TECHNISCHE UNIVERSITÄT DRESDEN

CENTER FOR INFORMATION SERVICES AND HIGH PERFORMANCE COMPUTING

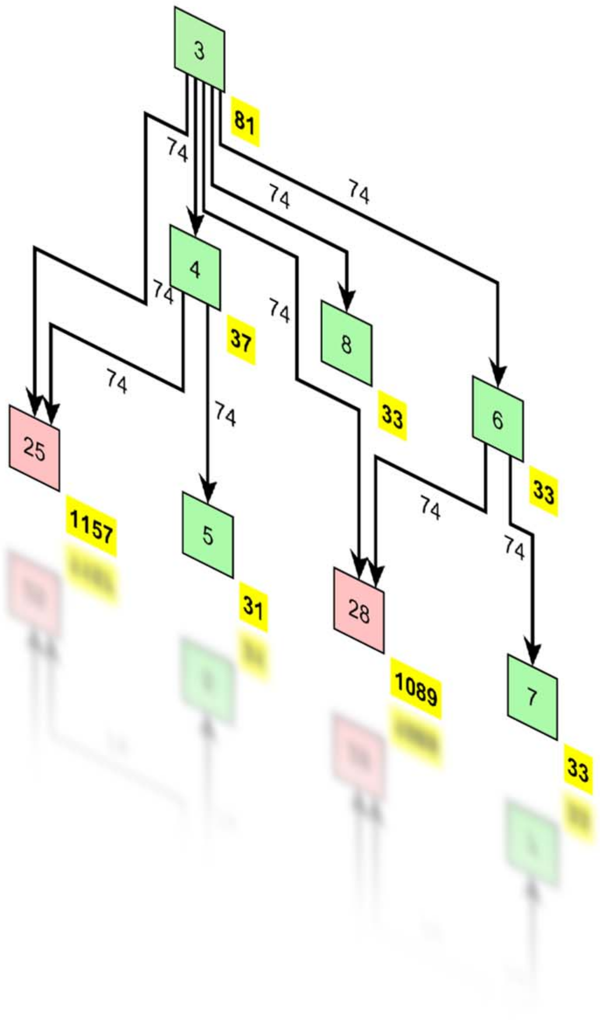
# Equation based parallelization of Modelica models

Linköping, 03/02/2014

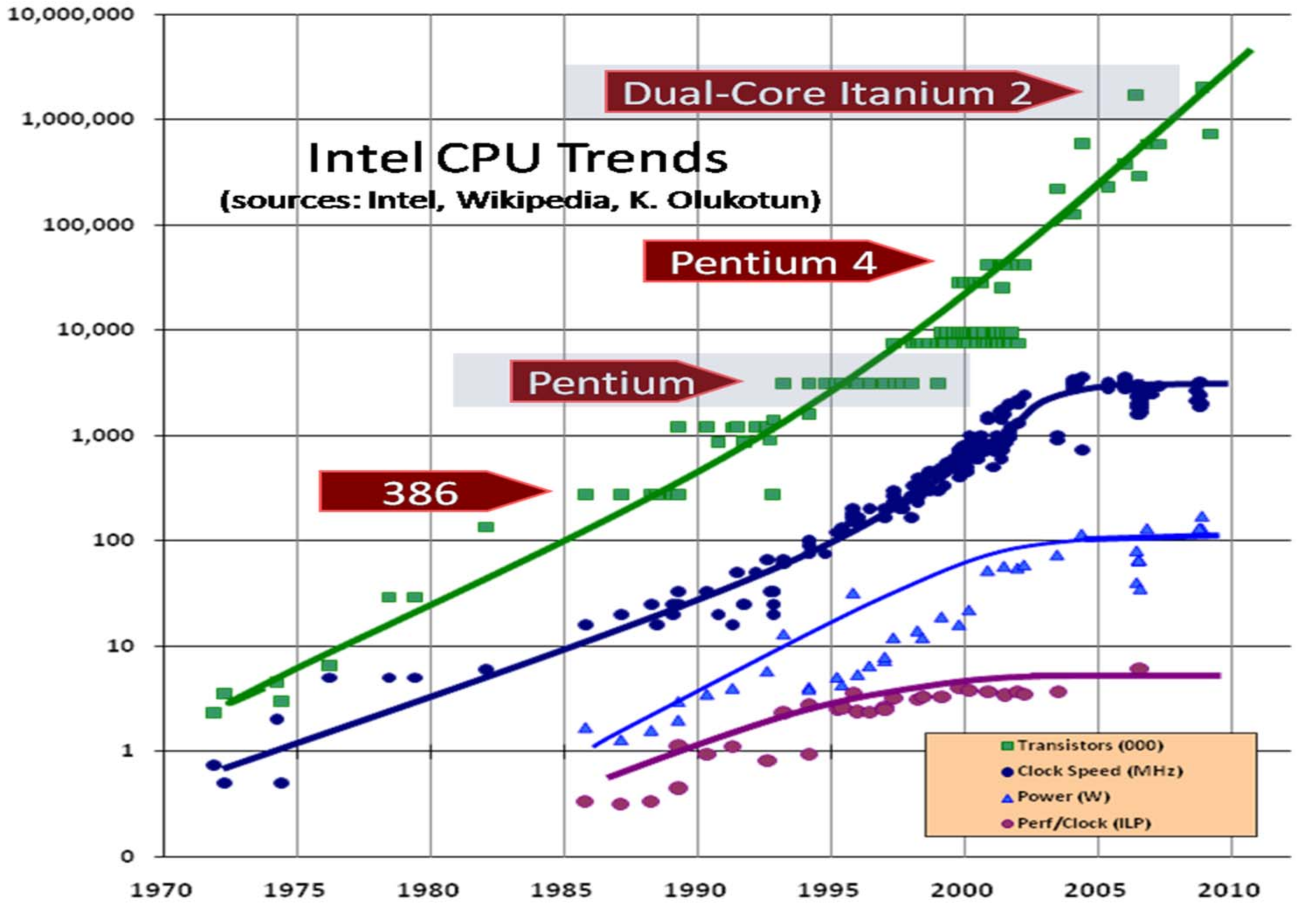
DRESDEN concept
Exzellenz aus Wissenschaft und Kultur

