Ilkogretim Online - Elementary Education Online, 2019; Vol 18 (Issue 4): pp. 2428-2444
http://ilkogretim-online.org
doi: 10.17051/ilkonline.2019.641238

Advanced Computational Elasticity For Real-Time Systems In Fog

SHAHBAZ HUSSAIN , Supervisor Dr. A. SUHASINI

ANNAMALAI UNIVERSITY.

Abstract: Real-time systems such as industrial robotsand autonomous navigation vehicles integrate a wide range of algorithms to achieve their functional behavior. In certain systems, these algorithms are deployed on dedicated single-core hardware plat- forms that exchange information over a real-time network. With theavailability of current multi-core platforms, there is growing interest in an integrated architecture where these algorithms can run on a shared hardware platform. In addition, the benefits of virtualization-based cloud and fog architectures for non-real-time applications have prompted discussions about the possibility of achieving similar benefits for real-time systems. Although manyuseful solutions such as resource reservations and hierarchical scheduling have been proposed to facilitate hardware virtualization for real-time applications, the current state of the art is mainly concerned with applications whose timing requirements can be modelled according to the periodic or the sporadic task model. Since the computational demand of many real-time algorithms can be flexibly adjusted at runtime, e.g., by changing the periods, they can be better abstracted with the elastic task model in the context of virtualized hardware platforms. Therefore, in this paper, scheduling framework with reservations based on periodic resource supply for real-time elastic applications with single-core workloads has been proposed, and then extend this solution for applications with multi-core workloads whereреservations are based on the minimum-parallelism model. Since many existing applications run on dedicated single-core platforms, simultaneously provided a systematic methodology for migrating an existing real- time software application from a single-core to a multi- core platform. In doing so, focused on recovering the architecture of the existing software and transforming it for implementation on a multi-core platform. Next, explored the advantages of a fog-based architecture over an existing robot control architecture and identify the key research challenges that must be addressed for the adoption of the fog computing architecture.

functions to hardware resources is accomplished at designtime. While this approach allows software performance to be optimized on the assigned hardware resources, it limits the ability to provide continuous improvements to users.

Keywords: Real time systems, Multicore platforms, Singlecore platforms, Fog computing architechture.

Introduction

The mapping of software functions to hardware resourcesin many real-time sys- tems follows a distributed model in which system functionality is executed on a network ofdedicated hardware

devices and the mapping of software.

In the form of software upgrades, which is typically possible with cloud-based systems. However, the unpredictable communication latencies associated with cloud computing prevent the direct deployment of cloud-based architectures for real-time systems. Fog computing[1], on the other hand, complements the cloud computing model [2] by bringing additional computing infrastructure geographically closer to the users. Thisarchitecture aims to meet the needs of latency-sensitive, real-time applications by reducing communication delaysand providing the potential benefits of cloud computing, such as on-demand resource availability, to real-time systems that have long run their software on their own hardware.

However, migrating existing real-time systems designed

for optimal perfor- mance on dedicated hardware resources to a fog-based infrastructure is a non-trivial undertaking. In particular, the fog infrastructure is assumed to consist of multi-core computing systems that are virtualized and shared among unrelated applications. Existing systems, on the other hand, are designed to run on single- processor hardware that is entirely dedicated to the application. In addition, current state-of-the-artsolutions such as reservation-based hierarchical scheduling [3] that address the resource allocation problem in virtualized systems assume that application workloads can be specified according to the sporadic or periodic task model. However, many real-time systems such as industrial robots and autonomous vehicles integrate a wide range of algorithms that have different computational requirements depending on the state of their operating environment. In the case of autonomous driving software, execution times for different functions such as perception and planning have been shown to vary significantly [4]. In an industrial robot, it has been

shown that periods of some tasks were dynamically adjusted while still achieving control goals [5]. The need to adjust the frequency of some tasks of a mobile robot to respond to dynamically changing environments of the robot was highlighted in [6]. In some scenarios, variability can be capped and the computational demand of the application can be modeled as a periodic or sporadic set of tasks while maintaining schedulability on a high-performance computing system. For some applications, it may not be possible to meet therequirements without overloading the resources. In such cases, it may be more appropriate to consider mode- based schedulability techniques [7, 8, 9]. In scenarios where such an approach is not possible, e.g., because it isdifficult to define different modes, or when schedulability cannot be guaranteed for a specific mode.

However, migrating existing real-time systems designed

for optimal perfor- mance on dedicated hardware resources to a fog-based infrastructure is a non-trivial undertaking. In particular, the fog infrastructure is assumed to consist of multi-core computing systems that are virtualized and shared among unrelated applications. Existing systems, on the other hand, are designed to run on single- processor hardware that is entirely dedicated to the application. In addition, current state-of-the-artsolutions such as reservation-based hierarchical scheduling [3] that address the resource allocation problem in virtualized systems assume that application workloads can be specified according to the sporadic or periodic task model. However, many real-time systems such as industrial robots and autonomous vehicles integrate a wide range of algorithms

that have different computational requirements depending on the state of their operating environment. In the case of autonomous driving software, execution times for different functions such as perception and planning have been shown to vary significantly [4]. In an industrial robot, it has been shown that periods of some tasks were dynamically adjusted while still achieving control goals [5]. The need to adjust the frequency of some tasks of a mobile robot to respond to dynamically changing environments of the robot was highlighted in [6]. In some scenarios, variability can be capped and the computational demand of the application can be modeled as a periodic or sporadic set of tasks while maintaining schedulability on a high-performance computing system. For some applications, it may not be possible to meet therequirements without overloading the resources. In such cases, it may be more appropriate to consider mode- based schedulability techniques [7, 8, 9]. In scenarios where such an approach is not possible, e.g., because it isdifficult to define different modes, or when schedulability cannot be guaranteed for a specific mode due to insufficient computational resources, it may be necessary to introduce some form of overload management techniques [10, 11], where the computational demand is adjusted at runtime to keep the application schedulable.

