Probabilistic Application Interfaces for Hierarchical Scheduling

Nima Moghaddami Khalilzad, Meng Liu, Moris Behnam, Thomas Nolte
MRTC/Mälardalen University, Sweden
nima.m.khalilzad@mdh.se

***Abstract*—The concept of hierarchical scheduling is widely used for scheduling complex real-time systems that are composed of a number of components. In the conventional hierarchical scheduling framework, targeting hard real-time systems, the size of processor capacities assigned to components is derived based on the worst case execution times of tasks. In this paper, we present our ongoing work on bringing the notation of probabilistic execution times in the context of hierarchical scheduling and deriving probabilistic component processor requirements. When dealing with soft real-time systems, this approach can eliminate the unaffordable pessimism that exists in worst-case timing analyses.**

I. Introduction

Real-time embedded systems are increasingly growing in complexity. Previously separated independent systems are integrated to form new and more complex systems. To deal with such systems, compositional hierarchical scheduling techniques provide a modular approach in which the timing properties of federated real-time systems are inferred from the timing characteristics of their components [1], [2], [3]. The conventional approach in analyzing the timing properties of hierarchically scheduled systems is to derive the worst case processor demand of components (applications) based on the worst case processor demand of their inner sub-components (tasks). The Worst Case Execution Time (WCET) of tasks are the basic blocks of the analysis which are used in calculating the processor demand of applications. A processor capacity that can guarantee the worst case processor demand of the corresponding application is assigned to each application.

While deterministic timing analysis based on WCET is used in hard real-time systems, in soft real-time systems stochastic analysis based on probabilistic execution time distributions can reduce the pessimism and hence the processing capacity can be better utilized [4], [5]. In essence, the flexibility of soft real-time systems in allowing a limited amount of timing violations makes it possible to use probabilistic models of execution time instead of WCET. Hence, in the probabilistic domain the type of guarantees that the analysis provides is not deterministic anymore, i.e, instead of studying the schedulability of the system, the probability of having a deadline miss ratio below a certain level is of interest in the probabilistic analysis. In soft real-time systems, the Quality of Service (QoS) can be inferred from the probabilistic guarantees.

In our work, we bring the probabilistic analysis to the context of a hierarchical scheduling framework. This approach is especially beneficial in open real-time systems in which applications are added and removed during run-time. For instance, when the processor is overloaded, we can leverage the probabilistic model of applications' processor requirements to decide the processor capacities assigned to the applications such that the overall timing violations are minimized. In fact, our aim is to propagate the uncertainty in task execution times to the application interfaces and to utilize this information at the integration phase.

The research leading to these results has received funding from the Swedish Research Council (Vetenskapsrådet) under the project ARROWS.

Previous works in the area of probabilistic analysis mainly target flat systems, i.e, non-hierarchical systems. The closest work to our work is presented by Santinelli *et al.* which introduced a component based framework for probabilistic analysis of real-time systems [6]. Our work is different from theirs in that we assume deterministic processor provision through conventional reservation based scheduling techniques, and we use the probabilistic processor requirements of applications to derive the reservation sizes assigned to them. While in [6] the probabilistic component demands and probabilistic processor provisioning is used to analyze the probability of schedulability.

II. System model

We assume a single processor system consisting of N applications. The applications are scheduled on the processor using a "global scheduler". Applications, in turn, are responsible for scheduling their inner tasks. Although we start by investigating single processors, our ultimate goal is to extend the work and address multiprocessors.

A. System development model

We target a component based software development model in which the following two roles are defined: (i) application developer (ii) system integrator. The application developer is responsible for developing real-time tasks and selecting an appropriate scheduling policy for them. Then, the application requirement is abstracted using a number of interface parameters. The system integrator on the other hand, receives a number of applications and he/she is responsible for integrating the applications such that the requirements specified in the interface parameters are respected. The integrators' task involves assigning sufficient processor capacities to applications and choosing an appropriate global scheduler.

B. Task model

In our work we assume a sporadic task model in which task τ_i is represented with the following parameters: minimum inter arrival time T_i, deadline D_i and execution time $\mathcal{C}_i$. The execution time is assumed to be a random discrete variable with a known Probability Function (PF) $f_{C_i}(.)$.

C. Application interface

Each application A_j is a set of n_j sporadic tasks $\{\tau_1^j, \cdots, \tau_{n_j}^j\}$. Applications express their processor requirements using the following two parameters called interface parameters: P_j and $\mathcal{Q}_j$ where P_j is the application period,

while $\mathcal{Q}_j$ is a random discrete variable with a known PF $f_{\mathcal{Q}_j}(.)$ which denotes the amount of processor requirement of A_j every P_j time units. The application interface is provided by the application developer to the system integrator.

