Mälardalen University Press Licentiate Theses
No. 167

ADAPTIVE HIERARCHICAL SCHEDULING FRAMEWORK FOR REAL-TIME SYSTEMS

Nima Moghaddami Khalilzad

2013

School of Innovation, Design and Engineering

ISBN 978-91-7485-111-3
ISSN 1651-9256
Printed by Mälardalen University, Västerås, Sweden

Populärvetenskaplig sammanfattning

Moderna datorsystem är ofta utformade för att spela en mångsidig roll. De är därför kapabla att köra ett antal programvaror (program) samtidigt och parallellt. Dessa programvaror bör dela processorn så att alla av dem kan köra och avsluta sina beräkningar som förväntat. Å andra sidan, i de fall där programvaror har tidskrav innebär det att de inte bara ska slutföra beräkningarna som förväntat, utan även att dessa beräkningarna ska vara utförda i rätt tid. Således finns det ett behov av att i tid dela processorn bland olika program. Denna tidsdelning realiseras ofta genom att tilldela en fast och förutbestämd beräkningsresurs till varje program. Dock finns det en grupp av programvaror där, (i) deras efterfrågan på beräkningskraft förändras under körning, och/eller (ii) det kan tolereras att vissa beräkningar inte alltid utförs i tid. För system som innehåller program med dessa egenskaper är det inte effektivt att tilldela programmen med fasta beräkningsresurser. Om vi fördelar processorns beräkningsresurs baserad på den maximalt efterfrågade resursen för ett program, då kommer processorns beräkningskapacitet att gå till spillo för de fall då ett program kräver mindre än sin maximalt efterfrågade beräkningsresurs. För detta ändamål föreslår vi i denna avhandling ett adaptivt angreppsätt. I vårt föreslagna adaptiva system säkerställer vi, vid varje tidpunkt, att vi ger tillräckliga beräkningsresurser för varje program genom att följa den faktiska efterfrågan av beräkningsresurser. På så sätt kan vi integrera fler program i ett datorsystem som delas av flera programvaror, och samtidigt ge dessa programvaror garantier med avseende på timing.

Abstract

Modern computer systems are often designed to play a multipurpose role. Therefore, they are capable of running a number of software tasks (software programs) simultaneously in parallel. These software tasks should share the processor such that all of them run and finish their computations as expected. On the other hand, a number of software tasks have timing requirements meaning that they should not only access the processing unit, but this access should also be in a timely manner. Thus, there is a need to timely share the processor among different software programs (applications). The time-sharing is often realized by assigning a fixed and predefined processor time-portion to each application. However, there exists a group of applications where, (i) their processor demand is changing in a wide range during run-time, and (ii) their occasional timing violations can be tolerated. For systems that contain applications with the two aforementioned properties, it is not efficient to assign the applications with fixed processor time-portions. Because, if we allocate the processor resource based on the maximum resource demand of the applications, then the processor's computing capacity will be wasted during the time intervals where the applications will require a smaller portion than the maximum resource demand. To this end, in this thesis we propose adaptive processor time-portion assignments. In our adaptive scheme, we monitor the actual demand of the applications during run-time, and we provide sufficient processor time-portions for each application based on their monitored demand. In doing so, we are able to integrate more applications on a shared and resource constrained system, while at the same time providing the applications with timing guarantees.

To Arefeh, my kind-hearted wife
and my loving companion

Acknowledgments

First of all, I would like to offer my special thanks to my supervisors Prof. Thomas Nolte and Dr. Moris Behnam who have been supervising me from my master thesis. This licentiate thesis would not be possible without their support and encouragement. Thomas has always inspired me by his positive attitude that he brings to work, I also appreciate his incomparable support. I am particularly grateful for the useful critiques of Moris which have always improved my work.

Next, I wish to express my appreciation to the lecturers and professors at MDH, Damir, Ivica, Frank, Cristina, Thomas, Dag, Lars, Gordana, Jan, Ning, Andreas, Björn, Giacomo, Emma, Monica, Paul and Hans who I have learned a lot from during my graduate courses. I would also like to thank Carola, Susanne and other IDT administration staff for their help with practical issues.

Furthermore, many thanks goes to my office-mate Mikael who has guided me in evaluating my research through Linux implementations. In addition, I would like to thank Farhang, Lu, Hang, Mohammad, Sara, Meng, Daniel, Hamid, Sina, Dana, Rafia, Aida, Adnan, Leo, Svetlana, Saad, Abhilash, Mahnaz, Andreas, Mehrdad, Farhad, Huseyin, Ali, Gabriel, Eduard, Omar, Henrik, Raluca, Apala, Irfan, Jagadish, Kan, Jiale for the good company during courses, conference trips, PhD schools and/or lunches.

Last but not least, I would like to express my very great appreciation to my beloved wife, Arefeh, for the endless energy and love that she brings to my life, and acknowledge my parents' unwavering support.

Nima M. Khalilzad
Västerås, June, 2013

This work has been supported by the Swedish Research Council (Vetenskapsrådet) under the project ARROWS.