Concretely, the work in this thesis extends current techniques for integrat- ing real-time applications on a shared hardware platform through a hierarchical scheduling framework [3], where the application is encapsulated in a reservation server and uses the elastic task compression algorithms for overload manage- ment

[12] within the reservation. The framework proposes the use of the periodic resource supply reservation [13] for applications whose workload does not exceed that of a single core, and the minimum parallelism resource supplyform [14, 15] combined with the periodic resource supplymodel for applications with multipro- cessor workload. Concurrently, since there are considerable differences between the existing deployment model utilizing the single-core hardware platforms and the multi-core hardware platforms of fog computing, we provide a systematic methodology to support the transformation of single-core software systems into multi-core software systems, focusing on the recovery of the architecture of the existing software system. Furthermore, we propose a fog-based architecture for an industrial robot control software, compare it with an existing architecture of the control, and identify some of the research challenges that need to be solved for the deployment of the fog architecture in practice.

I. Background

This chapter provides an overview of the relevant background and work related to the contributions of the thesis.

A. Real-Time Systems

Real-time systems are characterized by the presence of some software functions where the computed results are useful only if they are available within certain time intervals [16]. To ensure the correctness of such functions(called tasks), it is nec- essary to estimate their computational demand and to provide adequate resources during the required time intervals. Resources can be provisioned based on worst- case computational demand or on probabilistic and average-case requirements [16]. The computational demand is usually expressed in terms of the execution time, i.e., the time required for the taskto execute on a processor, and the minimum interval between two successive instances of a task (also referred to as a period). The execution time of a task is estimated using timing analysis techniques [17] and the period

is usually defined at design time. A scheduling strategy thenallocates computational resources among the various tasks to meet the timing requirements.

B. Dynamic Computational Demand

Many real-time algorithms have computational demand that varies depending on their operating environment. The variability can be observed both in the execution times as well as in the periodicity of the tasks. For example, the execution time variability can be due to the different conditional branches in the code triggered by external events and the period variability can be due to different control laws. Mode-based analysis [7, 8, 9] isan approach to provision resources in systems with dynamic computational demand. In this approach, each mode can be defined by a specific set of tasks. When tasks exist in different modes, they can be distinguished by different execution times and periods. Schedulability is then evaluated for each mode as well as for the transition intervals. For systems where such an approach is not possible, e.g., because it is difficult to define different modes, or when schedulability within a mode cannot be guaranteed due to insufficient computational resources, it may be necessary to introduce some form ofoverload management to maintain the schedulability of the application [10, 11].

The Elastic Task Model

The elastic task model and associated workload modification algorithms [10, 12] provide an approach fordealing with overload situations in real-time systems.This model captures dynamic variability by allowing task parameters to be specified as a range of values rather than a single value as in the periodic and sporadic task models. It assumes that task parameters, such as the period and/or execution time, can be changed at runtime to any value within the predefined range. An additional parameter, the elastic coefficient, is associated with each task to indicate the relative flexibility of the task to changes in its timing parameters. At runtime, the workload modification algorithms attempt to keep the application schedulable by updating the timing parameters to be as close as possible to the desired values. In this work, we assume that real-time algorithmswith dynamic computational requirements can be modeled according to this elastic task model.

Reservation Servers

In addition to overload management, resource provisioning approaches should also provide mechanisms to manage overruns and ensure temporal isolation. A task overrun occurs when the actualexecution time exceeds the estimated worst-case execution times or when the the actual arrival times of consecutive instance of a task is less than the specified minimum inter-arrival time. In [11], the authors define temporal isolation/protection as:

The property of temporal protection requires that the temporal behav- ior of one task not be affected by the temporal behavior of the other tasks running in the system.

Reservation servers manage overruns by allocating a fixed amount of compu- tation time (called budget) to a task during certain time intervals (called a server period) to provide temporal isolation.

Reservation servers allow each task to run for the duration of its budget within a server period. If a task has not completed

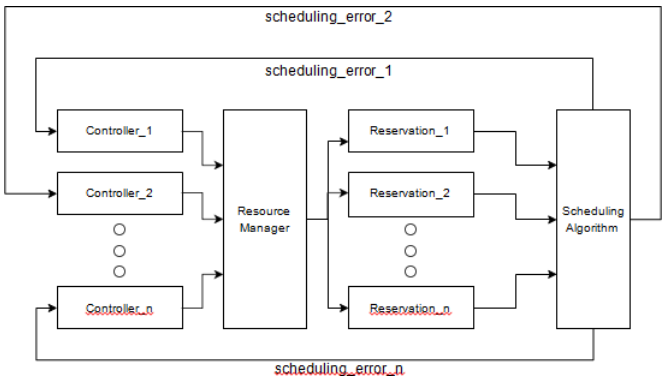

Fig. 2.1. Adaptive Reservation Architecture from [18,19].

its execution at the end of its budget, it is blocked untilthe next server period. If a task is released before the end of its minimum inter-arrival time, it will not be processed until the start of its next server period. This approach ensures that tasks that violate their timing specifications do not affect the temporal behavior of other tasks in the system [11]. For applications with dynamic computational demand, static allocation of budget and theperiod may not meet the performance needs of applications with dynamic computational demands[18].

Adaptive Reservation Servers

Adaptive reservation schemes address the needs of applications with dynamic computational demands by adjusting the budget allocated to a task at runtime. These schemes include a control mechanism that takes into account the spare capacity in the system and takes task execution times as feedback to change the budget assigned to a task[19, 18]. Fig.2.1 shows a general architecture of the adaptive reservation schemes based on the solutions in [19, 18]. Here, each task is assigned a reservation server and is connected to a controller that receives as input a scheduling error parameter (see [19]). The controller requests new reservation bandwidths to minimize the associated task's scheduling error. A system-level resource manager takes the controller's output and updates the reservation param- eters of the reservations in a way that minimizes the scheduling error and ensures schedulability of the changed reservations.