# Outline

1. Introduction
2. BLT parallelization
3. Task Graphs
4. HpcOm implementation
5. Benchmarks
6. Summary

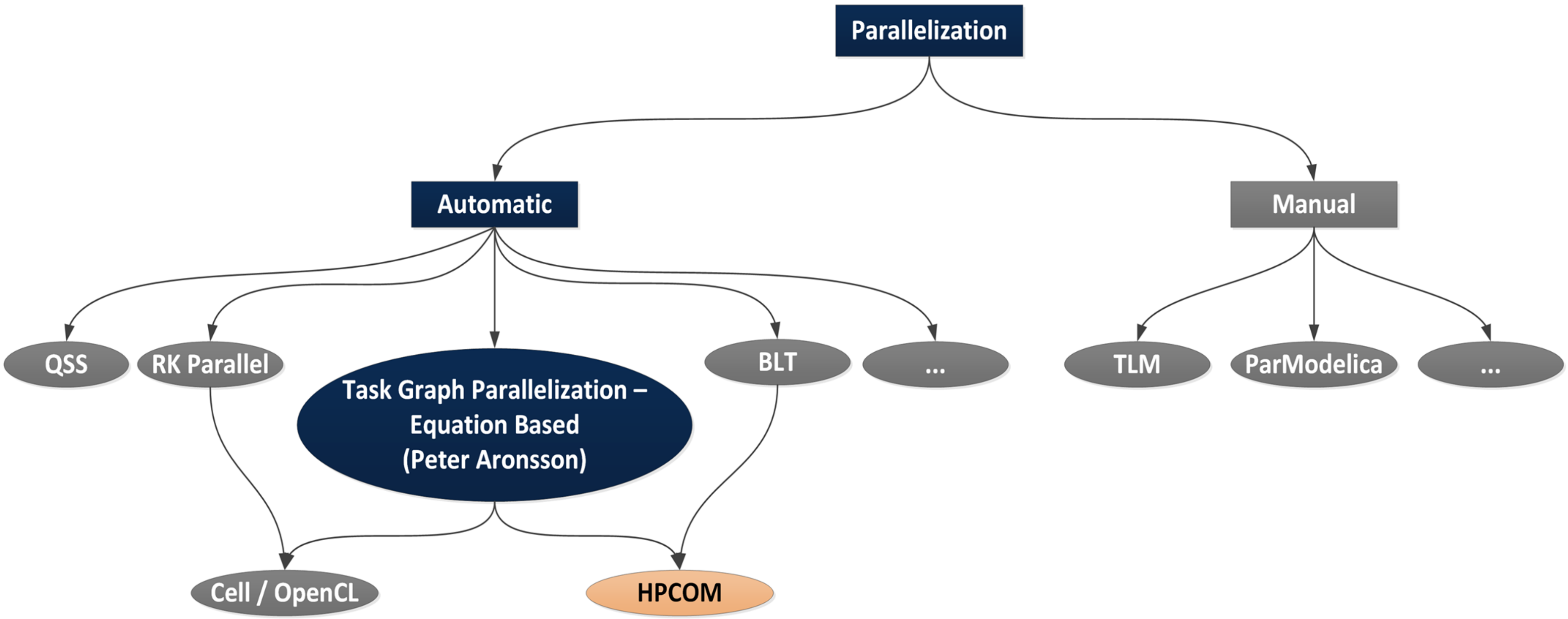


# 1. Introduction

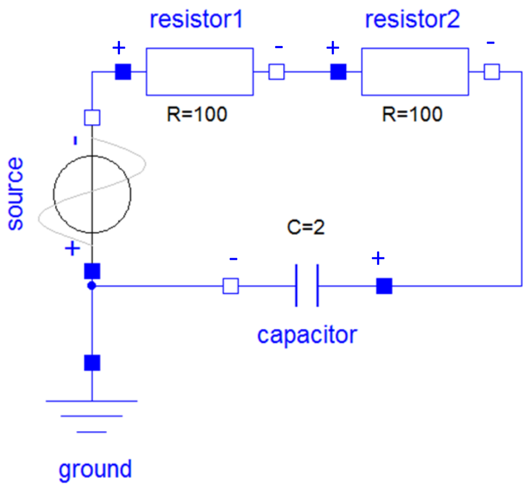



http://www.gotw.ca/publications/concurrency-ddj.htm

# Introduction



# 2. BLT parallelization




$$f_1: \quad u_s = offset$$

$$f_2: \quad u_{R1} = R_1 \cdot i_c$$

$$f_3: \quad P_{R1} = u_{R1} \cdot i_c$$

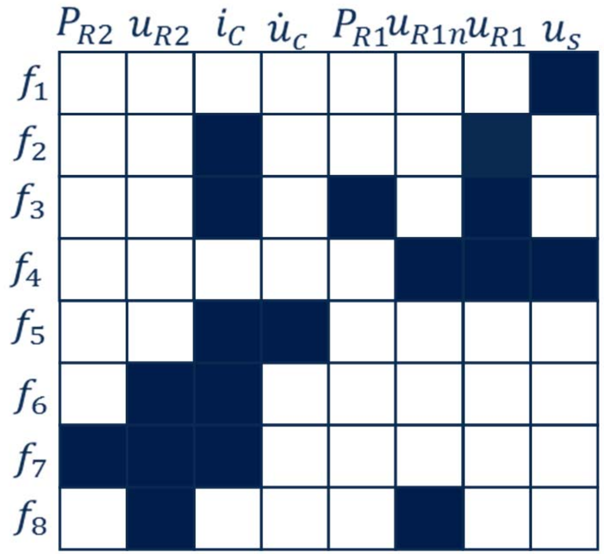
$$f_4: \quad u_{R1} = -u_s - u_{R1n}$$

$$f_5: \quad i_C = C \cdot \dot{u}_c$$

$$f_6: \quad u_{R2} = R_2 \cdot i_c$$

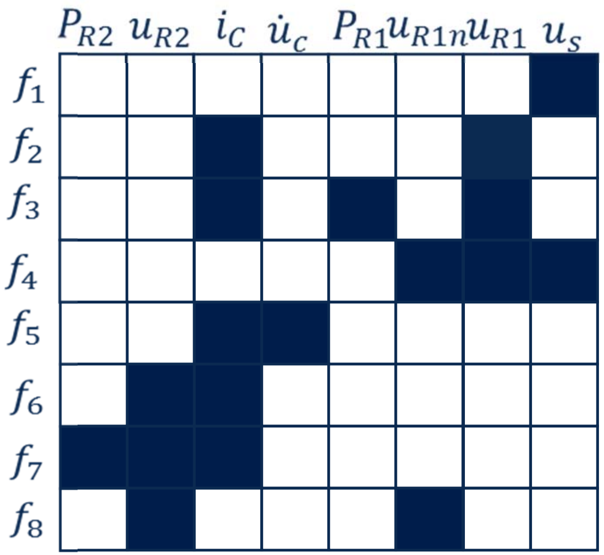
$$f_7: \quad P_{R2} = u_{R2} \cdot i_c$$

$$f_8: \quad u_{R2} = u_{R1n} - u_C$$

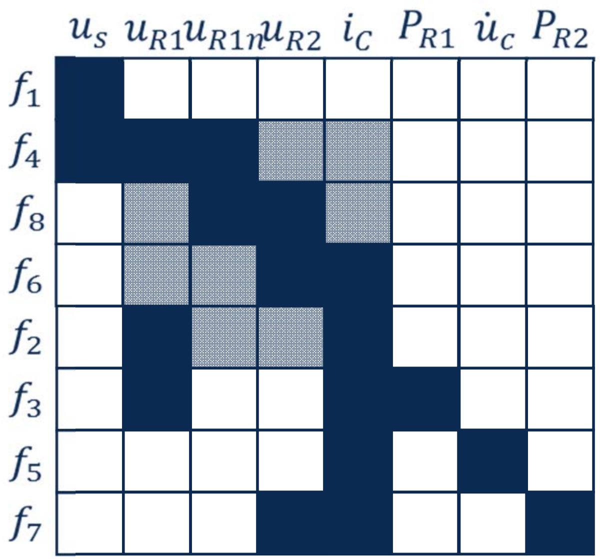


$$f_1: \quad u_s = offset$$
$$f_2: \quad u_{R1} = R_1 \cdot i_c$$
$$f_3: \quad P_{R1} = u_{R1} \cdot i_c$$
$$f_4: \quad u_{R1} = -u_s - u_{R1n}$$
$$f_5: \quad i_C = C \cdot \dot{u}_c$$
$$f_6: \quad u_{R2} = R_2 \cdot i_c$$
$$f_7: \quad P_{R2} = u_{R2} \cdot i_c$$
$$f_8: \quad u_{R2} = u_{R1n} - u_C$$

