A Framework for Compositional and Hierarchical Real-Time Scheduling

Shanmuga Priya Marimuthu Samarjit Chakraborty
Department of Computer Science
National University of Singapore
{shanmuga, samarjit}@comp.nus.edu.sg

Abstract

Hierarchical scheduling frameworks have lately received a lot of attention for component-based design of complex real-time systems. The specification of the resource reservation policy play a dominant role in such frameworks. In this context, the notion of real-time virtual resources results a very flexible representation of resource reservation schemes. We intend to combine the advantages offered by virtual resource scheduling with very general event models specified using real-time calculus. Our proposed framework permits resource partitioning to be extended to multiple levels and handles a wider range of scheduling algorithms and task models. In addition, it allows the handling of data dependencies between tasks from different task groups in the hierarchy.

1 Introduction

Hierarchical scheduling is increasingly being investigated as a flexible resource allocation mechanism for component-based design of complex real-time systems. Towards this, a component [6] $C(W, R, A)$ is characterized by the resource R, the workload W and the scheduling algorithm A. The *resource demand* of a component represents the collective resource requirements that its workload set W requests under a scheduling algorithm A. The *demand bound function* $dbf_A(W, t)$ of a component calculates the maximum possible resource demand that workload W requests under scheduling policy A for a time interval of length t. *The resource supply* of a resource R calculates the minimum possible resource capacity that R provides. The *supply bound function* $sbf_R(t)$ of a resource R calculates the minimum possible resource supplies that R provides during a time interval of length t. A resource R is said to satisfy the resource demand of W under A if $dbf_A(t) \leq sbf_R(t)$. In other words, a component is schedulable if its resource supply satisfies its resource demand.

In such a compositional scheduling framework, a major issue is to define a interface model in order to specify the collective real-time requirements of a component. Shin and Lee [5] developed a compositional scheduling framework using the periodic resource model as a scheduling interface model. When a component exports its periodic scheduling interface to the system, the component could be treated as a single periodic task at the system-level. The results presented in [5] were used in another framework [6] based on the bounded-delay resource model. The utilization bounds for the bounded delay resource model were derived for EDF and RM scheduling policies.

One of the drawbacks of these previous approaches is that the event models and scheduling algorithms are restrictive. They accommodate only standard event models like periodic or sporadic. To accommodate more generalized models, the event models are again abstracted as periodic event models for the purpose of analysis, which leads to loss of accuracy and pessimism in timing estimates. The schedulability conditions are derived for EDF and RM algorithms, although many other scheduling policies could be used in practice. In hierarchical schedulability analysis, the child scheduling models abstract the resource requirement depending on their workload demands. This resource requirement then forms the workload demand of the parent resource model and the parent model has to find the resource requirement to satisfy this demand. In this manner, finally the solution to the resource model at the physical-level, which would be sufficient to schedule all the applications, is derived. Thus the problem is to derive the resource requirement at the physical level, given the workloads and scheduling algorithms, using a bottom-up approach.

We would like to incorporate both top-down and bottom-up approaches to schedulability analysis in such a hierarchy. It would often be desirable to determine if a given physical resource could be partitioned according to some scheduling policy to schedule a set of workloads. Thus a top-down approach in the hierarchy would be required for such analysis. Moreover, it would be desirable to reuse unused computation capacity, if any, within the partition to schedule other task groups. In this paper we propose a hierarchical scheduling framework based on the real-time calculus framework presented in [2]. We utilize the generalized event and resource models from real-time calculus for the hierarchical framework thus accommodating a wider range of workloads and scheduling algorithms. We also handle data dependencies between task groups in the hierarchy.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

2 Resource models

The notion of real-time virtual resource was introduced in [4] for abstracting resource sharing where a physical resource such as a CPU is shared by application task groups that are subject to different types of timing constraints. Sharing is enforced by some kind of partitioning scheme that time-multiplexes the physical resource among the different application task groups, with the goal that each application task group may be programmed as if it had dedicated access to a physical resource. Tasks within the same application task group are scheduled by an application-task-level scheduler that is specialized to the real-time requirements of the tasks in the group. The problem of preventing interference between task groups cannot, in general, be solved by simply reserving a fixed portion of the physical resource's time for a task group. This is because different task groups may have different time granularity requirements. In contrast, the abstraction of a real-time virtual resource guarantees that a task group that has been promised a fraction α of a resource will receive at least $\alpha \times L$ units of the resource's time in any interval of length $L + \delta$ for any value of L, where δ is a design parameter, a non-negative number specified by the programmer.