List of publications

Papers included in the licentiate thesis[1]

Paper A *Towards Adaptive Hierarchical Scheduling of Real-time Systems*. Nima Moghaddami Khalilzad, Thomas Nolte, Moris Behnam and Mikael Åsberg. In Proceedings of the 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'11), pages 1-8, September, 2011.

Paper B *Adaptive Hierarchical Scheduling Framework: Configuration and Evaluation*, Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte. Accepted for publication in the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'13), September, 2013.

Paper C *Bandwidth Adaptation in Hierarchical Scheduling Using Fuzzy Controllers*, Nima Moghaddami Khalilzad, Moris Behnam, Giacomo Spampinato and Thomas Nolte. In Proceedings of the 7th IEEE International Symposium on Industrial Embedded Systems (SIES'12), pages 148-157, June, 2012.

Paper D *Multi-Level Adaptive Hierarchical Scheduling Framework for Composing Real-Time Systems*, Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte. Technical Report, Mälardalen University, April, 2013.

Paper E *Implementation of the Multi-Level Adaptive Hierarchical Scheduling Framework*, Nima Moghaddami Khalilzad, Moris Behnam, and Thomas

[1]The included articles have been reformatted to comply with the licentiate thesis layout.

Nolte. Accepted for publication in the 9th annual workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT'13), July, 2013.

Additional papers, not included in the licentiate thesis

1. *Towards Implementation of Virtual-Clustered Multiprocessor Scheduling in Linux*, Syed Md Jakaria Abdullah, Nima Moghaddami Khalilzad, Moris Behnam, Thomas Nolte, accepted for publication in the 8th IEEE International Symposium on Industrial Embedded Systems (SIES'13), Work-in-Progress (WiP) session, June 2013.

2. *Exact and Approximate Supply Bound Function for Multiprocessor Periodic Resource Model: Unsynchronized Servers*, Nima Moghaddami Khalilzad, Moris Behnam, Thomas Nolte, In Proceedings of the 5th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS'12), December, 2012.

3. *Towards Adaptive Hierarchical Scheduling of Overloaded Real-Time Systems*, Nima Moghaddami Khalilzad, Thomas Nolte, Moris Behnam, In Proceedings of the 6th IEEE International Symposium on Industrial Embedded Systems (SIES'11), Work-in-Progress (WiP) session, June, 2011.

4. *On Adaptive Hierarchical Scheduling of Real-time Systems Using a Feedback Controller*, Nima Moghaddami Khalilzad, Moris Behnam, Thomas Nolte, Mikael Åsberg, In Proceedings of the 3rd Workshop on Adaptive and Reconfigurable Embedded Systems (APRES'11), April, 2011.

Contents

I

Thesis

Chapter 1

Introduction

In order to deal with the increasing complexity of real-time systems, hierarchical scheduling and compositional analysis techniques have been proposed and investigated for scheduling complex real-time systems consisting of multiple real-time components (subsystems) on a shared underlying hardware platform [1, 2]. Using such techniques, components are developed independently and their timing behaviors are studied in isolation, while the correctness of the system is inferred from the correctness of its components. In hierarchical frameworks, each component is allocated a portion of the CPU and, in turn, it guarantees that with this portion, all its internal tasks will be scheduled such that their corresponding timing constraints are respected. In addition, hierarchical scheduling provides temporal isolation among components, i.e., timing anomalies in one component are not propagated to the rest of the system.

The CPU portions allocated to the components are often specified by the component period which represents the frequency of CPU provision, and the component budget which indicates the amount of provided CPU in each period [3]. These two parameters are called interface parameters which indeed abstract the CPU resource requirements of the internal tasks of the components. The interface parameters can be calculated based on the worst case CPU demand of the tasks that exist in the components, and the worst case CPU supply that is going to be provided to the component. This approach guarantees that the timing requirements of the hard real-time systems are met. However, since the average CPU demand of the component tasks might be far less than their worst case CPU demands, and since the actual CPU provision to the component might be more than the assumed worst case, allocating resource using this ap-

proach might not be a cost-justifiable design approach when dealing with soft real-time components in which occasional deadline misses can be tolerated.

To this end, targeting dynamic soft real-time systems, i.e., systems where the CPU demand of their tasks is fluctuating, we propose using adaptive schemes in which the CPU portions are adapted during run-time using feedback loops that monitor the current load state of the components. The feedback loops are designed to manipulate the interface parameters of the components. The dynamic resource allocation techniques are especially efficient when the system components are composed of dynamic tasks in which their execution times are changing significantly during run-time. For example, when a component consists of control tasks controlling a physical plant, the CPU demand of such tasks depends on the current state of the physical plant. Also, in case of video decoder tasks, their CPU demand depends on the content of the input video. Thus, in such systems, the fixed CPU allocation techniques, which are based on the worst case scenario, are not efficient and may result in underutilized systems and consequently the CPU resource will be wasted.

1.1 Goal of the thesis

The goal of the thesis is to provide mechanisms and approaches to enable the adaptability feature in hierarchically scheduled systems in which a number of dynamic soft real-time tasks coexist on a shared underling hardware.

