Towards Energy-aware Multiprocessor Hierarchical Scheduling of Real-time Systems

Nima Moghaddami Khalilzad[1], Juri Lelli[2], Giuseppe Lipari[3], Thomas Nolte[1]
[1]MRTC/Mälardalen University, Sweden
{nima.m.khalilzad, thomas.nolte}@mdh.se
[2]Scuola Superiore Sant'Anna, Italy
j.lelli@sssup.it
[3]Scuola Superiore Sant'Anna, Italy and LSV - ENS Cachan, France
g.lipari@sssup.it

***Abstract*—Multiprocessor platforms are becoming increasingly more popular. Providing more computation capacity on a single hardware platform, multiprocessors make it possible to integrate previously federated real-time systems onto a single platform. Multiprocessor hierarchical scheduling techniques provide the ground for composing real-time components, while guaranteeing the timing correctness of the composed system. A considerable deal of compositional real-time systems are embedded systems that operate on battery power. In such systems, reducing the power consumption is of paramount importance to increase the system lifetime.**

In this paper, we present our idea on reducing the energy consumption when performing hierarchical scheduling on multiprocessors. We formulate the problem, present the model and sketch the outline of the solution. Finally, we present a number of challenges which will be addressed in our work.

I. Introduction

Hierarchical scheduling is a technique used for composing real-time time software components on a shared underlying processor platform [1], [2]. In this technique, based on the processor requirements of each component, a sufficient portion of the processor capacity is assigned to the component. The processor portions are often represented using a processor supply model which abstracts the amount of processor capacity available for the component.

The Multiprocessor Periodic Resource (MPR) Model [5] is a processor supply model that abstracts the processor supply of underlying multiprocessor platforms. In this model, each component is assigned a share of the multiprocessor platform. The MPR model specifies a common period P, total budget Q and maximum number of available processor at each time m. However, the exact allocation of the processors to the components is not specified in this model. All assignments that are compliant with the MPR interface are allowed to be utilized by the component during run-time.

On the other hand, energy-aware scheduling is widely studied for flat systems, i.e non-hierarchical systems (e.g., [3], [4]). However, energy-aware scheduling is also applicable in hierarchical scheduling when the total processor capacity is not utilized by the components.

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 246556, and the Swedish Research Council (Vetenskapsrådet) under the project ARROWS.

In this paper, assuming that the underlying multiprocessor platform allows us to modify the number of active processors as well as their operation frequencies during run-time, we want to (i) model the relation between the processor frequency and the component's processor requirements (ii) provide a light-computation algorithm for selecting the most energy efficient component interfaces during run-time.

We are aiming to utilize both Dynamic Voltage Scaling (DVS) and Dynamic Power Management (DPM) techniques for managing the energy consumption. Hence, the processor speeds and the number of active processors are altered during run-time. To this end, we will first introduce an extended version of MPR model called Energy-aware MPR (EMPR) model that suits our goal. Thereafter, we will give an outline of our online algorithm which reduces the EMRP model to the MPR model.

II. System model

We assume multiprocessor systems with M processors in which a number of real-time components ($\mathcal{A}^l$) are composed on the same multiprocessor platform. The components may join and leave the system during run-time.

A. Processor speed model

We assume the processor frequencies and thus the processor speeds are discrete values and know a priori. The set of all available speeds are denoted using S. The processor speed is calculated through dividing the processor frequency f by the maximum processor frequency f_{max}:

$$s = \frac{f}{f_{max}}. \quad (1)$$

B. Component model

Each real-time component is composed of n implicit deadline periodic tasks $\{\tau_1, \ldots, \tau_n\}$. Tasks have a constant arrival time T_i, however depending on the processor speed s tasks will run for $C_i(s)$ time units at each arrival.

C. Execution time model

For modeling the task execution times, we use a similar approach introduced by Marinoni and Buttazzo in [6]. In this model, the execution time is a function of the processor speed:

$$C_i(s) = \frac{\phi_i C_{i_{max}}}{s} + (1 - \phi_i) C_{i_{max}}, \qquad (2)$$

where $C_{i_{max}}$ and ϕ_i are the execution time at the maximum speed and the percentage of the task code that scales with the processor speed. This parametric execution time captures both the part of task execution which scales with the processors frequency e.g., computation intensive task code, and the part that does not scale with the processor frequency e.g., memory access code.