Fog Computing Platforms

Fog computing can be viewed as an extension of the cloud computing model, sharing its key characteristics, such as on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [2]. Additionally, it addresses the needs of latency- sensitive applications by allocating them on geographically closer resources to reduce communication latencies. The fog computing model envisions provisioning resources in hierarchical intermediate layers, where applications with shorter latencies are deployed on resources closer to the user andthose that can tolerate longer latencies are deployed on resources in distant layers[20]. The resources in these

intermediate layers are provided by hardware devices called Fog Computing Platforms or Fog Nodes. Within afog network, fog nodes can range from high- performance multi-core platforms with hardware accelerators to resource-constrained single-coreprocessors. To support a multi-tenant model, these fog nodes are assumed to provide access to their resources via hypervisors [1, 21] or container-based approaches [22, 23, 24]. Processing resources can be allocated to applications through reservation-based hierarchical scheduling (see Section. 2.3).

Hierarchical Scheduling

Hierarchical scheduling [25, 26] is a reservation-based scheduling approach that allocates processing resources for each application according to its reservation parameters. It supports the fog computing characteristics of multi-tenancy by sharing the physical resources among different applications. When an application's reservation server is scheduled, the application's local scheduler selects the task to run on the processing resource. For example, in a fog computing platform with hypervisor-based virtualization, the hypervisor scheduler selects the operating system to run on a processor at any given time, while the operating system scheduler decides which of its tasks to run on the processor. In this framework, the reservation parameters are defined taking into account all the tasks of the application, as opposed to the one-to-one allocation in single-level scheduling, where each task is assigned its own reservation. This approach also ensures temporal isolation between unrelated applications and limits any overruns to those applications that violate the resource demand assumptions, rather than propagating the impact to all the applications running on the shared resource. The current state of the art in hierarchical scheduling approaches mainly deals with applications whose resource demand can be specified according to the periodic and sporadic task model. Applications with multiple modes and varying computational demands are managed by changing resource reservations and ensuring their schedulability during transition intervals [9, 27, 28]. In cases where sufficient spare capacity is available, the bandwidth changes can be made without affecting the performance of concurrently running applications or alternatively, by taking into account their quality-of-service requirements [29]. An alternative mechanism to manage varying computational demand is to adjust the timing parameters of an application without changing the reservation parameters. Hierarchical adaptive reservation systems have been proposed in [30, 31] for applications with distinct modes. The goal of these solutions is to change the reservation parameters so that all applications sharing the resource can run in optimal modes by considering the computational needs of all applications. Inthis thesis, we extend this idea to support applications that do not have distinct modes, but rather are specified according to the elastic task model to enable the execution of such applications in a fog- computing environment.

Legacy Software Systems

The software of many existing real-time systems is constantly evolving to improve performance or introduce new features to meet user needs[32]. However, the software evolution has mostly focused on single-core hardware platforms. The computational capacities offered by these platforms may not be sufficient to inte- grate additional functions or computationally intensive algorithms. This can be

addressed by considering multi-core hardware platforms and new architectural paradigms such as fog computing. Developing new software from scratch for multi-core and fogcomputing platforms may not be economical, so reusing the existing code may be a better option [33]. However, these legacy systems suffer from issues such as lack of documentation[34] that hinder the understanding of the system's behavior. Moreover, the differences between single-core and multi-core platforms in terms of caches, memory buses, scheduling algorithms and data syn- chronization strategies make the migration process considerably complex. In this thesis, we provide a systematic migrationmethodology that focuses on architecture recovery and transformation to enable reuse of existing code.

II. Related Work

Hierarchical Scheduling

In a hierarchical scheduling framework, the computational needs of an application are abstracted by a single interface that specifies the computational time to be
reserved along with the time period in which it should be provisioned [13]. The reserved computing time can be made available to the application through various reservation servers such as the periodic server and the deferrable server [35]. The mechanisms for defining suchan interface and the schedulability tests vary depending on the scheduling strategies used for both local and global schedulers.

[53] For single-processor systems, Davis and Burns [35]
provided an exact schedu- lability test for a hierarchical system with fixed-priority preemptive schedulers (FP) for both local and global scheduling. They evaluated the schedulability under periodic servers, sporadic servers, and deferrable servers and showed that the periodic servers provide better schedulability compared to the deferrable and the sporadic server models. Similarly, Zhang and Burns [36] provided schedulability analysis for a system with the earliest deadline first (EDF) policy as the local scheduling policy, and either FP or EDF as the global scheduling policy. Mok and Alex [37] proposed the regularity-based resource supply model andprovided schedulability conditions for applications where either FP or EDF was used as the local scheduler. Shin and Lee [38] proposed the periodic resource supply model to define the guaranteed resource time to be provided to an application along with the schedulability tests when FP or EDF is used as the local scheduling policy. Easwaran et al [39] generalized the periodic resource model and provided methods to generate the optimal bandwidth interfaces and improve resource utiliza- tion. Dewan and Fisher [40, 41] proposed analgorithm to determine the optimal server parameters for the periodic resource model while Kim et al. [42] proposed a method to reduce the overhead associated with the periodic resource model. Yang and Dong. [43] considered scheduling of mixed criticality tasks within a periodic resource model where each resource supply reservation had multiple bandwidth estimates.

