Status of OpenModelica Real-Time Simulation Profiling

Martin Sjölund, Linköping University
2011-02-07
3rd OpenModelica Workshop
Linköping, Sweden

OpenModelica 1.5.0 (2010)

```
>> simulate(TanksConnectedPI, ...)

record SimulationResult

    resultFile =
"TanksConnectedPI_res.plt",

    messages = "",

end SimulationResult;


>> timing(simulate(TanksConnectedPI,
...))

0.85
```

- The compiler is silent for a few seconds / minutes / hours and you get the result
- You don't even know what phase is slow

OpenModelica r6077 (2010-09)

```
>> simulate(TanksConnectedPI, ...)
record SimulationResult
    resultFile =
"TanksConnectedPI_res.plt",
    messages = "",
    totalTime = 0.825739127,
    timeFrontend = 0.007130054,
    timeBackend = 0.004097103,
    timeCodegen = 0.019479138,
    timeCompile = 0.493652825,
    timeSimulation = 0.301340639
end SimulationResult;
```

- Added a few real-time clocks so we can show users how long each phase takes
- What if the simulation takes 99% of the time

OpenModelica ~r7700 (2011-01)

```
>> buildModel(TanksConnectedPI,
method="dassl2", ...)

{"TanksConnectedPI","TanksConnectedP
I_init.txt"}

>> system("./TanksConnectedPI -mt")

Time to calculate initial values:
0.000134333 sec.

Total time to do event handling:
7.2713e-05 sec.

Total time to produce the output
file: 0.0109643 sec.

Total time to calculate simulation:
0.0185101 sec.
```

- The -mt (measure time) flag now shows accumulated time for event handling and the output file
- Switching to Matlab output format gives a big performance increase

OpenModelica ~r7800 (2011-01)

```
>> buildModel(TanksConnectedPI,
method="dassl2", ...)

{"TanksConnectedPI","TanksConnectedP
I_init.txt"}

>> system("./TanksConnectedPI -mt")

...

>> system("cat omc_mt.log | head -n
3")

step,time,solver time,limitValue,

0,0.002,7.0561e-05,94,1.2941e-05

1,0.004,1.0562e-05,12,1.675e-06
```

- -mt now also creates omc_mt.log
- Functions, linear / mixed / non-linear blocks execution count+time is shown
- Will change it to a binary format for low overhead real-time profiling

What does it find? (1)

```
class SimpleNonLinear
  Real x = cos(x);
end SimpleNonLinear;
```

```
step,residualFunc1
1, 3, 7.6e-07
...
n, 3, 7.6e-07
```

- Finds potential performance issues even in small models
- The non-linear solver used does not get a jacobian function as input
- The system is executed in every step

What does it find? (2)

```
class ArrayCall
//  {cos(1*time), ...
//  cos(10*time)}
  Real x[10] =
tenCos(time);
end ArrayCall;


Step,tenCos
0, 10, 1.3e-05
...
n, 10, 1.3e-05
```

- The following causes the function to be called 10 times in every step

What does it find? (3)

```
class ArrayCall
  Real x[10];
equation
 x = tenCos(time);
end ArrayCall;
```

```
Step,tenCos
0, 1, 1.3e-06
...
n, 1, 1.3e-06
```

- This will call the function only once every step
- The compiler should be improved as soon as possible