The objective of the adaptation is to keep the performance of the systems in an acceptable range while providing a minimum amount of resources to the systems. In doing so, more applications can be integrated on a single hardware platform. Hence, the production cost will be decreased. We are targeting the CPU resource in this thesis. The adaption is done through modification of the scheduling parameters used for expressing the CPU resource supply to the subsystems. The parameter modification essentially alters the amount of the CPU provision at run-time.

Following the main goal of the thesis, we have the following research subgoals, all in the context of hierarchical scheduling:

1. Designing adaptation schemes for adapting the bandwidth of the subsystems.

2. Providing a mechanism to deal with overload situations when the subsystems are not schedulable.

3. Finding an efficient configuration for the controller used in the adaptation scheme.

4. Designing a multi-level adaptive scheme and enabling the coexistence of hard real-time and soft-real time tasks through a mode change protocol.

5. Investigating the implementation complexity and the overhead of the adaptation scheme.

1.2 Research method

To achieve the goal of the thesis, we take the following two steps: (i) we design adaptive frameworks suitable for the context of hierarchical scheduling (ii) we evaluate the efficiency of the designed framework. For the first step we use the common methods available in the control theory literature and apply the available analysis to study some properties of the controlled systems.

The second step is to evaluate the designed scheme to examine the efficiency of the scheme. Two types of evaluations are used: (i) simulation based evaluation, and (ii) evaluation based on real implementations. For the simulation based studies, we simulate the scheduling of a number of dynamic applications which are equipped with feedback loops in an extended version of the TrueTime simulation tool [4]. We use parameters from real applications in the simulations to be able to generalize the results. For instance, we use the execution times gathered from running a video decoder application, and we feed them into our simulations. While, the implementation based evaluations are done using a Linux kernel loadable module which implements the adaptive hierarchical scheduler. Based on the simulation and experiment results, we often went back to the design and refined it to improve the performance of our adaptive scheme.

1.3 What is adaptive hierarchical scheduling?

In this section, we elaborate the two initial words of the thesis title (i) adaptive (ii) hierarchical.

We define adaptive scheduling as follows: adaptive scheduling is a scheduling scheme in which the scheduling mechanism, i.e., the policy and/or the parameters that affect the schedule, can be changed during run-time to deal with

different resource demand situations. From the definition, the workload variation is the property that causes the adaptation. Furthermore, fulfilling the tasks' real-time requirements is the objective of the adaptation. We achieve the scheduling behavior changes by modifying the scheduling parameters, more specifically, the interface parameters that abstract the resource demand of the components are adjusted.

The second term in the title is the word "hierarchical" which means that we are interested in systems where the scheduling is performed hierarchically. Although the available schedulability analysis for hierarchical systems are not directly used in our adaptive framework, we utilize the same presented models in this area. Therefore, hierarchical scheduling can be considered as the second dimension of the focus of the thesis.

1.3.1 Adaptive versus non-adaptive

A "non-adaptive" scheduling scheme is a scheme in which the scheduling policy and its corresponding scheduling parameters are decided at design-time and kept unchanged during run-time. The scheduling parameters are the parameters that are used for one of the following two reasons: (i) making scheduling decisions, i.e., selecting the next task that should be assigned to the CPU resource, (ii) triggering a scheduling event, e.g., releasing a task. For instance, when the scheduling policy is fixed priority scheduling, task priorities and task periods are the corresponding parameters that influence the scheduling behavior.

In contrast, in an adaptive scheduling scheme, the corresponding scheduling parameters and/or the scheduling policy itself can be changed during run-time. In the context of soft real-time systems, adaptive schemes are designed to increase the system performance while reducing the development costs. The performance metric can be timeliness, for example the deadline miss ratio of the application, while cost can be seen as the hardware requirement of systems. For example, on a given hardware platform we can run more real-time applications, correctly and concurrently, using adaptive schemes compared with using non-adaptive schemes.

1.3.2 Use cases of adaptive schemes

We now present a number of use cases in which adapting the scheduling behavior plays a key role in increasing the overall system performance.

An important use case of adaptive schemes can be found in scenarios that have the following properties.

- The system is composed of multiple applications that share the underling resources.
- The system is resource constrained, i.e, it has limited amount of resources.
- The applications utilizing shared resources have dynamic resource demands.

In such systems, in order to meet the real-time requirements of the applications, the scheduling behavior has to be adapted during run-time. Another property of the systems that might hold is that it may be very difficult or impossible to capture the system behavior at design-time (e.g., the task execution times may be difficult to estimate). This property makes the available schedulability analysis unsuitable. Because, in such analysis it is assumed that the worst case CPU demand is given. Therefore, when the system behavior is unknown, the system designer should either make unrealistic assumptions and then use the available analysis to design the system, or utilize adaptive schemes. The unrealistic assumptions can either be overestimations of the real parameters (e.g., real execution times) or underestimations of them. In either case, the system performance would be degraded at run-time. For instance, consider a video decoder task which is supposed to decode an unknown video stream. The system designer can not make realistic estimations on the execution times of such a task. Thus, for utilizing the analysis based CPU allocation techniques, the designer either has to overestimate the real execution times or underestimate them. In the former case, the CPU resource will be wasted, while in the latter case the task will suffer and miss its deadlines.