[54] For multiprocessor reservations, Leontyev and Anderson proposed the minimum-parallelism supply model to schedule soft real-time [14]. Yang and Anderson [15] provided conditions to preserve the optimality of the minimum- parallelism supply model for hard real-time tasks. Easwaran et al. [44] extended the periodic resource model with an additional parameter that specifies the max- imum number of physical processors that can be used to supply the reserved computation time at a given time. They investigated the schedulability of sporadic tasks within such a reservation using the global EDF [45] scheduling policy. In addition,they provided a transformation to generate equivalent periodic tasks for multiprocessor resource reservations (MPR) to be scheduled at the system level. Burmaykov et al [46]

proposed a generalization of the MPR model to reduce bandwidth allocation pessimism and provided schedulability conditions for both global EDF and globalFP scheduling policies. Pathan et al [47] proposed an overhead-aware interface generation method for multiprocessor reservations with global FP scheduling policies.

[55] Khalilzad et al. [48, 49, 27] proposed a feedback- based adaptive resource reservation scheme that adjusts the bandwidth of resource supply for variable tasks to minimize deadline overruns. Groesbrink et al. [29] considered a similar approach to variable taskmanagement in which bandwidth is adjusted so that eachserver receives a guaranteed minimum supply, while the remaining spare capacity is used to meet the demands of applications whose bandwidth should be increased. Cucinotta et al. [30] provided an adaptive scheme similarto the work presented in this thesis for applications with a finite set of modes and also consider power consumption as a constraint.

Legacy Software Migration

i.e., the adaptation of software to meet various requirements, such as the intro- duction of newer algorithms and the change of hardware platform. Menychtas et al. [50] presented a framework called ARTIST , a three-phase approach to software modernization with a focus on migration to the cloud. They divided migration efforts into three main phases, Pre-migration, Migration and Modernization, and Post-migration. For the pre-migration phase, they proposed the investigation of feasibility to considering the technical and economic constraints. The changes are made in the migration and modernization phase, and finally the system is deployed and validated in the post-migration phase. Erraguntla and Carver [51] discussed a three-phasemigration method consisting of analysis, synthesis, and transformation phases to migrate single-core to multi-coreparallel environments. The analysis and synthesis phases recover the design of the existing software, while the transformation phase of the migration method makes

recommendations for the multicore environment. They

have also provided a reverse engineering toolkit called RETK for the analysis and synthesis phases. Battaglia et al. [52] presented the RENAISSANCE method for re- engineering a legacy system. The method focuses on planning and managing the evolution process. Forite and Hug

[56] proposed the FASMM approach to reuse knowledgegained during migra- tion for reuse in other projects. Reussner et al. [54] and Wagner [34] proposed model- driven approaches for software migration. Their approachrequires reverse engineering the system with automated tools and capturing the information in modeling languages and using model-driven techniques to further maintain the system. Most of these works provide a general approach to software migration. In this thesis, we adapt these approaches for the migration of real-time software to multi-core platforms.

Fog Computing for Industrial Systems

Mohamed et al. [55] discussed fog-based solutions for multi- robot systems and Gudi et al. [56] highlighted the advantages of using a fog-based architecture for robotic applications that require human-robot interactions. Hao et al. [57] provided a generic software architecture for fog computing, while Faragardiet al. [58] provided a time predictable framework for a smart factory that integrates the fog and cloud layers. Skarin et al. [59]developed a testbed to investigate the feasibility of a fog-based approach for control applications, while Pallasch et al. [60] and Mubeen et al. [61] demonstrated the

feasibility of an edge-basedsolution for combining cloud and end devices. Ning et al. [62] considered fog computing in the context of smart traffic management and Barzegaran et al. [63] provided an industrial use-case for fog computing in which the electric drives functionas fog nodes.

III. Research Objectives

The objectives of this papers is to propose and evaluate solutions to integrate inde- pendent elastic real-time applications on fog computing platforms while satisfying their temporal requirements. To achieve this, define the following subgoals are defined:

RG1**:** *Provide a solution to schedule elastic real-time applications on virtual-ized single-core and multi-coreplatforms.*

The fog computing platforms are expected to host a wide range of applications, including those with real-time requirements. Since the resource requirements of applications may vary between those that require limited computation time on a singlecore and those that require multiple processors for their functional behavior, we formulate the research goal RG1 to address the scheduling of applications on both single-core and multi-core platforms.

RG2**:** *Propose solutions to guide the migration of existingreal-time software applications from single-core to multi-core platforms.*

Since many existing real-time applications are optimized for single-core and tworked system architectures, it may be necessary to transform their architec- tures for multi-core platforms before they can be deployed in a fog computing architecture. To support this transformation, we formulate the research goal RG2.

RG3**:** *Evaluate the advantages of a fog-based architecture for industrial robotic systems and identify research challenges.*

Industrial robots are used in many different fields, especiallyin the automotive industry. The system architecture of some existing robot controllers follows a net- worked approach, with software distributed across multiple single-core platforms. To investigate the advantages of fog computing architecture over such existing architectures, we formulate the research goal RG3. Furthermore, we extend the scope of this goal to identify the research challenges associated with fog computing architectures so that such an architecture can be realized in practice.

IV. Research Process

The thesis results were developed following the hypothetico-deductive method and consisted of the following four steps.

- Problem Definition - Understand the field of the topic through literature review, state-of-art and state- of-practice studies, and defining the scope of the problem.

- Idea Development - Iterative development of solutions to the defined prob- lem.

- Implementation - Converting the idea and theory into an artefact.

- Evaluation - Evaluate the idea and its implementation and draw conclusions.

Problem Definition In this thesis, we define theproblems for our research to achieve the research goals described in Section. 3.1. The problems were based on literature review and inputs from industrial experts.

Idea Development and implementation The solutions to the identified problems were iteratively refined by evaluating the solutions using different research methods. A mapping of the different research methods used to achieve the research goals is provided in Table 3.1.

Evaluation The evaluation of the solutions developed foreach of the research goals was done through a comparative analysis of existing state-of-art solutions for RG1, and state-of-practice for RG3. The evaluation ofthe methodology developed for RG2 was evaluated usinga survey-based approach, which is described in detail in Paper C (Chapter 7).