Equations

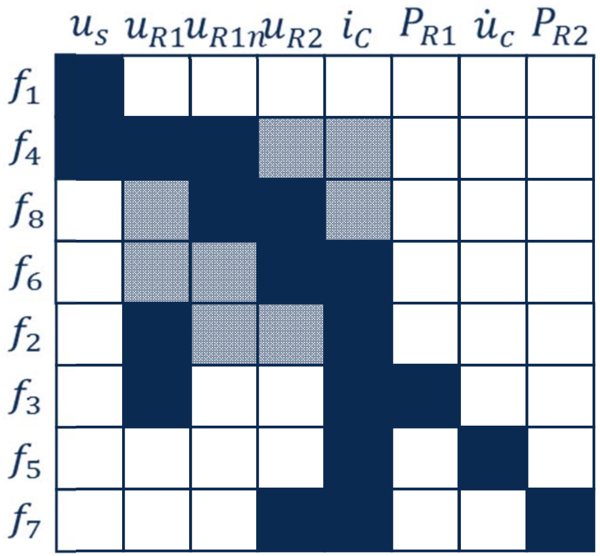



Incidence matrix

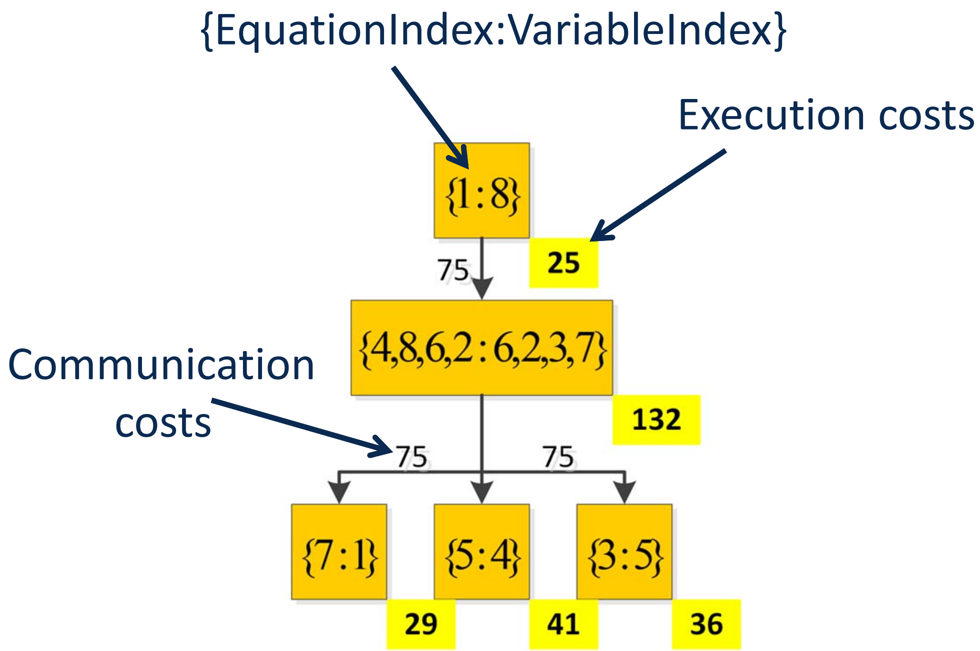



BLT-transformation

# BLT parallelization




BLT-
transformation




Task-Graph

# 3. Task Graphs

- Critical Path: Longest path through graph
- Serial execution time:

$$t_s = \sum_{n \in G} t_n$$

- *Theoretical parallel time for j threads:*

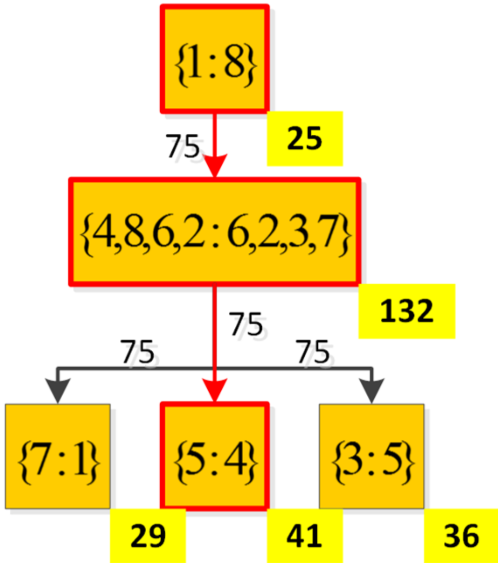
$$t_{P(j)}$$

- *Theoretical minimal parallel time:*

$$t_{Pmin} = \sum_{t \in crit} t_n$$

- Max. theoretical speedup:

$$n_{max} = \frac{t_s}{t_{Pmin}}$$

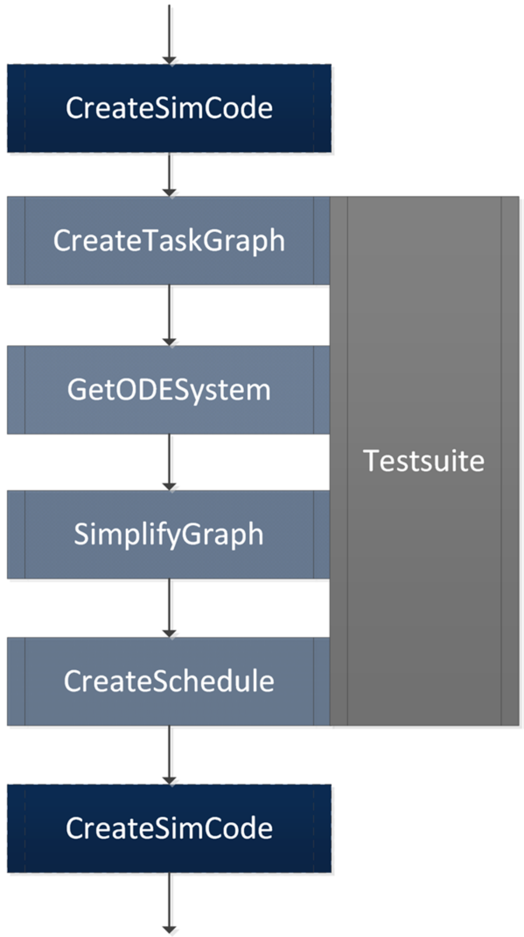


# 4. HpcOm Implementation

- Compiler backend module

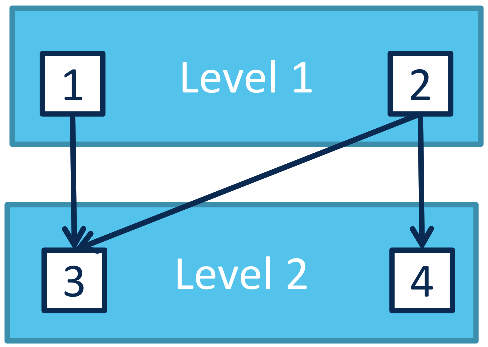
## Some Features

- Parallel code generation
- Task graph export as *.graphml
- Simple rewriting rules
- Multiple schedulers



