STORM: a Simulation Tool for Real-time Multiprocessor Scheduling Evaluation

Richard Urunuela, Anne Marie Delaplanche, Yvon Trinquet

STORM: a Simulation Tool for Real-time Multiprocessor Scheduling Evaluation

Richard Urunuela, Anne-Marie Dplanche, Yvon Trinquet
IRCCyN, UMR CNRS 6597
University of Nantes
1 rue de la Noe, BP 92101, 44321 Nantes cedex 3, France
{richard.urunuela — anne-marie.deplanche — yvon.trinquet}@irccyn.ec-nantes.fr

Abstract

The increasing complexity of the hardware multiprocessor architectures as well as of the real-time applications they support makes very difficult even impossible to apply the theoretical real-time multiprocessor scheduling results currently available. Thus, so as to be able to evaluate and compare real-time multiprocessor scheduling strategies on their schedulability performance as well as energy efficiency, we have preferred a simulation approach and are developing an open and flexible multiprocessor scheduling simulation and evaluation platform called STORM. This paper presents the simulator on which STORM relies and that is able to simulate accurately the behaviour of those (hardware and software) elements that act upon the performances of such systems.

1. Introduction

Multiprocessor architectures are becoming more and more attractive. From single-core designs with heat and thermal problems, chip makers are now shifting to multicore technologies. While making them decreasingly expensive, they are making very powerful platforms available. Such platforms are desirable for embedded system applications with high computational workloads such as robot controls, image processing, or streaming audio. The additional characteristic of such applications is to meet some real-time constraints; consequently they require predictable performance guarantees from the underlying operating system. This has led to renewed interest in multiprocessor real-time scheduling: scheduling algorithms together with schedulability analysis. So many theoretical results are available but mainly based on simple hardware and software architecture models [2]. These results are difficult to exploit if the assumptions are not satisfied which is the case for modern hardware and software architectures (processor heterogeneity, cache management, inter-processor communications, complex software behaviours, system overheads, etc.).Furthermore, due to the specific application fields where energy efficiency is critical (battery-based embedded systems), power management techniques (Dynamic Voltage and Frequency Scaling or Dynamic Power Management) are inescapable and must be taken into account. Indeed, in such a context, it is very difficult to evaluate and compare scheduling algorithms on their schedulability performance as well as energy efficiency. So we think that a more global approach is needed in order to explore the adequacy between scheduling policies and (hardware and software) architectures. It is not for us a matter of implementing scheduling algorithms on true multiprocessor platform(s) but rather of designing an open and flexible framework able to simulate accurately the behaviour of those (hardware and software) elements that act upon the performances of such systems.

2. The STORM simulator

For the time being, engineering efforts on STORM have concerned its simulator component since it is the retaining element of the platform. For a given problem i.e. a software application that has to run on a (multiprocessor) hardware architecture, this simulator is able to play its execution over a specified time interval while taking into account the requirements of tasks, the characteristics and functioning conditions of hardware components and the scheduling rules, and with the highest timing faithfulness. As shown by Figure 1, the STORM simulator needs the specification of the studied problem as input. It is based on some reference components and their characteristics available in attached libraries. It computes a lot of outputs either user readable in the form of diagrams or reports, or machine readable intended for a subsequent analysis tool. More information is made available in [1]..

Inputs : The system to be simulated is specified in a xml file. As shown by the example of Figure 2, some specific tags delineate the parts of the specification and some specific attributes characterise their elements. As regards the given example, the hardware architecture is composed of two identical processors. Here a reference is made to the

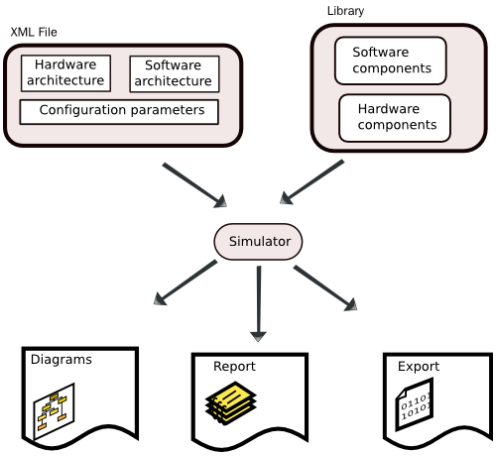

Figure 1. The STORM simulator.