Research Methods	**Research Goals**
State of art study	RG_1,RG_2,RG_3
State of Practicestudy	RG_2,RG_3
Simulation	RG_1
Case Study	RG_2, RG_3
Survey	RG_2

Table 3.1. Mapping of Research Methods and ResearchGoals

	RG1	RG2	RG3
C1	X		
C2	X		
C3		X	
C4			X

Table 3.2. Mapping between the Contributions C1 through C4 and the research goals RG1 through RG3.

Technical Contributions

Here we outline the technical contributions of the thesis and then summarize each contribution. A mapping of the research goals to the technical contributions is shown in Table 3.2.

C_1: A reservation-based scheduling framework for executing elastic real-time applications on a uniprocessor system.

C_2: A reservation-based scheduling framework for executing elastic real-time applications on a multiprocessor system.

C_3: A systematic methodology to migrate from a single-core to a multi-core architecture with maximum software reuse and minimal reengineering effort for real-time systems.

C_4: A fog-based architecture for industrial robotic systems and identification of research challenges.

C_1: A reservation-based scheduling framework for executing elastic real-time applications on a uniprocessor system.

This contribution addresses the research goal RG1 and is provided in Paper A. Here we consider a system model in which an application is modeled after the elastic task model[10] and is assumed to execute within a reservation server. The reservation server is designed according to the periodic resource supply form (PRM) [13]. The application can include either the fixed-priority rate monotonic scheduler or the dynamic earliest-deadline-first priority scheduler. The design parameters of the reservation server, i.e., the budget and the period of the server, are set taking into account the initial desired utilization and period of the application tasks. Whenever an application task requests a change in its period during runtime, the reserved server bandwidth must be updated. Such a bandwidth update can be performed without affecting the bandwidth of other reservations if there is sufficient spare capacity in the system. However, if there is insufficient spare capacity, there may be no choice but to adjust the bandwidth of all other concurrent servers. To minimize such bandwidth changes, a utilization modification algorithm, a variant of the original elastic compression algorithm[10], is integrated within the application to generate a new set of periods for the application's tasks that satisfy the constraints of each of these tasks. Since the reservation is provided according to the PRM form, there is a possibility that, in the worst case, the resource will not be available to the application for a certain duration. If the newly generated task periods are shorter than the duration of unavailability, there may be no other option but to adjust the server bandwidth. However, if no tasks have their periods less than or equal to the unavailable duration and the newly generated periods satisfya utilization-based schedulability test, the server's bandwidth may remain unchanged, limiting the need to adjust the server's bandwidth. The evaluation in Paper A shows that the proposed approach works well for many task sets, i.e., the utilization adjustment within the application can keep the tasks schedulable within the reservation, but forcertain task sets, the bandwidth adjustments may be unavoidable and the bandwidth needs to be readjusted for each period change request.

C_2: A reservation-based scheduling framework for executing elastic real-time applications on a multiprocessor system.

This contribution addresses the research goal RG1 and is provided in Paper B. Here we consider a system model where an application is modeled after the elastic task model and assumed to execute within a reservation server. The reservation server is designed according to the multiprocessor minimum-parallelism resource supply form[14, 15]. This reservation model provides an application with a fixed number of fully dedicated processors and at most one partial processor that is available according to the periodic resource supply model. Each reservation includes a local scheduling policy to schedule the application tasks. In Paper B, the partitioned EDF scheduling policy is considered as the local scheduling policy. Similar to the uniprocessor scheduling framework discussed earlier, the reservation parameters are set based on the initially desired utilization and period of the application's tasks. Once the initial reservation parameters are set, the goal is to minimize the frequency of reservation bandwidth changes. To do so, each time a task requests a

change in its period, the utilization modification algorithm considers only the tasks that share the processor with the task requesting the change and attempts to generate new periods that satisfy the constraints of each of these tasks. If no solution is found, the algorithm re-partitionsthe tasks among the processors, including the partial processor. The re-partitioning can be done according to any reasonable allocation scheme [65]. If the re-partitioning approach fails, the reservation bandwidth is considered for modification. The change may take the form of a modified partial processor bandwidth or, if possible, the allocation of a new, fully dedicated processor. The evaluation in Paper B shows that up to ninety-four percent of requests can be satisfied with the per-core utilization modification scheme and one hundred percent of requests when combined with there-partitioning step.

C_3: A software engineering methodology to migrate legacy real-time systems to multi-processor platforms

This contribution addresses the research goal RG_3 and constitutes the content of Paper C. Here we provide a migration methodology as many existing real- time industrial systems have software designed to run on single-core platforms, and they typically rely on a network of multiple single-core devices to realize complex functions. With multiprocessor platforms, the network-based system architecture can be replaced by an integrated approach in which a multiprocessor hardware platform is shared by the software to reduce communication latencies. For some systems where such an integrated approach is being considered, a complete redesign for multiprocessor platforms may be beneficial; for other systems, reuse of previously developed code may be preferred to a complete redesign.

Doing so successfully, however, requires a well-defined methodology. In Paper C, we propose a systematic approach that defines a set of processes required to enable the reuse of existing code on a multiprocessor platform. We also present some of the available tools that can be used in this transition. The proposed approach is tailored to systems with real-time requirements and provides a three-step workflow that starts with architecture migration, followed by implementation- level processes, and finally validation of the transformed architecture. The validity of this approach is evaluated using a survey developed following the guidelines in [66]. The survey questions were intended to assess the feasibility, easeof use, and usefulness of the proposed methodology. The results of the survey indicated that the proposed approach met the evaluation objectives and that the methodology would need to be adapted for individual cases to address the constraints of different systems.