# Scheduling

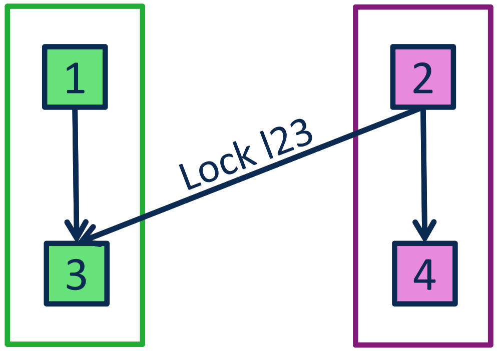
- Scheduling: Mapping between tasks and threads (NP-hard)
- 1. **Level Scheduler** (OpenMP)




```
static void solveOde(data) {
  //Level 1
  #pragma omp parallel sections
  {
    #pragma omp section
    {
       eqFunction_1(data);
    }
    #pragma omp section
    {
       eqFunction_2(data);
     }
  }
  //Level 2
 #pragma omp parallel sections
 {
     ...
}}
```

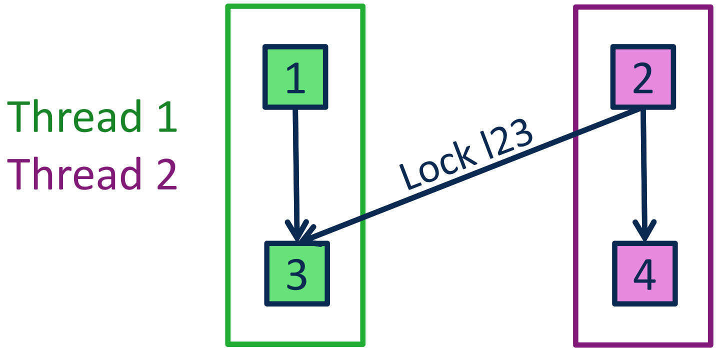
- 2. **List Scheduler** (HLF – heuristic, OpenMP)

Thread 1
Thread 2




```
static void solveOde(data) {
  INIT_LOCK(l23,true);

  #pragma omp parallel sections num_threads(2)
  {  //Thread 1 -- Green
    #pragma omp section {
       eqFunction_1(data);
       SET_LOCK(l23);
       eqFunction_3(data);
    }
    //Thread 2 -- Violett
    #pragma omp section {
       eqFunction_2(data);
       UNSET_LOCK(l23);
       eqFunction_4(data);
     }
  }}
```

- 2. **List Scheduler** (HLF – heuristic, pThreads)