D. Component demand model

The component demand is calculated based on the parametric execution times of the component's inner tasks ($C_i(s)$). The demand bound function ($\mathtt{dbf}^s(t)$) represents the worst case processor demand of the tasks in a given time interval t when the component's allocated processors are operating at speed s.

E. Resource supply model

Energy-aware Multiprocessor Periodic Resource (EMPR) model $\Gamma = \langle P, Q(s), m \rangle$ indicates that when operating at speed s, at each P time units, m processors collectively provide $Q(s)$ time units to the corresponding component.

The processor supply to component $\mathcal{A}^i$ is calculated using the component's EMPR interface Γ_i. We will use the supply bound function ($\mathtt{sbf}^s(t)$) for calculating the supply to the components. The $\mathtt{sbf}^s(t)$ specifies the minimum processor supply to each component in a given time interval t when the component's allocated processors are operating at speed s. Assuming that the component periods are known, we will provide an analysis to derive the minimum number of processors m and also minimum amount of budget Q that a component can guarantee the schedulability of its inner tasks. The budget that is derived using this analysis should fulfill the following condition:

$$\forall t \qquad \mathtt{sbf}^s(t) \geq \mathtt{dbf}^s(t).$$

This analysis is performed offline and the result is saved in the memory as a table to be used by the resource manager. Table I illustrates an example parametric budget for a system with four processors where the processors can operate in three processor speeds ($S^1 > S^2 > S^3$). At each processor speed s^j, $Q(s)$ represent the minimum budget that guarantees the schedulability of the component's inner tasks. The table indicates that the component requires two processors at speed s^1 and s^2, while it requires three processors at speed s^3.

At each energy management event, we will run our resource manager algorithm (explained in Section III) and we will choose one of the records of this table for each component.

m \ s	s^1	s^2	s^3
1	-	-	-
2	$Q(s^1)$	$Q(s^2)$	-
3	-	-	$Q(s^3)$
4	-	-	-

TABLE I: An example of the parametric budget in EMPR.

III. Resource manager algorithm

The input to the algorithm is a number of EMPR interfaces Γ_i. The steps that the algorithm will take is as follows.

1) The algorithm first derives the minimum number of processors (M') that the can guarantee the schedulability of the components.
2) Afterwards, the algorithm chooses the minimum processor speed (s') in which all components are schedulable. Hence, for each component one record from its budget table will be selected. Therefore, at the end of this step, the EMPR model is reduced to the MPR model.
3) Switches off $M - M'$ processors.
4) Switches the processor speeds to s'.

IV. Challenges

We will address the following challenges in our proposed solution.

First of all, we will identify some points in time that the resource manager algorithm should be triggered. For instance, the resource management can be performed when a new component is joining or leaving the system.

Secondly, we would like to make it possible for the system to change its operation mode, i.e the number of active processors and their speed, while the components are running. In doing so, one part of the task execution will be performed in the first operation mode while the rest will be executed in the second mode. Hence, we should provide a safe mode change protocol.

References

[1] Z. Deng and J. W.-S. Liu, "Scheduling real-time applications in an open environment," in *Proceedings of the 18th IEEE Real-Time Systems Symposium (RTSS'97)*, December 1997, pp. 308–319.

[2] G. Lipari and S. Baruah, "A hierarchical extension to the constant bandwidth server framework," in *Proceedings of the 7th IEEE Real-Time Technology and Applications Symposium (RTAS'01)*, May 2001, pp. 26–35.

[3] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics," in *Proceedings of the 13th Euromicro Conference on Real-Time Systems (ECRTS'01)*, June 2001, pp. 225–232.

[4] W. Kim, D. Shin, H.-S. Yun, J. Kim, and S.-L. Min, "Performance comparison of dynamic voltage scaling algorithms for hard real-time systems," in *Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02)*, September 2002, pp. 219–228.

[5] I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering of multiprocessors," in *Proceedings of the Euromicro Conference on Real-Time Systems, (ECRTS'08)*, July 2008, pp. 181–190.

[6] M. Marinoni and G. Buttazzo, "Elastic DVS management in processors with discrete voltage/frequency modes," *IEEE Transactions on Industrial Informatics*, vol. 3, no. 1, pp. 51–62, 2007.