Bounded-delay resource model: The real-time virtual resource abstraction describes a resource model that bounds the output jitter of a task group. This gives rise to the notion of the bounded-delay resource model. Such a model is specified as $\Phi(\alpha, \delta)$, where α is the availability factor (resource capacity) $(0 < \alpha \leq 1)$ and δ is a partition delay bound $(0 \leq \delta)$. A bounded-delay model $\Phi(\alpha, \delta)$ with the supply function $S_\Phi(t)$ (specifying the resource supply from time 0 to t) satisfies the following property:

$$\begin{array}{l} \forall t_1, \quad \forall t_2 \geq t_1, \quad \forall d \leq \delta, \\ (t_2 - t_1 - d)\alpha \leq (S_\Phi(t_1) - S_\Phi(t_2)) \leq (t_2 - t_1 + d)\alpha \end{array} \quad (1)$$

The supply bound function $sbf_\Phi(t)$ of a bounded-delay model specifies the minimum resource supply over any interval length t:

$$sbf_\Phi(t) = \begin{cases} \alpha(t - \delta) & \text{if } (t \geq \delta) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Service curve based resource model: The resource model based on real-time calculus is characterized by the service function $C(t)$ analogous to the supply function $S_\Phi(t)$ mentioned above. The *service function* $C(t) \geq 0$ represents the amount of computation that could be delivered up to time t and is specified as follows.

Definition 1 (Service function) A resource can be described by a service function $C(t)$ denoting the number of events that can be served during the interval $[0, t)$.

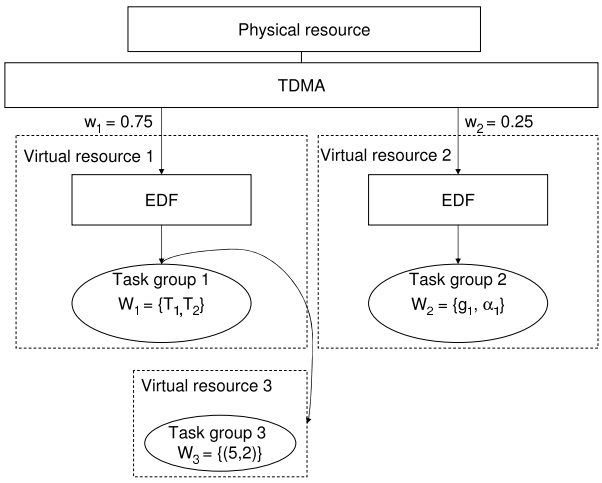

Figure 1. Hierarchical scheduling.

Definition 2 (Service Curve) The service curve $\beta = (\beta^l, \beta^u)$ of a service function C is defined as:

$$\beta^l(t - s) \leq C(t) - C(s) \leq \beta^u(t - s), \forall s, t : 0 \leq s \leq t$$

$\beta^u(\Delta)$ and $\beta^l(\Delta)$ can be interpreted as the minimum and maximum computing resource available within any time interval of length Δ, respectively. Therefore, given the service function C, we have $\beta^l(\Delta) = \min_{u \geq 0}\{C(\Delta + u) - C(u)\}$ and $\beta^u(\Delta) = \max_{u \geq 0}\{C(\Delta + u) - C(u)\}$.

Similarly, the triggering of task graphs also need not be restricted to periodic or sporadic event models. Instead, they can be defined using similar *arrival curves* (see [2] for details).

3 Hierarchical scheduling

The main advantage of using the above service curve based resource model is that it allows the modeling of the resource demand for a variety of task models, e.g. the recurring real-time task model, the stream-based task model (see [1, 3]) and others. Many of these models do not naturally lent themselves to be modeled using the bounded-delay resource model.

