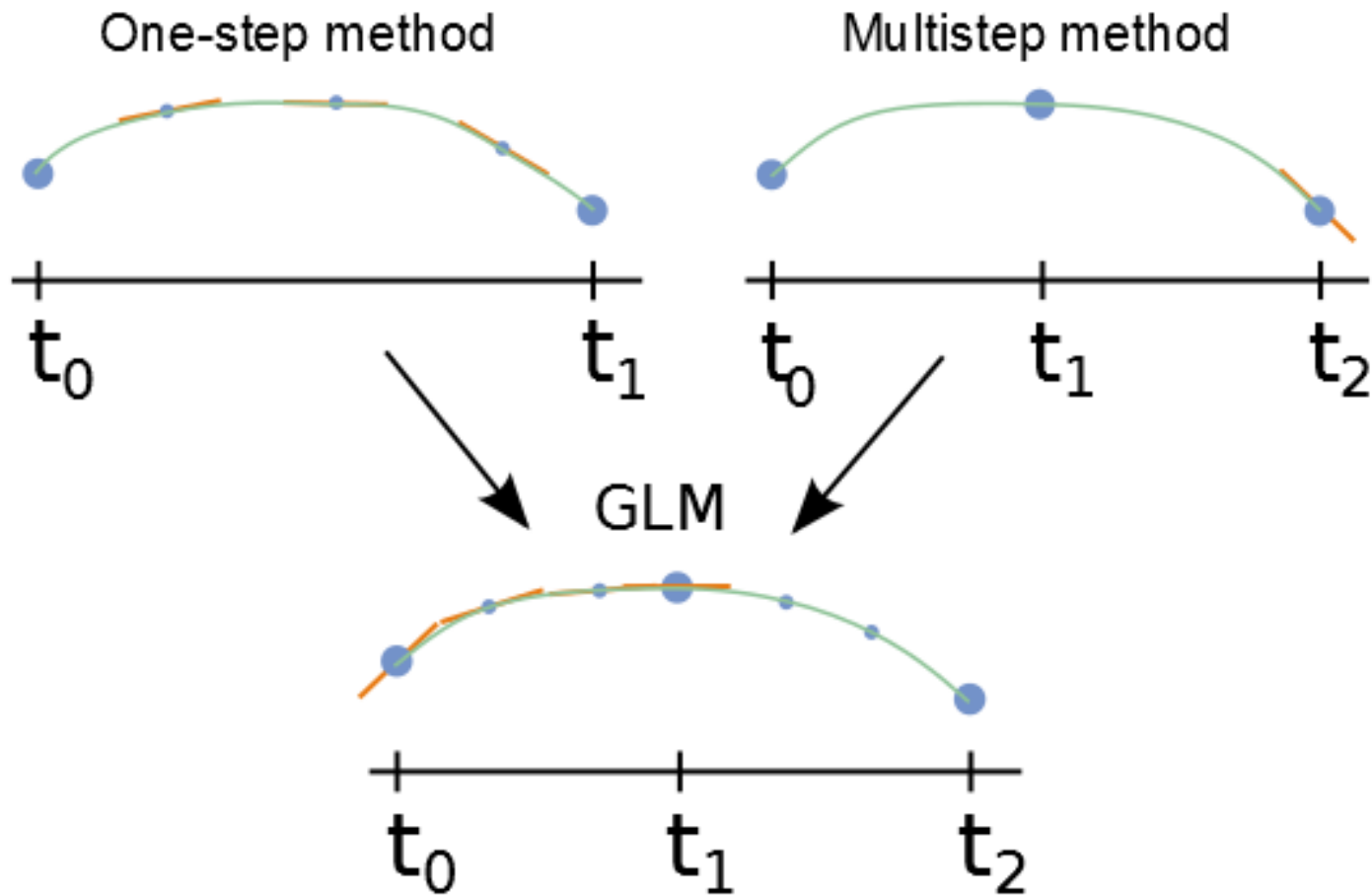
Parallel Time Integration

Linköping, 02/02/2015



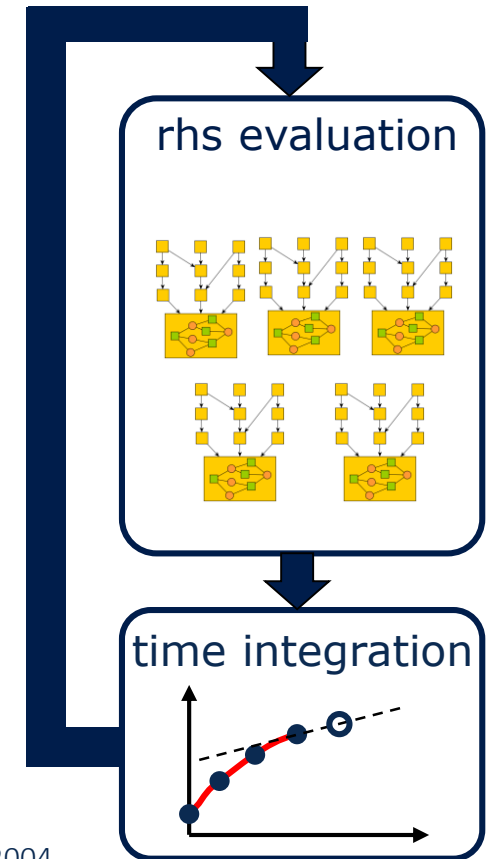
DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur





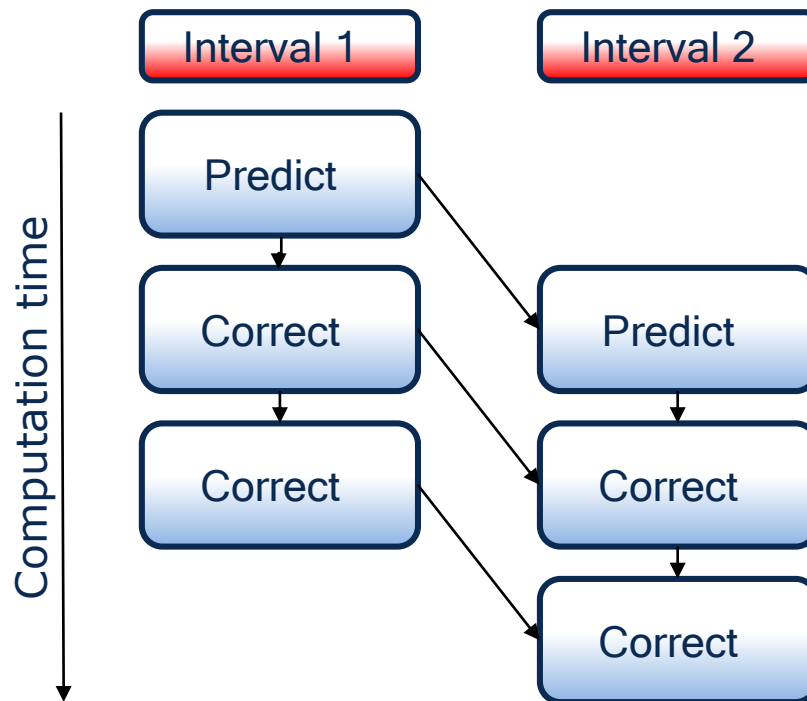
Linear implicit PEER-W-Method¹

- Method of order n runs on n parallel clusters
- Order 5 Peer-Method will be implemented
- Can efficiently employ up to 80 processors



1) Schmidt & Weiner, Parallel two-step W-methods with peer variables, SIAM J Numer. Ana., 2004

Parallelism across the steps

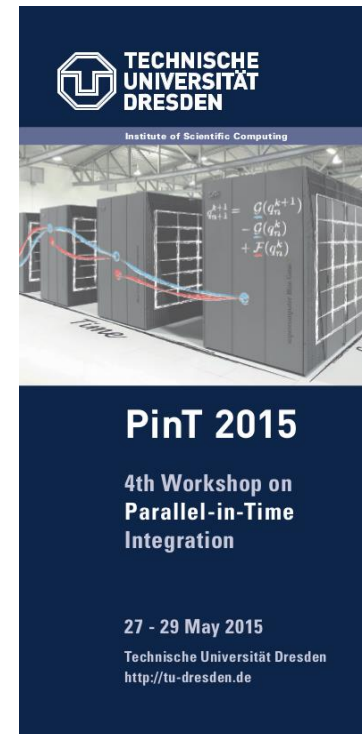


- Reported speedups of up to 15
- Methods of arbitrary order are possible
- High speedups only reached for very high precision requirements
- Tuning method parameters is difficult, needs in-depth knowledge of simulated model

1) v. d. Houwen et al. - Parallel iteration across the steps of high-order Runge-Kutta methods for nonstiff initial value problems, JCAM 95

Summary

- Several approaches to parallelization exist
- Parallel GLMs showed promising performance in previous tests
- Peer method will be implemented in the near future
- Parallel-in-Time method are not suitable for usage in OpenModelica



Performance Analysis

Linköping, 02/02/2015



DRESDEN
concept
Exzellenz aus
Wissenschaft
und Kultur

Motivation

„What gets measured gets improved.“

Peter F. Drucker

To improve HPC-OM speedups we have tried some things

- Switched from C to C++ runtime
- Implemented different scheduling and code generation techniques
- Embedded different solvers