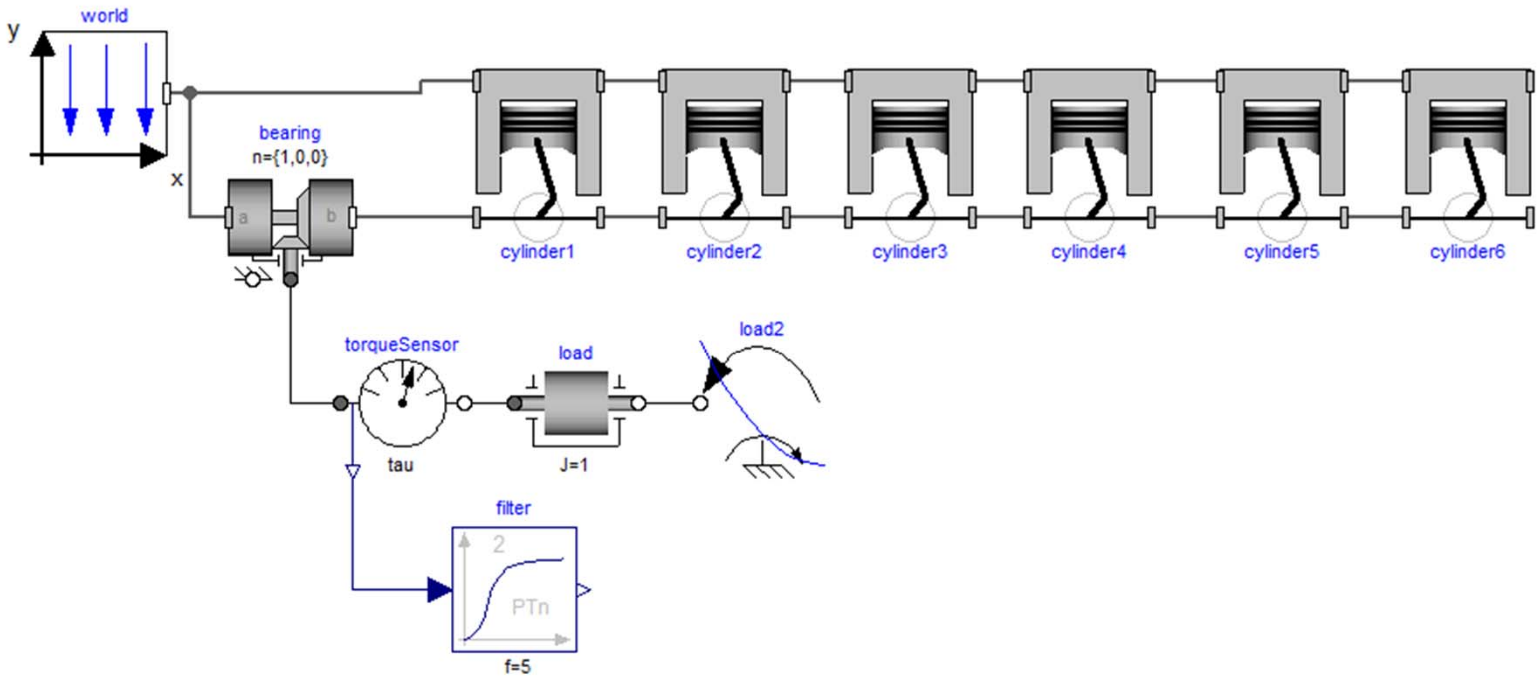


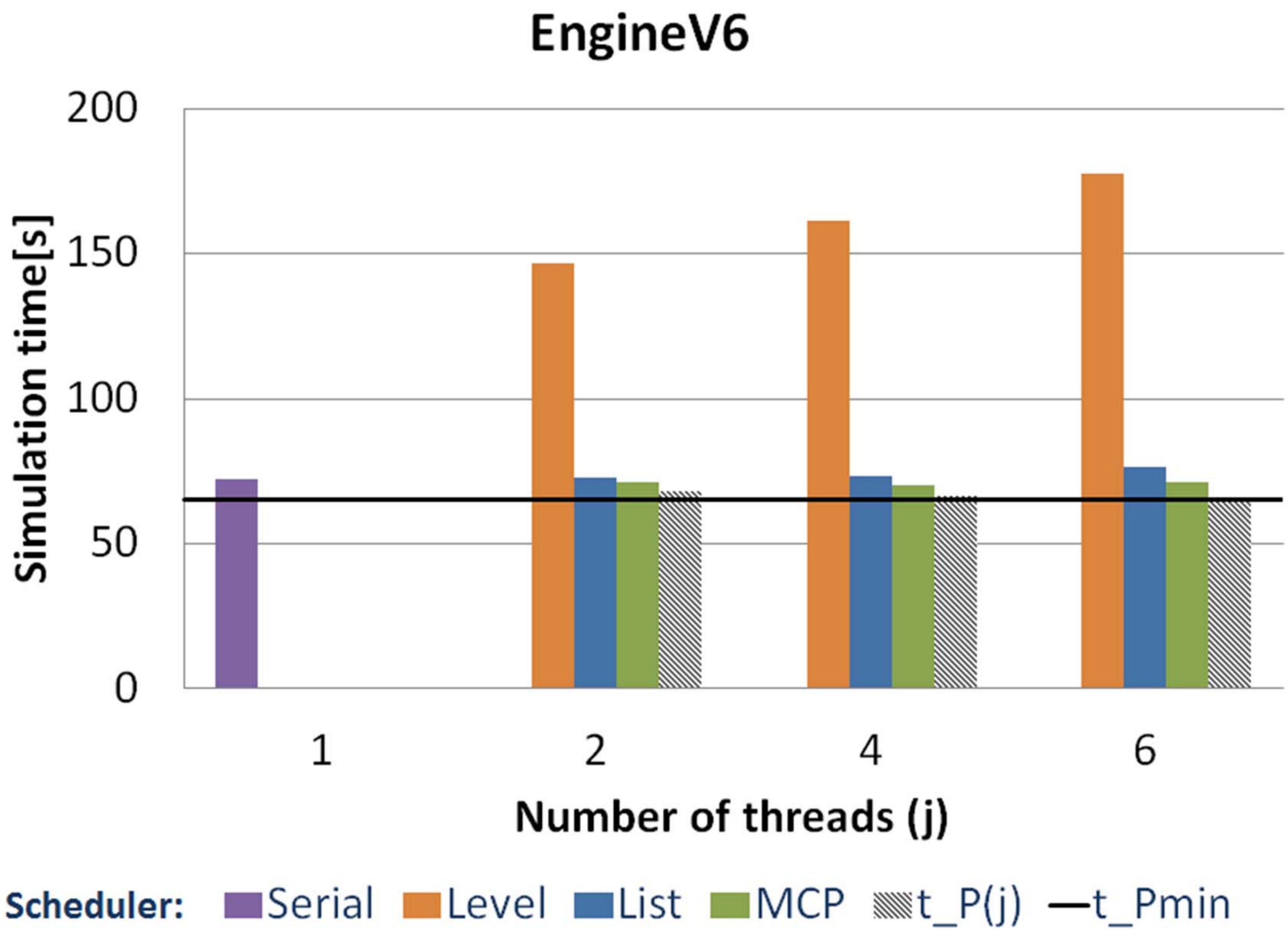

```
static void thread1Ode(data) { //Function of thread1
  while(1) {
    pthread_mutex_lock(&th_lock_0);
    eqFunction_1(data);
    SET_SPIN_LOCK(l23);
    eqFunction_3(data);
    pthread_mutex_unlock(&th_lock1_0);
  }
}

static void solveOde(data) {
  INIT_SPIN_LOCK(l23,true); //pthread_spinlock_t
  INIT_LOCKS();
  if(firstRun)
    CREATE_THREADS(...);
  //Start threads
  pthread_mutex_unlock(&th_lock_0);
  pthread_mutex_unlock(&th_lock_1);
  //"join"
  pthread_mutex_lock(&th_lock1_0);
  pthread_mutex_lock(&th_lock1_1);
}
```

- 3. **External Scheduler**
    - Schedule by hand
    - Graph partitioning (Metis)
- 4. **MCP Scheduler**

# 4. Benchmarks

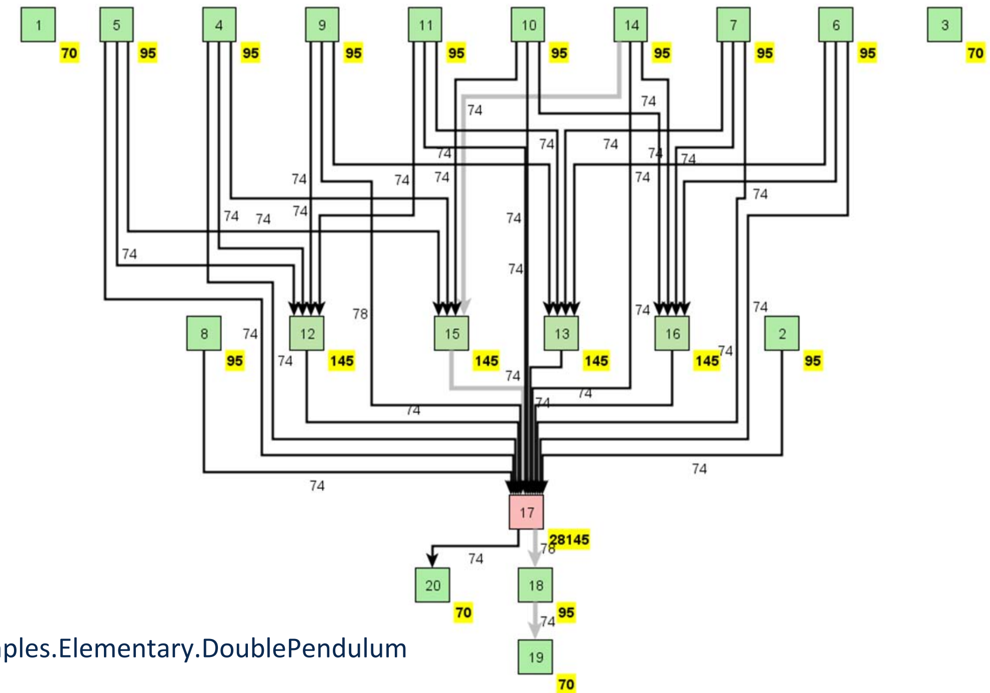
- **Engine V6**




EngineV6
200
150
100
50
0
Simulation time[s]
1
2
4
6
Number of threads (j)
Scheduler: Serial Level List MCP t_P(j) t_Pmin