1.4 Contributions of the thesis

Following the main goal of this thesis, to provide mechanisms and approaches to enable the adaptability feature in hierarchically scheduled systems, we have addressed 5 sub-goals. Addressing Sub-goal 1 and 2, we first introduce an adaptive scheme based on conventional PI controllers in Paper A. Thereafter, Sub-goal 3 is addressed in Paper B which provides a statistical approach for configuring the adaptation scheme presented in Paper A. We investigate replacing the PI controller approach by the fuzzy controller approach in Paper C. This paper mainly addresses Sub-goal 1, however, Sub-goal 3 is also addressed in this paper. Paper D addresses Sub-goal 4 by introducing a multi-level adaptive scheme that targets composing hard and soft real-time systems. Finally,

Sub-goal 5 is addressed in Paper E in which the adaptive framework presented in Paper D is implemented in the Linux kernel.

1.5 Outline of the thesis

The thesis outline is as follows. In Chapter 2 we provide a brief background on our work. Chapter 3 presents the basics of our adaptive hierarchical scheduling framework which is common in all of the included papers. In Chapter 4, we summarize the contributions of this thesis and we provide a prospect of our work. An overview of included papers is presented in Chapter 5, while the included papers are presented in Chapter 6 to 10.

Chapter 2

Background

2.1 Real-time systems

Computational systems in which their correctness depend on both time and function are called *real-time systems*. In such systems, the timing behavior of the system is carefully analyzed to ensure its correctness. Thus, real-time systems have timing requirements that need to be fulfilled.

In real-time systems, different functionalities are realized through concurrent programs which are called *tasks*. Tasks often perform the same functionality repeatedly throughout the system's life-time. Each instance of a task execution is called a *job*. At each point in time, more than one job may be ready for execution. Therefore, the jobs should be scheduled in such way that the timing requirements of the real-time tasks are met.

2.1.1 Hard real-time systems

A group of real-time systems in which violation of timing requirements result in a catastrophic consequence are called hard real-time systems. Hence, when dealing with such systems, the corresponding timing analysis will ensure that there will be absolutely zero timing violations in the system. The Anti-lock Braking System (ABS) is an example of a hard real-time system in which incorrect timing may result in human losses.

2.1.2 Soft real-time systems

In soft real-time systems, the timing requirement violations only result in performance degradation. Although timing violations are not desirable, occasional violations can be tolerated in such systems. For instance, video players are considered as soft real-time systems. Because in such systems timeliness is crucial with respect to performance while incorrect timing has no catastrophic consequences such as loss of human lives.

2.2 Integrated real-time systems

In traditional computational systems, a single hardware platform is used for performing a single specific functionality. In other words, such systems are single purpose. However, recent advances in hardware technology enable the integration of multiple applications on a single hardware [5, 6]. When composing different systems on a single hardware, previously independent systems become subsystems of the new system. In such integrated systems, the timing analysis is performed compositionally, i.e., the timing correctness of the system is inferred from the timing correctness of its subsystems [3, 7, 8].

2.3 Hierarchical real-time systems

Hierarchical scheduling is a scheduling scheme in which the scheduling of real-time task is performed hierarchically. In such systems, the system is composed of a number of subsystems. The subsystems may also be composed of further sub-subsystems and/or tasks. The global scheduler schedules the subsystems. Whenever, a subsystem is assigned to the CPU, the scheduling is delegated to the subsystem and it will be responsible to schedule its inner sub-subsystems/tasks.

Hierarchical real-time systems [1, 2, 9, 10] are a group of integrated real-time systems in which (i) the composition of real-time tasks is performed in a hierarchical manner, (ii) the timing analysis is performed compositionally. Figure 2.1 illustrates a two-level hierarchical scheduling framework. The global scheduler schedules the subsystems according to the global scheduling policy. When a subsystem is assigned to the CPU, the local scheduler starts scheduling the subsystem's corresponding tasks according to the local scheduling policy.

In modeling hierarchical real-time systems, the system model often consists of two parts: resource supply model and task demand model. The re-

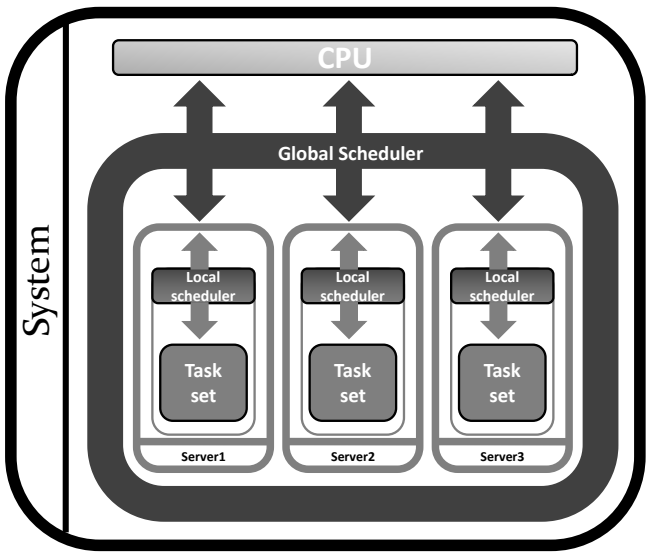

Figure 2.1: Hierarchical Scheduling Framework.