→ Often not as fast as expected

→ Better analyses required

Profiling

„A profile provides an inventory of performance events and timings for the execution as a whole.“

IPM-HPC















```
"name": "Modelica.Electrical.Analog.Examples.CauerLowPassSC",
"cname": "CauerLowPassSC",
"code": [
  {"id": "calcFunction", "ncall": 202716, "time": 5888891159, "maxTime": 305709, "meanTime": 29049},
  {"id": "solve", "ncall": 600, "time": 7722982737, "maxTime": 25416490, "meanTime": 12871637},
  {"id": "writeOutput", "ncall": 120068, "time": 183756486, "maxTime": 998646, "meanTime": 1530},
  {"id": "evaluateZeroFuncs", "ncall": 121282, "time": 44150038, "maxTime": 10737, "meanTime": 364}
]
```

- Profiling already available in C runtime, but now in C++ runtime as well
- Besides simple time measurements, hardware counters can be analyzed
- Results stored in JSON-file

Continuous Benchmarking

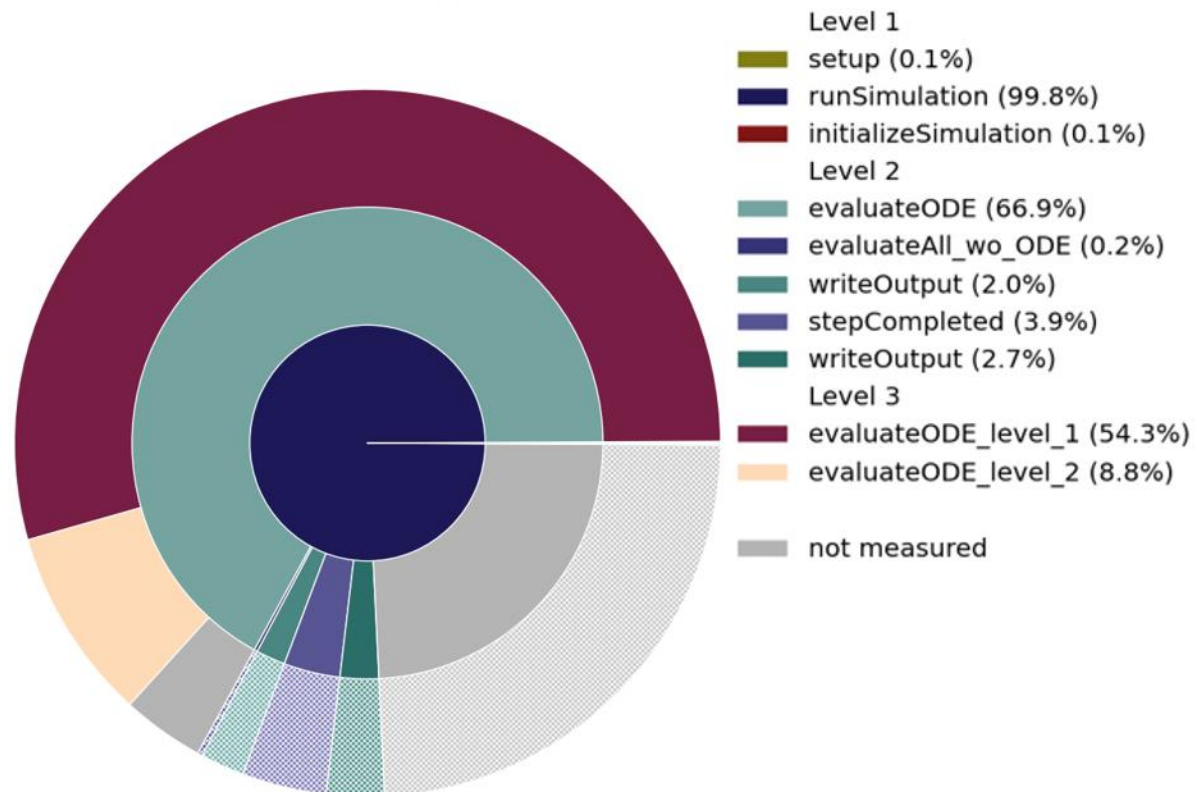
- Open Source fork "Jenkins" of "Hudson"
- Used to run various parallel HPCOM simulations daily