C_4: A fog-based architecture for industrial robotic systems

This contribution addresses the research goal RG3 and is the main content of Paper It demonstrates the advantages of a fog- based architecture for an industrial robotic system. The advantages are discussed through a comparative study in which an existing robotic system was analyzed and some of its limitations were identified. Based on this analysis, a fog-based architecture was proposed that eliminates these limitations. The proposed fog architecture shifts much of the existing system functionality from a dedicated hardware platform to computational resources in the fog layer. However, since the fog computing paradigm is relatively new, there are still significant issues that need to be explored for implementing such an architecture in practice. Therefore, a number of research challenges have been identified related to isolating and virtualizing resources to deploy real-time applications in the fog layer and orchestrating real-time workloads among fog computing

resources to improve efficiency and performance.

V. Conclusion

In this thesis, we addressed the problem of scheduling real- time applications with variable timing requirements on virtualized hardware platforms and proposed a scheduling framework that minimizes the frequency of system-level bandwidth changes on single-processor and multiprocessor platforms. Simultaneously, we ad- dressed the reuse of existing application code developed for single-core platforms on multi-core platforms and proposed a systematic methodology for managing migration, focusing on architecture recovery and its transformation. The method- ology was evaluated for feasibility, usefulness, and usability through a survey. In addition, we considered the limitations of an existing architecture for industrial robot control systems and proposed a fog-based architecture that addresses these limitations and identified some of the research challenges that should be addressed to implement such an architecture in practice.

For future work, we will investigate the worst-case performance of the proposed heduling framework and consider alternative reservation strategies such as the GMPRinterface [46] and the possibility of virtualized access to accelerators such as FPGAs among elastic real-time applications.

VI. References

[1] IEEE standard for adoption of openfog reference architecture for fog com- puting. IEEE Std 1934-2018, pages 1–176, 2018.

[2] Peter Mell and Tim Grance. The NIST definition of cloud computing. 2011.

[3] Z. Deng and J.W.-S. Liu. Scheduling real-time applications in an open environment. In Proceedings Real-Time Systems Symposium, pages 308–319, 1997.

[4] Miguel Alcon, Hamid Tabani, Leonidas Kosmidis, Enrico Mezzetti, Jaume Abella, and Francisco J Cazorla. Timing of autonomous driving software: Problem analysis and prospects for future solutions. In 2020 IEEE Real- Time and Embedded Technology and Applications Symposium (RTAS), pages 267–280. IEEE, 2020.

[5] D. Simon, D. Robert, and O. Sename. Robust control/scheduling co-design: application torobot control. In 11th IEEE Real Time and Embedded Tech- nology and Applications Symposium, pages 118–127, 2005.

[6] Ala' Qadi, Steve Goddard, Jiangyang Huang, and Shane Farritor. Mod- elling computational requirements of mobile robotic systems using zones and processing windows. Real-Time Systems, 42(1):1–33, 2009.

[7] P. Pedro and A. Burns. Schedulability analysis for mode changes in flexible real-time systems. In Proceeding. 10th EUROMICRO Workshop on Real- Time Systems (Cat. No.98EX168), pages 172–179, 1998.

[8] Jorge Real and Alfons Crespo. Mode Change Protocols for Real-Time Sys- tems: A Survey and a New Proposal. Real-Time Systems, 26(2):161–197, 2004.

[9] Luca Santinelli, Giorgio C. Buttazzo, and Enrico Bini. Multi-moded resource reservations. In IEEEReal-Time and Embedded Technology andApplications Symposium, pages 37–46. IEEE Computer Society, 2011.

[10] Giorgio C. Buttazzo, Giuseppe Lipari, and Luca Abeni. Elastic task modelfor adaptive ratecontrol. In

Proceedings - Real-Time Systems Symposium, pages 286–295. IEEE, 1998.
[11] Giorgio Buttazzo, Giuseppe Lipari, Luca Abeni, and Marco Caccamo. Soft Real-Time Systems. Springer, 2005.
[12] Giorgio C. Buttazzo, Giuseppe Lipari, Marco Caccamo, and Luca Abeni. Elastic scheduling for flexible workload management. IEEE Transactions on Computers, 51(3):289–302, 2002.
[13] Insik Shin and Insup Lee. Compositional real- time scheduling framework with periodic model. ACM Trans. Embedded Comput. Syst.,7(3):30:1–30:39, 2008.
[14] Hennadiy Leontyev and James H. Anderson. A hierarchical multiproces- sor bandwidth reservation scheme with timing guarantees. Proceedings - Euromicro Conference on Real- Time Systems, pages 191–200, 2008.
[15] Kecheng Yang and James H. Anderson. On the Dominance of Minimum- Parallelism Multiprocessor Supply. Proceedings - Real-Time Systems Sym-posium, 0:215–226, 2016.
[16] Hermann Kopetz. The Real-Time Environment, pages 1–28. Springer US, Boston, MA, 2011.
[17] Claire Maiza, Hamza Rihani, Juan M. Rivas, Joël Goossens, Sebastian Alt- meyer, and Robert I. Davis. A survey of timing verification techniques for multi-core real-time systems. ACM Comput. Surv., 52(3):56:1–56:38, 2019.
[18] Rodrigo Santos, Giuseppe Lipari, Enrico Bini, and Tommaso Cucinotta. On- line schedulability tests for adaptive reservations in fixed priority scheduling. Real-Time Systems, 48(5):601–634,2012.
[19] Luca Abeni, Tommaso Cucinotta, Giuseppe Lipari, Luca Marzario, and Luigi Palopoli. Qos management through adaptive reservations. Real-TimeSystems, 29(2-3):131–155, 2005.
[20] Michaela Iorga, Larry Feldman, Robert Barton, Michael J Martin, Nedim S Goren, Charif Mahmoudi, et al. Fog computing conceptual model. 2018.
[21] Paul Pop, Bahram Zarrin, Mohammadreza Barzegaran, Stefan Schulte, SasikumarPunnekkat, Jan Ruh, and Wilfried Steiner. The fora fog computing platform for industrial iot. Information Systems, 98:101727, 2021.
[22] Václav Struhár, Moris Behnam, Mohammad Ashjaei, and Alessandro V Papadopoulos. Real-time containers: A survey. In 2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020). Schloss Dagstuhl-Leibniz- Zentrum für Informatik, 2020.
[23] Zeinab Bakhshi, Guillermo Rodriguez-Navas, and Hans Hansson. Fault- tolerant permanent storage for container-based fog architectures. In 2021 22nd IEEE International Conference on Industrial Technology (ICIT), vol- ume 1, pages 722–729, 2021.
[24] Volkan Gezer. A Modular and Scalable Software Reference Architecture for Decentralized Real-Time Execution on Edge Computing. A Modular and Scalable Software Reference Architecture for Decentralized Real-TimeExecution on Edge Computing, 2021.
[25] Moris Behnam, Insik Shin, Thomas Nolte, and Mikael Nolin. SIRAP: A Synchronization Protocol for Hierarchical Resource Sharing in Real-Time Open Systems. In Christoph M. Kirsch and Reinhard Wilhelm, editors, Proceedings of the 7th ACM & IEEE international conference on Embedded software - EMSOFT '07, pages 279–288, New York, New York, USA, 2007. ACM Press.
[26] Rafia Inam. Hierarchical scheduling for predictable execution of real-time software components and legacy systems. PhD thesis, Mälardalen University, December 2014.
[27] Nima Khalilzad, Fanxin Kong, Xue Liu, Moris Behnam, and Thomas Nolte. A feedback scheduling framework for component-based soft real-time sys- tems. In 21st IEEE Real- Time and Embedded