source supply model abstracts the underlying hardware resource such that each application has the illusion of running solo on an independent hardware. This virtual hardware is often realized through *servers* [1]. Servers are containers that contain tasks and/or servers. The resource supply model represents the minimum amount of resource that a server provides in a given time interval t. The amount of provided resource is often represented using a Supply Bound Function ($\texttt{sbf}(t)$). The resource demand model, however, represents the resource demand of real-time tasks. Similarly, the maximum demand in a given time interval t is often represented using a Demand Bound Function ($\texttt{dbf}(t)$) [11]. Consequently, the schedulability test is performed using the $\texttt{sbf}(t)$, which is dependent on the resource model, and the $\texttt{dbf}(t)$ which is dependent on the server's scheduling policy. An Earliest Deadline First (EDF) based scheduled system, for instance, is schedulable if the demand of its tasks does not exceed the resource supply at all time points ($\forall t \ : \texttt{dbf}(t) \leq \texttt{sbf}(t)$).

[1] Throughout the thesis, we use the terms "server" and "subsystem" interchangeably.

There are two sources of pessimism in the aforementioned schedulability analysis.

1. The pessimism that stems from the $\texttt{sbf}(t)$.

2. The pessimism that is originated from the $\texttt{dbf}(t)$.

When deriving the $\texttt{sbf}(t)$, the worst case scenario of resource supply is assumed. For instance, in the periodic resource model [3], the following three facts hold in the worst case scenario: (i) the first budget provision happens at the beginning of the server period (ii) the first task release happens exactly after the first budget depletion, (iii) the rest of the budget provisions happen as late as possible (at the end of server periods). This worst case scenario is illustrated in Figure 2.2.

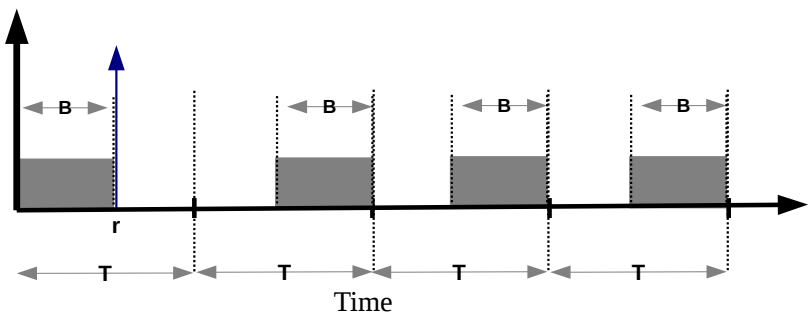

Figure 2.2: Worst case resource provision scenario. r, P and B represent task release, server period and server budget respectively.

On the other hand, the $\texttt{dbf}(t)$ is highly dependent on the tasks' worst case execution times. Figure 2.3 presents the execution times gathered from running a video decoder task. The figure shows that the CPU demand of such a task varies significantly from job to job. The figure also illustrates the vast difference between the worst case and average case execution times.

When dealing with hard real-time systems, considering the absolute worst case is necessary due to the potential catastrophic consequences of timing violations. However, in reality, the worst case is very unlikely to happen, therefore, the analysis is known to be pessimistic. Our goal is to remove the sources of pessimism and allow the soft real-time subsystems to efficiently utilized the CPU resource.

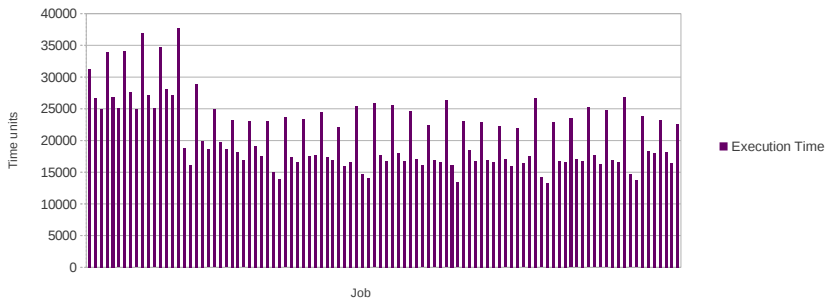

Figure 2.3: Execution times of a video decoder task.

2.4 Adaptive scheduling

In the context of soft real-time systems in which occasional timing violations can be tolerated, using conventional hierarchical scheduling analysis is not cost-justifiable due to the two sources of pessimism discussed in Section 2.3.

