Exploiting OpenMP in the Initial Section of Modelica Models
- Work in Progress -

Javier Bonilla\(^{(a)}\), Luis J. Yebra\(^{(a)}\) and Sebastián Dormido\(^{(b)}\)

\(^{(a)}\) CIEMAT-PSA, Centro de Investigaciones Energéticas, MedioAmbientales y Tecnológicas
Plataforma Solar de Almería, Almería, Spain, \{javier.bonilla, luis.yebra\}@psa.es

\(^{(b)}\) UNED, Universidad Nacional de Educación a Distancia, Madrid, Spain, sdormido@dia.uned.es

5\(^{th}\) of September, 2011
Index

1. Introduction
2. OpenMP
3. The DISS test facility
4. Initial Section Parallelization
5. Simulation Results
6. Conclusions and Future Work
Modern equation-based object-oriented (EOO) modelling languages are continuously increasing their expressiveness to model complex systems. However, the required computational effort to simulate is also increasing.

Commonly, EOO models are compiled as single-threaded executables not taking advantage of the newest multi-core processors.

This work describes a straightforward parallelization method in the initial section of the resulting C source code obtained from a Modelica model.
OpenMP - Introduction

OpenMP (Open Multi-Processing) is an application programming interface (API) for multi-platform (Unix and Microsoft Windows platforms) shared-memory parallel programming in C/C++ and Fortran.

<table>
<thead>
<tr>
<th>Version</th>
<th>Year</th>
<th>Languages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1997</td>
<td>Fortran</td>
</tr>
<tr>
<td>1.0</td>
<td>1998</td>
<td>C/C++</td>
</tr>
<tr>
<td>2.0</td>
<td>2000</td>
<td>Fortran</td>
</tr>
<tr>
<td>2.0</td>
<td>2002</td>
<td>C/C++</td>
</tr>
<tr>
<td>2.5</td>
<td>2005</td>
<td>Fortran,C/C++</td>
</tr>
<tr>
<td>3.0</td>
<td>2008</td>
<td>Fortran,C/C++</td>
</tr>
<tr>
<td>3.1</td>
<td>2011</td>
<td>Fortran,C/C++</td>
</tr>
</tbody>
</table>

OpenMP is maintained by the OpenMP Architecture Review Board (ARB).
OpenMP - Features

Key concept - Multithreading

Core elements

- Parallel control directives. Control the flow execution (i.e. parallel).
- Work Sharing. Distribute the code among cores (i.e. parallel for dir.).
- Synchronization. critical, atomic and barrier directives.
- Run-time functions. Set and get run-time information.
La Plataforma Solar de Almería (PSA)

CIEMAT-PSA, a Spanish government research and test center, is the biggest European research center devoted to solar concentrating technologies.
The DISS test facility

The DISS test facility is the first facility built in the world where two-phase-flow steam-water processes in parabolic-trough collectors can be studied under real solar conditions.
Parabolic-Trough Collectors (PTCs)

Main elements

- Parabolic-trough receiver.
- Absorber tube.
DISS Modelica model

Model inputs
- Ambient temperature
- Direct solar radiation
- 1st PTC inlet flow
  - Mass flow
  - Flow temperature
  - Flow pressure
- 11th PTC injector
  - Mass flow
  - Flow temperature
  - Flow pressure

Initialization
- Outlet pressure
OpenMP in Modelica Models

- The DISS test facility

Simulator scheme

DISS project DataBase

Web Application

Sensor Data File

Results

Data File

Results and Comparison

File conversion ("dissTodsu" Application)

Trajectory File

Inputs

Real Data

Initial values (C library )

Parametrizable classes and Parameters

- Transfer Coefficient Model
- Medium Model
- Optical Reflectivity
- Pressure Drop model
- ...........

DISS test facility

DISS Modelica Model

J. Bonilla et al. (CIEMAT-PSA)
OpenMP in Modelica Models

The DISS test facility

Model Calibration

Calibration process

- Exporting the DISS Modelica model to Matlab/Simulink.
- Using the Matlab Genetic Algorithm Toolbox.
- Multi-objective approach, minimizing the output temperature difference in each PTC.

J. Bonilla et al. (CIEMAT-PSA)
Procedure

1. Reference the OpenMP header (omp.h) in the dsmodel.c file (Listing 1).
2. Set the maximum number of threads (omp_set_num_threads) (Listing 2).
3. Manually distribute the sentences among the threads (Listing 3).
4. Compile including the OpenMP library (-fopenmp in GNU compilers).

```c
#include <omp.h>

Listing 1. OpenMP header inclusion.

constant int NCores = 4;
omp_set_num_threads(NCores);

Listing 2. Setting the maximum number of threads.
```
#pragma omp parallel
{
    #pragma omp sections
    {
        #pragma omp section
        {
            ........
            T[0]= getIniValue(iniTime,"Tem PTC1");
            T[1]= getIniValue(iniTime,"Tem PTC2");
            ........
        }
        #pragma omp section
        {
            ........
            p[0]= getIniValue(iniTime,"Pres PTC1");
            p[1]= getIniValue(iniTime,"Pres PTC2");
            ........
        }
        ........
    }
    #pragma omp section
    {
        ........
        r[0]= getIniValue(iniTime,"Rad PTC1");
        r[1]= getIniValue(iniTime,"Rad PTC2");
        ........
    }
    #pragma omp section
    {
        ........
        m[0]= getIniValue(iniTime,"Mflow PTC1");
        m[1]= getIniValue(iniTime,"Mflow PTC2");
        ........
    }
    ........

Listing 3. OpenMP parallel sections.
Simulation Statistic

The tests were performed using an Intel® Core™ i5 CPU M540, 2.53 GHz. Several tests were performed and mean execution times have been calculated.

<table>
<thead>
<tr>
<th>Kind of model</th>
<th>Original</th>
<th>Parallelized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution time (s)</td>
<td>170.727</td>
<td>151.415</td>
</tr>
<tr>
<td>Speedup</td>
<td>1</td>
<td>1.13</td>
</tr>
<tr>
<td>Initialization section exec. time (s)</td>
<td>31.231</td>
<td>11.127</td>
</tr>
<tr>
<td>Initialization section speedup</td>
<td>1</td>
<td>2.80</td>
</tr>
<tr>
<td>Calibration execution time</td>
<td>3 days 9 hours</td>
<td>3 days 4.5 hours</td>
</tr>
</tbody>
</table>
Simulation Result - output DISS field temperature

Calibration

Validation
Conclusions

- OpenMP can be easily used to parallelize the initial section in the resulting C source code obtained from Modelica models.

- A mean speedup of 1.13 was obtained in the whole simulation.

- A mean speedup of 2.80 was obtained in the initial section.

- The gain in speed was useful in simulation and calibration.
Future Work

- **Study** how to implement the use of OpenMP directly in the Modelica code.

- **Study** how to take advantage of OpenMP in the integration process.

- **Take advantage of a 13-node cluster.**

- **Consider the parallelization not only in simulation but in the genetic algorithm calibration.**