S	W	Name ↓	Last Success	Last Failure
		Taurus-Build-OMC-MKL	7 hr 0 min - #614	12 days - #597
		Taurus-Test-Backend	6 hr 44 min - #23	5 days 20 hr - #16
		Taurus-Test-Marc	6 hr 44 min - #7	1 day 6 hr - #4
		Taurus-Test-SimpleModels	6 hr 43 min - #17	1 day 15 hr - #13
		Taurus-Test-Spring	6 hr 40 min - #23	1 day 15 hr - #19
		Taurus-Update-HPCOMBenchmark-Repo	6 hr 44 min - #436	6 mo 9 days - #106
		Taurus-Update-OpenModelica-Repo	7 hr 1 min - #551	6 mo 19 days - #53

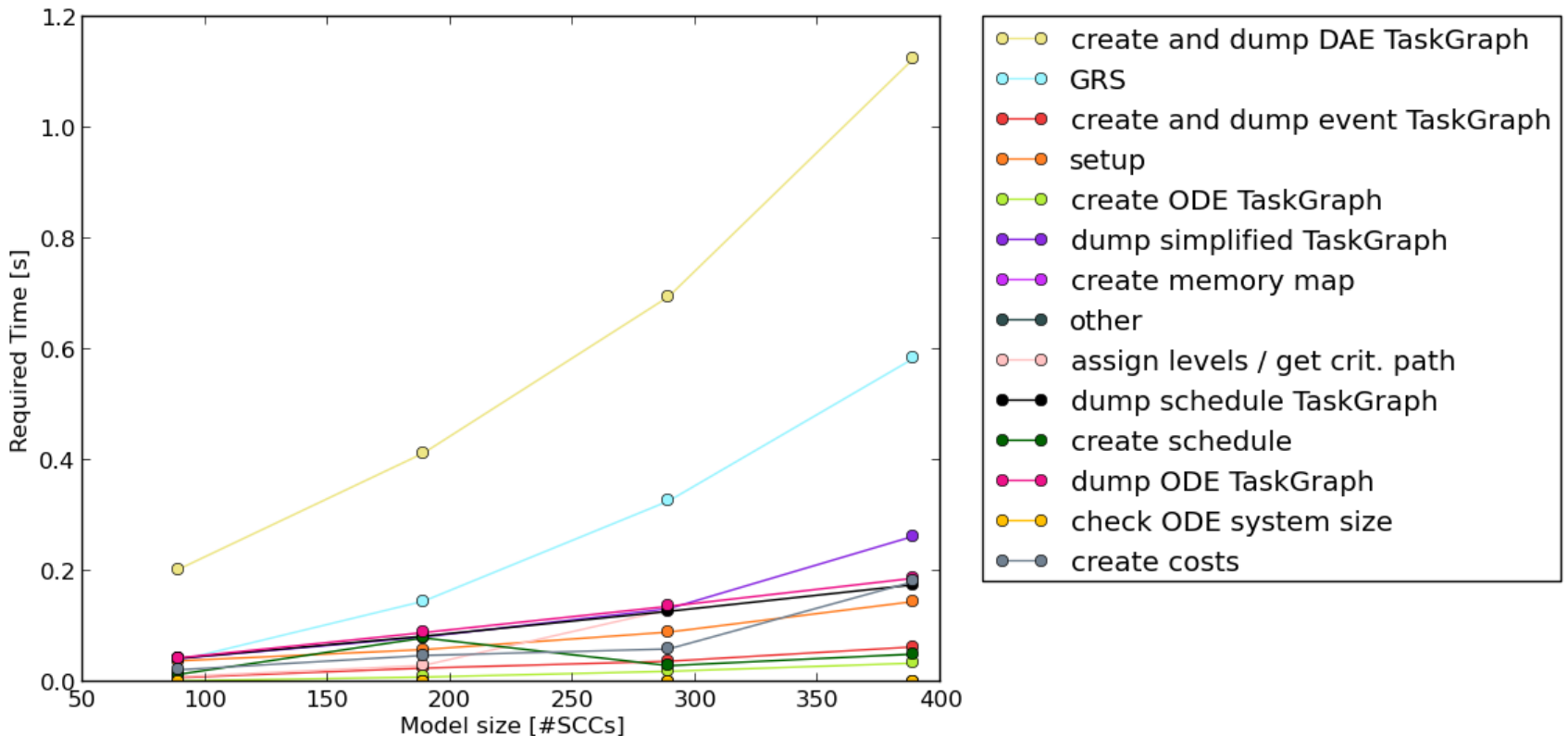
Continuous Benchmarking

- Analysis of JSON profiling data



Continuous Benchmarking

- Performance analysis of OMC backend algorithms



Summary

- Profiling available in C++ runtime
- Continuous benchmarking with Jenkins used in HPC-OM

Thank you for your attention.



»Wissen schafft Brücken.«

Volker Waurich

volker.waurich@tu-dresden.de

Michael Klöppel

michael.kloepfel@tu-dresden.de

Marcus Walther

marcus.walther@tu-dresden.de