D. Server model

The processor capacity becomes available to the applications through periodic servers compliant with the periodic resource model introduced in [3]. The periodic servers are expressed using the following parameters: Π_j and Θ_j. The periodic servers provide Θ_j time units of the processor time to application A_j each Π_j time units. In fact, the system integrator designs servers based on the application interfaces.

Although we are targeting soft real-time applications, it is still beneficial to use hard reservations such as periodic servers to (i) handle the system complexity through processor partitioning (ii) keep the timing isolation among applications (iii) reason about the system performance at design stage.

E. From a task set to application interface

In a deterministic hierarchical framework, the application interfaces often express the minimum amount of processor capacity that can guarantee the schedulabilty of the application's inner tasks. Hence, the system integrator has to choose a server that provides a processor capacity at least equal to the application requirement. In fact, the server interfaces (Π, Θ) express both processor requirement of applications and processor supply of the servers. In contrast, the application requirement and server supply are two separate interfaces in our probabilistic hierarchical scheduling framework.

Moreover, the processor requirement of the applications is often abstracted using a demand bound function $\texttt{dbf}(t)$ which denotes the maximum processor time needed by the task set inside one application in time interval t. On the other hand, the processor supply through servers is often abstracted using a supply bound function $\texttt{sbf}(t)$ that denotes the minimum amount of processor capacity provided to the application in time interval t. The application developer has to provide an interfere such that the minimum processor supply through that interface fulfills the following inequality for all time intervals:

$$\texttt{sbf}(t) \geq \texttt{dbf}(t). \tag{1}$$

In the deterministic analysis, $\texttt{dbf}(t)$ returns one value for each given input t, and the objective is to find the minimum $\texttt{sbf}(t)$ (and hence the optimal server parameters) that fulfills the above inequality. However, in our probabilistic framework, we redefine the demand bound function ($\texttt{dbf}^*$) such that for each input t the output is a probability function with a known distribution that expresses the probability distribution of the processor demand in interval t:

$$\texttt{dbf}^* : t \rightarrow f_{\mathcal{D}}(.),$$

where $f_{\mathcal{D}}(x) = \mathbb{P}(\texttt{dbf}(t) = x)$. Besides, the objective of our probabilistic framework is to understand for a given server budget Θ_j to what extent Inequality 1 holds.

The probabilistic interface is derived from the probabilistic execution times. We suggest the following algorithm for calculating the probabilistic budgets. In the algorithm, Q_i is the application budget that is under analysis. The goal is to derive the probability of application A_j being schedulable given that $\Theta_j = Q_i$. We intend to analytically derive a range for Q_i. t is the time interval that has to be taken into account in the analysis. Similarly, we will bound t to bound the number of iterations in the algorithm. For bounding t, we will use techniques analogous to the ones used in deterministic compositional analysis which use the scheduling policy and task parameters for confining t.

Algorithm 1: calculating $f_{\mathcal{Q}}(.)$

```
1: for all Q_i ∈ [Q^min, Q^max] do
2:    for all t_κ ∈ [0, t^max] do
3:       f_D = dbf*(t);
4:       p(Q_i, t_κ) = f_D(Q_i);
5:    end for
6:    f_Q(Q_i) = min(p(Q_i, t_κ));
7: end for
```

In Line 3 of Algorithm 1, the probability distribution of the demand bound function for a given time interval t is calculated. In Line 4, the probability of the demand being equal to the budget under investigation (i.e., Q_i) when the time interval is equal to t_κ is stored.

F. From application interfaces to periodic servers

Given a set of application interfaces, the system integrator has to derive a set of periodic servers such that the overall deadline miss ratio is minimized. When the system is underloaded, the servers can be designed according to the applications' worst case processor requirements. However, when the system is overloaded, some applications have to receive smaller processor portions than their worst case requirement. Hence, the probabilistic budget $\mathcal{Q}$ provides the system integrator with valuable information to decide which application that should be sacrificed such that minimal damage is imposed to the system.

This mechanism can also be used by an online admission controller which decides whether or not a new application should enter the system and if yes how should the overall processor capacity be redistributed.

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] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in *Proceedings of the 24th IEEE Real-Time Systems Symposium, (RTSS'03)*, December 2003, pp. 2–13.

[4] J. L. Díaz, D. F. García, K. Kim, C.-G. Lee, L. Lo Bello, J. M. López, S. L. Min, and O. Mirabella, "Stochastic analysis of periodic real-time systems," in *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02)*, December 2002, pp. 289–300.

[5] A. Burns, G. Bernat, and I. Broster, "A probabilistic framework for schedulability analysis," in *Embedded Software*. Springer Berlin Heidelberg, 2003, vol. 2855, pp. 1–15.

[6] L. Santinelli, P. Yomsi, D. Maxim, and L. Cucu-Grosjean, "A component-based framework for modeling and analyzing probabilistic real-time systems," in *Proceedings of the 16th IEEE Conference on Emerging Technologies Factory Automation (ETFA'11)*, September 2011, pp. 1–8.