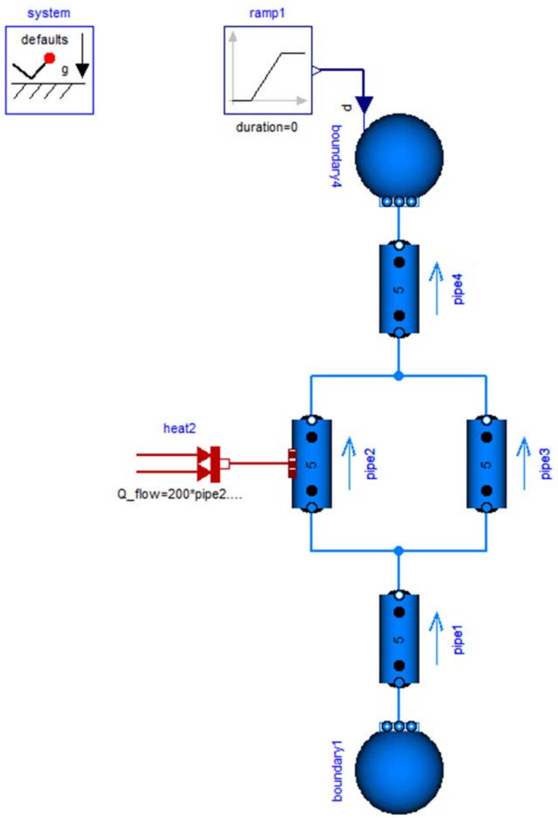
# Mechanics domain

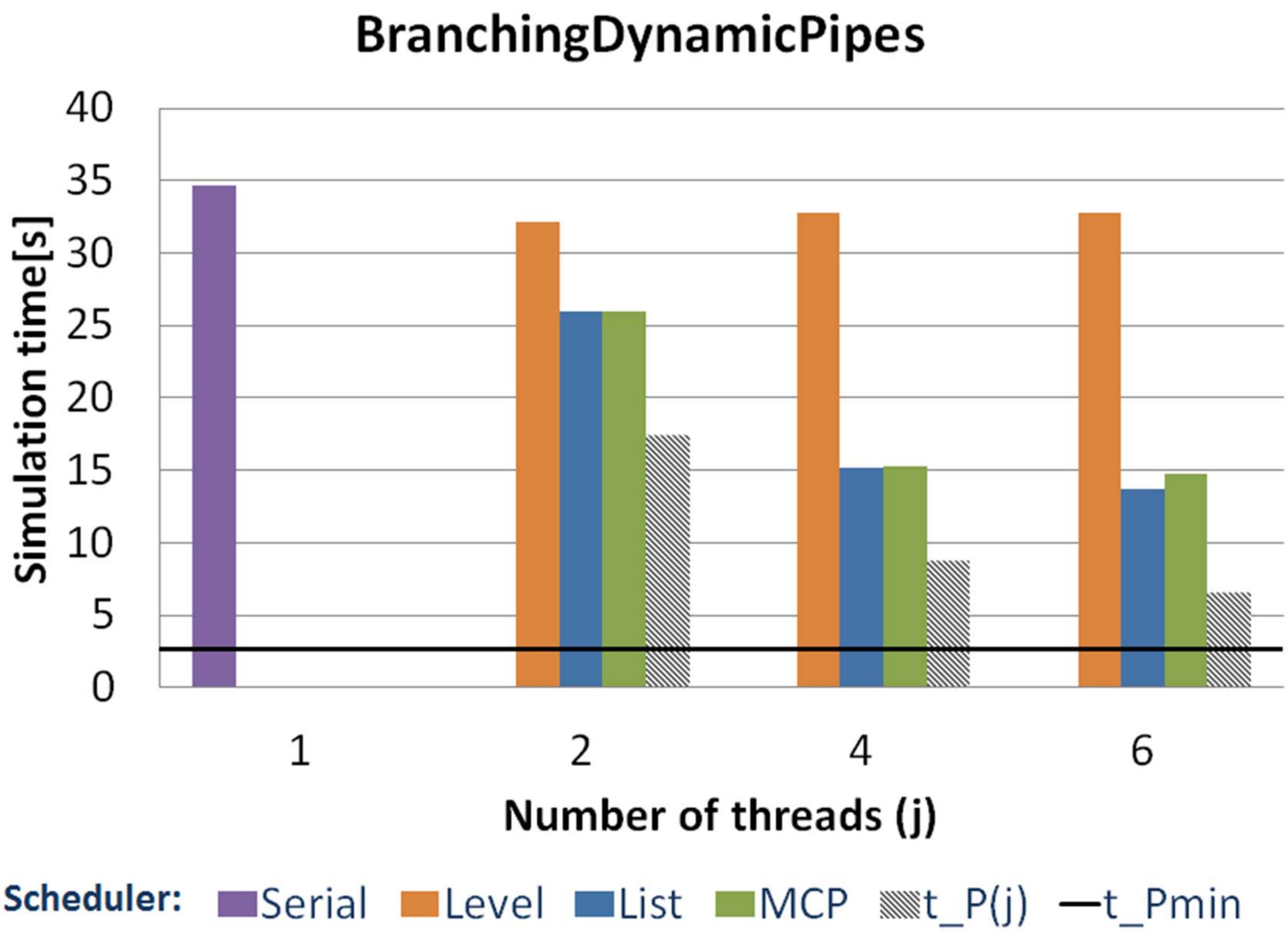
- Graph contains one „big“ task
    - Calculation of accelerations




Modelica.Mechanics.MultiBody.Examples.Elementary.DoublePendulum

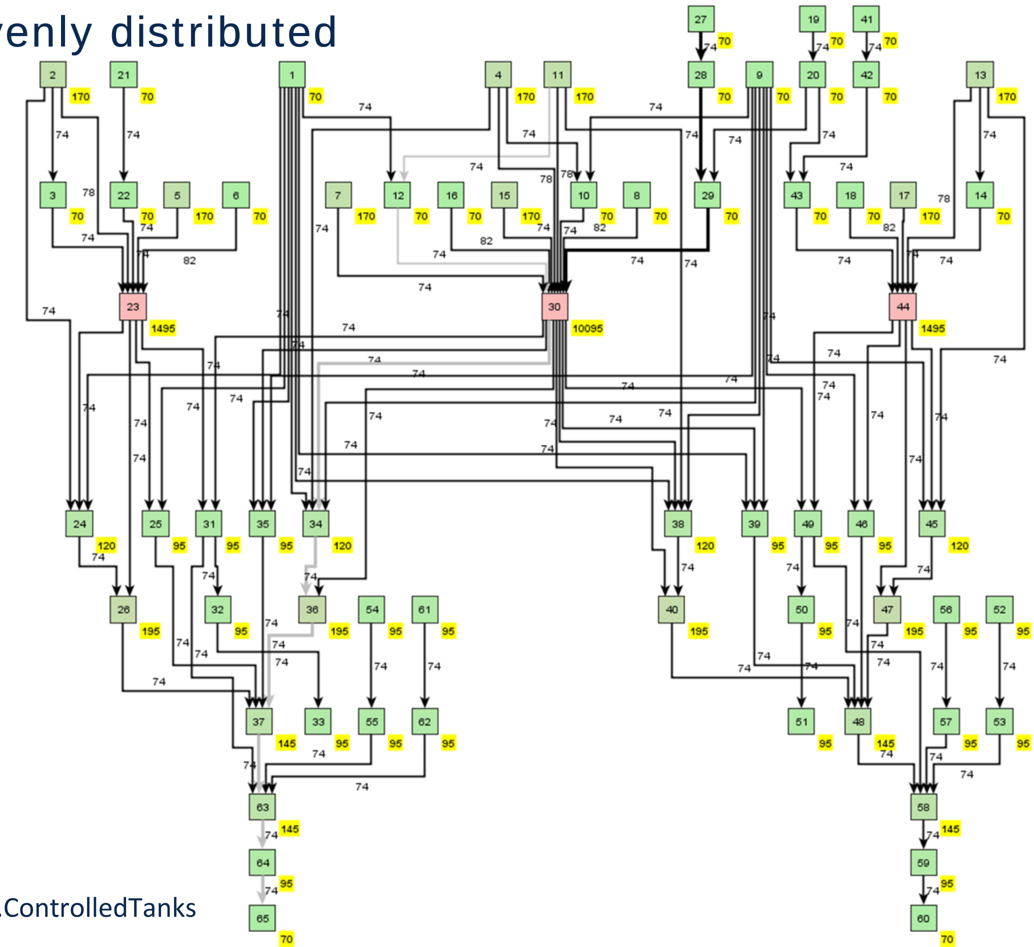
- **Branching Dynamic Pipes**




BranchingDynamicPipes
40
35
30
25
20
15
10
5
0
Simulation time[s]
1
2
4
6
Number of threads (j)
Scheduler:
Serial
Level
List
MCP
t_P(j)
t_Pmin