Technology andApplicationsSymposium, pages 182–193, 2015.

[28] Wei-Ju Chen, Peng Wu, Pei-Chi Huang, AloysiusK Mok, and Song Han. On- line reconfiguration of regularity-based resource partitions in cyber- physical systems. Real-Time Systems, pages 1–44, 2021.

[29]Stefan Groesbrink, Luis Almeida, Mario DeSousa, and Stefan M. Pet- ters. Towards certifiable adaptive reservations for hypervisor- based vir- tualization. Real-Time Technology and Applications - Proceedings, 2014- October(October):13–24, 2014.

[30]Tommaso Cucinotta, Luigi Palopoli, LucaAbeni, Dario Faggioli, and Giuseppe Lipari.On the integration of application level and resource level QoS control for real-time applications. IEEE Transactions on Industrial Informatics, 6(4):479–491, 2010.

[31] Vladimir Nikolov, Stefan Wesner, Eugen Frasch, and Franz J. Hauck. A hierarchical scheduling model for dynamic soft-realtime systems. Leibniz International Proceedings in Informatics, LIPIcs, 76(01):71–723, 2017.

[32]Ned Chapin, Joanne E. Hale, Khaled Md. Kham,Juan F. Ramil, and Wui-Gee Tan. Types of software evolution and software maintenance. Journal of Software Maintenance, 13(1):3–30, January 2001.

[33]Irune Yarza Perez. Legacy software migrationbased on timing contract aware real-time execution environments. PhD thesis, Universität Oldenburg, 2020.

[34]Christian Wagner. Model-Driven Software Migration: A Methodology. Springer Fachmedien Wiesbaden, Wiesbaden, 2014.

[35]R. I. Davis and A. Burns. Hierarchical fixed priority pre-emptive scheduling. Proceedings - Real-Time Systems Symposium,2005.

[36] Fengxiang Zhang and Alan Burns. Analysis of hierarchical edf pre-emptive scheduling. In 28th IEEE International Real-Time Systems Symposium (RTSS 2007), pages 423–434, 2007.

[37] Aloysius K Mok and Xiang Alex. Towards compositionality in real-time resource partitioning based on regularity bounds. In Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001)(Cat. No. 01PR1420), pages 129–138. IEEE, 2001.

[38] Insik Shin and Insup Lee. Periodic resource model for compositional real- time guarantees. InRTSS, pages 2–13. IEEE Computer Society, 2003.

[39] Arvind Easwaran, Insup Lee, Insik Shin, and Oleg Sokolsky. Compositional schedulability analysis of hierarchical real-time systems. In ISORC, pages274–281. IEEE Computer Society, 2007.

[40] Nathan Fisher and Farhana Dewan. A bandwidth allocation scheme for compositional real-time systems with periodic resources. Real Time Syst., 48(3):223–263, 2012.

[41] Farhana Dewan and Nathan Fisher. Bandwidth allocation for fixed-priority- scheduled compositional real-time systems. ACM Trans. Embed. Comput. Syst., 13(4):91:1–91:29, 2014.

[42] Jin Hyun Kim, Kyong Hoon Kim, Arvind Easwaran, and Insup Lee. Towards overhead- free interface theory for compositional hierarchical real-time sys- tems. IEEE Trans. on CAD of Integrated Circuits and Systems,37(11):2869– 2880, 2018.

[43] Kecheng Yang and Zheng Dong. Mixed-criticality scheduling in compo- sitional real- time systems with multiple budget estimates.In 2020 IEEE Real-Time Systems Symposium (RTSS), pages 25–37. IEEE, 2020.

[44] Arvind Easwaran, Insup Lee, Oleg Sokolsky, and Steve Vestal. A composi- tional schedulingframework for digital avionics systems. In RTCSA, pages 371–380. IEEE Computer Society, 2009.

[45] Sanjoy Baruah, Marko Bertogna, and Giorgio **C.** Buttazzo. Multiprocessor scheduling for real-time systems. Embedded systems. Springer, Cham, 2015.

[46] Artem Burmyakov, Enrico Bini, and Eduardo Tovar. Compositional multi- processor scheduling: the GMPR interface. REAL-TIME SYSTEMS, 50(3, SI):342–376, MAY 2014.