In such systems, in order to eliminate the two sources of pessimism, the resource demand and resource utilization can be monitored on-line while the system is running. Thereafter, based on the actual system state, realistic resource provisions can be performed. In doing so, we can avoid wasting the CPU resource.

Adaptive scheduling has been explored in the context of the AQuoSA [12] project. In addition, the ACTORS [13] project has added some multi-core flavor to adaptive scheduling. We would like to highlight that our proposed framework in this thesis is different from the ones that are presented in the context of AQuoSA and ACTORS in the sense that the scheduling is performed hierarchically. This new dimension strengthens our framework for being more suitable when dealing with complex compositional real-time systems.

Chapter 3

Adaptive Hierarchical Scheduling Framework

3.1 System model

In our framework, we assume single processor platforms in which a real-time system is composed out of multiple subsystems. The subsystems are scheduled using a so called "global scheduler". Furthermore, each subsystem consists of a number of sub-subsystems and/or real-time tasks together with a "local scheduler". The local scheduler schedules the inner sub-subsystems/tasks of the subsystems. The CPU portion of the components are adapted during run-time using budget (CPU portion) controllers that are attached to the subsystems. For example, Figure 3.1 illustrates the structure of our two-level Adaptive Hierarchical Scheduling Framework (AHSF). We extend this framework to a multi-level AHSF in Chapter 9 and Chapter 10.

3.2 Server model

In this thesis, we use the periodic servers [9] that are compliant with the periodic resource model [3]. The periodic servers provide their inner tasks with B units of the CPU time each P time units. The periodic servers idle their budget if there is no active task inside the server. The server budget is replenished to B at each server release event, thereafter, whenever the server is active the budget is decreased. When the budget is depleted, the server becomes inactive. Hence,

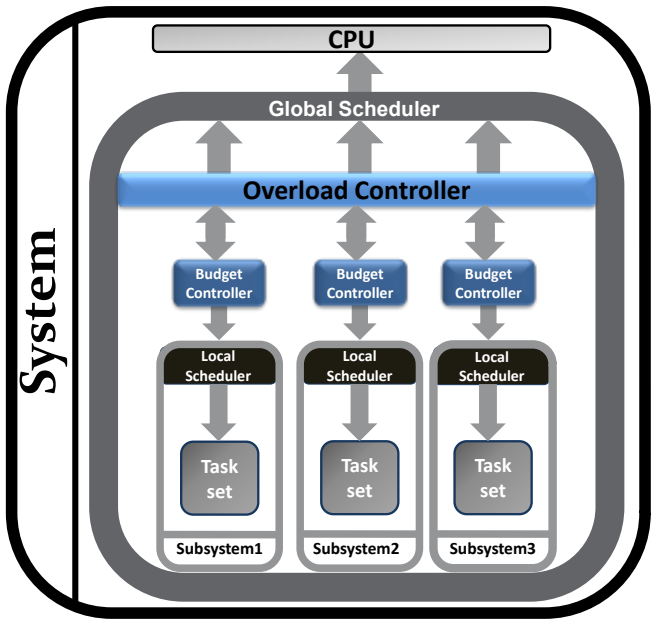

Figure 3.1: The Adaptive Hierarchical Scheduling Framework.

its inner tasks can not access the CPU. The relative importance of a server with respect to the other servers is denoted using the importance parameter ζ. This parameter is used in *overload* situations, that is when the subsystems are not schedulable.

3.3 Task model

Each subsystem consists of a set of tasks. We assume the periodic task model in which tasks are released periodically and at each period they execute for a number of time units. If the servers do not provide enough CPU resource, the tasks may miss their corresponding deadlines. However, the task deadline misses are observed by the budget controller component and as a consequence the budget controller will provide more resource (if available) to the subsystem when needed. Each task is a sequence of jobs.

3.4 Adaptation model

The adaptation is performed periodically by the budget adapter component. At each adaptation event, according to the load situation of the subsystems, a new budget is assigned to the subsystems. As shown in Figure 3.1, each server has its own budget controller component. Thus, the budgets are adapted independently in the different subsystems. However, there is an overload controller component which monitors the budgets and prevents an overload situation from happening by providing a budget ceiling for each server.

Throughout the thesis, we use different approaches for deriving the new budget at budget adaptation events. However, we use the same overload control technique in all papers which is based on the subsystem's importance (also called criticality) value. In short, in overload situations, we assign the budgets to the servers in their importance order, i.e, the low importance subsystems will be sacrificed for the purpose of serving high importance subsystems.

3.5 Performance metrics

As we use different budget adaptation schemes throughout the thesis, in order to compare their performance, we use two performance metrics. First of all we count the number of deadline misses during the experiment. The lower the number of deadline misses the better the performance. The number of deadline misses often is expressed in the form of deadline miss ratio, which is the number of missed deadlines divided by the number of successful jobs.

The second performance metric is the amount of wasted (idled) bandwidth by the servers. Recall that the periodic servers idle their budget if there is no active task. This metric can also be expressed by the amount of utilized budget since the sum of utilized budget and wasted budget is equal to the amount of the assigned budget.

