Equation based parallelization of Modelica models

Linköping, 03/02/2014
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
2. BLT parallelization

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

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

Incidence matrix

BLT transformation

BLT parallelization
BLT parallelization

Equation based parallelization of Modelica models

\[ u_s \ u_{R1} u_{R1p} u_{R2} \ i_c \ P_{R1} \ \dot{u}_c \ P_{R2} \]

BLT transformation

Task-Graph

\{EquationIndex:VariableIndex\}

Execution costs

Communication costs

22.01.14

Equation based parallelization of Modelica models

Slide 7
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)

```java
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
    {
        ...
    }
}
```
• **List Scheduler** (HLF – heuristic, OpenMP)

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**
Benchmarks

EngineV6

Simulation time [s]

Number of threads (j)

Scheduler: Serial, Level, List, MCP, t_P(j), t_Pmin

22.01.14
Equation based parallelization of Modelica models
Mechanics domain

- Graph contains one "big" task
  - Calculation of accelerations

• Branching Dynamic Pipes
Fluid domain

- Execution costs evenly distributed
4. Benchmarks

- **Cauer Low Pass SC**
Benchmarks

CauerLowPassSC

Simulation time [s]

Number of threads (j)

Scheduler: Serial, Level, List, MCP, t_P(j), t_Pmin

22.01.14
Equation based parallelization of Modelica models
Benchmarks

### CauerLowPassSC

<table>
<thead>
<tr>
<th>Number of threads (j)</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation time [s]</td>
<td>14</td>
<td>12</td>
<td>10</td>
<td>12</td>
</tr>
</tbody>
</table>

**Scheduler:**
- Serial
- List
- MCP
- $t_P(j)$
- $t_{Pmin}$
Cauer Low Pass SC

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

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
Equation based parallelization of Modelica models

Questions?

Linköping, 03/02/2014