An important difference between the previously studied hierarchical virtual resource model and our framework is that it is possible to calculate the computation capacity that remains from a partition after processing the workload scheduled under that partition. If dbf_W is the demand imposed by workload W on the resource with minimum supply of β^l, then the remaining computation capacity $\beta^{l\prime}$ is given by:

$$\beta^{l'}(t) = \sup_{0 \leq \lambda \leq t} \{\beta^l(\lambda) - dbf_W(\lambda)\} \quad (3)$$

This computation capacity can be used up by another workload, possibly even in a different partition. Thus,

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

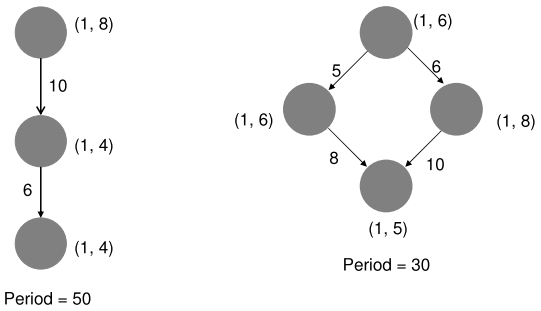

Figure 2. Two recurring real-time tasks (see [1]).

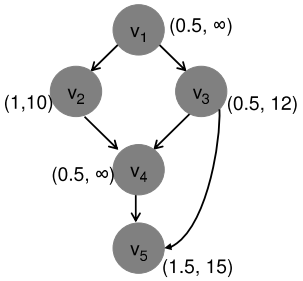

Figure 3. A stream-based task graph (see [3]).

there could be workloads and partitions scheduled on the same level. While the real-time virtual resource framework requires every task group to be scheduled on a partition created by abstracting its demand, our framework allows task-groups to be scheduled on partitions in the same level. Due to space restrictions, we illustrate our framework using the following example.

Example: Consider the resource partitions VR_1 and VR_2 shown in Figure 1. Suppose that the top-level TDMA scheduler assigns weights $w_1 = 0.75$ and $w_2 = 0.25$ to the two partitions, and it has a period P that is infinitesimally small. Given that $W_1 = \{T_1, T_2\}$ and $W_2 = \{(G_1, \alpha_1)\}$, where T_1 and T_2 are two recurring real-time tasks shown in Figure 2 and G_1 is a stream based task graph shown in Figure 3. G_1 is triggered by the arrival function α_1 shown in

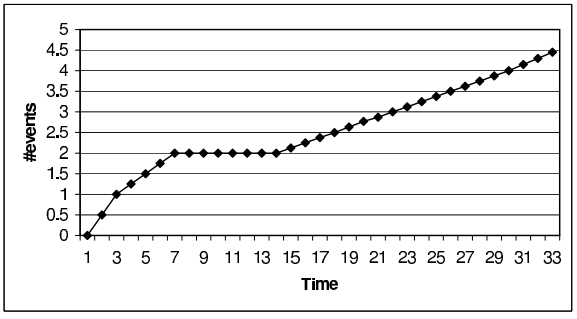

Figure 4. Arrival function defined using piecewise linear segments, with three pieces starting at t = 0, 2, 6, 13 with slopes 30, 15, 0 and 5 respectively, triggering the arrival of the stream-based task model in Figure 3.

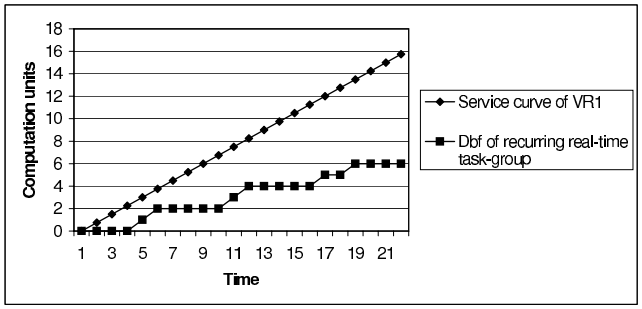

Figure 5. Schedulability of W_1 under VR_1.