3.6 Evaluation environment

We have evaluated the proposed framework using two approaches: (i) simulation studies (ii) implementation studies. Since there is no available framework which implements/simulates the behavior of an adaptive hierarchical scheduling framework, we had to either extend the available frameworks or to implement a new framework. In the case of using a simulation framework, we extended the TrueTime [4] simulator. However, for the implementation studies

we implemented a new framework from scratch. The evaluation environment setup comprises approximately 20 % of the time spent for completing this thesis. In this section, we provide an overview of the evaluation environments used in the thesis.

3.6.1 TrueTime

TrueTime is a simulator that uses the infrastructure available in MATLAB and Simulink. The tool is suitable for simulating the behavior of control tasks. TrueTime supports different scheduling algorithms. When it comes to reservation based algorithms, TrueTime supports the Constant Bandwidth Server (CBS) [14] and also the hard CBS [15]. We use the hard CBS to implement the periodic server [9]. Basically, we run an idle task inside each reservation to mimic the behavior of periodic servers. In addition, we have added a local scheduler inside the hard CBS such that the tasks attached to a hard CBS are scheduled according to the EDF scheduling policy.

3.6.2 `AdHierSched`

`AdHierSched` is the name of our Linux loadable kernel module that implements our multi-level adaptive hierarchical scheduler in the Linux kernel. When the real-time tasks attach themselves to `AdHierSched`, then the Linux kernel delegates the scheduling of these tasks to `AdHierSched`. For more details about this module refer to Chapter 10 (Paper E).

Chapter 4

Conclusion

4.1 Summary

In this thesis, we have provided a number of approaches for adapting the allocated CPU portions in a hierarchically scheduled system using on-line CPU demand monitoring.

First we proposed a PI controller-based approach in which the periodic server budgets are adapted as a result of the number of deadline misses and the amount of idled budget (Paper A). Thereafter, we have evaluated our framework through extensive simulations and we have proposed a statistical approach for configuring the framework (Paper B).

Furthermore, we have suggested a second approach which is based on fuzzy controllers, and we have tuned the controllers using a genetic algorithm (Paper C). Thereafter, we have introduced a multi-level scheme in which the budgets are adapted based on the predictions of a so called "workload estimator" component. This scheme is evaluated using our Linux implementation (Paper D).

Finally, we have presented the implementation details of our multi-level adaptive hierarchical scheduling framework (Paper E).

4.2 Discussion

The PI controller based adaptation approach relies on the approximate model of the system. Since the approximate model might not be valid for all systems,

we have proposed a model-free control scheme which uses fuzzy controllers for the adaptation purposes. Although, this controller does not require any model of the system, it has to be tuned for different scenarios. To overcome the tuning problem, we have proposed a third approach which neither needs a model of the system, nor does it require tuning. The third adaptation approach uses a workload estimator and, assuming that the next workload is known, it derives a new budget for the servers. The third adaptation approach also supports multi-level hierarchical systems.

Overall, we have approached the budget adaptation problem from different angles, and we believe that the proposed solutions have evolved over time. Nonetheless, each approach has its own advantages and disadvantages. For instance, for a known scenarios, the fuzzy controller scheme gives a good performance, while when the scenario is unknown then the predictor based solution is a better choice.

4.3 Future work

There are a number of paths that can be taken in the future.

- The adaptation scheme may be improved by using more sophisticated workload estimators than the one that we are currently using in Paper D. We are planning to explore this direction in the future.
- We are currently contemplating to extend our framework to incorporate multiprocessor architectures. More specifically, we are investigating using the multiprocessor periodic resource model [16] in which the interface parameters are being adapted during run-time.
- For the multiprocessor extension of our framework, we have to first implement the non-adaptive version of the multiprocessor hierarchical scheduler. Thus, we will implement the multiprocessor hierarchical scheduler in the Linux kernel.
- We would like to incorporate energy aspects into our framework and perform adaptations in such a way that in addition to the performance enhancement, the energy consumption is also optimized.
- We consider incorporating the network into our solution such that in addition to the performing adaptations with respect to the CPU resource, we also perform adaptation for controlling the network resource.

- If the probabilistic model of the dynamic workloads is available, we can provide probabilistic guarantees to the applications. Hence, it would be interesting to model the application workloads and use the probabilistic model to (i) better perform the adaptation, (ii) provide better performance guarantees.

Chapter 5

Overview of the papers

5.1 Contribution

The contribution of the thesis is the introduction of adaptation mechanisms for adapting the CPU portion of the components that are composed in a hierarchical manner on a shared processor. The proposed mechanisms are evaluated by either simulation studies or by experiments implemented in the Linux kernel. In the rest of this section, the contribution of each paper included in this thesis is explained individually.

5.1.1 Paper A

Towards Adaptive Hierarchical Scheduling of Real-Time Systems, Nima Moghaddami Khalilzad, Thomas Nolte, Moris Behnam, Mikael Åsberg, 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'11), pages 1-8, September, 2011.

In this paper we introduce an adaptation mechanism which is based on measuring the number of task deadline misses along with the unused bandwidth of subsystems, and based on this controlling the CPU portions using a PI controller.