CT11MPCore component which is the identifier of a processor component in library. It is the name of a Java class which implements such a processor. The scheduling strategy is a global pre-emptive EDF one (as for processors, it is the Java class name of this scheduler). Three tasks are present in the software architecture. All of them reference the same periodic task type (Java class) in library. Their own period, first activation date, worst case execution time and deadline (if not equal to the period) are given. Moreover, a data of 10 byte size is declared which is exchanged from task T1 to task T2. Its rate means that 2 executions of task T1 are necessary for task T2 to be allowed to begin. As it will be explained later, whatever it is a matter of processor, task, or scheduling strategy, other types are of course available and those libraries are extensible as often as wished.

```
<SIMULATION duration="200">
    <SCHED className="storm.Schedulers.EDF_NP_Scheduler" ></SCHED>
    <CPUS>
      <CPU className="storm.Processors.CT11MPCore" name ="CPU B" id="10">
      </CPU>
      <CPU className="storm.Processors.CT11MPCore" name ="CPU A" id="20">
      </CPU>
    </CPUS>
    <TASKS>
      <TASK className="storm.Tasks.PTask_NAM" name ="TASK T1" period="5"
            id="1"  WCET="2"  deadline="3"></TASK>
      <TASK className="storm.Tasks.PTask_NAM" name ="TASK T2" period="10"
            id="2"  WCET="5" activationdate="10"></TASK>
      <TASK className="storm.Tasks.PTask_NAM" name ="TASK T3" period="10"
             id="3"  WCET="5"> </TASK>
    <TASKS>
    <DATAS>
      <DATA source="1" destination="2" rate="1" size="10"></DATA>
    </DATAS>
</SIMULATION>
```

Figure 2. . A xml file as input of the simulator.

Graphical User Interface : The STORM simulator offers a user-friendly graphical interface which is composed of various windows and enables him, on the one hand, to order the simulator and on the other hand, to visualize some simul ation results (as it will be discussed in the next section). All those windows can be easily handled through some key combinations or function keys.

Outputs: Over the simulation process, a very large set of events that have punctuated the life of the studied system are recorded with their occurrence date and some other context information. Such events are for example: the release of a job, its start of execution on a processor, its pre-emption of a job, its completion, a frequency processor change, etc. From such execution tracks, either directly or with some further computation, it is thus possible to compute the required real-time metrics for analysis.

3. Conclusion

Presently, the STORM simulator is able to simulate accurately the execution of a set of tasks over a multiprocessor system. Tasks may exhibit various behaviors and be independent or not. Various scheduling strategies are supported too. As a result, the simulator is not only able to state about the schedulability of the studied system but also to characterize its behavior with some measurements for further analysis. It is a flexible, portable, and open tool. At first, the STORM software is freeware under Creative Commons License; subsequently it will be available as an open source. At the present time, our work on STORM is concerned with The energy and power aspects: we are expanding the hardware component library with processors supporting DVFS and DPM; The memory architecture modelling: since memory time access can act significantly upon the global performances of a multiprocessor system, we are investigating how to model the behaviour of (different levels of) cache and external memory and in particular the overhead induced by their management; The evaluation process automation: we are defining a simulation language to drive the experiments. Thus the user would be able to describe a domain for the simulation inputs and to specify the metrics to be measured as a result. Then the tool would automatically build a statistically representative set of compliant test cases, run their simulations, and output the statistical results.

References

[1] http://storm.rts-software.org/. may 2009.

[2] J. H. Anderson. *Handbook of scheduling:Algorithms, Models, and Performance Analysis*. J.Y-T Leung, Chapman & Hall/CRC, 2004.