[47] Risat Mahmud Pathan, Per Stenström, Lars Göran Green, Torbjörn Hult, and Patrik Sandin. Overhead-aware temporal partitioning on multicore processors. Real-Time Technology and Applications - Proceedings, 2014- Octob(October):251–262, 2014.

[48] Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte. Adaptive hierarchical scheduling framework: Configuration and evaluation. In ETFA,pages 1–10. IEEE, 2013.

[49] Nima Moghaddami Khalilzad, Moris Behnam, and Thomas Nolte. Multi- level adaptive hierarchical scheduling framework for composing real-time systems. In RTCSA, pages 320–329. IEEE Computer Society, 2013.

[50] Andreas Menychtas, Kleopatra Konstanteli, Juncal Alonso, Leire Orue- Echevarria, Jesus Gorronogoitia, George Kousiouris, Christina Santzaridou, Hugo Bruneliere, Bram Pellens, Peter Stuer, Oliver Strauss, Tatiana Senkova, and Theodora Varvarigou. Software modernization and cloudification using the ARTIST migrationmethodology and framework. Scalable Computing: Practice and Experience, 15(2), 2014.

[51] Ravi Erraguntla and Doris L. Carver.Migration of sequential systems to parallel environments by reverse engineering. Information & Software Tech- nology, 40(7):369–380, 1998.

[52]M. Battaglia, G. Savoia, and J. Favaro.Renaissance: a method to migrate from legacyto immortal software systems. In Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering,pages197–200, 1998.

[53] Louis Forite and Charlotte Hug. FASMM: Fast and Accessible Software Mi- gration Method. In 2014 IEEE Eighth International Conference on Research Challenges in Information Science, pages 1–12. IEEE, 2014.

[54] Ralf Reussner, Michael Goedicke Wilhelm Hasselbring, Birgit Vogel-Heuser, Jan Keim, Lukas Märtin, editor. Managed Software Evolution. Springer Nature Switzerland AG, 2019.

[55] Nader Mohamed, Jameela Al-Jaroodi, and Imad Jawhar. Fog-Enabled Multi- Robot Systems. In 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), pages 1–10. IEEE, 2018.

[56] Siva Leela Krishna Chand Gudi, Suman Ojha, Benjamin Johnston, Jesse Clark, and Mary-Anne Williams. Fog Robotics for Efficient, Fluent and Ro- bust Human-Robot Interaction. In 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), pages 1–5. IEEE, 2018.

[57] Zijiang Hao, Ed Novak, Shanhe Yi, and Qun Li. Challenges and Software Architecture for Fog Computing. IEEE Internet Computing, 21(2):44– 53, 2017.

[58] Hamid Reza Faragardi, Saeid Dehnavi, Mehdi Kargahi, Alessandro V. Pa- padopoulos, and Thomas Nolte. A time-predictable fog-integrated cloud framework: One step forward in the deployment of a smart factory. In 2018 Real-Timeand Embedded Systems and Technologies (RTEST), pages 54–62.IEEE, 2018.

[59] Per Skarin, William Tarneberg, Karl-Erik Å rzén,and Maria Kihl. TowardsMission-Critical Controlat the Edge and Over 5G. In 2018 IEEE Interna- tional Conference on Edge Computing (EDGE), pages 50–57. IEEE, 2018.

[60] Christoph Pallasch, Stephan Wein, Nicolai Hoffmann, Markus Obdenbusch, Tilman Buchner, Josef

Waltl, and Christian Brecher. Edge Powered Industrial Control: Concept for Combining Cloud and Automation Technologies. In 2018 IEEE International Conference on Edge Computing (EDGE), pages130–134. IEEE, 2018.

[61] S. Mubeen, P. Nikolaidis, A. Didic, H. Pei-Breivold, K. Sandström, and M. Behnam. Delay mitigation in offloaded cloud controllers inindustrial iot. IEEE Access, pages 4418–4430, 2017.

[62] Zhaolong Ning, Jun Huang, and Xiaojie Wang.Vehicular fog computing: Enabling real-time traffic management for smart cities. IEEE Wireless Com- munications, 26(1):87–93, 2019.

[63] Mohammadreza Barzegaran, Nitin Desai, JiaQian, Koen Tange, Bahram Zarrin, Paul Pop, and Juha Kuusela. Fogification of electric drives: An industrial use case. In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA),volume 1, pages 77–84, 2020.

[64] Gordana Dodig-crnkovic. Scientific methodsin computer science. In In Proc. PROMOTE IT2002, 2nd Conference for the Promotion ofResearch in IT at New Universities and at University Colleges in, pages 22–24, 2002.

[65] José María López, José Luis Díaz, and Daniel F. García. Utilization bounds for EDF scheduling on real-time multiprocessor systems. RealTime Syst., 28(1):39–68, 2004.

[66] Barbara A Kitchenham and Shari L Pfleeger. Personal opinion surveys. In Guide to Advanced Empirical Software Engineering, pages 63–92.Springer London, London, 2008.

[67] Shaik Mohammed Salman, Saad Mubeen, FilipMarković, Alessandro V Papadopoulos, and Thomas Nolte. Scheduling elastic applications in compo- sitional real-time systems. In 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pages 1–8. IEEE,2021.

[68] Shaik Mohammed Salman, Alessandro V Papadopoulos, Saad Mubeen, and Thomas Nolte. Multi-processor scheduling of elastic applications in com- positional real-time systems. Journal of Systems Architecture,122:102358,2022.

[69] Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. A systematic methodology to migrate complex real-time software systems to multi- core platforms. Journal of Systems Architecture,117(March):102087, 2021.

[70] Shaik Mohammed Salman, Vaclav Struhar, Alessandro V. Papadopoulos, Moris Behnam, and Thomas Nolte. Fogification of industrial robotic systems: Research challenges. IoT-Fog 2019 - Proceedings of the 2019 Workshop on Fog Computing and the IoT, pages 41–45, 2019.