# Fluid domain

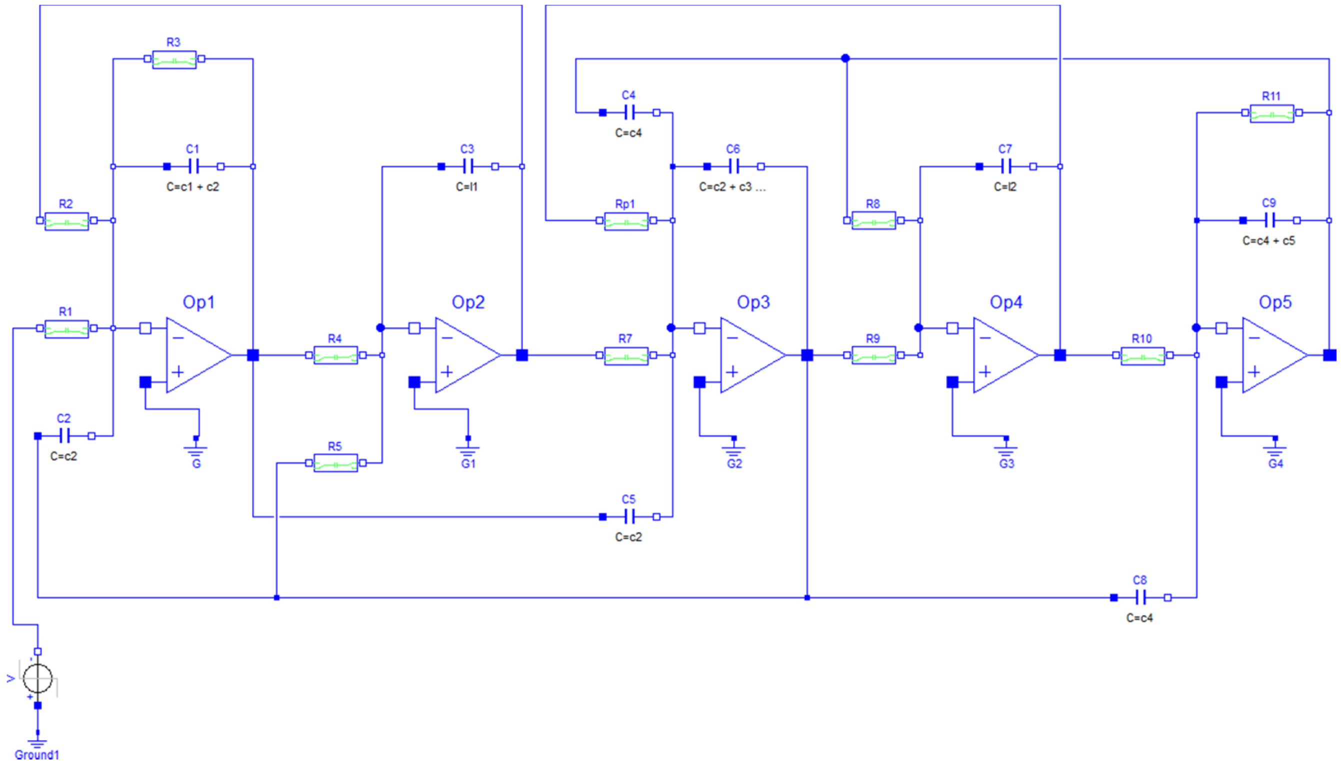
- Execution costs evenly distributed

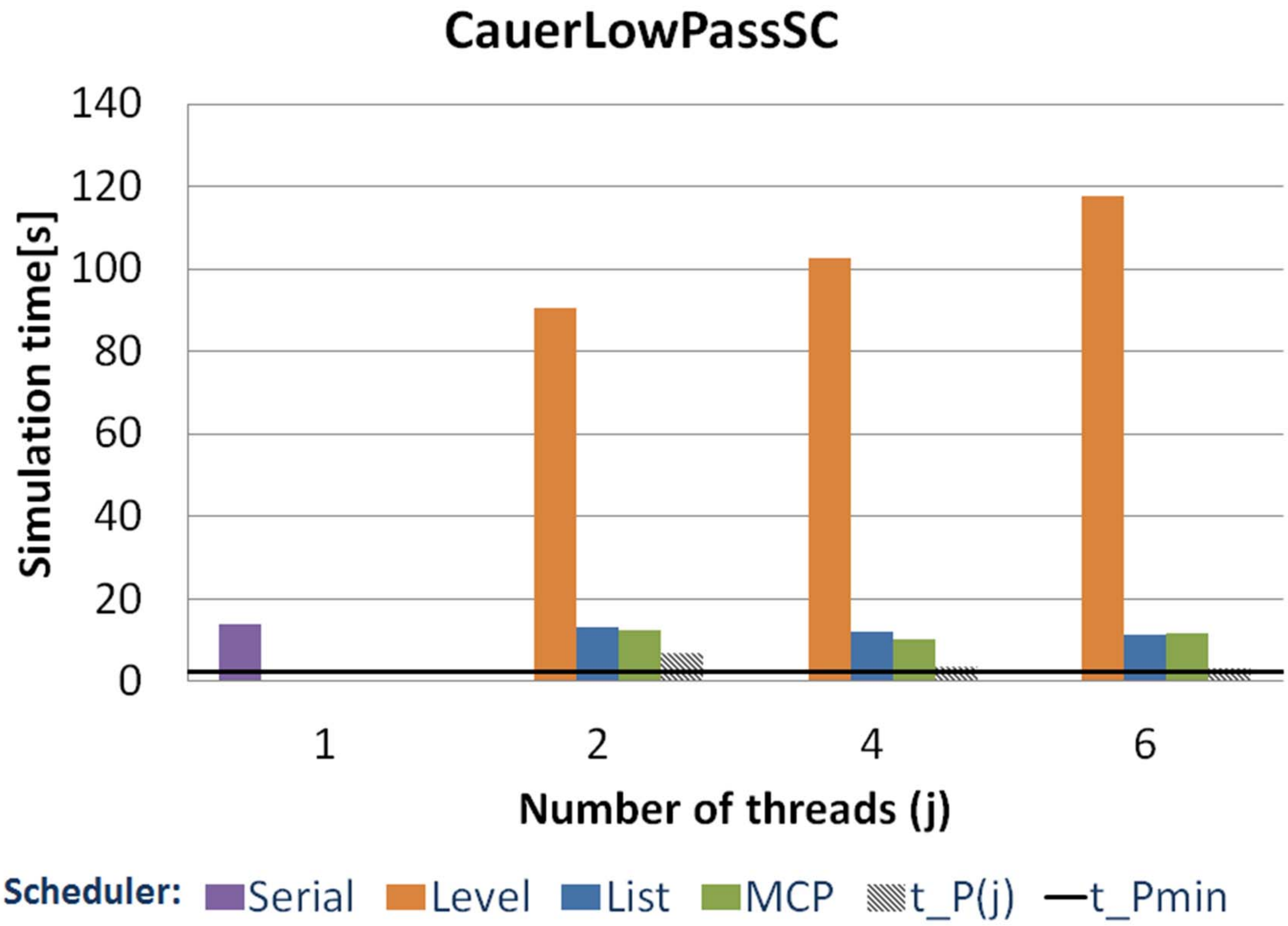


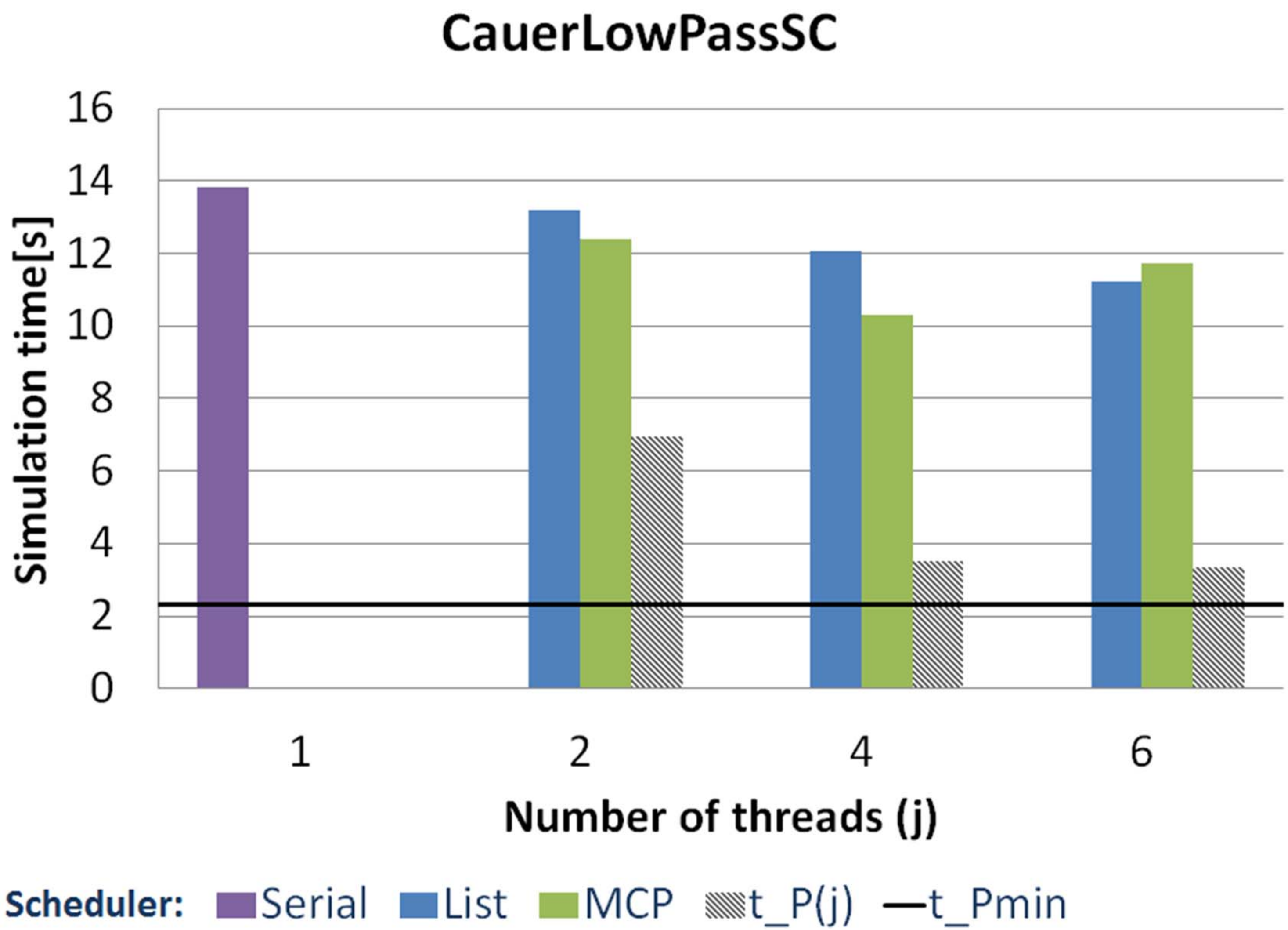
Modelica.Fluid.Examples.ControlledTankSystem.ControlledTanks

# 4. Benchmarks

- **Cauer Low Pass SC**

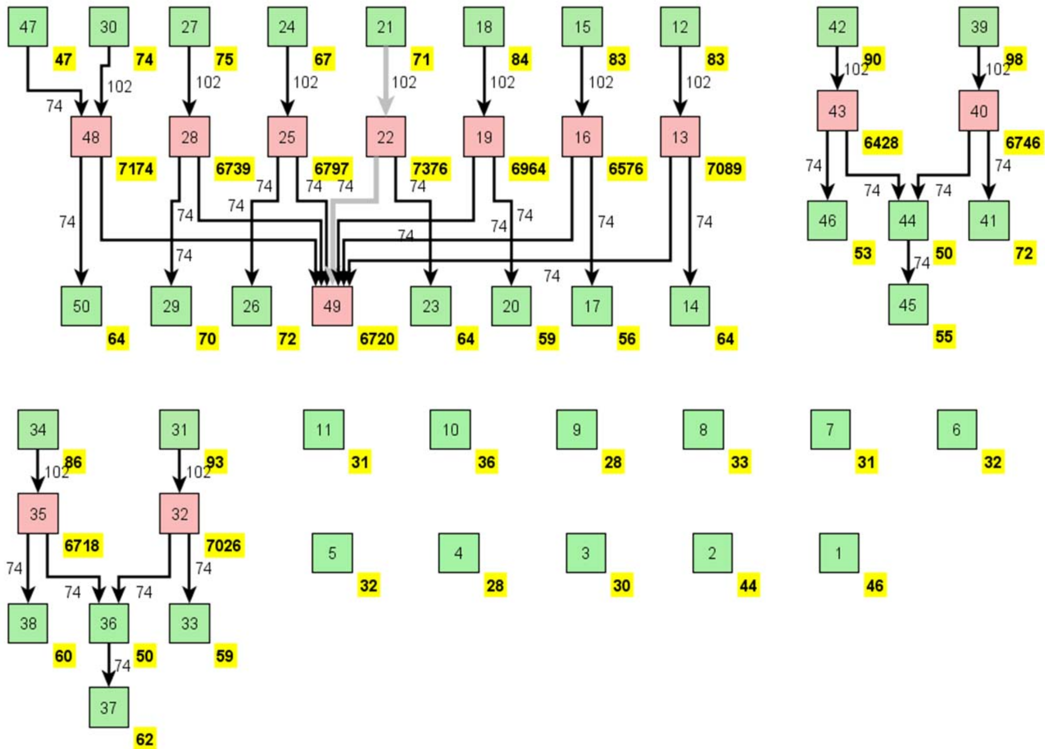



CauerLowPassSC
140
120
100
80
60
40
20
0
Simulation time[s]
1
2
4
6
Number of threads (j)
Scheduler:
Serial
Level
List
MCP
t_P(j)
t_Pmin


CauerLowPassSC
16
14
12
10
8
6
4
2
0
Simulation time[s]
1
2
4
6
Number of threads (j)
Scheduler: Serial List MCP t_P(j) t_Pmin

# Cauer Low Pass SC

- Execution costs evenly distributed, too
- Fast ODE calculation
    - Overhead too big




Modelica.Electrical.Analog.Examples.CauerLowPassSC

# 5. Summary

- Theoretical speedups look promising
- So far good results for fluid models (2.2 with 4 cores)

  → Without optimized code!

- But further work required:
  - Improve parallel code (reduce cache invalidation)
  - Split „big“ tasks
  - Combine with other approaches

TECHNISCHE UNIVERSITÄT DRESDEN

CENTER FOR INFORMATION SERVICES AND HIGH PERFORMANCE COMPUTING

# Equation based parallelization of Modelica models

## Questions?

Linköping, 03/02/2014

DRESDEN concept
Exzellenz aus Wissenschaft und Kultur