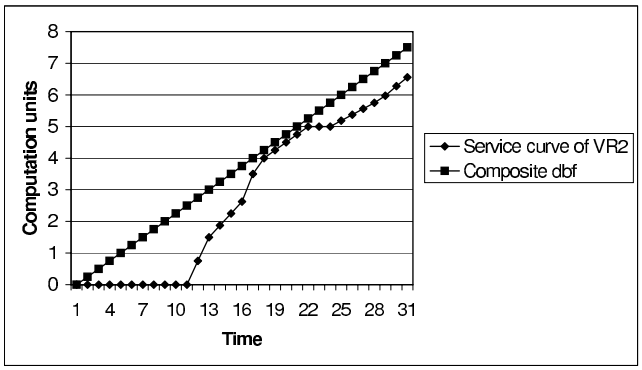

Figure 6. Schedulability of W_2 under VR_2.

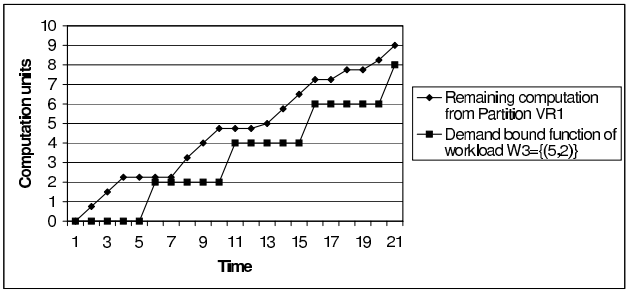

Figure 7. Schedulability of W_3 using the unused computation capacity from VR_1.

Figure 4. All the execution requirements and deadlines are assumed to be specified in time units. We intend to schedule W_1 on VR_1 and W_2 on VR_2. The *demand bound functions* of these workloads are calculated based on algorithms for the respective task models. The demand bound functions and the supply bound functions for partitions VR_1 and VR_2 are shown Figures 5 and 6. As evident from these figures, VR_1 easily schedules the task group W_1 with sufficient scope to reclaim the unused computational bandwidth, but VR_2 is just enough to schedule the task group W_2. The unused capacity from VR_1 can be reclaimed following Eq. (3) to form partition VR_3. A task group W_3 consisting of a single periodic task $W_3 = \{(5, 2)\}$ can be scheduled on this reclaimed resource as shown in Figure 7.

Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'06)
0-7695-2676-4/06 $20.00 © 2006 IEEE

This example shows how our framework can model heterogeneous scheduling policies and a mix of virtual resources and tasks at any level of the hierarchy.

In general if there are n partitions of a resource β^l, with task groups each with demand dbf_{W_i} scheduled on them, then the remaining computation capacity of the parent partition can be computed as:

$$\beta^{l'}(t) = sup_{0 \leq \lambda \leq t}\{\beta^l(\lambda) - \sum_{i=1}^{n} dbf_{W_i}(\lambda)\}$$

This is in contrast to the real-time virtual resource framework, where the schedulability of partitions is done by considering the resource allocations to the child scheduling models, as the workload of the parent model. In this framework, we would like to consider the cumulative workload demand imposed on the child scheduling models to be the workload demand of the parent scheduling model in order to calculate the unused computation capacity. In this manner we avoid over-provisioning of resources.

4 Handling data dependencies among tasks

Our framework can be used to model both inter- as well as intra-group task dependencies. We illustrate this using an example scheduler shown in Figure 8.

Example: Task T_{10} is data dependent on T_9, which in turn is data dependent on task T_8. While tasks T_8 and T_{10} are scheduled on one partition using resource β_{2-1}, task T_9 is scheduled on a different partition using resource β_{4-1} that is also shared by tasks T_{11} and T_{12}. The arrival curves α triggering these tasks and their grouping under respective partitions is shown in Figure 9. All tasks are assumed to be represented as stream based task models. Task T_8 is triggered by an arrival curve α_8 and the processed stream from T_8, denoted by α'_8, is the arrival curve that triggers T_9. The processed stream from T_9 denoted by α'_9 is the arrival curve for T_{10}. The composite demand bound functions (see [3] for details) of tasks T_8 and T_{10} denoted by $dbf^C_8(t)$ and $dbf^C_{10}(t)$ are determined using α_8 and α'_9 respectively, and the schedulability condition is given by $(dbf^C_8(t) + dbf^C_{10}) \leq \beta_{2-1}(t),\ \forall t \geq 0$. Similarly, the composite demand bound functions of tasks T_9, T_{11} and T_{12} denoted by $dbf^C_9(t)$, $dbf^C_{11}(t)$ and $dbf^C_{12}(t)$ are determined using α'_8, α_{11} and α_{12} respectively and the schedulability condition in this case is:

$$(dbf^C_9(t) + dbf^C_{11}(t) + dbf^C_{12}(t)) \leq \beta_{4-1}(t),\ \ \forall t \geq 0$$