My contribution

The idea of enabling adaptability in such a setting was suggested by Thomas and Moris. The research was conducted by me. Mikael provided the evaluation

framework. The evaluation framework was adapted by me to incorporate the budget adaptation mechanism.

5.1.2 Paper B

Adaptive Hierarchical Scheduling Framework: Configuration and Evaluation, Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte, accepted for publication in the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'13), September, 2013.

In this paper we focus on configuring our framework in such a way that the performance is enhanced. The paper suggests a statistical approach for finding the parameters that play an important role in increasing the performance of our framework.

My contribution

I was the main driver of the work. The co-authors contributed by discussions and reviewing the paper.

5.1.3 Paper C

Bandwidth Adaptation in Hierarchical Scheduling Using Fuzzy Controllers, Nima Moghaddami Khalilzad, Moris Behnam, Giacomo Spampinato, Thomas Nolte, 7th IEEE International Symposium on Industrial Embedded Systems (SIES'12), pages 148-157, June, 2012.

In this paper we replace the PI controller in the previous work with a fuzzy controller and instead of counting the number of deadline misses we measure the amount of used bandwidth after missing the deadline by the tasks. We evaluate this framework using an extended version of the TrueTime [4] simulation tool and we compare it against the PI controller approach presented in the previous work.

My contribution

I was the main driver of the work. The co-authors contributed by discussions and reviewing the paper.

5.1.4 Paper D

Multi-Level Adaptive Hierarchical Scheduling Framework for Composing Real-Time Systems, Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte, Technical Report, Mälardalen University, Västerås, Sweden, April, 2013.

In this paper we present a multi-level adaptive hierarchical scheduling framework in which we compose hard and soft real-time tasks together into the same system. The evaluations are performed using our Linux kernel loadable module which implements our adaptive framework.

My contribution

I was the main driver of the work. The co-authors contributed by discussions and reviewing the paper.

5.1.5 Paper E

Implementation of the Multi-Level Adaptive Hierarchical Scheduling Framework, Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte, accepted for publication in the 9th annual workshop on Operating Systems Platforms for Embedded Real-Time applications (OSPERT'13), July, 2013.

In this paper we reveal the details of the design decisions and implementation techniques of our Linux loadable module that implements our multi-level hierarchical scheduler with independent local schedulers. The framework is equipped with feedback loops which adapt the CPU bandwidth of the subsystems.

My contribution

I was the main driver of the work. The co-authors contributed by discussions and reviewing the paper.

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)*, pages 308–319, December 1997.

[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)*, pages 26–35, May 2001.

[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)*, pages 2–13, December 2003.

[4] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzen. How does control timing affect performance? analysis and simulation of timing using Jitterbug and TrueTime. *Control Systems, IEEE*, 23(3):16–30, June 2003.

[5] M. Di Natale and A.L. Sangiovanni-Vincentelli. Moving from federated to integrated architectures in automotive: The role of standards, methods and tools. *Proceedings of the IEEE*, 98(4):603–620, 2010.

[6] R. Obermaisser, C. El-Salloum, B. Huber, and Hermann Kopetz. From a federated to an integrated automotive architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(7):956–965, 2009.

[7] I. Shin and I. Lee. Compositional real-time scheduling framework with periodic model. *ACM Transactions on Embedded Computing Systems (TECS'08)*, pages 30:1–30:39, April 2008.

[8] A. Easwaran, M. Anand, and I. Lee. Compositional analysis framework using EDP resource models. In *Proceedings of the 28th IEEE Real-Time Systems Symposium, (RTSS'07)*, pages 129–138, December 2007.

[9] R.I. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. In *Proceedings of the 26th IEEE Real-Time Systems Symposium (RTSS'05)*, pages 10–398, December 2005.

[10] F. Zhang and A. Burns. Analysis of hierarchical EDF pre-emptive scheduling. In *Proceedings of the 28th IEEE International Real-Time Systems Symposium (RTSS'07)*, pages 423–434, December 2007.

[11] S.K. Baruah, A.K. Mok, and L.E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium (RTSS'90)*, pages 182 –190, December 1990.

[12] L. Palopoli, T. Cucinotta, L. Marzario, and G. Lipari. AQuoSA-adaptive quality of service architecture. *Softw. Pract. Exper.*, 39(1):1–31, January 2009.

[13] E. Bini, G. Buttazzo, J. Eker, S. Schorr, R. Guerra, G. Fohler, K.-E. Årzen, V. Romero, and C. Scordino. Resource management on multicore systems: The ACTORS approach. *Micro, IEEE*, 31(3):72–81, May-June 2011.

[14] L. Abeni and G. Buttazzo. Resource reservation in dynamic real-time systems. *Real-Time Systems*, 27(2):123–167, July 2004.

[15] L. Abeni, C. Scordino, and L. Palopoli. Serving non real-time tasks in a reservation environment. In *Proceedings of the 9th Real-Time Linux Workshop*, November 2007.

[16] 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)*, pages 181–190, July 2008.