Tasks T_4 and T_5 are involved in a precedence relationship such that task T_5 is data dependent on T_4. These two tasks along with T_3 and T_6 are scheduled on an EDF scheduler with resource β_{1-1}. As shown in Figure 10, the composite demand bound function of each of these tasks denoted by $dbf^C_3(t)$, $dbf^C_4(t)$, $dbf^C_5(t)$ and $dbf^C_6(t)$ are computed using α_3, α_4, α'_4 and α_6 respectively, and the schedulability condition is given by: $\sum_{i=3}^{6} dbf^C_i \leq \beta_{1-1},\ \ , \forall t \geq 0$. Other schedulers can be modeled using a similar approach.

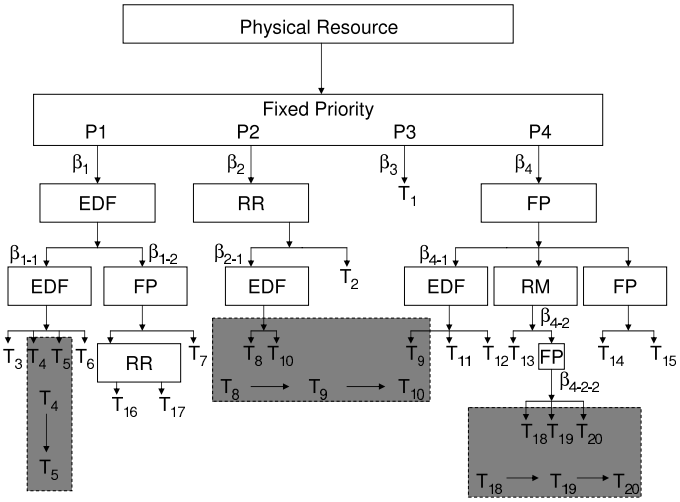

Figure 8. A scheduler with complex task dependencies.

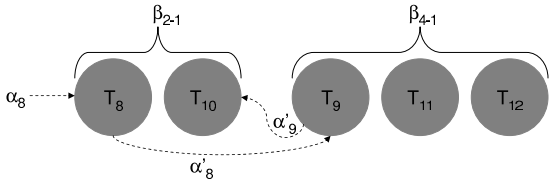

Figure 9. Inter-group data dependency.

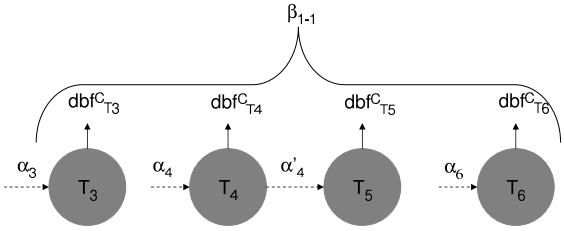

Figure 10. Intra-group data dependency.

References

[1] S. K. Baruah. Dynamic and static priority scheduling of recurring real-time tasks. *Real-Time Systems*, 24:93–128, 2003.

[2] S. Chakraborty, S. Künzli, and L. Thiele. A general framework for analysing system properties in platform-based embedded system design. *IEEE Design, Automation and Test in Europe (DATE)*, 2003.

[3] S. Chakraborty and L. Thiele. A new task model for streaming applications and its schedulability analysis. *IEEE Design, Automation and Test in Europe (DATE)*, 2005.

[4] A. Mok and D. Chen. Resource partition for real-time systems. *IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2001.

[5] I. Shin and I. Lee. Periodic resource model for compoisitional real-time guarantees. *IEEE Real-Time Systems Symposium (RTSS)*, 2003.

[6] I. Shin and I. Lee. Compositional real-time scheduling framework. *IEEE Real-Time Systems Symposium (RTSS)*, 2004.